

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ TEZİ

“**BERT Tabanlı Metin Steganografisi Yöntemleri: Karakter Tabanlı, Sabit Bitli ve Huffman Kodlama Yaklaşımları”**

**“Text Steganography Methods Based on BERT: Character-Based, Fixed-Bit, and Huffman Coding Approaches”**

“Mert KESKİN 21120205001”

DANIŞMAN

Zeynep Turgut AKGÜN

Mayıs, 2025,

İstanbul Medeniyet Üniversitesi, İstanbul

**ÖNSÖZ**

Proje boyunca bana destek olan danışman hocama Zeynep Turgut Akgün’e teşekkürlerimi sunuyorum.

**İÇİNDEKİLER**

**Sayfa No**

[ÖNSÖZ i](#_heading=h.2et92p0)

[İÇİNDEKİLER i](#_heading=h.2et92p0)i

[TABLO LİSTESİ iii](#_Toc199595429)

[ŞEKİL LİSTESİ iv](#_Toc199595430)

[ÖZET v](#_Toc199595431)

[ABSTRACT vi](#_Toc199595432)

[2. GİRİŞ 1](#_Toc199595433)

[3. GENEL KISIMLAR 2](#_Toc199595434)

[4. MATERYAL VE YÖNTEM 4](#_Toc199595435)

[5. BULGULAR 7](#_Toc199595436)

[6. TARTIŞMA VE SONUÇ 18](#_Toc199595437)

[KAYNAKLAR 19](#_Toc199595438)

[EKLER 21](#_Toc199595439)

### TABLO LİSTESİ

**Sayfa No**

[Tablo 5.5: Deney Sonuçları 17](#_heading=h.lnxbz9)

### ŞEKİL LİSTESİ

**Sayfa No**

[Şekil 5.1: Karakter Tabanlı Yöntemde Başlık Gömme İşlemi 10](#_heading=h.lnxbz9)

Şekil 5.2: Karakter Tabanlı Yöntemde Gizli Mesaj Gömme İşlemi 10 Şekil 5.3: Sabit Genişlikli Bit Kodlama Yönteminde Gizli Mesaj Gömme İşlemi 12 Şekil 5.4: Huffman Tabanlı Yöntemde Gizli Mesaj Gömme İşlemi 15

### ÖZET

Bu çalışmada, metin steganografisi alanında BERT tabanlı üç farklı algoritma geliştirilmiş ve karşılaştırılmıştır. İlk yöntem, her kelimenin belirli bir karakter pozisyonuna gizli bir harf yerleştirmeye dayanan karakter tabanlı bir yaklaşımdır. İkinci yöntem, sabit bit genişliğinde kodlama kullanarak her kelimeye belirli sayıda bit gizlemeyi hedeflerken; üçüncü yöntem, tahmin olasılıklarına dayalı Huffman kodlaması ile daha esnek ve anlamlı kelime seçimlerine olanak tanımaktadır. Algoritmalar Python diliyle geliştirilmiş ve BERT’in Masked Language Modeling (MLM) özelliğinden yararlanılmıştır. Her yöntem için gömme ve çözme işlemlerine ait pseudo kodlar sunulmuş ve örnek uygulamalarla açıklamalar yapılmıştır. Uygulanan deneylerde, kapasite, anlamsal bozulma ve başarısız gömme oranları gibi metrikler üzerinden karşılaştırmalar yapılmış; karakter tabanlı yöntemin en yüksek kapasiteye ulaşmasına rağmen semantik bütünlük açısından en çok bozulmaya yol açtığı, Huffman tabanlı yöntemin ise en düşük anlamsal bozulmayı sağladığı gözlemlenmiştir. Bu bağlamda çalışma, derin öğrenme modellerinin doğal dil işleme temelli steganografi uygulamalarında sunduğu potansiyeli ve sınırlılıkları ortaya koymaktadır.

**Anahtar Kelimeler:** Steganografi, Maskeli Dil Modelleme, BERT, Huffman Kodlama, Sabit Bit Kodlama, Karakter Tabanlı Kodlama

### ABSTRACT

In this study, three different BERT-based algorithms were developed and compared in the field of text steganography. The first method is a character-based approach that embeds a single hidden character into a fixed position within each word. The second method aims to embed a fixed number of bits per word using fixed-width bit encoding, while the third method leverages Huffman coding based on prediction probabilities to enable more flexible and semantically appropriate word selection. The algorithms were implemented in Python and utilized BERT’s Masked Language Modeling (MLM) capability. Pseudo-code for embedding and extraction processes is provided for each method, accompanied by illustrative examples. Experimental evaluations focused on metrics such as capacity, semantic distortion, and failed embedding rates. Results showed that while the character-based method achieved the highest embedding capacity, it also led to the most semantic degradation. In contrast, the Huffman-based method yielded the lowest semantic distortion. This work highlights the potential and limitations of deep learning models in natural language processing-based steganographic applications.

**Keywords:** Steganography, Masked Language Modeling, BERT, Huffman Coding, Fixed-Bit Encoding, Character Based Encoding

### GİRİŞ

**2.1. Steganografinin Tanımı ve Amacı**

Steganografi, bilgiyi gizleme sanatıdır. Kelime kökeni Yunanca "steganos" (gizli) ve "graphein" (yazmak) kelimelerinden türemiştir ve "gizli yazı" anlamına gelir. Temel amacı, gizlenmiş bilginin varlığını üçüncü taraflara fark ettirmeden saklamak ve iletmektir. Kriptografi, bilgiyi şifreleyerek yetkisiz kişilerin anlamasını engellemeye çalışırken, steganografi doğrudan verinin gizlenmesine odaklanır. Böylece mesajın yalnızca belirlenen alıcı tarafından bilinmesi ve okunması sağlanır.

Steganografi, yalnızca güvenli iletişim sağlamak için değil, aynı zamanda telif hakkı koruma, veri bütünlüğü sağlama ve kimlik doğrulama gibi alanlarda da kullanılmaktadır. Ancak kötü niyetli kişiler tarafından da kullanılabileceği için, siber güvenlik uzmanları ve dijital adli bilişim araştırmacıları steganografik teknikleri tespit etmeye yönelik analizler yapmaktadır.

**2.2 Metin Steganografisi**

Metin steganografisi (*text steganography*), taşıyıcı olarak dijital metinlerin kullanıldığı bir steganografi türüdür. Görsel ve ses tabanlı steganografiye kıyasla, taşıyıcı kapasitesinin sınırlılığı ve dilin yapısal bütünlüğünü koruma zorunluluğu nedeniyle daha karmaşık ve zorlu bir problem alanı sunar (Chapman et al., 2001). Buna rağmen, metinlerin dijital iletişimde yoğun kullanımı ve sade yapısı, metin steganografisini hem pratik hem de dikkat çekici bir konu hâline getirmiştir.

Metin steganografisinde kullanılan teknikler genel olarak üç ana başlık altında toplanabilir: **dilbilimsel yöntemler**, **biçim tabanlı yöntemler**, ve **istatistiksel veya rastgele üretim yöntemleri** [1]. Dilbilimsel yöntemler, mevcut bir metin içinde eşanlamlı kelime değişimi, cümle yeniden yapılandırması gibi doğal dil işleme (NLP) tekniklerine dayanır. Biçim tabanlı yöntemlerde ise metnin görsel veya yapısal özellikleri (örneğin boşluklar, sekmeler, yazı tipi) üzerinde oynama yapılarak bilgi saklanır. Diğer yandan, rastgele üretim tabanlı yöntemler ya da yapay metin üretimine dayalı sistemler, baştan aşağıya mesaj içeren ancak anlamlı ve doğal görünen yeni metinler üretmeyi hedefler.

Bu alandaki en büyük zorluklardan biri, gizlenen bilginin hem taşıyıcı metnin anlamını bozmadan yerleştirilmesi hem de tespit edilmesinin zorlaştırılmasıdır. Özellikle son yıllarda doğal dil işleme alanında yaşanan gelişmelerle birlikte, metin steganografisinde daha gelişmiş ve bağlama duyarlı yöntemler geliştirilmeye başlanmıştır. Literatürde eşanlamlı kelime değişimi, makine çevirisi, biçimsel işaretleme, n-gram modelleri, hatta dil modeli tabanlı yaklaşım (örneğin BERT, GPT) gibi birçok teknik önerilmiştir. Bu yöntemler, hem taşıyıcı metnin anlam bütünlüğünü korumaya hem de gizli mesajların taşınabilirliğini artırmaya odaklanmaktadır.

**2.3 Çalışmanın Amacı**

Bu tezde, maskelenmiş dil modeli (MLM) olan BERT’in tahmin sıralamasını kullanarak sabit uzunluklu veri bloklarını ve hufman ağacı kullanılarak farklı uzunluktaki veri bloklarını doğal görünümlü metinlere gömebilen iki farklı yöntem önerilmektedir. Önerilen yaklaşım, karakter eşleşmesi veya kelime içi pozisyon aramaya dayanan geleneksel yöntemlerden farklı olarak, veri bloklarını tahmin edilen kelimelerin sırasına göre kodlayarak hem uygulama kolaylığı hem de daha doğal metin üretimi sağlamaktadır.

### GENEL KISIMLAR

**3.1 Biçim Tabanlı Yöntemler (Format-based Methods)**

**3.1.1. Satır ve Kelime Kaydırma Yöntemleri**

* "A Novel Approach to Format Based Text Steganography"  
  Bu çalışmada, satır ve kelime kaydırma teknikleri birleştirilerek metin belgelerinde gizli bilgi saklama yöntemi geliştirilmiştir. Yöntem, metin düzenini değiştirerek bilgi gizlemeyi hedeflemektedir [2].

**3.1.2. Boşluk ve Sekme Karakteri Kullanımı**

* "Text Steganographic Approaches: A Comparison"  
  Bu makalede, metin steganografisinde boşluk ve sekme karakterlerinin kullanımıyla ilgili çeşitli yöntemler karşılaştırılmıştır. Yazar, bu tekniklerin avantajlarını ve sınırlamalarını tartışmaktadır [3].

**3.1.3. Karakter Biçimlendirme ile Bilgi Gizleme**

* "New Text Steganography Technique by Using Mixed-Case Font"  
  Bu çalışmada, büyük ve küçük harflerin belirli bir düzenle kullanılmasıyla metin içerisinde bilgi gizleme yöntemi önerilmiştir. Yöntem, metnin görsel yapısını değiştirerek bilgi saklamayı amaçlamaktadır [4].

**3.2 İstatistiksel ve Rastgele Üretim Yöntemleri (Statistical and Random Generation Methods)  
  
3.2.1. Markov Zinciri Tabanlı Metin Steganografisi**

Hernán Moraldo'nun çalışmasında, Markov zincirleri kullanılarak otomatik olarak metin üretilmesi ve bu metinlere gizli bilgi gömülmesi ele alınmıştır. Bu yöntem, dilin istatistiksel özelliklerini modelleyerek doğal görünümlü metinler oluşturmayı hedefler [5].

**3.2.2. Markov Zinciri ve Huffman Kodlama ile Otomatik Stego Metin Üretimi**

Yang ve arkadaşları (2018), Markov zinciri ve Huffman kodlama tekniklerini birleştirerek otomatik steganografik metin üretimi üzerine bir yöntem geliştirmişlerdir. Bu yaklaşım, doğal dil örneklerinden öğrenilen istatistiksel modellerle gizli bilgileri içeren akıcı metinler üretmeyi amaçlamaktadır [6].

**3.2.3. Rastgele Sayı Üreteçleri ve Sıkıştırma Kullanarak Metin Steganografisi**

Mohamed Yousif Elmahi ve çalışma arkadaşları, Pseudo-Random Number Generators (PRNG) kullanarak rastgele metinler oluşturmuş ve bu metinlere gizli bilgileri entegre etmişlerdir. Ayrıca, oluşturulan stego-metni sıkıştırarak veri boyutunu azaltmayı hedeflemişlerdir [7].

**3.3 Dilbilimsel Yontemler**  
  
**3.3.1. Eşanlamlı Kelime Değişimi (Synonym Substitution) Tabanlı Yöntemler**

* **"Practical Linguistic Steganography using Contextual Synonym Substitution and Paraphrasing"**  
  Bu makalede, bağlama duyarlı eşanlamlı kelime değişimi ve yeniden ifade etme (paraphrasing) teknikleri kullanılarak metin steganografisi yöntemleri geliştirilmiştir. Google n-gram verileri kullanılarak kelime seçimlerinin bağlama uygunluğu değerlendirilmiştir [8].

**3.3.2. Antonym Tabanlı Yer Değiştirme (Antonym-Based Substitution)**

* **"Substitution-based linguistic steganography based on antonyms"**  
  Bu makalede, eşanlamlı kelime değişimi yerine zıt anlamlı kelimelerin kullanıldığı bir steganografi yöntemi önerilmiştir. Bu yaklaşım, metnin anlamını koruyarak gizli bilgilerin entegre edilmesini hedeflemektedir [9].

**3.3.3. Derin Öğrenme ve Dil Modeli Tabanlı Yöntemler**

* **"Neural Linguistic Steganography"**  
  Bu çalışmada, büyük ölçekli dil modelleri kullanılarak gizli bilgilerin metin içerisine entegre edilmesi üzerine bir yöntem önerilmiştir. Yöntem, aritmetik kodlama ve dil modeli dağılımlarını kullanarak gerçekçi görünen metinler üretmeyi amaçlamaktadır [10].
* **"Autoregressive Linguistic Steganography Based on BERT and Consistency Coding"**  
  Bu makalede, BERT tabanlı bir otomatik regresif model kullanılarak metin steganografisi yöntemi geliştirilmiştir. Yöntem, gizli bilgilerin metin içerisine entegre edilmesini sağlarken metnin akıcılığını ve doğal yapısını korumayı hedeflemektedir [11].
* **"A Character Based Steganography Using Masked Language Modeling"** Bu çalışma, metin steganografisi alanında, BERT tabanlı Maskeli Dil Modelleme (Masked Language Modeling - MLM) kullanarak karakter düzeyinde bilgi gizleme yöntemini önermektedir. Yöntem, mevcut bir metindeki belirli kelimeleri, gizli mesajın karakterlerini temsil edecek şekilde değiştirmeyi amaçlamaktadır [12].

### MATERYAL VE YÖNTEM

**4.1 Kullanılan Dil ve Geliştirme Ortamı**

Bu çalışmada, tüm algoritmalar Python programlama dili kullanılarak geliştirilmiştir. Python’un zengin doğal dil işleme (NLP) kütüphaneleri ve açık kaynaklı derin öğrenme araçları, metin işleme ve dil modeli tabanlı gizleme işlemleri için oldukça uygun bir altyapı sağlamaktadır.

Geliştirme ortamı olarak Google Colaboratory (Colab) tercih edilmiştir. Colab, GPU/TPU destekli çalışabilen bulut tabanlı bir platform olduğundan, transformer tabanlı modellerin kolayca eğitilmesi ve test edilmesine olanak tanımıştır. Projede kullanılan başlıca kütüphaneler şunlardır:

* transformers (HuggingFace): BERT ve GPT-2 gibi önceden eğitilmiş dil modellerine erişim
* torch (PyTorch): Derin öğrenme modeli yükleme ve çalışma ortamı
* scikit-learn: Vektörleştirme, benzerlik hesaplama ve istatistiksel metrikler
* matplotlib ve numpy: Görselleştirme ve sayısal işlemler

**4.2 BERT Tabanlı Dil Modeli Kullanımı**

Bu tez kapsamında geliştirilen steganografi algoritmaları, transformer tabanlı bir önceden eğitilmiş dil modeli olan BERT üzerine inşa edilmiştir. Özellikle, HuggingFace kütüphanesinden sağlanan “bert-base-cased” versiyonu kullanılmıştır. Bu model, İngilizce büyük/küçük harf duyarlılığına sahip olup, cümle bazlı bağlam anlamını güçlü biçimde modelleyebilen 12 katmanlı bir transformer mimarisine sahiptir.

BERT modeli, doğrudan metin üretimi yerine, maskelenmiş konumlara en uygun kelimeyi tahmin etme (Masked Language Modeling - MLM) yapısında çalışmaktadır. Bu özellik, gizli bilginin metne entegre edilmesinde oldukça önemlidir. Geliştirilen algoritmalarda, BERT modelinin verdiği olasılık tahminlerinden oluşan top-k kelime listesi kullanılarak, gizli bilgiye karşılık gelen kelimeler seçilmiştir.

**4.3 Softmax Fonksiyonu**

BERT modeli, maskelenmiş bir kelime için olası tahminleri bir logits vektörü (ham skorlar) olarak üretir. Bu skorlar doğrudan olasılık olarak yorumlanamaz. Bu nedenle logits değerleri softmax fonksiyonu ile normalize edilerek her kelime için bir olasılık değeri elde edilir. Bu dağılım, Huffman kodlama tabanlı algoritmada kullanılmak üzere Huffman ağacını oluşturmak için temel teşkil eder.

Softmax fonksiyonu şu şekilde tanımlanır:

Burada:

* i. kelimenin model tarafından atanan logit değeridir.
* , softmax sonrası elde edilen normalize edilmiş olasılığı ifade eder.

**4.4 Deterministik Rastgelelik için SHA-256 Kullanımı**

Bu tez kapsamında geliştirilen algoritmalarda, gizli bilginin metne entegre edileceği kelimelerin belirlenmesi için deterministik rastgele sayı üretimi kullanılmaktadır. Bu sayede, aynı gizli mesajın aynı örtü metne gömülmesi durumunda, hem gönderici hem de alıcı taraf aynı indeksleri hesaplayabilir.

Bu amaçla **SHA-256 (Secure Hash Algorithm 256-bit)** kriptografik hash fonksiyonu kullanılmıştır. Python’da hashlib.sha256() fonksiyonu ile uygulanan bu yapı, aşağıdaki biçimde kullanılmıştır:

Burada:

* seed, mesajın başında gömülü olarak bulunan 12 bitlik rastgelelik değeridir,
* index, gömülmek istenen bitin/karakterin sırasını belirtir,
* N, toplam kelime sayısıdır (mod alınan değer),
* Bu işlem hem gönderici hem de alıcı tarafından aynı şekilde uygulanarak gömme ve çıkarma süreçlerinin senkronize çalışması sağlanır.

**4.5 Pencere Boyutu(HALF\_WINDOW)**

Bu çalışmada kullanılan tüm algoritmalarda, metin içindeki kelime seçim aralığını belirleyen en önemli parametrelerden biri HALF\_WINDOW değeridir. Bu parametre, hem BERT modeline verilecek maskeleme penceresinin boyutunu hem de veri gömme sıklığını doğrudan etkiler. Örneğin, HALF\_WINDOW = 5 olduğunda, veri yalnızca her 6 kelimede bir gömülebilir. Bu durum, algoritmanın kapasitesini doğrudan belirlerken; pencere boyutunun küçültülmesi kapasiteyi artırabilir, ancak bağlam bütünlüğü açısından anlamlı tahminler yapılmasını zorlaştırabilir.

**4.6 Değerlendirme Metrikleri**

Geliştirilen steganografi algoritmalarının çıktılarının dil modeli açısından doğal görünüp görünmediğini ve gizli veriyi taşıma kapasitesini değerlendirmek için dört farklı metrik kullanılmıştır. Bu metrikler, metin düzeyinde hem anlamsal hem de istatistiksel farklılıkları ortaya koymayı amaçlamaktadır.

**4.6.1 Gömme Oranı (Embedding Rate - ER)**

Gömme oranı, birim metin başına kaç bit gizli bilgi yerleştirildiğini ölçer. Bu çalışmada, gizlenen toplam bit sayısı metindeki toplam değiştirilen kelime sayısına oranlanarak hesaplanmıştır. Kullanılan formül:

​, gömülen toplam bit sayısını, ise metindeki toplam değiştirlen kelime sayısını ifade eder.

**4.6.2 Kullback-Leibler Divergence (KLD)**

KL divergence, iki metnin kelime dağılımı arasındaki farkı ölçmek için kullanılmıştır. Özellikle örtü (cover) metin ile stego metin arasındaki sözcük frekans farkını ölçerek gizleme işleminin iz bırakıp bırakmadığını belirlemeye yardımcı olur. Hesaplama sırasında skikit-learn kütüphanesiyle elde edilen kelime frekansları normalleştirilmiş ve aşağıdaki formül uygulanmıştır:

Burada örtü metindeki kelimenin normalize edilmiş frekansını, ise stego metindeki karşılık gelen değeri ifade eder. Sayısal kararsızlıkları önlemek amacıyla frekans değerlerine

ε = 1e-10 alt sınırı uygulanmıştır.

**4.6.3 Perpleksite (Perplexity - PPL)**

Perpleksite, bir dil modelinin bir metni ne kadar öngörülebilir bulduğunu gösterir. Bu metrik, örtü ve stego metinlerinin akıcılığını kıyaslamak için kullanılmıştır. GPT-2 dil modeli (gpt2) üzerinden hem örtü metin hem de stego metin için aşağıdaki yöntemle hesaplanmıştır:

Burada L, GPT-2 modelinin verdiği ortalama cross-entropy loss değeridir. Metin uzunluğu modelin sınırlamalarını aştığında, metin parçalara bölünerek parça başına loss alınmış ve bu değerlerin ortalaması üzerinden PPL hesaplanmıştır.

**4.6.4 Anlamsal Benzerlik (Semantic Similarity - SIM)**

Gizli veri yerleştirme işleminden sonra metnin anlamında ne kadar bozulma olduğunu ölçmek için örtü ve stego metinlerinin vektörel temsilleri üzerinden cosine benzerliği hesaplanmıştır. Bu işlem CountVectorizer ile kelime frekanslarına dayalı bir vektör uzayı oluşturularak gerçekleştirilmiştir. Cosine similarity şu şekilde hesaplanır:

Burada A ve B, sırasıyla örtü ve stego metinlerinin vektör temsilleridir. Değerin 1'e yakın olması metinlerin yüksek düzeyde anlamsal olarak benzer olduğunu gösterir.

### BULGULAR

**5.1 Maskeli Dil Modellemesi Kullanarak Karakter Tabanlı Metin Steganografisi**

Bu bölümde, bert-base-cased modelinden yararlanarak geliştirilen bir karakter tabanlı metin steganografisi yöntemi anlatılmaktadır [12]. Bu yöntem, gizli bir mesajı verilen bir kapak metin içerisine, her kelimenin içinde yalnızca bir karakter gizleyecek şekilde, doğal ve dikkat çekmeyecek biçimde gömmeyi amaçlamaktadır. Alıcı taraf, aynı model ve parametreler ile gizli mesajı tekrar elde edebilir.

**Genel Bakış**

Yöntem, BERT modelinin maskelenmiş dil modelleme (Masked Language Modeling - MLM) özelliğinden faydalanır. Belirli kelimeler, küçük bir bağlam penceresi içinde maskelenerek modele verilmekte ve modelin önerdiği yüksek olasılıklı tahminlerden bir tanesi ile değiştirilerek içine gizli karakter kodlanmaktadır. Her tahminin belirli bir karakter konumundaki (örneğin 1. ya da 2. harfi) karakteri, gizli mesajdaki bir karakter ile eşleşirse, bu tahmin stego kelime olarak seçilir.

**5.1.1 Başlık (Header) Gömme**

Gizli mesaja geçmeden önce, kapak metnin başına 30 bitlik bir başlık (header) bilgisi gömülür. Bu başlık aşağıdaki bileşenlerden oluşur:

* **12 bitlik bir tohum (seed):** Rastgele kelime seçimlerin, alıcı tarafından da aynen yapılabilmesini sağlar.
* **12 bitlik karakter sayısı (char count)**: Gizli mesajın kaç karakter içerdiğini belirtir.
* **6 bitlik döngü indeksi (loop index)**: Her kelimede gizli karakterin hangi konumda (0. karakter, 1. karakter, vb.) saklandığını belirtir.

Bu 30 bitlik bilgi, 8 bitlik 3 parçaya ayrılarak, metin içinde sabit aralıklarla (örneğin her HALF\_WINDOW + 1 kelimede bir) seçilen üç kelimeye gömülür. Her kelime, etrafındaki küçük bir pencereyle birlikte maskelenerek modele verilir ve modelin ilk 257 tahmini (top-k=257) içinden, ilgili 8-bitin ondalık karşılığının değerine karşılık gelen kelime ile değiştirilir.

**5.1.2 Gizli Mesaj Gömme**

Asıl gizli mesaj gömme işlemi karakter karakter gerçekleştirilir. Her karakter için şu adımlar izlenir:

1. **Deterministik rastgele seçim**: Her karakter, metindeki bir kelimeye saklanmalıdır. Seed ve karakterin sırası birlikte kullanılarak bir sayı üretilir. Bu sayı daha sonra **SHA-256** adlı bir şifreleme algoritmasından geçirilir. Bu algoritma sayesinde rastgele bir sayı üretilmiş olur ve bu sayı metindeki hangi indeksdeki kelimeyi seçeceğimize karşılık gelir. Bu yöntem sayesinde aynı seçim, alıcı tarafından da yapılabilir.
2. **Atlama kontrolü**:
   * Üretilen indeks değeri, başlık gömme esansında kullanılan kelimelerle çakışıyorsa veya
   * Daha önce kullanılan indekslere çok yakınsa (HALF\_WINDOW kadar yakınsa), bu indeks atlanır. Eğer bu atlama kontrolu yapılmaz ise BERT çözümleme esnasında farklı tahmin listeleri üreteceğinden algoritma hatalı sonuç verir.
3. **Maskelenmiş pencere oluşturma**: Seçilen kelime çevresinde sabit boyutta bir pencere (örneğin 5 kelime) oluşturulur ve merkezdeki kelime maskelenir ( [MASK] ). Ardından bu cümle BERT modeline verilir.
4. **BERT tahmini alma**: Model maskelenmiş konuma gelebilecek anlamlı 257 tahminde bulunur.
5. **Karakter eşleştirme**: Tahmin edilen kelimeler listesinden, birinci indeksten itibaren kontrol edilir ve herhangi bir kelimenin loop\_index konumundaki karakteri, gizli karakter ile eşleşirse, bu kelime stego kelime olarak seçilir ve metne yerleştirilir.
6. **Eşleşme bulunamazsa**: Tahminler içinde eşleşen bir kelime yoksa, sıfırıncı indeksteki tahmin yerleştirilir. Bu kelime gizli karakter taşımaz. Saklanamayan harf bir sonraki iterasyonda tekrar denenir.

Bu işlem tüm gizli mesaj bitene kadar devam eder. loop\_index değeri her karakter için döngüsel olarak değiştirilebilir (örneğin 0–3 arasında döner) ve bu da gizleme sürecine çeşitlilik kazandırır.

**5.1.3 Mesaj Çözümleme (Gizli Mesajı Geri Çekme)**

Alıcı taraf, metni analiz ederek şu adımlarla mesajı geri elde eder:

1. **Başlığı çözümleme**: İlk üç kelime kullanılarak 30 bitlik başlık elde edilir.
2. **Aynı seed ile indeks üretimi**: Gömmede kullanılan indeksler aynı şekilde yeniden hesaplanır.
3. **Kelime analizleri**:
   * Maskelenmiş pencere oluşturulur ve BERT tahminleri alınır.
   * Skip kelimeler atlanır.
   * Diğer kelimelerde loop\_index konumundaki karakter alınarak mesaj oluşturulur.

Bu şekilde, gizli mesaj tamamen geri elde edilir.

**5.1.4 Örnek Üzerinden Algoritma Açıklaması**

Aşağıda algoritmanın çalışması adım adım örnekli şekilde anlatılmıştır:

**Örnek:**

* Örtü metni (cover text):  
  "Forms of steganography have been used for centuries and include almost any technique for hiding a secret message in an otherwise harmless container.”
* Gizli mesaj: "ok"
* Parametreler:
  + HALF\_WINDOW = 2 → Maskeleme penceresi yarıçapı.
  + HEADER\_BITS = 30 → Başlık 12 bitlik rastgelelik tohum değeri, 12 bit karakter sayısı ve 6 bit döngü indeksi içerir.

**Adım 1: Gizli Mesajın Binary Temsili**

Gizli mesaj "ok" önce ASCII karakterlerine, ardından ikili (binary) biçime dönüştürülür:

* "o" → 01101111
* "k" → 01101011
* Mesajın toplam ikili karşılığı: 0110111101101011

Bu ikili dizinin başına mesajın çözülmesi için gerekli başlık (header) bilgisi eklenir.

**Adım 2: Header Bilgisinin Kodlanması**

Header, aşağıdaki üç bilgiyi içerir:

* Rastgelelik tohumu (seed) → Örneğin 12 → 000000001100
* Karakter sayısı (“ok”) → 2 → 000000000010
* Döngü indeksi (loop\_index) → Örneğin 1 → 000001

Toplam header binary:  
000000001100000000000010000001 (30 bit)

Bu 30 bit, 8-bitlik bloklara bölünerek örtü metne gömülecek 4 kelime belirlenir. 30 sayısı 8’e tam bölünemediği için 2 tane “0” biti sona eklenir.

**Adım 3: Header’ın Örtü Metne Gömülmesi**

Header'ı gizlemek için örtü metin kelimeleri üzerinde bir plan yapılır. Gömme işlemi HALF\_WINDOW + 1 kadar aralıklarla yapılmalıdır. Örneğin, HALF\_WINDOW = 2 ve ilk index = 3’ten başlarsa Sonraki header kelimeleri: index = 3 + 3 = 6, index = 6 + 3 = 9, index = 9 + 3 = 12 şeklinde belirlenir.

Header’ın örtü metne gömülme süreci Şekil 5.1’de gösterilmiştir. Bu indexlerdeki kelimeler maskeleme (masking) işlemine tabi tutulur. Örneğin, index = 3 için pencere "of steganography have been used " olur ve "have" kelimesi [MASK] ile değiştirilerek "of steganography [MASK] been used" elde edilir. Bu cümle BERT modeline verilir ve ilk 257 tahmin içerisinden, header bit bloğuna karşılık gelen kelime seçilir. Örneğimiz için ilk 8 bitlik blok “00000000” olduğundan tahminler içerisinden sıfırıncı indeksteki kelime seçilir.

**Adım 4: Mesaj Karakterlerinin Gömülmesi**

Gizli mesajın örtü metne gömülme süreci Şekil 5.2’de gösterilmiştir

İlk mesaj karakteri 'o'

1. **SHA-256 ile Rastgele Seçim:**

**SHA-256** seed ve karakter numarası (iterasyon numarası) parametreleriyle çalıştırılır ve rastgele sayı üretilir. Örneğimiz için bu parametreler 12 ve 0 sayılarıdır. Ardından atlama kontrolu başlığında anlatılan işlemler yapılır. Örneğin üretilen sayı 15 den küçükse bu sayı atlanır çünkü bu kısım header için ayrılmıştır. Bu örneğimiz için 16 sayısının üretildiğini farz edelim.

1. **Maskelenmiş Cümle Üretimi:**

16. kelime maskelenir. Örneğin: “hiding a [MASK] message in”

1. **BERT Tahmini ve Gömme:**

BERT modeli maskelenmiş yere gelebilecek anlamlı kelimeleri verir. Örnek olarak şu tahminleri yaptığını farz edelim: “hidden”, “secret”, “coded”, “encrypted” … Loop\_index değerimizin 1 olduğunu göz önünde bulundurursak bu kelimelerden birinci indeksi “o” olan kelime seçilir yani “coded” kelimesi. Bu durumda yeni cümlemiz “hiding a coded message in” olur. “o” harfi başarılı bir şekilde saklanmıştır ve “k” harfi içinde aynı işlemler yapılır. Fakat bu sefer **SHA-256** 12 ve 1 parametreleriyle çalıştrılır ve ayrıca üretilen sayı “o” harfi için seçilen indeksle arasında en az HALF\_WINDOW kadar mesafe olmalıdır. Yani 14, 15, 16, 17 ve 18 indeksleri seçilmemlidir. Ayrıca her döngü sonunda olduğu gibi loop indeks değeri 1’den 2 ye arttırılmıştır.

**Adım 5: Mesajın Çözülmesi**

Gizli mesajı ortaya çıkarmak için aşağıdaki adımlar izlenir:

* Header verisi, örtü metnin HALF\_WINDOW + 1 konumundan başlayarak HALF\_WINDOW + 1 aralıklarla okunan kelimelerden geri elde edilir.
* Bu header sayesinde:
  + Kullanılan seed,
  + Mesaj karakter sayısı,
  + Başlangıç loop\_index bilgileri elde edilir.
* Mesaj karakterleri, tıpkı gömülme sürecinde olduğu gibi, aynı hash tabanlı index üretimi ile seçilen kelimelerden, loop\_index pozisyonundaki karakter okunarak elde edilir. Yani **SHA-256** yine 16 sayısını üretir. Bu pozisyondaki kelime “coded” maskelenir ve BERT modeline cümle gönderilir. Örneğimiz için bu “hiding a [MASK] message in”.  
  Ardından yine buraya gelebilecek anlamlı kelimelerin listesi üretilir ve “coded” kelimesinin sıfırıncı indekste yer alıp almadığı control edilir. Tahminler yine gömme işleminde aynen olduğu gibi “hidden”, “secret”, “coded”, “encrypted” … olacaktır. “coded” kelimesi ilk indekste yer almadığında mesaj saklama işlemi başarılı demektir. Yani bu durumda “coded” kelimesinin loop\_index pozisonundaki indeksinde gizli karakterimiz mevcuttur. Loop\_index değerimiz 1 olduğunda gore “coded” kelimesinin birinci indeksi “o” karakteri başarılı bir şekilde bulunmuştur. Aynı işlemler diğer karakterler içinde devam eder.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Şekil 5.1: Karakter Tabanlı Yöntemde Başlık Gömme İşlemi**

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Şekil 5.2: Karakter Tabanlı Yöntemde Gizli Mesaj Gömme İşlemi**

**5.1.5 Karakter Tabanlı Yöntemde Gömme İşlemi (Pseudo-code)**

**Input: cover\_text, secret\_message, seed, loop\_index**

**Output: stego\_text**

**1. cover\_text → kelimelere böl (words[])**

**2. secret\_message → 8-bit binary dizisine dönüştür**

**3. Header oluştur: seed (12b), karakter sayısı (12b), loop\_index (6b)**

**4. Header’ı sabit aralıklarla (HALF\_WINDOW + 1) göm:**

**a. Maskeli pencere oluştur**

**b. BERT tahminleri al (top-k=257)**

**c. İlgili bit blok değerine karşılık gelen kelime ile değiştir**

**5. Başla: i = 0, hidden = 0**

**6. Her karakter için aşağıdaki işlemleri yap:**

**a. index = SHA256(seed + i) mod len(words)**

**b. index, başlık aralığına çok yakınsa veya daha önce kullanılmışsa → i++**

**c. Seçilen indeks etrafında pencere oluştur, merkez kelimeyi [MASK] yap**

**d. BERT tahmin listesi al**

**e. Tahmin edilen kelimelerin loop\_index konumundaki harfi gizli karakterle eşleşirse kelimeyi yerleştir**

**f. Eşleşme yoksa skip kelimesi yerleştir**

**g. hidden++ yalnızca eşleşme başarılıysa**

**h. i++**

**7. stego\_text olarak güncellenmiş kelimeleri döndür**

**5.1.6 Karakter Tabanlı Yöntemde Çözme İşlemi (Pseudo-code)**

**Input: stego\_text**

**Output: secret\_message**

**1. stego\_text → kelimelere böl**

**2. Header’ı çöz: sabit aralıklarla okunarak 30 bit elde edilir**

**3. seed, char\_count, loop\_index elde edilir**

**4. Başla: i = 0, found = 0**

**5. Her karakter için:**

**a. index = SHA256(seed + i) mod len(words)**

**b. index geçersizse (header alanında veya çok yakınsa) → i++**

**c. index geçerliyse:**

**i. pencere oluştur, merkez kelimeyi maskele**

**ii. BERT tahminlerini al**

**iii. Eğer kelime skip kelimesiyse → i++**

**iv. Aksi takdirde, loop\_index konumundaki karakteri al → secret\_message += char**

**v. found++**

**d. i++**

**6. Gizli mesaj döndürülür**

**5.2 Sabit Genişlikli Bit Kodlama ile Metin Steganografisi (Fixed-Bit Yöntemi)**

Bu bölümde, karakter tabanlı steganografi yöntemine alternatif olarak geliştirilen sabit bit genişlikli (fixed-bit) bir yaklaşım tanıtılmaktadır. Bu yöntem, her bir kelimeye yalnızca bir karakter gizlemek yerine, kelime seçim sürecinde daha az tahmin arasından seçim yaparak doğrudan sabit genişlikte bitler (örneğin 2 bit) gömmeye olanak tanır. Böylece, mesaj başına gizlenebilen bilgi miktarı azalırken daha anlamlı tahminler yapılması hedeflenmektedir.

**5.2.1 Yöntem Özeti ve Karakter Tabanlı Yöntemden Farkları**

**1. Gizli Mesaj Kodlaması: Karakter Yerine Bit Dizileri**

İlk algoritmada her kelimeye yalnızca bir karakter (8 bit) gizleniyordu. Bu yeni yöntemde ise her kelimeye sabit uzunlukta n bit gömülür. Örneğin, bit\_width=2 olduğunda, her kelimede yalnızca 2 bit bilgi saklanır. Bu sayede tahmin edilen kelimeler doğrudan bu 2 bitlik değeri taşıyacak şekilde seçilir.

**2. Tahmin Seçimi: Karakter Eşleştirme Yerine Sıralı Bit Kodlama**

İlk yöntemde, BERT'in tahmin ettiği kelimelerin loop\_index konumundaki karakteri gizli mesajdaki karakter ile eşleşene kadar arama yapılırdı. Bu yöntemde ise karakter eşleştirme yapılmaz. Bunun yerine BERT’in ilk 2^bit\_width (örneğin 4) tahmini arasından, gömülecek bit değerinin ondalık karşılığına denk gelen indeks seçilir. Örneğin gömülecek bit dizisi “10” (ondalıkta 2) ise, tahmin listesinin üçüncü kelimesi (index=2) doğrudan seçilir.

**3. Başlık (Header) Yapısı ve Gömme**

Bu yöntemde daha az bilgiye ihtiyaç duyulduğu için başlık (header) sadece 24 bitliktir. Bunun 12 biti seed diğer 12 biti karakter uzunluğu içindir.

Karakter tabanlı yöntemde ek olarak loop\_index bilgisi de header’da tutuluyordu. Ancak bu yöntem karakter pozisyonuna göre eşleme yapmadığı için buna gerek yoktur.

Header, belirlenen bit\_width boyutunda parçalara bölünerek HALF\_WINDOW + 1 adım aralıkla metindeki kelimelere gömülür. Gömme işlemi, header bit bloğunun ondalık karşılığı olan indeksle BERT tahminleri içinden doğrudan kelime seçilerek yapılır.

**4. Gizli Mesaj Gömme: Bit Blokları ve Belirli Tahmin Seçimi**

* Gizli mesaj metne çevrilip bit dizisine dönüştürülür.
* Bu bit dizisi, bit\_width kadar parçalanır (örneğin her biri 2 bit olacak şekilde).
* Her 2-bitlik blok için:
  + SHA-256 tabanlı deterministik bir indeks seçilir.
  + İlgili kelime maskelenir ve BERT'e verilir.
  + İlk 2^bit\_width tahmin içerisinden, bit blok değerine karşılık gelen kelime seçilir.

**5. Mesaj Çözümleme (Dekodlama)**

* Header’dan seed ve mesaj uzunluğu alınır.
* Her bit\_width kadar bitlik blok için:
  + Aynı hash tabanlı indeks üretilir.
  + Maskelenmiş cümle hazırlanarak BERT’e verilir.
  + İlk 2^bit\_width tahmin listesi içinde örtü metindeki kelimenin kaçıncı sırada olduğu kontrol edilir.
  + Sıradaki indeks, karşılık gelen bit olarak kabul edilir ve böylece orijinal bit dizisi geri elde edilir.
* Bit dizisi daha sonra metne çevrilerek orijinal gizli mesaj geri elde edilir.

**5.2.2 İki Bitlik Örnek**

Gizli mesajın örtü metne gömülme süreci Şekil 5.3’de gösterilmiştir.

Gizli Mesaj: “ok”

ASCII → Binary:

* “o” → 01101111
* “k” → 01101011

Toplam: 0110111101101011 → 16 bit

Bu mesaj, 2 bitlik bloklara ayrılır:  
01 10 11 11 01 10 10 11

Her blok için, örneğin “01” (ondalık: 2), BERT’in tahminlerinden index=1 seçilir. Örneğin “hiding a [MASK] message in” maskeli cümlesi için tahminler şunlar olabilir:  
['hidden', 'coded', 'secret', 'encrypted']

01 → 2 → 'secret' kelimesi metne yerleştirilir. Bu seneryoda olduğu gibi bazı durumlarda tesadüfen aynı maskelenen yere yine aynı kelimede gelebilir. Bu işlem her blok için tekrarlanır.

metin, ekran görüntüsü, yazı tipi, çizgi içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**Şekil 5.3: Sabit Genişlikli Bit Kodlama Yönteminde Gizli Mesaj Gömme İşlemi**

**5.2.3 Sabit Bit Genişlikli Yöntemde Gömme İşlemi (Pseudo-code)**

**Input: cover\_text, secret\_message, seed, bit\_width**

**Output: stego\_text**

**1. secret\_message → bit dizisine dönüştür**

**2. Header oluştur: seed (12b) + bit\_length (12b)**

**3. Header bitlerini sabit aralıklarla göm (BERT tahmin listesi üzerinden)**

**4. Başla: i = 0, hidden = 0**

**5. Her bit\_width’lik blok için:**

**a. index = SHA256(seed + i) mod len(words)**

**b. index uygun değilse (header’a yakın veya kullanılmış) → i++**

**c. pencere oluştur, merkez kelimeyi maskele**

**d. BERT tahmin listesi al (ilk 2^bit\_width tahmin)**

**e. bit bloğunun ondalık karşılığına denk gelen kelimeyi yerleştir**

**f. hidden += bit\_width**

**g. i++**

**6. Gömülmüş metin döndürülür**

**5.2.4 Sabit Bit Genişlikli Yöntemde Çözme İşlemi (Pseudo-code)**

**Input: stego\_text**

**Output: secret\_message**

**1. stego\_text → kelimelere böl**

**2. Header'ı çöz → seed ve bit\_length elde edilir**

**3. Başla: i = 0, extracted\_bits = ''**

**4. Bit dizisi tamamlanana kadar:**

**a. index = SHA256(seed + i) mod len(words)**

**b. index uygun değilse → i++**

**c. pencere oluştur, merkez kelimeyi maskele**

**d. BERT tahminlerini al**

**e. Kelimenin tahmin listesinde kaçıncı sırada olduğunu bul → bit dizisi olarak ekle**

**f. i++**

**5. Bit dizisi → metne çevrilerek mesaj elde edilir**

**5.3 Huffman Tabanlı Steganografi Yöntemi**

Bu çalışmada geliştirilen üçüncü algoritma, gizli veriyi gizlemek için Huffman kodlaması tabanlı bir yaklaşım kullanır. Daha önce geliştirilen karakter tabanlı ve sabit bit genişlikli yöntemlerde sabit uzunlukta bit desenleri ile seçim yapılırken, bu yöntemde kullanılan her kelimenin kod uzunluğu, modelin o kelimeye verdiği olasılığa bağlı olarak değişkenlik göstermektedir. Bu sayede daha yüksek olasılıklı kelimelere daha kısa kodlar atanırken daha düşük olasılıklı kelimelere daha uzun kodlar atanarak daha etkili ve esnek bir gizleme mekanizması oluşturulması hedeflenmektedir.

**5.3.1 Temel Farklar ve Yöntemin Açıklaması**

* **Değişken Uzunlukta Kodlama:** Huffman yöntemi, modelin [MASK] yerine önerdiği en olası TOP\_K=16 kelimeye karşılık gelen olasılıkları kullanarak bir Huffman ağacı oluşturur. Böylece yüksek olasılığa sahip kelimelere daha kısa, düşük olasılıklılara daha uzun bit dizileri atanır. Ayrıca TOP\_K değeri yerine farklı sayılar yazarakta algoritmayı çalıştırabiliriz.
* **Bit Eşleştirme:** Gizlenecek bit dizisi, bu Huffman sözlüğü üzerinden kodlarla eşleştirilerek kelime seçimleri yapılır. Eğer kalan bit sayısı ağaçtaki minimum uzunluktaki bit bloğundan kısa ise (‘0000’) ile pad edilir.

**Başlık ve Mesaj Saklama**

* **Başlık Saklama:** Gizli mesajın uzunluğu ve seed bilgisi, karakter tabanlı yöntemle benzer şekilde gömülür. Kelimeler sabit indeks aralıklarında maskelenir ve her bir pozisyonda modelden gelen tahminler ile oluşturulan Huffman ağacı kullanılarak ilgili bit desenine karşılık gelen kelime seçilir.
* **Mesaj Saklama:** Başlıktan sonra gelen kelimelerde, rastgele hash fonksiyonu ile seçilen pozisyonlarda maskelenmiş kelime üzerinden tahminler alınır ve yine Huffman kodlama ile eşleşme sağlanarak gizli bitler kelimelere gömülür.

**Çözümleme Süreci**

Gizli mesajın çözülmesi için:

1. **Başlık Çözümü:** Başlıkta kullanılan kelimeler ve onların Huffman kodları çözülerek seed ve gizli mesajın uzunluğu elde edilir.
2. **Mesaj Çözümü:** seed kullanılarak aynı rastgele indeksler hesaplanır, her pozisyondaki kelime tahmin listesi yeniden oluşturulur ve orijinal kelimenin bulunduğu Huffman sözlüğündeki karşılığı okunarak bitler geri elde edilir.
3. **Yazı Çıkarımı:** Elde edilen bit dizisi, 8-bit ASCII karakterlerine dönüştürülerek orijinal mesaj geri elde edilir.

**5.3.2 Örnek Huffman Ağacı Uygulaması**

Gizli mesajın örtü metne gömülme süreci Şekil 5.4’de gösterilmiştir

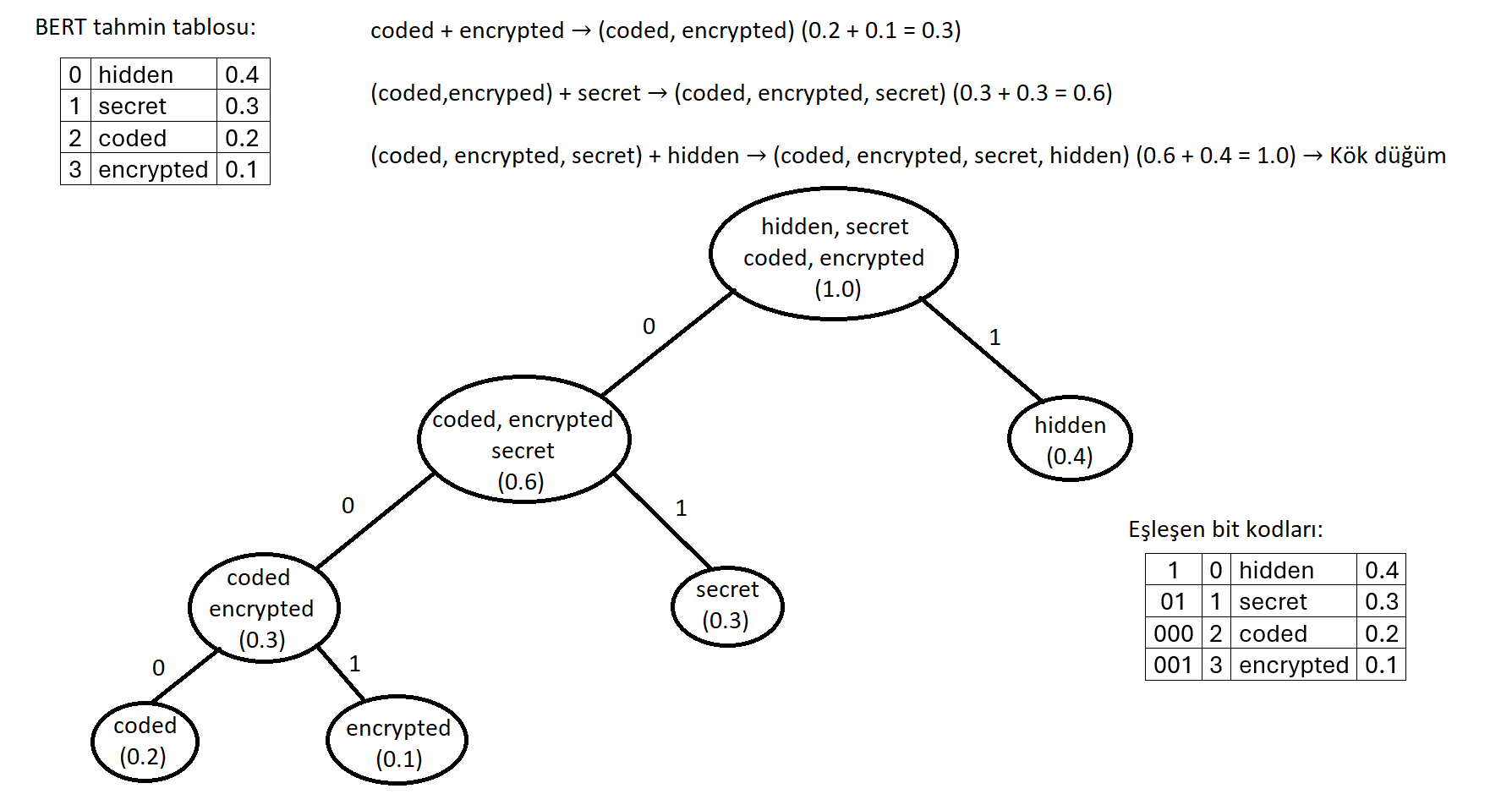
Örnek olarak, “hiding a [MASK] message in” cümlesinin BERT modeline verildiğini ve modelin en yüksek olasılıkla önerdiği dört tahminin sırasıyla ['hidden', 'coded', 'secret', 'encrypted'] olduğunu varsayalım. Bu işlem sırasında BERT, modelin tüm kelime dağarcığı için her kelimenin o konumda gelme olasılığını temsil eden bir logits vektörü üretir.

Logits, her kelime için modelin "ham güvenini" ifade eden gerçek sayılardır ve doğrudan olasılık olarak yorumlanamazlar. Bu nedenle, bu değerler üzerinde softmax fonksiyonu uygulanır. Softmax, tüm logits değerlerini normalize ederek bunları [0,1] aralığında değerlere dönüştürür ve toplamlarını 1 yapar. Böylece her kelime için anlamlı bir olasılık değeri elde edilir.

Bu örnekte, ilgili kelimelere ait normalize edilmiş olasılıkların sırasıyla 0.4 ('hidden'), 0.3 ('coded'), 0.2 ('secret') ve 0.1 ('encrypted') olduğunu varsayalım. Bu olasılıklar kullanılarak Huffman ağacı şu şekilde oluşturulur:

1. En küçük iki olasılık olan ‘encrypted’ (0.1) ve ‘coded’ (0.2) birleştirilerek ara bir düğüm oluşturulur (0.1 + 0.2 = 0.3).
2. Oluşan bu düğüm ‘secret’ (0.3) ile birleştirilerek daha büyük bir düğüm oluşturulur (0.3 + 0.3 = 0.6).
3. Son olarak, bu düğüm ‘hidden’ (0.4) ile birleştirilerek ağacın kök düğümü elde edilir (0.6 + 0.4 = 1.0).

Huffman ağacı tamamlandıktan sonra, geleneksel olarak her sol dal için '0' ve sağ dal için '1' biti atanır. Böylece her kelimeye karşılık gelen benzersiz bir Huffman kodu elde edilir.



**Şekil 5.4: Huffman Tabanlı Yöntemde Gizli Mesaj Gömme İşlemi**

**5.3.3 Huffman Yönteminde Gömme İşlemi (Pseudo-code)**

**Input: cover\_text, secret\_message, seed**

**Output: stego\_text**

**1. secret\_message → bit dizisine dönüştür**

**2. Header oluştur: seed (12b) + bit\_length (12b)**

**3. Header bitlerini göm:**

**a. sabit indekslere git**

**b. pencere oluştur, tahmin listesi al, softmax uygula**

**c. Huffman ağacı oluştur, header bitleriyle eşleşen kelimeyi yerleştir**

**4. Başla: i = 0, hidden = 0**

**5. Bit dizisi boyunca:**

**a. index = SHA256(seed + i) mod len(words)**

**b. index geçersizse → i++**

**c. pencere oluştur, merkez kelimeyi maskele**

**d. tahmin listesi al, softmax ile olasılıklar elde et**

**e. Huffman ağacı oluştur, bit dizisinin başıyla eşleşen kodu bul**

**f. eşleşen kelimeyi yerleştir, hidden += kod uzunluğu**

**g. i++**

**6. stego\_text döndürülür**

**5.3.4 Huffman Yönteminde Çözme İşlemi (Pseudo-code)**

**Input: stego\_text**

**Output: secret\_message**

**1. stego\_text → kelimelere böl**

**2. Header çözülür: seed ve bit\_length elde edilir**

**3. Başla: i = 0, bits = ''**

**4. Bit dizisi tamamlanana kadar:**

**a. index = SHA256(seed + i) mod len(words)**

**b. pencere oluştur, merkez kelimeyi maskele**

**c. tahmin listesi al, softmax uygula, Huffman ağacı oluştur**

**d. Orijinal kelimenin karşılık geldiği Huffman kodunu bul**

**e. bits += code**

**f. i++**

**5. bits → karakter dizisine çevrilerek mesaj elde edilir**

**5.4 Algoritmaların Değerlendirilmesi**

**1. Karakter Tabanlı Yaklaşım**

**Kapasite:**  
Bu yöntem, her bir kelimeye bir karakter gömecek şekilde tasarlanmıştır. Bu da kelime başına 8 bitlik bir veri saklama kapasitesi anlamına gelir ve test edilen diğer yöntemlere göre teorik olarak en yüksek kapasiteyi sunar. Ancak bu yaklaşım, gömülecek karakterin BERT modelinin tahmin listesinde, belirli bir konumda (örneğin loopIndex) yer aldığı kelimelerin eksikliği nedeniyle bazı durumlarda başarısız olabilir. Özellikle, gizli mesajda İngilizcede nadiren geçen karakterler (örneğin ‘z’, ‘q’, ‘x’ gibi) sıkça bulunuyorsa, algoritma bu karakterleri gömecek uygun kelimeyi bulamayabilir. Bu durum, toplam kapasiteyi düşüren atlamalara neden olabilir.

**Anlamsal Tutarlılık:**  
Bu algoritma, BERT modelinden alınan en iyi 257 tahmin arasından, yalnızca hedef karakteri istenilen konumda içeren kelimeleri seçer. Tahmin listesinde oldukça aşağı sıralarda (örneğin 240. sırada) yer alan kelimeler de kullanılabildiğinden, bağlama uymayan veya anlamı bozan kelimeler seçilebilir. Bu da semantik bütünlüğün zaman zaman bozulmasına neden olabilir. Ancak dikkat edilmesi gereken önemli bir fark, bu yöntem kelime seçiminde BERT’in tahmin listesinin başından aşağıya doğru arama yaptığı için yüksek olasılıklı kelimeleri öncelikli olarak değerlendirme eğilimindedir. Bu, bazı durumlarda semantik açıdan daha anlamlı kelimelerin seçilmesini sağlayabilir.

**Ek Not:** Bu yöntemin başarısı, gizli mesajdaki karakterlerin İngilizce kelimelerde ne sıklıkta geçtiğine büyük ölçüde bağlıdır.

**2. Sabit Bit Genişlikli (Fixed Bit) Yaklaşım**

**Kapasite:**  
Bu yöntem, her kelimeye sabit sayıda bit (örneğin 2-bit veya 4-bit) gömerek çalışır. Bu, karakter tabanlı yaklaşıma kıyasla kelime başına daha az veri saklama kapasitesi anlamına gelir. Ancak, sabit bit genişliği kullandığı için her kelimeye mutlaka bir bit dizisi gömülebilir ve karakter tabanlı yöntemde olduğu gibi atlamalar yaşanmaz. Bu nedenle, pratikte dahadüşük teorik kapasiteye sahip olsa da gizli mesajın tamamını başarıyla gömmekte daha kararlı olabilir.

**Anlamsal Tutarlılık:**

Bu yöntem, yalnızca 2^n (örneğin 4 veya 8) adet en yüksek olasılıklı tahmin arasından seçim yaptığı için, kelime bağlamına daha uygun ve semantik olarak anlamlı kelimeler tercih edilme olasılığı yüksektir. Ancak seçim doğrudan bir bit dizisi ile eşleştiği için kelime listesinin sırasına göre önceliklendirme yapılmaz. Bu da bazen yüksek olasılıklı ama semantik olarak daha uygun olan kelimenin göz ardı edilmesine neden olabilir.

**3. Huffman Kodlama Tabanlı Yaklaşım**

**Kapasite:**  
Bu yöntem, gizli mesajı BERT’in tahmin ettiği kelimelere Huffman kodlaması uygulayarak gömer. Huffman algoritması, kelimelere tahmin olasılıklarına göre kısa veya uzun bit dizileri atar. Yüksek olasılıklı kelimeler kısa bit dizileri ile eşleştirilirken, düşük olasılıklı kelimeler daha uzun bit dizileri alır. Bu, kelime başına düşen ortalama bit miktarını azaltır; dolayısıyla karakter tabanlı yönteme göre daha düşük kapasite sunar. Ancak tıpkı sabit bit yöntemde olduğu gibi, bu algoritma da kelime seçimi konusunda daha kararlıdır ve karakter tabanlı yöntemdeki gibi gömülememe sorunları yaşanmaz.

**Anlamsal Tutarlılık:**

Bu algoritma, yalnızca önceden belirlenmiş en iyi tahminleri kullanır ve bu tahminlerin tahmin listesinde üst sıralarda yer alması beklenir. Bu nedenle semantik tutarlılığı görece yüksektir. Ancak, karakter tabanlı yöntemde olduğu gibi tahmin listesinin başından aşağıya doğru bir arama yapılmaz; burada da sabit bit yaklaşımında olduğu gibi doğrudan eşleşmeye dayalı bir işlem söz konusudur. Yani bazı durumlarda bağlama en uygun kelime seçilemeyebilir.

**5.5 Deney Sonuçları**

Aşağıdaki tablo 5.5’te, dört farklı algoritmanın (Character, Fixed-Bit-2, Fixed-Bit-4, Huffman) 50 gizli mesajla yapılan testler sonucunda elde edilen ortalama metriklerini göstermektedir. Her mesaj yaklaşık 17.68 karakter uzunluğundadır. Gizli mesajlar sadece karakterlerden oluşmaktadır çünkü karakter tabanlı yöntem sadece karakter gömebilmektedir. Kapak metninin değiştirilmemiş halinin perplexity değeri 20.41778946’dır. Her sütundaki en iyi sonuç yeşil, en kötü sonuç kırmızı arka plan ile gösterilmiştir.

**Tablo 5.5: Deney Sonuçları**

metin, yazı tipi, çizgi, sayı, numara içeren bir resim

Yapay zeka tarafından oluşturulan içerik yanlış olabilir.

**1. Kapasite**

* Karakter tabanlı yöntem, ortalama 21.68 kelime değiştirerek yalnızca 0. 48 kelimeyi atlamaktadır. Yaklaşık 22 kelime değiştirerek en yüksek kapasiteye ulaşan algoritmadır.
* Sabit 2-bit kodlama yöntemi ise ortalama 82.72 kelime değiştirerek en düşük kapasiteye sahip algoritmadır.

**2. Anlamsal Bozulma:**

* Genel olarak metindeki anlamsal bozulmaya bakıldığında, en az bozulan algoritma karakter tabanlı yöntem, en fazla bozulan algoritma ise sabit 2-bit kodlama yöntemi olmuştur. Bu durum doğrudan algoritmaların kapasitesiyle ilişkilidir. Yüksek kapasiteli algoritmalar daha az kelime değiştirerek veri gömebildikleri için, genel anlamda daha az semantik bozulma meydana gelmektedir.
* Ancak daha doğru bir analiz için kelime başına semantik bozulma da göz önünde bulundurulmalıdır. Az sayıda kelime değiştiriliyor olsa bile, bu kelimeler bağlamdan çok uzaksa, özellikle insanlar tarafından metnin içinde gizli bir mesaj olduğu fark edilebilir.
* Kelime başına anlamsal kayıp incelendiğinde, en iyi sonucu Huffman tabanlı algoritma, en kötü sonucu ise karakter tabanlı yöntem vermiştir.

### TARTIŞMA VE SONUÇ

Bu tez çalışmasında, BERT tabanlı Maskeli Dil Modelleme yapısından yararlanılarak üç farklı metin steganografi algoritması geliştirilmiştir: karakter tabanlı, sabit bit genişlikli ve Huffman kodlamalı yöntem. Geliştirilen bu yöntemler, klasik tekniklerin aksine bağlama duyarlı, dil modeli destekli ve daha doğal görünümlü stego metinler üretmeyi amaçlamaktadır.

Yöntemler, kapasite ve semantik bozulma açısından değerlendirilmiştir. Karakter tabanlı yöntem teorik olarak en yüksek kapasiteyi sunarken, kelime başına semantik kayıp en yüksek olan yöntem de olmuştur. Huffman kodlamalı yöntem ise anlamsal bütünlüğü en iyi koruyan yaklaşım olarak öne çıkmaktadır. Sabit bit genişlikli yöntem, düşük kapasiteye rağmen kararlı performansıyla dengeli bir çözüm sunmuştur.

Deneysel sonuçlar, önerilen yöntemlerin başarıyla çalıştığını ve farklı senaryolara uygun çözümler sunduğunu göstermektedir. Özellikle Huffman tabanlı yaklaşım, düşük semantik bozulma ile bilgi gömmenin mümkün olduğunu ortaya koymuştur. Bununla birlikte karakter tabanlı yöntemin daha yüksek kapasiteli uygulamalarda tercih edilebileceği değerlendirilmektedir.

Gelecekteki çalışmalar, Türkçe gibi farklı dillerde bu yöntemlerin test edilmesi, BERT yerine daha yeni dil modelleri ile performans karşılaştırması yapılması, semantik doğruluğu artırmaya yönelik bağlam optimizasyon tekniklerinin entegre edilmesi ve daha kapsamlı ön işleme işlemlerinin uygulanması yönünde ilerleyebilir.

### KAYNAKLAR

[1] M. Nithya and V. Kavitha, “A comprehensive survey on text steganography: Approaches, challenges, and future directions,” *Journal of Theoretical and Applied Information Technology*,

vol. 100, no. 5, pp. 1212–1226, 2022. [Çevrimiçi]. Erişim adresi: <https://www.jatit.org/volumes/Vol100No5/18Vol100No5.pdf>

[2] A. R. Pathak and A. K. Yadav, “Text steganography: A novel approach,” in *Proc. Int. Conf. Communication, Computing & Security (ICCCS '11)*, Rourkela, India, 2011, pp. 458–463. [Çevrimiçi]. Erişim adresi: <https://dl.acm.org/doi/10.1145/1947940.1948046>

[3] H. Murdoch, “A survey of steganographic techniques for text documents,” *arXiv preprint arXiv:1302.2718*, 2013. [Çevrimiçi]. Erişim adresi: <https://arxiv.org/pdf/1302.2718>

[4] A. M. Ali ve A. S. Saad, “New text steganography technique by using mixed-case font,” *Int. J. Comput. Appl.*, cilt. 62, sayı. 3, ss. 6–10, 2013. [Çevrimiçi]. Erişim adresi: <https://research.ijcaonline.org/volume62/number3/pxc3884650.pdf>

[5] H. H. Moraldo, “An approach for text steganography based on Markov chains,” *41 JAIIO - WSegI 2012*, ss. 21–35, 2014. [Çevrimiçi]. Erişim adresi: <https://arxiv.org/abs/1409.0915>

[6] Z. Yang, S. Jin, Y. Huang, Y. Zhang ve H. Li, “Automatically generate steganographic text based on Markov model and Huffman coding,” *arXiv preprint arXiv:1811.04720*, 2018. [Çevrimiçi]. Erişim adresi: <https://arxiv.org/abs/1811.04720>

[7] M. Y. Elmahi, T. M. Wahby ve M. H. Sayed, “Text steganography using compression and random number generators,” *Int. J. Comput. Appl. Technol. Res.*, cilt. 6, sayı. 6, ss. 259–263, 2017. [Çevrimiçi]. Erişim adresi:

<https://www.researchgate.net/publication/317587216_Text_Steganography_Using_Compression_and_Random_Number_Generators>

[8] C.-Y. Chang ve S. Clark, “Linguistic steganography using automatically generated paraphrases,” *Proc. Human Lang. Technol.: 2010 Annu. Conf. North Amer. Chapter Assoc. Comput. Linguistics (NAACL-HLT 2010)*, pp. 591–599, 2010. [Çevrimiçi]. Erişim adresi: <https://aclanthology.org/N10-1084.pdf>

[9] F. Z. Mansor, A. Ismail, R. Din, A. Mustapha ve N. A. Samsudin, “Substitution-based linguistic steganography based on antonyms,” *Indonesian Journal of Electrical Engineering and Computer Science*, cilt. 16, sayı. 1, ss. 530–538, 2019. [Çevrimiçi]. Erişim adresi: <https://www.researchgate.net/publication/339806276_Substitution-based_linguistic_steganography_based_on_antonyms>

[10] Z. M. Ziegler, Y. Deng ve A. M. Rush, “Neural linguistic steganography,” *Proc. 2019 Conf. Empirical Methods Natural Lang. Process. (EMNLP 2019)*, ss. 3157–3167, 2019. [Çevrimiçi]. Erişim adresi: <https://arxiv.org/abs/1909.01496>

[11] X. Zheng ve H. Wu, “Autoregressive linguistic steganography based on BERT and consistency coding,” *Security and Communication Networks*, 2022. [Çevrimiçi]. Erişim adresi: <https://arxiv.org/abs/2203.13972>

[12] E. Öztürk, A. Ş. Mesut ve Ö. A. Fidan, “A character-based steganography using masked language modeling,” *IEEE Access*, cilt. 11, ss. 1–1, 2024. [Çevrimiçi]. Erişim adresi: <https://www.researchgate.net/publication/377430731_A_character_based_steganography_using_masked_language_modeling>

### EKLER

Projenin Google Colab linki:

<https://colab.research.google.com/drive/1m9xiKv9KIBKosa2LnSMy7hUJgyNx_gVh?usp=sharing>