MÜHENDİSLİK FAKÜLTESİ

Uluslararası Mühendislik Araştırma ve Geliştirme Dergisi
International Journal of Engineering Research and Development

UMAGD, (2023) 15(3), 16-27.

10.29137/umagd.1294273

Cilt/Volume:15 Sayı/Issue:03 Aralık/December 2023 Özel Sayı / Special Issue

Araştırma Makalesi / Research Article

İstatiksel Kodlama Yöntemlerinin Türkçe ve İngilizce Metinlerde Sıkıştırma Başarımı Karşılaştırma Örneği

A Comparison of Text Compression Performance of Statistical Coding Methods in Turkish and English

İbrahim Öztürk[1], **Hakan Celil Kaya** [2]

[1,2]*Osmaniye Korkut Ata University,Department of Electrical and Electronics Engineering,Osmaniye,TURKEY*

***Başvuru/Received**: 09/05/2023* ***Kabul / Accepted**: 12/07/2023* ***Çevrimiçi Basım / Published Online:** 31/12/2023*

***Son Versiyon/Final Version**: 31/12/2023*

Öz

Veri sıkıştırma, dijital ortamda bulunan verilerin hafızada olduğundan daha az yer kaplayabilmesi için yapılan işlem adımları bütünüdür. Bu işlemler dosya türlerine göre değişen az ya da çok tekrar eden veri öbeklerinden yararlanarak gerçekleştirilir. Sıkıştırma hafızanın ve veri iletişim hattının taşıma kapasitesini daha verimli kullanımına olanak sağlamaktadır. Sıkıştırma teknikleri kayıplı ve kayıpsız olarak iki gruba ayrılmaktadırlar. Bu çalışmada sıkıştırma için istatiksel kodlamayı kullanan Huffman, Shannon-Fano ve Aritmetik kodlama yöntemlerinin İngilizce ve Türkçe metinler üzerindeki başarımları karşılaştırılmıştır. Çalışmada kullanılmak üzere İngilizce için Calgary külliyatı içerisinde bulunan metin tabanlı dosyalar, Türkçe için gazetelerde yayımlanmış köşe yazılarından derlemeler yapılmıştır. Sonuçlar Aritmetik kodlamanın tasarruf oranı ölçütüne göre İngilizce metin dosyaları ortalaması %37,92, Türkçe metin dosyaları ortalaması %36,66 ile en yüksek başarımı sağladığı göstermektedir. Huffman kodlaması iki dil için sıkıştırma-açma işlemlerinde en hızlı sonucu vermiştir. İngilizce metin dosyalarının toplam sıkıştırma süresi 2,303 s, açma işlemi 2,988 s sürmüştür. Türkçe metin dosyalarının toplam sıkıştırma için geçen süre 8,142 s, açma işlemi de 10,555 s olarak ölçülmüştür. Kodlamalarda diller arasında sıkıştırma ve açma süreleri açısından başarım farkı görülmemiştir. İngilizce içerikli dosyaların ikisinde en düşük BPC değerini Huffman kodlaması verirken diğer dosyalarda Aritmetik kodlama vermiştir. Türkçe içerikli dosyaların tamamında en düşük BPC değerini Aritmetik kodlama vermiştir.

Anahtar Kelimeler

"Veri sıkıştırma, İstatiksel kodlama, Huffman kodlama, Shannon-Fano kodlama, Aritmetik kodlama"

Abstract

Data compression is a set of process steps performed to make data in the digital environment take up less space than in memory. These operations are performed using more or less repetitive data phrases that vary according to file types. Compression allows more efficient use of memory and data communication capacity. Compression techniques are divided into two types: lossy and lossless. In this study, the performance of Huffman, Shannon-Fano and Arithmetic coding methods based on statistical coding for compression is compared on English and Turkish texts. Text-based files from the Calgary corpus were used for English and newspaper columns for Turkish. The results show that arithmetic coding achieves the highest performance with an average of 37.92% for English text files and 36.66% for Turkish text files according to the saving rate criterion. Huffman coding gave the fastest compression-decompression results in both languages. The total compression time for English text files was 2.303 s and decompression took 2.988 s. For Turkish text files, the total compression time was 8.142 s and decompression time was 10.555 s. There was no performance difference between the languages in terms of compression and decompression times. Huffman coding produced the weakest BPC value in two English files, while Arithmetic coding produced the lowest BPC value in the other files. Arithmetic coding gave the lowest BPC value in all Turkish files.

Key Words

"Data compression, Statistical coding, Huffman coding, Shannon-Fano coding, Arithmetic coding"

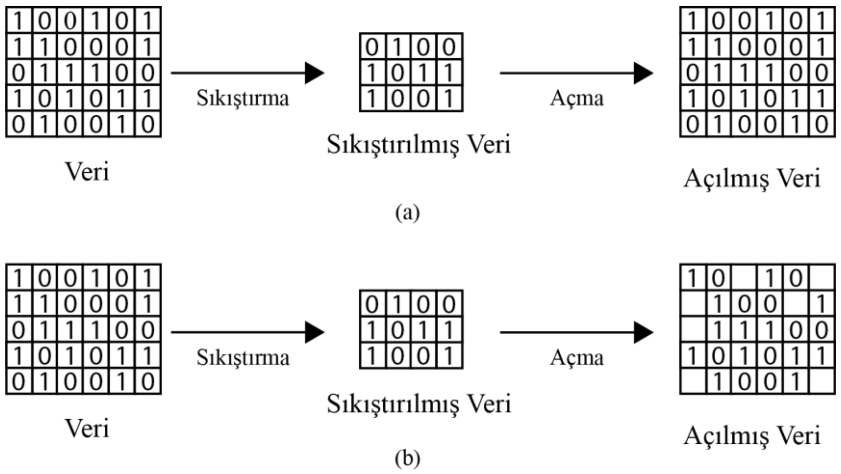
***Sorumlu Yazar: ibrahimozturk@osmaniye.edu.tr**

1. Giriş

Veri sıkıştırma, veriyi olduğundan daha az sayıda bit ile ifade etmektir (Çölkesen, 2021). Günümüzde birçok uygulama ve sistem, büyük miktarda veri işleme gereksinimine ihtiyaç duymaktadır. Verilerin daha az depolama alanı kullanacak şekilde sıkıştırılması, kaynakların verimli kullanılması açısından önem arz etmektedir. Ancak, sıkıştırma işleminin dezavantajları da bulunmaktadır. Sıkıştırma ve açma işlemleri, ilave işlem gücü gereksinimlerine ve gecikmelerine neden olmaktadır. Ayrıca, açma işlemi sırasında meydana gelebilecek veri kayıplarının tespiti ve onarımı sistem kaynaklarına ek yük de getirmektedir.

Sıkıştırma teknikleri kayıplı (lossless) ve kayıpsız (lossy) olarak iki ana gruba ayrılmaktadır (Öztürk & Mesut, 2021). Kayıplı sıkıştırma veri kaybının tolere edilebildiği durumlarda kullanılırken kayıpsız sıkıştırma ise tek bir bitin bile kaybolmasının veya değişmesinin göze alınamayacağı altyapılarda kullanılmaktadır. Kayıplı sıkıştırmada veri kaybı kabul edilebilir seviyelerde tutulurken sıkıştırma oranının arttırılması hedeflenmektedir (Ince et al., 2022). Video, ses ve fotoğraf gibi veri kaybının olası etkisinin daha az olduğu durumlarda genellikle kayıplı sıkıştırma yöntemleri kullanılmaktadır. Program kaynak dosyaları, metin dosyaları ve veri tabanı gibi hassas verilerin saklandığı durumlarda ise genellikle kayıpsız sıkıştırma yöntemleri tercih edilmektedir.

Kayıpsız sıkıştırma yöntemi, veri sıkıştırma işlemi sırasında hiçbir veri kaybına neden olmadığı için sıkıştırılmış veri açıldığında tam olarak orijinal veri tekrar elde edilebilmektedir. Şekil 1.a üzerinde bu durum gösterilmektedir. Diğer yandan, kayıplı veri sıkıştırma yöntemi veri sıkıştırılırken bazı veri kayıplarına neden olabilmektedir. Bu durumda sıkıştırılmış veri açıldığında orijinal verinin tamamı elde edilememektedir. Şekil 1.b üzerinde bu durum gösterilmektedir.



Şekil 1. Veri sıkıştırmanın **(a)** kayıpsız **(b)** kayıplı sıkıştırma üzerinden gösterimi.

Kayıpsız sıkıştırma yöntemleri sözlük tabanlı kodlama ve istatistiksel kodlama olarak ikiye ayrılmaktadır (Mesut, 2006). Sözlük tabanlı kodlama, sıkıştırılacak olan verinin içerisinde tekrar eden karakter veya karakter öbeklerinin belirlenmesiyle başlar. Daha sonra bu öbeklerden ve bunları ifade eden kod kelimelerinden (code word) oluşan bir sözlük oluşturulur. Ardından verideki öbekler, sözlükteki belirlenen kod kelimesi ile ifade edilerek kodlama gerçekleştirilir. Sözlük tabanlı kodlamada sık tekrar eden karakter öbeklerini içeren verilerin sıkıştırılması, daha düzensiz karakter öbekleri içeren verilerden daha başarılıdır (Sayood, 2006). Sözlük tabanlı kodlama yöntemlerinden bazıları LZ77 (Ziv & Lempel, 1977), LZ78 (Ziv & Lempel, 1978), LZSS (Storer & Szymanski, 1982) ve LZW (Lempel-Ziv-Welch) (Welch, 1984) olarak sıralanabilecektir.

İstatiksel kodlama yönteminde, verilerin içerisindeki karakter veya karakter öbekleri veri içerisinde kullanım olasılıkları belirlenerek kodlama gerçekleştirilir. Bu çalışmada Türkçe ve İngilizce metinlerdeki sıkıştırma başarımlarını karşılaştırmak için istatiksel kodlamayı kullanan Huffman (Huffman, 1952), Shannon-Fano (Shannon, 1948) ve Aritmetik kodlama (Abramson, 1963; Witten et al., 1987) kullanılmaktadır.

Çalışmanın Materyal ve Metot bölümünde derlenen metin dosyalarının ayrıntıları, kullanılan programlama dili ve yazılan programların çalıştırıldığı bilgisayarın özellikleri tanıtılmıştır. Huffman, Shannon-Fano ve Aritmetik kodlama yöntemlerine ek olarak karşılaştırma ölçütleri ve ölçütlerin seçilme amaçları irdelenmiştir. Yapılan ölçüm ve hesaplamalar Bulgular başlığı altında görselleştirilerek Sonuçlar kısmında değerlendirilmiştir.

2. Materyal ve Metot

Bu çalışmada kullanılmak üzere İngilizce için Tablo 1'de özellikleri verilen Calgary külliyatından bibliyografi, kitap, haber ve makale içerikli dosyalar (Bell et al., 1989), Türkçe için Tablo 2' de özellikleri verilen farklı alanlardaki on farklı yazarın spor, ekonomi, siyaset ve edebiyat içerikli köşe yazılarından derleme yapılmıştır (Github, 2023). Farklı karakter sayısı, bir metindeki benzersiz karakterlerin sayısını ifade etmektedir. Sıkıştırılacak metindeki bu farklı karakter sayısı, sıkıştırma için önemli bir ölçüt olup sıkıştırılmış veri boyutunu önemli ölçüde etkileyebilmektedir (Sayood, 2006). Dosyalarda kullanılan farklı karakter sayı ortalaması İngilizce 88, Türkçe'de ise 102'dir. Kullanılan veride her iki dil için onar adet dosya mevcut olup toplam dosya boyutu

İngilizce için 2,01 MB, Türkçe için 7,7 MB'dır. Çalışmalar Python dili kullanılarak gerçekleştirilmiştir (Github, 2023). Bütün testler ve ölçümler Core i5-1135G7 @2.40 GHz işlemci, 8 GB Ram, Nvme SSD'ye sahip dizüstü bilgisayarda yapılmıştır.

Tablo 1. Calgary külliyatındaki İngilizce içerikli dosyalara ait özellikler

Dosya Adı	Dosya Boyutu (Byte)	Karakter Sayısı	Farklı Karakter Sayısı
bib	111.261	104.981	80
book1	768.771	752.149	81
book2	610.856	595.222	95
news	377.109	367.050	97
paper1	53.161	51.911	94
paper2	82.199	80.468	90
paper3	46.526	45.426	83
paper4	13.286	12.992	79
paper5	11.954	11.634	90
paper6	38.105	37.086	92

Tablo 2. Köşe yazılarından derlenen Türkçe içerikli dosyalara ait özellikler

Dosya Adı	Dosya Boyutu (Byte)	Karakter Sayısı	Farklı Karakter Sayısı
yazar1	747.412	671.671	103
yazar2	584.588	529.554	97
yazar3	860.366	782.918	102
yazar4	608.155	550.021	108
yazar5	834.856	757.877	99
yazar6	787.361	715.794	104
yazar7	557.230	503.906	107
yazar8	1.451.909	1.319.568	97
yazar9	1.058.118	959.438	104
yazar10	590.482	532.234	98

2.1. İstatiksel kodlama

ASCII (American Standard Code for Information Interchange - Bilgi Alışverişi için Standart Amerikan Kodu) bilgisayarda karakterleri ifade etmek için kullanılan kodlama standardıdır. Bu standartta kullanım sıklığından bağımsız olarak bütün karakterlerin kod kelimesi uzunluğu 8 bittir. İstatistiksel kodlamada veri içerisindeki yüksek olasılıklı karakter veya karakter öbeklerini daha kısa; düşük olasılıklı olanı daha uzun kod kelimesi kullanarak temsil edilmektedir (Öztürk & Mesut, 2021). Kullanım sıklığı belirlenmesi statik ve dinamik olmak üzere iki şekilde belirlenir. Statik olan yöntem kullanılan dilin karakteristik yapısına göre harflerin kullanım sıklıklarının her durumda aynı olduğu ön kabulüne dayanır. Tablo 3 üzerinde İngilizce ve Türkçe alfabelerde yer alan harflerin kullanım sıklığı alfabetik sırayla verilmiştir. Dinamik yöntemde ise veri dizisinde kullanılan harflerin kullanım sıklıkları ölçülerek her veri dizisine özgü olarak tekrardan hesaplanmaktadır (Diri, 1999). Bu yöntem dahilinde kodlanan veriye ek olarak sıkıştırılmış dosyanın içerisine sıkıştırılmış veriyi açma işlemi esnasında kullanılmak üzere harflerin kullanım sıklıkları da eklenmelidir.

Tablo 3. Türkçe ve İngilizce dillerinde harflerin kullanım sıklığı (Çelikel Çankaya et al., 2010; Güneş & Işık, 2018)

Harf	İngilizce	Türkçe	Harf	İngilizce	Türkçe
A	8,17%	13,26%	N	6,75%	4,92%
B	1,49%	2,10%	O	7,50%	2,17%
C	2,78%	1,16%	Ö	-	0,83%

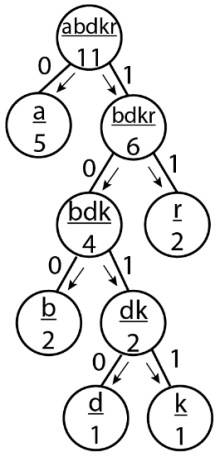
Tablo 3 (devam). Türkçe ve İngilizce dillerinde harflerin kullanım sıklığı (Çelikel Çankaya et al., 2010; Güneş & Işık, 2018)

Harf	İngilizce	Türkçe	Harf	İngilizce	Türkçe
Ç	-	1,38%	P	1,93%	1,23%
D	4,25%	2,46%	Q	0,10%	-
E	12,70%	8,96%	R	5,99%	5,51%
F	2,23%	0,81%	S	6,33%	3,37%
G	2,02%	1,46%	Ş	-	1,92%
Ğ	-	0,89%	T	9,06%	4,50%
H	6,09%	1,21%	U	2,76%	2,92%
I	6,97%	4,46%	Ü	-	2,03%
İ	-	6,59%	V	0,98%	0,99%
J	0,15%	0,09%	W	2,36%	-
K	0,77%	7,88%	X	0,15%	-
L	4,03%	6,51%	Y	1,97%	2,52%
M	2,41%	6,06%	Z	0,07%	1,81%

2.2. Huffman kodlama

David Huffman tarafından 1952 yılında geliştirilmiştir. Huffman kodlama GZIP, JPEG veya MPEG gibi sık tercih edilen sıkıştırma yöntemlerinin işlem basamaklarında kullanılır (Mesut, 2006). Kodlama işlemi veri dizisi içindeki karakterlerin kullanım sıklığına bağlı olarak oluşturulan Huffman ağacından, karakterler için tanımlanmış kod kelimesinin karakter yerine kullanılmasıdır. Veri dizisindeki karakter sayısına ve karakterlerin kullanım sıklıklarına bağlı olarak Huffman kodlaması %10 ile %90 oranında sıkıştırma başarımı sağlayabilmektedir. (Güzeldereli, 2012).

Örnek "abrakadabra" veri dizisinde "a" karakteri 5, "b" karakteri 2, "r " karakteri 2, "k" karakteri 1 ve "d" karakteri 1 defa kullanılmıştır. Bu değerler karakterlerin kullanım sıklığı tablosunu oluşturmaktadır. Bundan sonra hangi karakterin hangi kod kelimesi ile ifade edileceğini belirlemek için Şekil 2'de gösterildiği üzere Huffman ağacı oluşturulacaktır (Bulut, 2016). Huffman ağacını en düşük kullanım sıklığına sahip iki karakter olan [d,1] ve [k,1] karakterleri birleştirilip kullanım sıklığı toplanacaktır. Oluşan [dk,2] tabloda yerini alacaktır. Tabloda eklenen-çıkan karakterler ve değerleri ile sıralama değişmiş olacaktır. Tekrar tablodaki en düşük kullanım sıklığına sahip olan [b,2] ile [dk,2] birleştirilerek [bdk,4] tabloya eklenecektir. Bu işlemler tablodaki tek karakter kalana kadar devem edecek ve benzetim olarak Şekil 2'de gösterilen ağaca benzeyen yapı meydana gelmiş olacaktır.



Şekil 2. Huffman ağacıyla kodlama aşamalarının gösterimi

İlk oluşturulan kullanım sıklığı tablosundaki karakterler için kod kelimesinin belirlenmesi Huffman ağacından yukarıdan aşağıya doğru olmak üzere sağa ve sola ilerleyerek belirlenir. Sağa doğru giden dallar "1" ile işaretlenirken sola gidenler "0" ile işaretlenir (Oral & Aşşık, 2019). Her bir karaktere ulaşmak için geçilen dallar üzerindeki değerler birleştirilerek o karakterin kod kelimesi tanımlanmış olacaktır. Şekil 2'de belirtilen oklar takip edilerek 'a' karakteri "0", b karakterini "100", r karakterini "11", k karakterini "1011" ve d karakterinin de "1010" olacak şekilde kod kelimeleri belirlenmiş olacaktır.

2.3. Shannon-Fano kodlama

1940 yılında C. Shannon ve R. M. Fano tarafından ortaya konulmuştur. Huffman kodlamaya benzerlikler gösterse de işlem basamakları farklıdır. Ağaca benzer yapı yerine kodlanacak olan veri dizisinde kullanılan karakterlerin kullanım sıklıkları/oranı

azalan sırada olacak şekilde tablo oluşturulur (Çölkesen, 2021). Daha sonra oluşturulan tablo kullanım sıklıkları toplamı eşit ya da eşite en yakın olan yerden ikiye ayrılır. Üst grupta kalan karakterler "0" alt grupta kalan karakterler için kod kelimesine "1" eklenir. Daha sonra grupların elemanları birden fazla ise tekrar bölünür. Aynı şekilde üst gruptaki karakterlerin kod kelimesine "0", alt grupta kalan karakterlere "1" eklenir. Bu işleme gruplarda tek karakter kalana kadar devam edilir (Çölkesen, 2021; Diri, 1999). İşlem sonunda aşağıdan yukarı doğru her karakter için kod kelimeleri belirlenmiş olacaktır. Örnek olarak yine "abrakadabra" veri dizisi üzerinden devam edilecektir. Tablo 4 üzerinde gösterilen kullanım sıklıklarına göre ilk adımda olasılık toplamları tablonun eşit ya da eşite en yakın yer olan 0,45 ile 0,55 arasında yani "a" ile "b" arasında tablo ikiye ayrılacaktır. Üst grupta kalan "a" karakteri için kod kelimesi "0" olarak belirlenirken alt tarafta kalan gruptaki kod kelimeleri "b","r","k" ve "d" ye "1" eklenerek kodlamaya devam edilir. 2. adımda alt grup tekrar ikiye toplam oranları 0,18 ile 0,36 olacak şekilde "b" ile "r" arasından ikiye ayrılır. Üstte kalan "b" için "0" eklenerek kod kelimesi "10" olarak belirlenmiş olacaktır. Altta kalan gruptaki harflerin kod kelimelerine "1" eklenerek kodlamaya devam edilir. Kodlamaya 4. adımdan sonra Tablo 4 üzerinde gösterildiği üzere "a" karakteri "0", "b" karakteri "10", "r" karakteri "110", "k" karakteri "1100" ve "d" karakteri "1111" kod kelimesiyle ifade edilebilecektir.

Tablo 4. Shannon-Fano kodlamasının işlem basamaklarının gösterimi

Karakter	Olasılık	1. adım		2. adım		3. adım		4. adım		Kod Kelimesi
		Kod	Olasılık	Kod	Olasılık	Kod	Olasılık	Kod	Olasılık	
a	5/11=0,45	0	*0,45*							0
b	2/11=0,18	1		0	*0,18*					10
r	2/11=0,18	1	*0,55*	1		0	*0,18*			110
k	1/11=0,09	1		1	*0,36*	1	*0,18*	0	*0,09*	1110
d	1/11=0,09	1		1		1		1	*0,09*	1111

2.4. Aritmetik kodlama

Aritmetik kodlama, veri dizisi içerisindeki karakterlerin kullanım olasılıklarını kullanarak kodlanacak veri dizisini [0,1) sınırlar içerisinde gerçek bir sayı tarafından gösterilmesine dayanmaktadır (Çölkesen, 2021). Veri dizisindeki her bir karakter bu veri aralığını daraltır. Aralık daraldıkça veriyi ifade eden bit sayısı da artacaktır (Aktaner, 1995). Kullanım sıklığı yüksek olan karakterler ise aralığı daha az daralttığından onu gösteren bit sayısı da azalır. Dolayısıyla tekrar eden karakterler ne kadar çok ise veri dizisi de çok daha az bit ile kodlanabilecektir (Çölkesen, 2021; Mesut, 2006). Aritmetik kodlamada, veri dizisinin diğer dizilerden ayrılması için, her veri dizisinin benzersiz belirleyici ile ifade edilmesi gerekir. Bu belirleyici ise işlemlerin sonucunda hesaplanır ve etiket (tag) olarak adlandırılır (Mesut, 2006).

Örnek "abrakadabra" veri dizisini beş harfe sahip A=[a, b, r, k, d] alfabesini kullanır. Harflerin A üzerindeki olasılık dağılımları $P(a)$ = 0,45, $P(b)$ = 0,18, $P(r)$ = 0,18, $P(k)$ = 0,09 ve $P(d)$ = 0,09 olarak gösterilebilecektir. Etiketin hesaplanması için öncelikle alt sınırı 0 ile üst sınırı 1 olan sayı aralığı olasılık oranları baz alınarak alfabedeki harf sayısı olan 5 parçaya bölünür. 0 ile 0.45 arası "a", 0.45 ile 0.63 arası "b", ve 0.63 ile 0.81 arası "r", 0.81 ile 0.9 arası "k" ve 0.9 ile 0.99 arası da "d" için tahsis edilir. Kodlama işlemi veri dizisinin ilk karakteri "a" için 0 ile 0.45 aralığı artık yeni alt ve üst sınır olarak belirlenmiş olacaktır. Yeni alt ve üst sınırlara göre olasılık dağılımları baz alınarak tekrar 5 parçaya bölünür. 0 ile 0,2025 arası "a", 0,2025 ile 0,2835 arası "b", 0,2835 ile 0,3645 arası "r", 0,3645 ile 0,405 arası "k" ve 0,405 ile 0,4455 arası da "d" için tahsis edilir. Sıradaki karakter olan "b" için ayrılan alt sınır 0 ile üst sınırı 0,2025 aralığı yeni alt ve üst sınır olarak belirlenecektir. Tablo 5' te gösterilen tüm bu işlemler veri dizisindeki karakterlerin tamamı için sırasıyla gerçekleştirilir.

En son adımda ise alt sınırı 0,116590329711294 olan ve üst sınırı 0,116590400318678'in aritmetik ortalaması alınarak 0,116590365014986 etiket değeri hesaplanır. Bu değerin "0." kısmı hesaplamaya dahil edilmez (Diri, 1999). Onluk sayı sistemindeki 116590365014986 sayısı ikilik sayı sistemine (binary) çevrilerek elde edilen "11010100000100111001111010010100111101111001-010" ile veri dizisi kodlanmış olur.

Tablo 5. Aritmetik kodlamanın işlem basamaklarının gösterimi

Alfabe	Sınırlar	İlk Dağılım	Veri Dizisi										
			a	b	r	a	k	a	d	a	b	r	a
a	Alt	0	**0,0000**	0,0000	0,09112500	…	…	…	…	…	…	…	**0,116590329711294**
	Üst	0,45	**0,2025**	0,0911	0,10752750	…	…	…	…	…	…	…	**0,116590400318678**
b	Alt	0,45	0,2025	**0,0911**	0,10752750	…	…	…	…	…	…	…	0,116590400318678
	Üst	0,63	0,2835	**0,1276**	0,11408850	…	…	…	…	…	…	…	0,116590428561632

Tablo 5 (devam). Aritmetik kodlamanın işlem basamaklarının gösterimi

Alfabe	Sınırlar	İlk Dağılım	Veri Dizisi										
			a	**b**	**r**	**a**	**k**	**a**	**d**	**a**	**b**	**r**	**a**
r	Alt	0,63	0,2835	0,1276	**0,11408850**	…	…	…	…	…	…	…	0,116590428561632
	Üst	0,81	0,3645	0,1640	**0,12064950**	…	…	…	…	…	…	…	0,116590456804585
k	Alt	0,81	0,3645	0,1640	0,12064950	…	…	…	…	…	…	…	0,116590456804585
	Üst	0,9	0,4050	0,1823	0,12393000	…	…	…	…	…	…	…	0,116590470926062
d	Alt	0,9	0,4050	0,1823	0,12393000	…	…	…	…	…	…	…	0,116590470926062
	Üst	0,99	0,4455	0,2005	0,12721050	…	…	…	…	…	…	…	0,116590485047539

Kodlama anlatımlarında kullandığımız 11 karakterden oluşan "abrakadabra" veri dizisi ASCII standardı ile 8*11=88 bit ile gösterilirken Huffman kodlamasıyla 23 bit, Shannon-Fano ile 23 bit ve Aritmetik kodlamayla 47 bitle gösterilebilmiştir. Bu değerler kodlamanın genel başarımını göstermemektedir. Örnek veri dizisi kodlama yöntemlerini açıklarken daha kolay anlaşılması için kısa ve benzer karakterler içermesine dikkat edilmiştir. Veri dizisinin uzunluğu, metindeki benzerlikler ve karakterlerin kullanım sıklıkları kodlamanın başarımını önemli ölçüde etkilemektedir.

3. Başarım Karşılaştırma Ölçütleri

Karşılaştırma ölçütleri dosya boyutundaki tasarruf oranı, sıkıştırma-açma işlemleri için geçen süre, kodlanmış verinin karakter başına düşen bit sayısı ve en iyi sıkıştırma oranına ne oranda yaklaşıldığı temel alınarak seçilmiştir (Bulut, 2016; Mantoro et al., 2017; Rahman & Hamada, 2019; Shanmugasundaram & Lourdusamy, 2011; Stecuła et al., 2022). Başarım karşılaştırma temelinde sıklıkla kullanılan tasarruf oranı, sıkıştırma-açma süreleri, BPC ve entropi ölçütleri takip eden bölümde detaylandırılacaktır.

3.1. Tasarruf oranı

Sıkıştırma sonrası dosyanın hafızada kapladığı alandan dosya boyutuna göre yüzdesel olarak ne oranda tasarruf edildiğini ifade etmektedir. Tasarruf oranı, Denklem 1 kullanılarak hesaplanır.

$$tasarruf\ oranı = \frac{odb - sdb}{odb} * 100 \quad (1)$$

odb: orijinal dosya boyutunu, sdb: sıkıştırılmış dosya boyutunu ifade etmektedir.

3.2. Sıkıştırma-açma süresi

Dosya sıkıştırma ve açma işlemleri esnasında geçen süredir. Açma işleminden sonra verinin kayıpsız bir şekilde elde edildiği kontrol edilmiştir. Bu işlem orijinal dosya ile açılmış dosyanın SHA-256 (Gilbert & Handschuh, 2004) algoritması kullanılarak üretilen değerlerin tutarlılığıyla doğrulanmıştır.

3.3. BPC

Karakter başına düşen bit (Bit per character - BPC), dosyadaki her bir karakteri ifade etmek için karakter başında düşen bit sayısıdır. BPC değerlerinin hesaplanması Denklem 2 üzerinde yapılır. Sıkıştırma yöntemlerinin etkinliğini değerlendirmek için kullanılmaktadır. Dolayısıyla düşük BPC değeri daha fazla veri tasarrufu sağlandığını ve sıkıştırma işleminin daha verimli olduğunu göstermektedir.

$$BPC = \frac{dosya\ boyutu\ (bit)}{dosyadaki\ karakter\ sayısı} \quad (2)$$

3.4. Entropi (H)

İlk defa 1948'de Claude E. Shannon tarafından tanımlanmıştır. Veri kaynağındaki rastgelelik ölçüsünü gösteren bit sayısını ifade etmektedir (Koşan et al., 2019; Shannon, 1948). Entropi dijital ortamda verinin işlenmesi ve saklanması süreçlerinde önemli bir ölçüt olarak kabul edilmektedir. Rastgele bir değişkenin değerini tahmin etme sürecindeki belirsizliği sayısal olarak ifade etmek için kullanılabilir (Bulut, 2017). Diğer bir ifadeyle entropi değeri en iyi kodlama işlemi sonucunda karakter başına düşebilecek en az bit sayısını vermektedir (Shannon, 1948). Entropi değerlerinin elde edilmesinde Denklem 3 kullanılmaktadır.

$$H = -\sum_{i=1}^{n} P_i \log_2 P_i \quad (3)$$

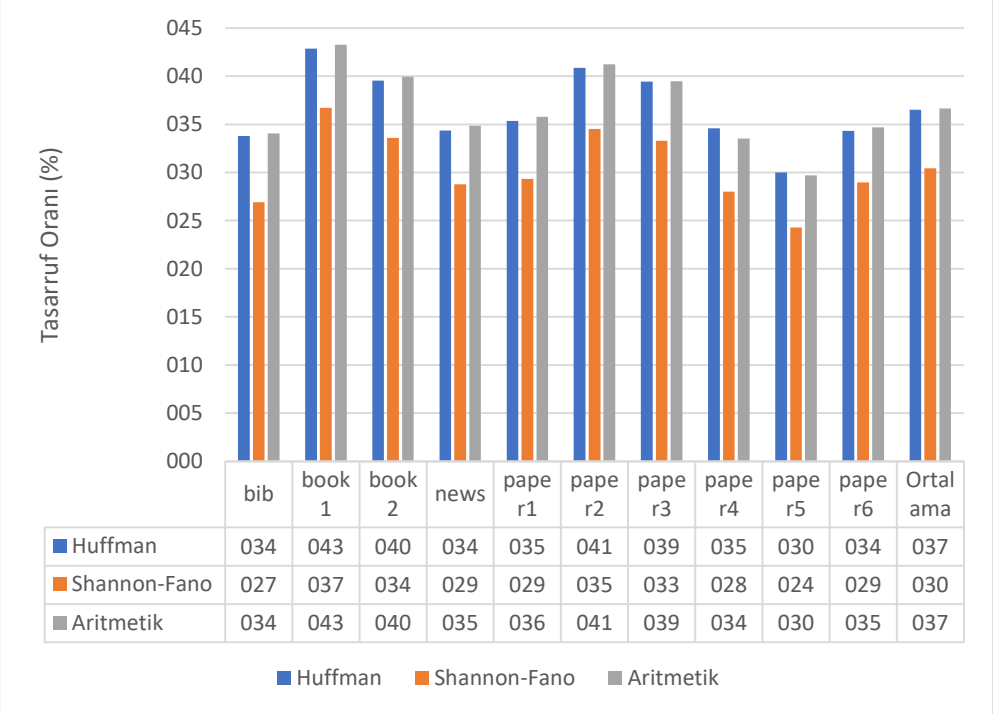
n: verideki toplam farklı karakter sayısı, i: verideki karakter sırası, H: entropi, P_i: i. karakterinin veri içerisinde bulunma olasılığını ifade etmektedir.

4. Bulgular

4.1. Tasarruf oranı

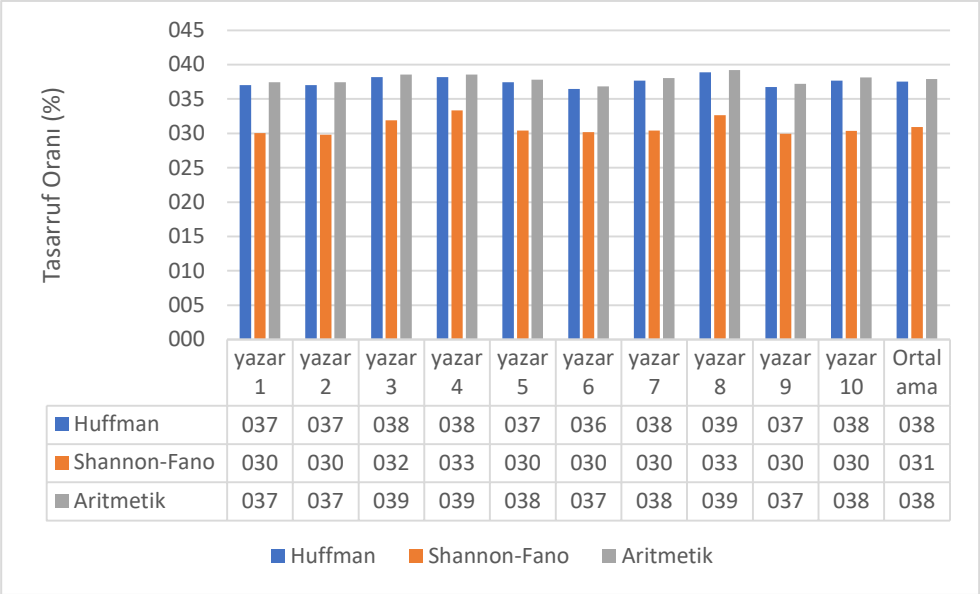
Metin dosyalarında tasarruf oranı hesaplamaları Denklem 1 kullanılarak elde edilmiştir. Tasarruf oranı başarımlarına bakıldığı zaman Huffman kodlama ve Aritmetik kodlamanın çok benzer sonuçlar ürettiği görülmüştür. Shannon-Fano kodlaması ise diğer iki kodlama yöntemine göre daha başarısız sonuçlar vermiştir. Bu durum çalışmada kullanılan her iki dil için de geçerlidir.

Kodlama işlemi ardından sıkıştırılmış İngilizce metin dosyalarının dosya boyutlarının ortalaması alındığında Huffman kodlama %36,51, Shannon-Fano kodlama %30,44, Aritmetik kodlama ile %36,66 oranında tasarruf sağlandığı görülmektedir. Türkçe metin dosyalarında ise Huffman kodlama %37,53, Shannon-Fano kodlama %30,91, Aritmetik kodlama ile %37,92 oranında tasarruf sağlanabildiği görülmektedir. İki dile ait metin dosyalarının ayrı ayrı tasarruf oranları Şekil 3 ve Şekil 4 üzerinde gösterilmektedir.



	bib	book 1	book 2	news	paper1	paper2	paper3	paper4	paper5	paper6	Ortalama
Huffman	034	043	040	034	035	041	039	035	030	034	037
Shannon-Fano	027	037	034	029	029	035	033	028	024	029	030
Aritmetik	034	043	040	035	036	041	039	034	030	035	037

Şekil 3. İngilizce içerikli dosyaların sıkıştırmayla sağlanan tasarruf oranları



Şekil 4. Türkçe içerikli dosyaların sıkıştırmayla sağlanan tasarruf oranları

4.2. Sıkıştırma-açma süresi

Süre ölçüm sonuçları farklı testler esnasında daha tutarlı olabilmesi için art arda yirmi defa tekrarlanarak aritmetik ortalama alınmıştır. Ölçümler sonucunda sıkıştırma-açma sürelerinde en iyi başarım Huffman kodlamasıyla elde edilmiştir. Bu durum iki dil için de geçerlidir. Bununla beraber Shannon-Fano kodlama Huffman kodlamaya çok yakın değerler üretmiştir. Aritmetik kodlama diğer iki kodlama yöntemlerine nazaran sıkıştırma-açma süresi açısında çok daha uzun sürmüştür. Aritmetik kodlama, Huffman ve Shannon-Fano kodlama yöntemlerine kıyasla daha yüksek bir tasarruf oranı sağlasa da daha karmaşık işlem adımları ve matematiksel hesaplamalar gerektirdiğinden dolayı süre açısından verimli değildir.

İngilizce metin dosyalarının sıkıştırma işlemi Huffman kodlamasıyla 2,303 s, Shannon-Fano kodlama ile 2,420 s, Aritmetik kodlama ile 11,802 s sürmüştür. Açma işleminde sıralama değişmeden Huffman kodlamasıyla 2,988 s, Shannon-Fano kodlama ile 3,199 s, Aritmetik kodlama ile 26,542 s olarak ölçülmüştür. Türkçe metin dosyalarının sıkıştırma için geçen süre Huffman kodlamasıyla 8,142 s, Shannon-Fano kodlama ile 8,819 s, Aritmetik kodlama ile 44,367 s olarak ölçülmüştür. Açma işleminde Huffman kodlamasıyla 10,555 s, Shannon-Fano kodlama ile 11,526 s, Aritmetik kodlama ile 101,609 s sürmüştür. Tablo 6 ve Tablo 7'de tüm dosyalar için sıkıştırma-açma süreleri detaylandırılmıştır.

Tablo 6. İngilizce içerikli dosyaların sıkıştırma-açma süreleri

	Sıkıştırma Süresi (s)			Açma Süresi (s)		
Dosya Adı	**Huffman**	**Shannon-Fano**	**Aritmetik**	**Huffman**	**Shannon-Fano**	**Aritmetik**
bib	0,155	0,155	0,667	0,188	0,202	1,441
book1	0,673	0,727	4,229	0,915	0,994	9,519
book2	0,568	0,602	3,257	0,772	0,836	7,551
news	0,391	0,415	2,133	0,531	0,567	4,760
paper1	0,095	0,098	0,323	0,111	0,117	0,699
paper2	0,115	0,120	0,463	0,140	0,148	1,042
paper3	0,093	0,095	0,282	0,102	0,108	0,623
paper4	0,063	0,062	0,105	0,066	0,065	0,202
paper5	0,063	0,059	0,098	0,065	0,063	0,188
paper6	0,085	0,086	0,243	0,097	0,099	0,517
Toplam	2,303	2,420	11,802	2,988	3,199	26,542

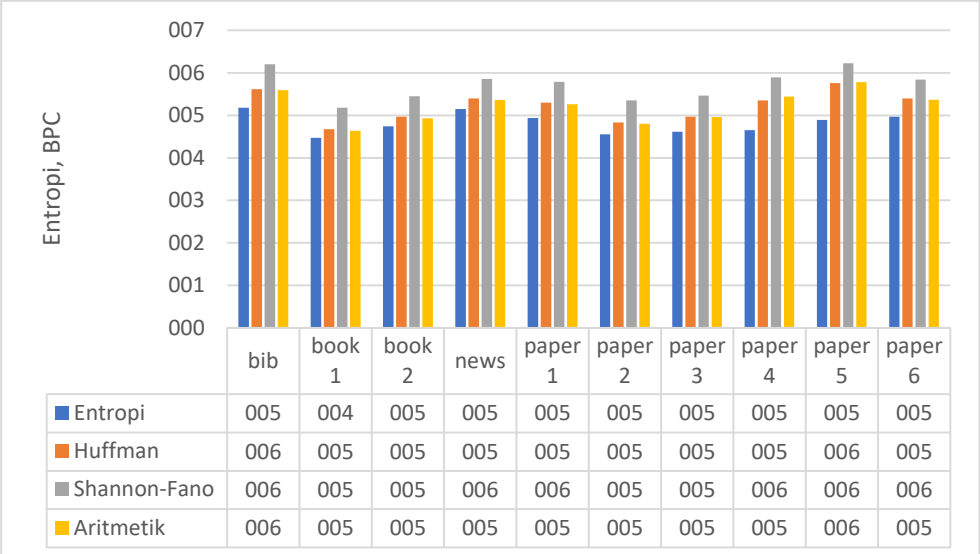
Tablo 7. Türkçe içerikli dosyaların sıkıştırma-açma süreleri

	Sıkıştırma Süresi (s)			Açma Süresi (s)		
Dosya Adı	**Huffman**	**Shannon-Fano**	**Aritmetik**	**Huffman**	**Shannon-Fano**	**Aritmetik**
yazar1	0,765	0,827	4,098	1,001	1,084	9,366
yazar2	0,615	0,654	3,238	0,781	0,866	7,373
yazar3	0,847	0,926	4,703	1,107	1,217	10,807
yazar4	0,617	0,649	3,316	0,796	0,848	7,725
yazar5	0,843	0,912	4,594	1,090	1,194	10,421
yazar6	0,806	0,871	4,370	1,050	1,136	10,001
yazar7	0,574	0,622	3,064	0,746	0,827	7,063
yazar8	1,410	1,545	7,915	1,818	1,982	18,096
yazar9	1,066	1,157	5,838	1,384	1,510	13,358
yazar10	0,599	0,655	3,229	0,781	0,862	7,400
Toplam	8,142	8,819	44,367	10,555	11,526	101,609

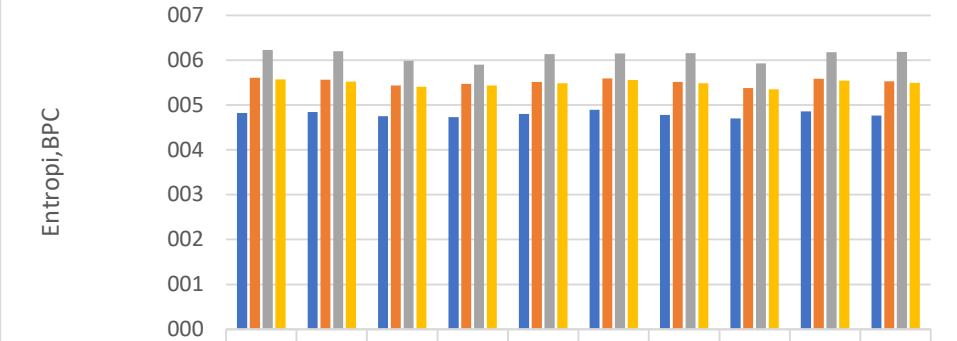
4.3. Entropi ve BPC

Entropi ile BPC değeri arasında doğru orantılı bir ilişki vardır. Verinin entropisi yüksekse yani veri düzensiz, rastgele ve bilgi içeriği yoğunsa sıkıştırma algoritması bu veriyi daha az etkili bir şekilde sıkıştırabilir. Bu durumda BPC değeri yüksek olacaktır. Verinin entropisi düşükse yani veri düzenli, tekrar fazla ve az bilgi içeriği barındırıyorsa sıkıştırma algoritması bu veriyi daha etkili bir şekilde sıkıştırabilir. Sonuç olarak, BPC değeri düşük olacaktır. Her bir dosya için ayrı ayrı, Entropi değeri Denklem 3 üzerinde ve BPC değeri ise Denklem 2 kullanılarak hesaplanmıştır. Sonuçlar Şekil 5 ve Şekil 6 üzerinde gösterilmiştir.

Entropi ortalama değerleri İngilizce içerikli dosyalarda 4,81, Türkçe içerikli dosyalarda ise 4,79 hesaplanmıştır. BPC ortalama değerleri İngilizce içerikli dosyalarda Huffman kodlamasıyla 5,22; Shannon-Fano kodlama ile 5,72, Aritmetik kodlamada ise 5,21 olarak hesaplanmıştır. Türkçe içerikli dosyalarda Huffman kodlama ile 5,52, Shannon-Fano kodlama ile 6,10, Aritmetik kodlama ile 5,48 olarak hesaplanmıştır.



Şekil 5. İngilizce içerikli dosyaların Entropi ve BPC değerleri



	yazar 1	yazar 2	yazar 3	yazar 4	yazar 5	yazar 6	yazar 7	yazar 8	yazar 9	yazar 10
Entropi	005	005	005	005	005	005	005	005	005	005
Huffman	006	006	005	005	006	006	006	005	006	006
Shannon-Fano	006	006	006	006	006	006	006	006	006	006
Aritmetik	006	006	005	005	005	006	005	005	006	005

Şekil 6. Türkçe içerikli dosyaların Entropi ve BPC değerleri

Türkçe içerikli dosyalarda en düşük BPC değerini Aritmetik kodlama vermiştir. İngilizce içerikli dosyalardan paper4 ve paper5 üzerinde en düşük BPC değerini Huffman kodlama verirken diğer dosyalarda en düşük değeri Aritmetik kodlama vermiştir. Bu durumun sebebi, paper4 ve paper5 dosyalarının içerisinde diğer dosyalara göre daha az karakter bulunması ve bu nedenle tekrar eden karakter sayısının da göreceli olarak daha az olmasıdır. Ortalama BPC değeri Türkçe metinlerde İngilizce metinlere göre Huffman kodlamasında %5,63, Shannon-Fano kodlamasında %6,63 ve aritmetik kodlama da ise %5,20 oranında daha yüksek olduğu görülmüştür. Ortalama entropi değerleri her iki dilde çok yakın sonuçlar vermiştir.

5. Sonuçlar

Bu çalışmada İngilizce ve Türkçe içerikli dosyaların istatiksel kodlamaları kullanarak sıkıştırma başarımları karşılaştırılmıştır. Kayıpsız veri sıkıştırma başarım testlerinde en çok tercih edilen dosyaları içermesi sebebiyle İngilizce içerikler için Calgary külliyatı kullanılmıştır. Türkçe için spor, ekonomi, siyaset ve edebiyat içerikli köşe yazılarından yapılan derleme kullanılmıştır. Çalışmada kullanılan tüm metinler ASCII kodlamaya sahiptir.

Tasarruf oranında Aritmetik kodlama her iki dildeki metinlerde de en iyi sonuçları vermiştir. Huffman kodlaması Aritmetik kodlamaya yakın sonuçlar göstermiş olsa da İngilizce metinlerde %0,15 Türkçe metinlerde %0,40 oranında daha düşük başarım sağlanabilmiştir. Shannon-Fano kodlaması ise Aritmetik kodlama'ya göre İngilizce metinlerde %6,22 Türkçe metinlerde %7,02 farkla en düşük tasarruf oranına sahiptir.

Huffman kodlaması, her iki dilde sıkıştırma ve açma süreleri açısından en iyi sonuçları verirken, Shannon-Fano kodlaması da sonuçlarıyla oldukça yakın bir başarım sergilemiştir. Bununla birlikte, Aritmetik kodlama yöntemi süre açısından en düşük başarımı göstermiştir. Türkçe metinlerin toplam dosya boyutunun İngilizce metinlere oranı 3,82'dir. Bu oran yaklaşık olarak iki dil arasındaki toplam sıkıştırma-açma sürelerinde gözlemlenmektedir. Dolayısıyla diller arasında sıkıştırma-açma süreleri arasında fark olmadığı anlaşılmaktadır.

Bu çalışmada test ve ölçümler için onar adet metin dosyası derlenmiştir. Sonraki çalışmalarda bu sayının daha fazla sayıda belirlenmesi ve içeriklerin zenginleştirilmesi ile doğruluk oranı daha isabetli tespit edilebilecektir. Bu çalışmada başarım karşılaştırmaları tasarruf oranı, sıkıştırma-açma süreleri, BPC ve entropi değerleri kullanılarak yapılmıştır. Bu ölçütlere bellek ve işlemci kullanımı gibi ölçütlerin eklenmesi kodlama yönteminin uygulama senaryosuna göre seçimine daha fazla belirleyicilik de sağlayabilecektir.

Bu çalışmada istatistiksel kodlama yöntemlerinden en çok bilinenleri tercih edilmiştir. Ayrıca, daha kapsamlı bir değerlendirme için seçilen kodlamaların yansıra Huffman kodlamanın varyasyonlarının, ANS (Asymmetric Numeral Systems- Asimetrik Sayı Sistemleri) kodlamanın performans odaklı iyileştirilmiş bir uygulaması olan FSE (Finite State Entropy - Sonlu Durum Entropisi) algoritmasının da dâhil edilmesi gelecekteki çalışmalarda isabetli olacaktır.

Kaynaklar

Abramson, N. (1963). *Information theory and coding.*

Aktaner, A. (1995). *Entropi Kodlama ile EKG Veri Sıkıştırma* İstanbul Teknik Üniversitesi]. Fen Bilimleri Enstitüsü.

Bell, T., Witten, I. H., & Cleary, J. G. (1989). Modeling for text compression. *ACM Comput. Surv.*, *21*(4), 557–591. https://doi.org/10.1145/76894.76896

Bulut, F. (2016). Huffman Algoritmasıyla Kayıpsız Hızlı Metin Sıkıştırma. *El-Cezerî Fen ve Mühendislik Dergisi*, *3*(2), 0-0. https://doi.org/10.31202/ecjse.264192

Bulut, F. (2017). Bilgi Kuramındaki Entropi Kavramıyla İlgili Farklı Matematiksel Modeller. *Bilge International Journal of Science and Technology Research*, *1*(2), 167-174.

Çelikel Çankaya, E., Palaniappan, V., & Latifi, S. (2010). Fazlalıktan Yararlanarak Kayıplı Metin Sıkıştırma Gerçekleştirimi. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi*, *16*(3), 235 - 245.

Çölkesen, T. F. (2021). *Veri Yapıları ve Algoritmalar*. Papatya Yayınları.

Diri, B. (1999). *Türkçe'nin biçimbilim yapısına dayalı bir metin sıkıştırma sistemi* Yıldız Teknik Üniversitesi]. Fen Bilimleri Enstitüsü

Gilbert, H., & Handschuh, H. (2004). Security Analysis of SHA-256 and Sisters. In M. Matsui & R. J. Zuccherato, *Selected Areas in Cryptography* Berlin, Heidelberg.

Github. (2023). https://github.com/hckaya/stcc

Güneş, F., & Işık, A. D. (2018). Türkçede Sık Kullanılan Harfler ve Öğretilmesi. *Sınırsız Eğitim ve Araştırma Dergisi*, *3*(1), 1-26.

Güzeldereli, E. A. (2012). *Veri Gizlemede Sıkıştırma Algoritması Kullanımı ve Uygulaması* Sakarya Üniversitesi]. Fen Bilimleri Enstitüsü.

Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes. *Proceedings of the IRE*, *40*(9), 1098-1101. https://doi.org/10.1109/JRPROC.1952.273898

Ince, I. F., Bulut, F., Kilic, I., Yildirim, M. E., & Ince, O. F. (2022). Low dynamic range discrete cosine transform (LDR-DCT) for high-performance JPEG image compression. *The Visual Computer*, *38*(5), 1845-1870. https://doi.org/10.1007/s00371-022-02418-0

Koşan, M. A., Coşkun, A., & Karacan, H. (2019). Yapay Zekâ Yöntemlerinde Entropi *Journal of Information Systems and Management Research*, *1*(1), 15-22.

Mantoro, T., Ayu, M. A., & Anggraini, Y. (2017, 23-25 Nov. 2017). The performance of text file compression using Shannon-Fano and Huffman on small mobile devices. 2017 International Conference on Computing, Engineering, and Design (ICCED),

Mesut, A. (2006). *Veri Sıkıştrmada Yeni Yöntemler* Trakya Üniversitesi]. Fen Bilimleri Enstitüsü.

Oral, M., & Aşşık, M. M. (2019). Kanonik Huffman Benzeri Kodlama için Kod Sözcüklerinin Uzunluklarını Cebirsel Olarak Hesaplayan Bir Algoritma. *Çukurova Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, *34*(4), 9-20. https://doi.org/10.21605/cukurova mmfd.702021

Öztürk, E., & Mesut, A. (2021). Kısa Metinlerin Sıkıştırılması için BERT Tabanlı bir Yöntem. *Avrupa Bilim ve Teknoloji Dergisi*(32), 177-182. https://doi.org/10.31590/ejosat.1039450

Rahman, M. A., & Hamada, M. (2019). Lossless Image Compression Techniques: A State-of-the-Art Survey. *Symmetry*, *11*(10), 1274. https://www.mdpi.com/2073-8994/11/10/1274

Sayood, K. (2006). *Introduction to Data Compression*. Elsevier Science. https://books.google.com.tr/books?id=044wLaqZ8twC

Shanmugasundaram, S., & Lourdusamy, R. (2011). A Comparative Study Of Text Compression Algorithms. *International Journal of Wisdom Based Computing*, *2*(4), 68-76. https://doi.org/10.21917/ijct.2011.0062

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379-423. https://doi.org/10.1002/j.1538-7305.1948.tb01338.x

Stecuła, B., Stecuła, K., & Kapczyński, A. (2022). Compression of Text in Selected Languages-Efficiency, Volume, and Time Comparison. *Sensors*, *22*(17), 6393. https://www.mdpi.com/1424-8220/22/17/6393

Storer, J. A., & Szymanski, T. G. (1982). Data compression via textual substitution. *J. ACM*, *29*(4), 928–951. https://doi.org/10.1145/322344.322346

Welch, T. A. (1984). A Technique for High-Performance Data Compression. *Computer*, *17*(6), 8-19. https://doi.org/10.1109/MC.1984.1659158

Witten, I. H., Neal, R. M., & Cleary, J. G. (1987). Arithmetic coding for data compression. *Commun. ACM*, *30*(6), 520–540. https://doi.org/10.1145/214762.214771

Ziv, J., & Lempel, A. (1977). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, *23*(3), 337-343. https://doi.org/10.1109/TIT.1977.1055714

Ziv, J., & Lempel, A. (1978). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, *24*(5), 530-536. https://doi.org/10.1109/TIT.1978.1055934