

Text-to-Speech (TTS) Teknolojisi: Gelişimi, Uygulamaları ve Geleceği

Dr. Öğr. Üyesi Funda AKAR
Bayram GÜNGÖR

İçindekiler

1. Giriş.....	1
2. Terimler ve Kısaltmalar.....	2
3. Text To Speech Teknolojisinin Tarihsel Gelişimi.....	3
3.1. Mekanik Dönemler	4
3.2. Elektronik Dönemi	8
3.3. Modern Dönem	15
4. Text To Speech Teknolojisinin Temel İşleyişi.....	16
4. 1. Metin Analizi	18
4.1.1. Hazırlayıcı (Pre-Processor)	18
4.1.2. Morfolojik Analiz Modülü (Morphological Analyzer)	21
4.1.3. Bağlamsal Analizör (Contextual Analyzer)	22
4.1.4. Söz Dizimsel – Prozodik Çözümleyici (Syntactic – Prosodic Parser)	23
4.2. Otomatik Fonetikleştirme.....	23
4.2.1. Giriş Metninin Hazırlanması:.....	23
4.2.2. Grafemleri Seslere Dönüştürme:	24
4.2.3. Ses Birimlerini Birleştirme	24
4.2.4. Dil Bilgisi ve Dil Bilgisel Kuralların Uygulanması.....	25
4.2.5. Çıktı Metninin Oluşturulması	25
4.3. Prozodi Üretici (Prosody Generator)	25
4.4. Ses Sentezi	26
4.4.1. Metin Birimleri Seslendirme	26
4.4.2. Dil ve Aksan Uyumu	26
4.4.3. Doğal Ses Modülasyonu	26
4.4.4. Ses Akışı ve Akıcılık.....	27
4.4.5. Sesli Özellikler ve Hız	27
4.4.6. Ses Kalitesi ve Yükseklik	27
4.5. Ses Çıkışı Oluşturma.....	27
5. Sentezleme Yöntemleri	27
5.1. Artikülasyon Sentezleme (Articulatory Synthesis)	28
5.2. Formant Sentezleme (Formant Synthesis)	29
5.3. Eklemeli Sentezleme (Concatenative Synthesis)	29

5.3.1. Birim Seçme Sentezleme (Unit Selection Synthesis)	30
5.3.2. İkili Ses Sentezleme (Diphone Synthesis).....	30
5.3.3. Uygulamaya Özgü Sentezleme (Domain-Specific Synthesis).....	30
5.3.4. Hece Tabanlı Sentezleme (Syllable-Based Synthesis).....	31
5.3.5. Dalga Formu Birleştirme Tekniği (Waveform Concatenation Technique).....	31
5.4. İstatistiksel Parametrik Sentezleme (Statistical Parametric Synthesis)	31
5.4.1. Hidden Markov Model Tabanlı Sentezleme (HMM)	32
5.4.2. Derin Sinir Ağı Tabanlı Sentezleme (DNN)	32
6. Text To Speech Teknolojisinin Uygulamaları	32
6.1. Eğitim Materyalleri	32
6.2. Sesli Asistanlar	33
6.3. Sesli Kitaplar.....	33
7. Text To Speech Teknolojisinin Geleceği	34
8. Text To Speech Uygulaması Geliştirme	35
8.1. Windows Form Uygulaması ve Speech Kütüphanesi ile Masaüstü Uygulaması Geliştirme	36
8.1.1. Microsoft Ses Paketleri	38
8.1.2. Seslendirmen Seçimi ve Özelliklerini Belirleme	39
8.1.3. Ses Parametrelerinin Ayarlanması	41
8.1.4. Sesi Duraklatma, Sesi Devam Ettirme ve Sesi Kaydetme Butonlarının Eklenmesi .	43
8.1.5. Harici Ses Paketlerinin Eklenmesi	44
8.1.6. Projenin Kodları.....	45
8.2. ASP.NET Core 6.0 MVC ve Azure Speech Studio ile Web Uygulaması Geliştirme	51
8.2.1. Microsoft Azure ile Konuşma Kaynağı Oluşturma	52
8.2.2. ASP.NET Core MVC Projesi Oluşturma.....	54
8.2.3. NuGet Paketinin Eklenmesi ve Metin Okuma Sistemine Giriş.....	57
8.2.4. Durdurma Butonunun ve Dil-Aksan Seçeneklerinin Eklenmesi	60
8.2.5. Desteklenen Tüm Ses Seçeneklerini Getirme	62
8.2.6. Dil Seçimine Göre Konuşmacıların Listelenmesi ve Tarayıcı Dilinin Otomatik Seçilmesi.....	66
8.2.7. Ses Dosyalarının MP3 ve WAV Formatlarında İndirilmesi	69
8.2.8. Adımların Değerlendirilmesi ve Projenin Kodlarının Tamamı.....	74
9. Referans.....	80

ŞEKİLLER LİSTESİ

Şekil 1. Kratzenstein'in sesli harf rezonatörleri	4
Şekil 2. "Satranç Oynayan Türk" Otomatının temsili görüntüsü.....	5
Şekil 3. "Satranç Oynayan Türk" Otomatının gizli bölmesinin temsili görüntüsü	6
Şekil 4. Kempelen'in ses sentezi makinesi.....	7
Şekil 5. Charles Wheatstone'ın konuşan makinesi.....	8
Şekil 6. VODER Kullanım Şeması	9
Şekil 7. Franklin Cooper'ın Desen Oynatma Makinesini Diyagramı.....	10
Şekil 8. Paralel Formant Sentezleyici Şeması.....	11
Şekil 9. DAVO Sentezleyicisinin Fotoğrafı	12
Şekil 10. Kurzweil'in Okuma Makinesi	13
Şekil 11. TTS teknolojisindeki önemli gelişmelerin tarihsel şeması.	15
Şekil 12. TTS Modern Dönem Gelişmeleri	16
Şekil 13. Örnek TTS Sentezleyici Modül Diyagramı	16
Şekil 14. Örnek Doğal Dil İşleme Modülü Diyagramı	17
Şekil 15. Sentezleme Yöntemleri Şeması	28
Şekil 16. Konuşma sırasında kullanılan organlar ve Organların modellenmesi	28
Şekil 17. Eklemeli Sentezleme Yöntemleri Şeması	30

1. Giriş

Günümüzde sesli iletişim teknolojileri hayatımızın hemen hemen her alanında giderek daha fazla önem kazanıyor. Akıllı cihazlar, dijital asistanlar ve daha birçok platform, metin tabanlı bilgileri sesli bir şekilde sunabilmek için Text to Speech (TTS - Metinden Konuşma Üretme) teknolojisine dayanıyor. TTS metin tabanlı bilgileri otomatik olarak sesli hale getiren bir dönüşüm sürecini ifade eder. Bu teknoloji, bilgisayarların yazılı metni insan benzeri seslere dönüştürmesini sağlar. TTS sadece metinleri duyulabilir hale getirmekle kalmaz, aynı zamanda metni anlamakta zorlanabilecek kişilere yardımcı olur. Özellikle görme engelliler, metin tabanlı içeriklere sesli bir şekilde erişerek bilgilere ulaşabilirler. Bu teknoloji ayrıca metin tabanlı içerikleri daha erişilebilir ve anlaşılabilir hale getirerek geniş bir uygulama yelpazesine sahiptir. Günümüzde TTS teknolojisindeki Türkçe kaynakların sınırlı olmasından dolayı çalışmamız bu teknolojinin tanıtılması için büyük önem taşımaktadır. Bu teknoloji öğrenmek ve kullanmak isteyen kişilere de ilham kaynağı olacaktır.

TTS dil sınırlarını aşmamıza ve kültürel çeşitliliği vurgulamamıza olanak tanır. Eğitim, erişilebilirlik ve dijital içerik oluşturma alanlarında TTS, bilgiye erişimi ve iletişimi dönüştürmek için güçlü bir araç olarak kabul edilir. Ayrıca, iş dünyası ve endüstriler için de büyük bir potansiyel sunar; sesli reklamlar, sesli kitaplar ve interaktif sanal asistanlar gibi alanlar, işletmelerin ve girişimlerin müşterileriyle daha yakın bir iletişim kurmalarını sağlar. Teknolojinin hızla evrildiği bir çağda, iletişim ve erişilebilirlik alanında ses teknolojileri, önemli bir değişimi tetikliyor. Bu değişimin önemli bir parçası olan TTS, metinden sese dönüşüm konusundaki ilerlemeleri ve uygulamalarıyla insanlar için yeni olanaklar sunuyor. Bu çalışma, TTS teknolojisinin geçmişi, çalışma prensipleri, sentezleme yöntemleri ve gelecekteki potansiyeli detaylı olarak ele alınacaktır.

Bu kitap, Text to Speech (TTS) teknolojisinin geniş bir perspektiften incelendiği kapsamlı bir kaynaktır. Kitap, TTS'nin tarihsel gelişimini, başlangıcından günümüze kadar olan evrimini ve farklı dönemlerdeki önemli aşamalarını detaylı bir şekilde ele almaktadır. Mekanik dönemlerden elektronik süreçlere ve modern yaklaşımlara kadar, TTS'nin nasıl geliştiği ve hangi aşamalardan geçtiği ayrıntılı olarak incelenmektedir. Ayrıca, kitap TTS teknolojisinin temel işleyişini açıklamakta ve metin analizinden ses sentezine kadar olan adımları adım adım açıklamaktadır. Bu adımlar, TTS'nin nasıl çalıştığını ve metin tabanlı bilgilerin nasıl seslendirildiğini anlamak için önemlidir. Metin analizi, otomatik fonetikleştirme, prozodi üretimi ve ses sentezi gibi aşamalar detaylı bir şekilde ele alınmaktadır. Sentezleme yöntemleri de kitabın önemli bir bölümünü oluşturmaktadır. Artikülasyon sentezlemesinden istatistiksel parametrik sentezlemeye kadar farklı sentezleme teknikleri incelenmektedir. Her bir yöntemin nasıl çalıştığı ve hangi durumlarda kullanılabileceği açıklanmaktadır. Kitap ayrıca, TTS teknolojisinin çeşitli uygulama alanlarını da ele almaktadır. Eğitim materyallerinden sesli asistanlara ve sesli kitaplara kadar geniş bir yelpazede TTS'nin kullanımı detaylı bir şekilde incelenmektedir. Ayrıca, TTS teknolojisinin geleceği ve potansiyeli hakkında da bir değerlendirme yapılmaktadır. Son olarak kitap Text to Speech uygulaması geliştirmeye yönelik pratik bilgiler içermektedir. Windows Form uygulamalarından ASP.NET Core MVC ve Azure Speech Studio ile web uygulamaları ile farklı platformlarda nasıl TTS uygulamaları geliştirilebileceği adım adım açıklanmaktadır.

Bu kitap, Text to Speech teknolojisinin geniş bir perspektiften ele alındığı, detaylı bir kaynak olup, okuyuculara TTS dünyasının kapılarını aralamayı amaçlamaktadır.

2. Terimler ve Kısaltmalar

Formant : Formantlar insan sesinin karakteristik özelliklerini oluşturan ses bileşenleridir. İnsanlar konuşurken, seslerin belirli frekansta yoğunlaştığı noktalara formant denir. Formantlar, sesin nasıl duyulduğunu ve anlaşıldığını etkiler. Örneğin, farklı vokal sesleri farklı formant yapılarına sahiptir. Bir vokal sesin yüksekliği ve şekli, formantlar tarafından belirlenir. Aynı şekilde, ünsüz seslerin farklılıkları da formantlar aracılığıyla oluşturulur. Formantlar, insan sesinin anlaşılabilirliğini sağlayan önemli bileşenlerdir.

Vokal Ses : Ses üretirken ses yolunun tıkanmadığı ve ses tellerinin titreşimine izin verdiği bir ses türüdür. Vokal sesler genellikle ünlü sesler olarak bilinir ve konuşma veya müziğin temel öğelerinden biridir. İnsan sesini oluşturan vokal sesler, ses tellerinin titreşimi ve ağız boşluğunun şeklinin değiştirilmesi ile oluşturulmaktadır.

Rezonans : Bir sistemin veya nesnenin titreşimlerin veya dalgaların etkisi altında özel bir frekansta tepki verme durumunu ifade eder. Yani, bir sistemin veya nesnenin doğal frekansında titreşimlere veya dalgaların etkisine karşılık verme yeteneğidir. Örneğin, bir müzik enstrümanının tellerinin belirli bir frekansta titreşmesi veya bir ses sisteminin odanın akustik özelliklerine uyum sağlaması gibi durumlar rezonansı ifade edebilir.

Rezanotör : Bir nesnenin veya sistemin titreşimlerini veya ses dalgalarını belirli bir şekilde etkileyen bir yapı veya bileşen olarak düşünülebilir. Yani, bir rezonatör, belirli bir frekans aralığında titreşimleri destekleyen veya belirli bir frekans aralığında ses dalgalarını güçlendiren bir yapı veya cihazdır. Örneğin, bir müzik enstrümanında sesi şekillendiren rezonans odası veya bir hoparlördeki ses odası birer rezonatördür.

Rezonans, Rezanotör Farkı: Rezonans, bir sistemin doğal frekansında titreşimlere veya dalgaların etkisine tepki verme yeteneğini ifade ederken, rezonatör bu titreşimleri veya dalgaları belirli bir şekilde etkileyen yapı veya cihazdır.

Fonotik : Konuşma seslerini inceleyen bir dil bilimi dalıdır. Fonotik dalı, konuşma seslerinin fiziksel özelliklerini (ses titreşimleri, ağız ve boğaz anatomisi gibi) ve nasıl üretildiğini inceler. Ayrıca, farklı seslerin nasıl duyulduğunu ve insanlar arasında nasıl farklılaştığını araştırır.

Akustik : Sesin üretimi, yayılması ve algılanmasıyla ilgilenen bilim dalıdır. Ayrıca sesin fiziksel özellikleri, titreşimleri, yankıları ve ses dalgalarının yayılma şekli gibi konuları da içerir. Akustik sesin nasıl oluştuğunu, nasıl taşındığını ve insanlar veya diğer canlılar tarafından nasıl algılandığını incelemekle ilgilenir.

Amplitüd : Elektrik ve manyetizma gibi alanlarda, amplitüd genellikle bir sinyalin maksimum veya minimum değerini ifade eder. Örneğin, bir elektrik sinyalinin amplitüdü, sinyalin en yüksek voltaj veya akım değeri olabilir. Amplitüd, bir dalganın genellikle daha büyük bir enerji veya güçle ilişkilendirildiği bir ölçüttür.

Spektrogram : Spektrogram, bir sinyalin frekans bileşenlerini ve zaman içindeki değişimini görsel olarak temsil eden bir grafik veya görüntüdür. Genellikle zaman frekans düzleminde bir ısı haritası şeklinde sunulur. Yatay eksen genellikle zamanı temsil ederken, dikey eksen genellikle frekansı temsil eder. Spektrogramlar ses işleme, konuşma tanıma, müzik analizi, radyo haberleşme ve diğer alanlarda sıklıkla kullanılır. Örneğin, bir ses dosyasının spektrogramu, sesin zaman içindeki frekans bileşenlerini görselleştirerek sesin özelliklerini analiz etmek için kullanılabilir.

Corpus : Belirli bir dil veya konuşma tarzı için toplanmış büyük ve sistemli bir metin koleksiyonunu ifade eder. Dil biliminde, bir dilin yapısını, kullanımını ve işleyişini anlamak için metinlerin toplanması ve analiz edilmesi önemlidir. Bu metin koleksiyonları, çeşitli kaynaklardan derlenmiş olabilir ve dil bilimsel çalışmalar, dil öğretimi, metin analizi ve bilgisayarla dil işleme gibi birçok alanda kullanılabilir. Örneğin, bir dilin gramer yapısını incelemek için büyük bir yazılı corpus kullanılabilir veya konuşma sentezi gibi yapay dil işleme uygulamaları için sesli bir corpus kullanılabilir.

Tablo 1. Kısaltmaların Açıklaması

Kısaltma	Açılımı
TTS	Text To Speech (Yazıdan Konuşmaya Çevirme)
STT	Speech To Text (Konuşmadan Yazıya Çevirme)
PAT	Parametric Artificial Talker (Parametrik Yapay Konuşmacı)
DAVO	Dynamic Analog of the VOcal tract (Vokal Kanalı Dinamik Analogu)
HMM	Hidden Markov Model (Gizli Markov Model)
DNN	Deep Neural Network (Derin Sinir Ağı)
SPSS	Statistical Parametric Speech Synthesis (İstatistiksel Parametrik Konuşma Sentezleme)
RNN	Recurrent Neural Network (Tekrarlayan Sinir Ağı)
LSTM	Long Short-Term Memory (Uzun Kısa Süreli Bellek)
NLP	Naturel Language Processing (Doğal Dil İşleme)
DSP	Digital Signal Processing (Dijital Sinyal İşleme)
LPC	Linear Predictive Coding (Lineer Tahmin Kodlaması)
JSRU	Joint Speech Research Unit (Ortak Konuşma Araştırma Birimi)
WCT	Waveform Concatenation Technique (Dalga Formu Birleştirme Tekniği)

3. Text To Speech Teknolojisinin Tarihsel Gelişimi

Günümüzde neredeyse ayırt edilemeyecek seviyede yapay konuşma üretebilen TTS sistemlerinin tarihi 18. yüzyıla dayanmaktadır. Mekanik cihazlarla başlayan evrim süreci günümüzde yapay zekâ kullanılarak devam etmektedir. Bu süreç, farklı dönemlerdeki teknolojik ilerlemeler, yazılım gelişmeleri ve kullanım alanlarının genişlemesiyle şekillenerek oluşmuştur. TTS teknolojisini bu üç döneme ayrılarak dönem içerisindeki gelişmeler anlatılmıştır. Zaman çizelgesi aşağıdaki gibidir:

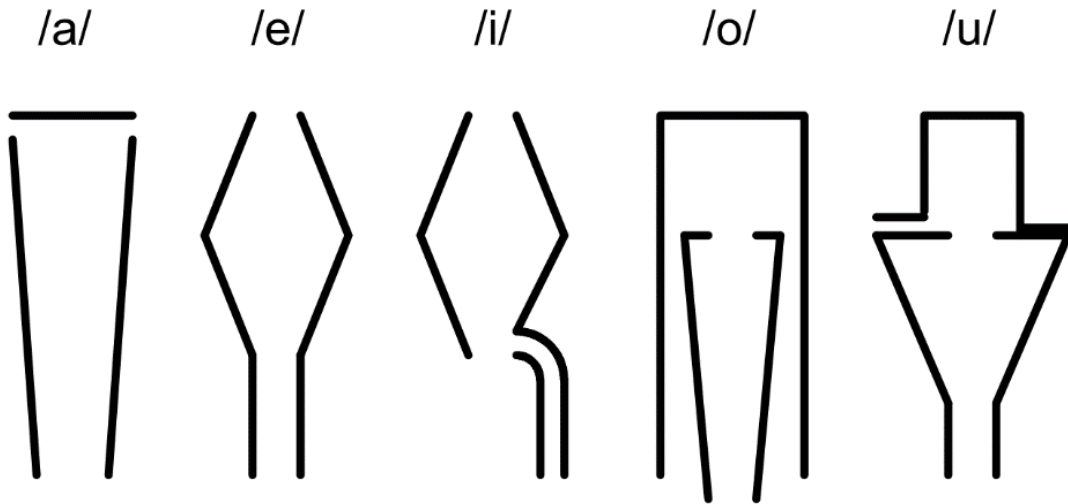
Mekanik Dönem: Yaklaşık olarak 1770'lerden 1939'a kadar olan dönem. Bu dönem, mekanik cihazların kullanıldığı ve ilk basit konuşma sentezleyicilerin geliştirildiği zaman aralığını kapsar.

Elektronik Dönem: Yaklaşık olarak 1940'lardan 1980'lere kadar olan dönem. Bu dönem, elektronik cihazların yaygın olarak kullanılmaya başlandığı ve daha karmaşık ses sentezleme tekniklerinin geliştirildiği zaman aralığıdır.

Modern Dönem: Yaklaşık olarak 1980'lerden günümüze kadar olan dönem. Bu dönem, dijital teknolojinin gelişimiyle birlikte ses sentezleme tekniklerinin daha da geliştiği ve daha doğal ve akıcı konuşma sentezleyicilerinin ortaya çıktığı zaman aralığını kapsar.

3.1. Mekanik Dönemler

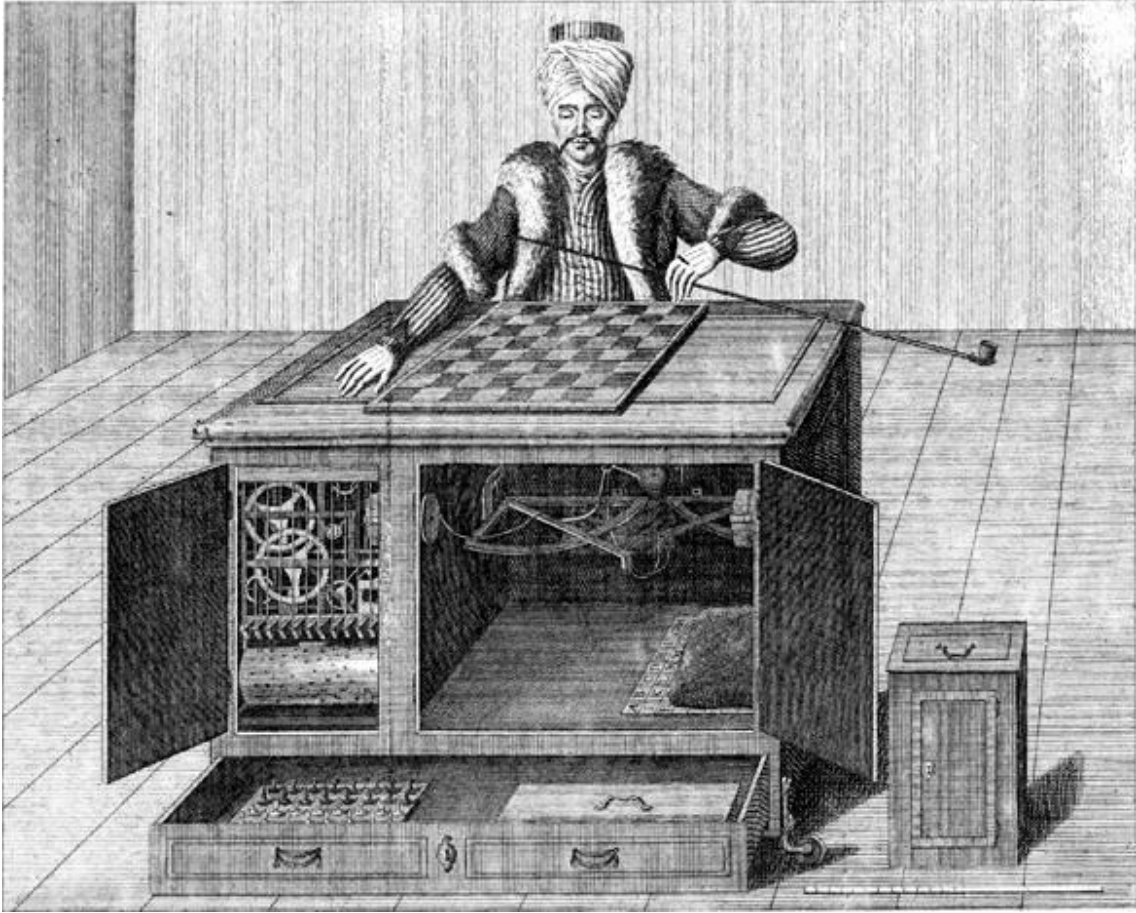
TTS teknolojisinin ilk adımlarının atıldığı mekanik dönemde, konuşma becerisi yaratıcının sağladığı bir yetenek mi yoksa insanlar tarafından geliştirilen bir beceri mi olduğu konusunda tartışmalar sürmekteydi. Bu tartışmalar, insanın doğal olarak mükemmel bir şekilde iletişim kurabilme yeteneğine mi dayandığını yoksa bu yeteneğin bilimsel ve teknolojik gelişmelerle taklit edilebileceğini mi vurguluyordu. İnsan sesinin karmaşıklığı ve çeşitliliği, onu taklit etmeye çalışan mekanik sistemler için büyük bir zorluk oluşturmuyordu. St. Petersburg Bilimler Akademisi, 1780 yılında sesli harflerin doğası, farkları ve seslerin yapay olarak nasıl üretilebileceği konusunda ödüllü bir bilimsel yarışma düzenlemeye karar verir. Bu yarışma, insan sesinin yapısını ve işleyişini anlamak için önemli bir fırsat sunacaktır. Profesör Christian Gottlieb Kratzenstein, bu yarışmaya çalışmalarını sunarak katıldı. Kratzenstein, İngiliz alfabesindeki beş ünlü harfin ('a', 'e', 'i', 'o', 'u') fizyolojik farklarını inceledi ve nefesli enstrümanlar gibi sesleri insan seslerine benzeten mekanik bir makine geliştirdi. Kratzenstein'in sesli harf rezonatör modelleri Şekil 1'de gösterilmiştir. Kratzenstein'in sesli harf rezonatörleri, dönemin bilim dünyasında büyük ilgi uyandırdı ve yapay ses üretimi konusundaki araştırmalara yeni bir ivme kazandırdı. Kratzenstein'in çalışmaları, ses ve konuşma teknolojisinin gelişiminde önemli bir dönüm noktası olarak kabul edilir. İnsan sesinin mekanik olarak taklit edilebileceğinin kanıtı olarak görülen bu çalışmalar, sonraki yıllarda TTS teknolojisinin gelişimine büyük katkı sağladı. Kratzenstein'in sesli harf rezonatörleri, günümüzde bile ses sentezi ve konuşma sentezi teknolojilerinin temelini oluşturan önemli bir adım olarak hatırlanır.



Şekil 1. Kratzenstein'in sesli harf rezonatörleri

Kratzenstein'in buluşundan hemen sonra, 1791'de Viyana'da Wolfgang von Kempelen, "Akustik-Mekanik Konuşma Makinesi" adlı icadını tanıttı. Bu, insan sesinin doğal taklidini gerçekleştirmeye yönelik önemli bir adımdı. Kempelen'in çalışmaları, o dönemde ses sentezi

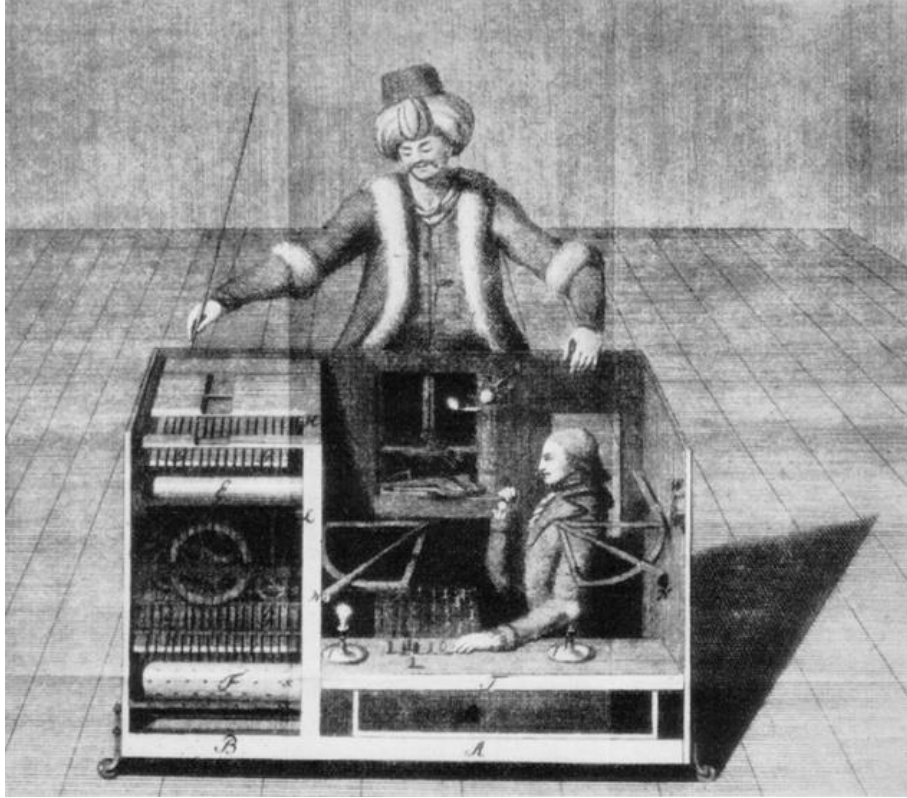
teknolojisinin evriminde kritik bir rol oynadı. Makine, insan sesinin farklı tonlarını ve ritimlerini üretebilmek için çeşitli mekanik parçaları kullanıyordu. Özellikle, Kempelen'in dil ve dudak modellerini eklemesi, makinenin insan anatomisine daha uygun hale gelmesini sağladı. Deri tüpünün şeklini manipüle ederek ünlü seslerin doğru bir şekilde taklidini gerçekleştirebilirken, ünsüzler için farklı daralmış geçitler ve hareketli dudaklar kullanarak daha gerçekçi bir konuşma üretimi sağladı. Şekil 4'te Kempelen'in konuşma makinesinin çizimi bulunmaktadır. Makinenin temel parçaları arasında, ses üretimini sağlamak için basınç odası, titreşen bir kamış ve ses yolunun eylemini taklit etmek için bir deri tüp bulunmaktaydı. Bu parçaların bir araya gelmesiyle, insan sesinin çeşitli tonları, vurguları ve ritimleri başarıyla taklit edilebiliyordu. Kempelen'in dikkatli tasarımı ve mekanik becerisi, o dönemdeki teknolojik sınırların ötesinde bir başarıyı mümkün kıldı. Ancak, Kempelen'in 1783'te sergilediği "Satranç Oynayan Türk" adlı otomat, kötü şöhreti nedeniyle konuşma makinesinin gereken ilgiyi görmemesine yol açtı. Dolayısıyla, Kempelen'in konuşma makinesiyle yaptığı önemli katkılar, zamanında hak ettiği değeri göremedi. Ancak, günümüzde ses ve konuşma sentezi teknolojisinin gelişiminde kritik bir role sahip olduğu kabul edilmektedir.



Şekil 2. "Satranç Oynayan Türk" Otomatının temsili görüntüsü

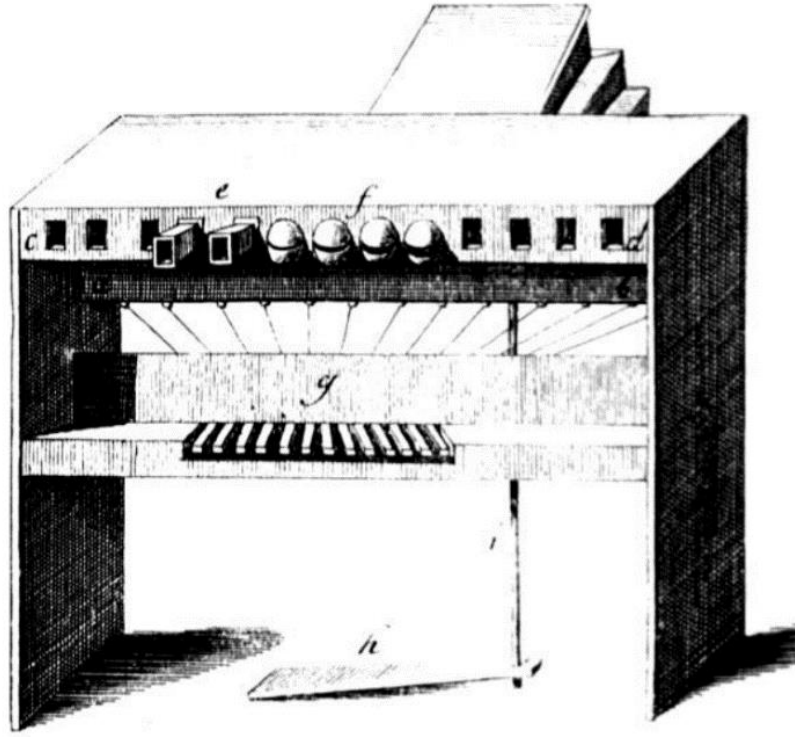
Kempelen'in "Satranç Oynayan Türk" adını verdiği otomatı, 6 ayda geliştirerek 1770'te ilk kez İmparatoriçe Maria Theresa için sergilenmiştir. Bu otomat, sadece satranç oynamakla kalmamış, aynı zamanda izleyicilere mekanik zekanın karmaşıklığını da göstermiştir. Üzerine satranç tahtası çizilen küçük, tekerlekli bir kabin bulunmaktadır. Kabinin yanında oturan bıyıklı, sarıklı ve pelerinli mekanik parçalardan oluşan Türk figürü oturmaktadır. Şekil 2'de Satranç Oynayan Türk otomatının temsili görüntüsü bulunmaktadır. Gösterici, gösteriye

başlamadan önce makinenin ve figürün iç bölmelerindeki makara, kaldıraç gibi karmaşık mekanik malzemeleri göstererek seyircilerde akıllı cihaz izlenimi oluştuyordu. Figür, rakibiyle satranç oynarken satranç tahtasını incelemekte, ara sıra başını sallayarak taşların yerini değiştirmekte ve maç bitiminde seyircilerden gelen soruları tepsideki harfleri birleştirerek yanıtlayabilmekteydi. Bu etkileyici performansıyla, otomat birçok ünlü ismin dikkatini çekmiştir. Napolyon Bonaparte, Benjamin Franklin gibi önemli isimlerle de satranç oynamıştır. Napolyon'un dahil olduğu 52 adet oyunun detayları "The Turk, Chess Automaton" (Gerald Levitt) isimli kitapta bulunmaktadır. Bu otomat, sadece satranç oynamakla kalmayıp aynı zamanda mekanik zekanın sınırlarını zorlamış ve insanlığın teknolojiye olan ilgisini artırmıştır.



Şekil 3. "Satranç Oynayan Türk" Otomatının gizli bölmesinin temsili görüntüsü

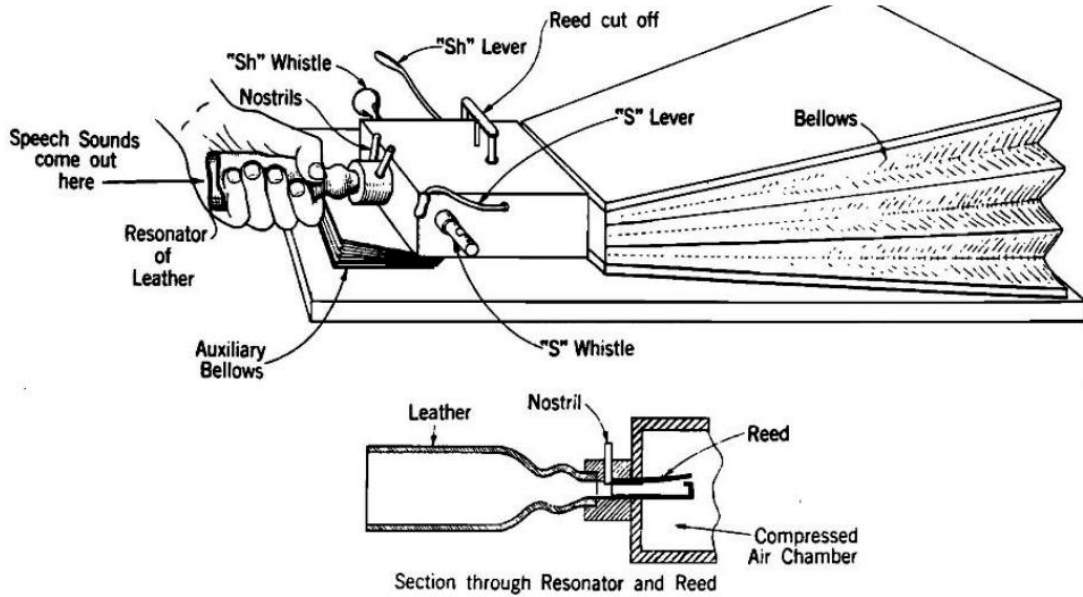
“Satranç Oynayan Türk” otomatı döneminde büyük ilgi uyandırarak pek çok yazıya konu olsa da aslında içerisine bir insanın saklandığı büyük bir aldatmacaydı. Bu aldatmaca, o dönemin teknoloji ve mühendislik anlayışına büyük bir meydan okuma olmuştur. Şekil 3’te Satranç Oynayan Türk otomatının gizli bölmesinin temsili görüntüsü bulunmaktadır. Mekanizmanın içerisindeki bacakları olmayan satranç ustası, mıknatıslar ve mum yardımıyla oyunu takip edebilmiş ayrıca mekanik kollar sayesinde de figürü yönetmiştir. Bu detaylar, otomatın arkasındaki insan kontrolünü gizlemek için ustaca tasarlanmıştır. Kempelen, her ne kadar bu cihazın akıllı olduğu iddia etmese de çalışma prensibini hakkında hiçbir yorum yapmamasından dolayı kötü şöhretle anılmıştır. Bu sebeple, “Akustik-Mekanik Konuşma Makinesi” adındaki gerçek ve önemli olan icadı yeterince ilgi görmemiştir. Ancak, otomatın ardındaki insan kontrolünün keşfi, o dönemin teknolojik sınırlarını aşma arzusunu ve insan zekasının yaratıcılığını yansıtan önemli bir dönüm noktası olmuştur. Bu olay, ilerleyen yıllarda mekanik ve elektronik alanlarda yapılan çalışmaları etkilemiş ve insan-makine etkileşimi konusunda derinlemesine düşünmemizi sağlamıştır.



Şekil 4. Kempelen'in ses sentezi makinesi

1837 yılında, teknolojinin hızla geliştiği bir dönemde, Charles Wheatstone isimli bilim insanı, tarihte önemli bir dönüm noktasını temsil eden bir adım attı. O dönemde, iletişim teknolojilerinde büyük bir ilerleme yaşıyordu ve bu ilerleme, insanlık tarihinde sesin önemini ve kullanımını yeniden tanımlayacak nitelikteydi. Wheatstone, bu dönemin ruhunu yakalayıp, von Kempelen'in konuşan makinesinin daha gelişmiş bir versiyonunu üretti. Bu makine, insan sesinin doğal tonlarını ve ritimlerini taklit edebilecek özelliklere sahipti. Ünlü seslerin üretilmesi için titreşen kamışlar kullanılarak, her harfin doğru şekilde çıkarılması hedefleniyordu. Bununla birlikte, makinenin karmaşıklığı, sadece ünlü sesleri değil, aynı zamanda çoğu ünsüz sesini de üretebilecek yetenekte olduğunu gösteriyordu. Şekil 5'te Wheatstone'un konuşma makinesinin çizimi bulunmaktadır. Özellikle, ses birleşimlerinin ve hatta tam kelimelerin üretilmesi, bu makinenin ses sentezi teknolojisindeki çığır açıcı niteliğini vurguluyordu. Wheatstone'un konuşma makinesi, o dönemin teknolojik sınırlarını zorlayarak, insan sesinin doğal çeşitliliğini ve zenginliğini yansıtmaya yönelik büyük bir adımdı. Bu icat, sadece mekanik bir cihaz değil, aynı zamanda insan sesinin inceliklerini anlama ve taklit etme çabasının bir ifadesiydi. Wheatstone'un geliştirdiği konuşma makinesinin sergilendiği bir etkinlik sonrasında genç Alexander Graham Bell, babasının da rehberliği ve teşvikiyle kendi konuşma cihazını oluşturmaya karar verdi. 1800'lerin sonlarına doğru, Alexander Graham Bell ve babası, Charles Wheatstone'un konuşma makinesinden ilham alarak benzer bir konuşma makinesi yapmaya karar verdiler. Bell, o dönemde ses teknolojilerindeki gelişmeleri takip ediyordu ve Wheatstone'un çalışmaları onun üzerinde derin bir etki bırakmıştı. Bu nedenle, babasıyla birlikte, insan sesinin üretimini anlamak ve taklit etmek için bir makine geliştirmeye karar verdiler. Ancak, Bell'in ilginç deneyleri ve metodları da bu sürece katkı sağladı. Bell, terrier köpeğini kullanarak ses sentezi üzerine deneyler yapmıştı. Köpeğini bacaklarının arasına alarak hırlatmasını sağladı ve ardından elleriyle ses yolunu değiştirerek köpeğin hırlamasını konuşma benzeri seslere dönüştürmeye çalıştı. Bu deneyler, Bell'in ses teknolojileri ve insan

sesinin doğasını anlama konusundaki merakını ve kararlılığını gösteriyor ve sonunda telefonun icadına kadar uzanan bir yolculuğun başlangıcını oluştuyordu.



Şekil 5. Charles Wheatstone'ın konuşan makinesi

1838 yılında Willis tarafından yapılan çalışmalar, ses biliminde önemli bir dönüm noktası oluşturdu. Willis, belirli bir ünlü ses ile ses yolunun geometrisi arasındaki ilişkiyi keşfetti. Bu keşif, insan sesinin üretimini anlama ve kontrol etme konusundaki anlayışımızı derinleştirdi. Çalışmaları sırasında, Willis farklı ünlü sesleri, org borularına benzer tüp rezonatörler kullanarak sentezledi. Bu yöntem, ünlü seslerin nasıl üretildiğini ve nasıl değiştirilebileceğini anlamak için önemli bir araç sağladı. Ayrıca Willis, ünlü kalitesinin sadece tüpün uzunluğuna değil aynı zamanda çapına da bağlı olmadığını keşfetti. Bu keşif, ses sentezi ve konuşma teknolojilerinin gelişiminde büyük bir adım olarak kabul edildi ve insan sesinin doğasını anlama konusunda önemli bir katkı sağladı.

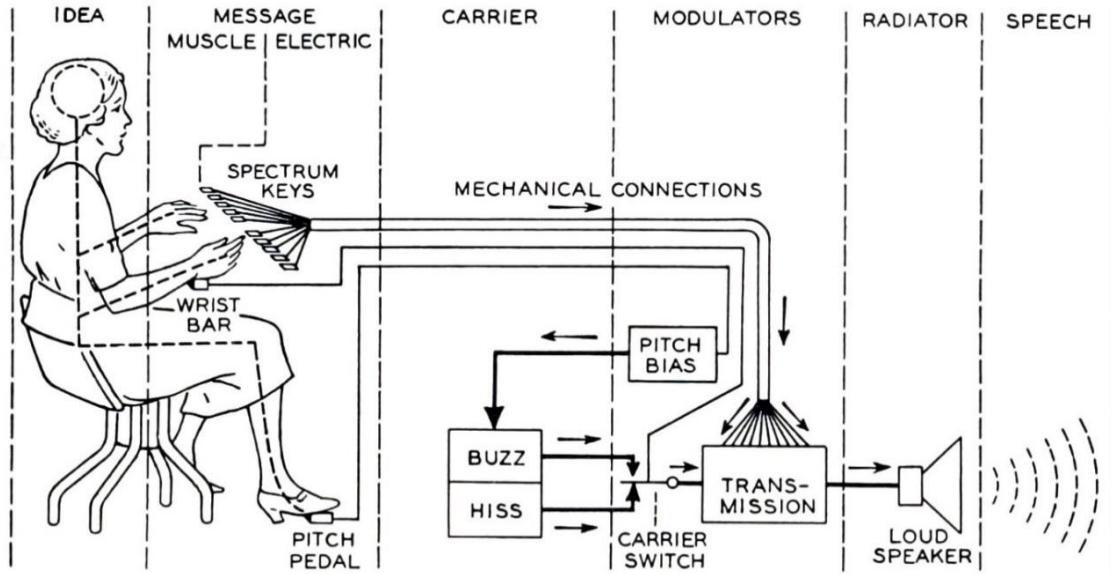
1960'lara kadar, insan sesinin mekanik ve yarı-elektriksel analoglarla araştırılması ve deneyleri, belirli bir başarı elde edilmeden devam etti. Önde gelen bilim insanlarından Herman von Helmholtz ve Charles Wheatstone gibi isimler, ses sistemini anlamak ve taklit etmek için çeşitli mekanik ve yarı-elektriksel cihazlar geliştirdiler. Helmholtz'un 19. yüzyılda yaptığı deneyler, insan sesinin fizyolojik özelliklerini anlamak için önemli bir adım olarak kabul edildi. Özellikle, Helmholtz rezonatörleri, insan sesinin frekans ve rezonans özelliklerini incelemek için kullanılan önemli bir araçtı. Ancak, bu dönemde yapılan deneylerin çoğu, insan sesini tam olarak taklit etmede yetersiz kaldı ve belirgin bir başarı elde edilemedi.

3.2. Elektronik Dönemi

1922 yılında Stewart tarafından tanıtılan ilk tam elektrik sentez cihazı, ses teknolojilerinde önemli bir ilerleme sağladı. Bu sentezleyici, insan sesinin akustik rezonanslarını modellemek için bir zil gibi bir uyarıcı ve iki rezonans devresi kullanıyordu. Ancak, bu cihaz yalnızca iki en düşük formantla sınırlı kalarak tek başına statik ünlü sesleri üretebiliyordu. Ünsüz sesler veya bağlı cümlelerin üretilmesi mümkün değildi. Benzer bir sentezleyici, Flanagan'ın çalışmalarına göre Wagner tarafından geliştirildi. Bu cihazda, dört elektriksel rezonatör paralel bağlanmıştı ve bir arı vızıltısı gibi bir kaynak tarafından uyarılmaktaydı. Dört rezonatörün çıkışları, uygun

amplitüdlerde birleştirilerek ünlü spektrumları oluşturuluyordu. Bununla birlikte, 1932'de Japon araştırmacılar Obata ve Teshima, ünlülerde üçüncü bir formantın varlığını keşfettiler. Bu keşif, sentetik konuşmanın daha doğal ve anlaşılır hale getirilmesinde önemli bir adım oldu. Genellikle ilk üç formantın, insanlar arasında anlaşılabilir sentetik konuşma için yeterli olduğu kabul edilir. Bu gelişmeler, ses sentezi teknolojilerinin ilerlemesine ve insan sesinin doğal yapısının daha iyi anlaşılmasına katkı sağladı.

1939'da New York Dünya Fuarı'nda Bell Laboratuvarı tarafından tanıtılan VODER, TTS teknolojisinin önemli bir gelişmesini temsil etti. Bu cihaz, VOCODER adlı başka bir cihazdan esinlenerek tasarlanmıştı ve elektronik bir konuşma sentezi sağlamak için kullanılıyordu. VODER, sesin frekansını ayarlamak için bir ayak pedalı ve harf, kelime ve kısa ifadeleri oluşturabilen tuşlara sahip bir piyano gibi görünüyordu. Şekil 6'da VODER Kullanım Şeması verilmiştir. Cihazı kullanabilmek için eğitilmiş bir uzmana ihtiyaç duyulmaktaydı. Bu cihazın sergisi, elektronik bir cihaz ile konuşma üretmenin mümkün olduğunu gösterdi ve konuşma sentezi konusundaki ilgiyi artırdı. Homer Dudley tarafından 1939'da New York Dünya Fuarı'nda tanıtılan VODER, ilk konuşma sentezleyicisi olarak kabul edilir. VOCODER, Bell Laboratuvarları'nda 1930'ların ortalarında geliştirilmişti ve konuşmayı akustik parametrelere analiz eden bir cihazdı. Orijinal VOCODER, konuşma sinyalini yaklaşık olarak yeniden oluşturmak için bir sentezleyiciyi yönlendirebiliyordu. VODER'ın ses kalitesi ve anlaşılabilirliği düşük olsa da yapay konuşma üretme potansiyelini göstermesi açısından önemliydi.

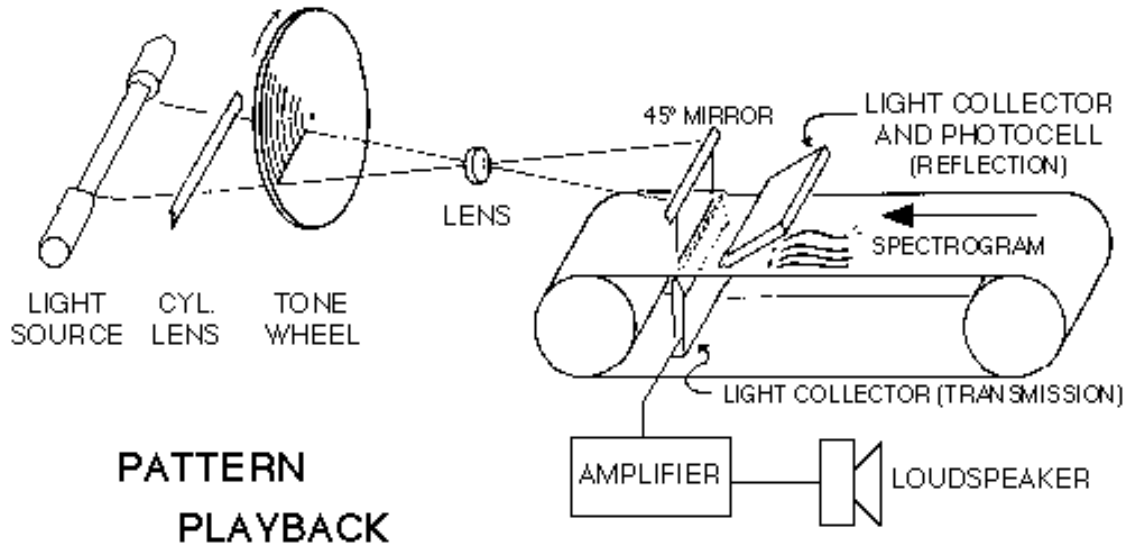


Şekil 6. VODER Kullanım Şeması

VODER'ın sergilenmesinin ardından, bilimsel toplumda konuşma sentezi konusuna olan ilgi arttı. Bu önemli gelişme, insan sesinin yapay olarak üretilebileceğini ve anlaşılabilir konuşmanın elde edilebileceğini gösterdi. VODER'ın temel yapısı ve fikri, günümüzde kullanılan konuşma sentezi sistemlerinin temelini oluşturan kaynak-filtre modeline dayanmaktadır. Bu model, konuşma üretim sürecini insan vokal sistemini taklit ederek gerçekleştirmeye çalışır. VODER, ses üretimi için bir kaynak sinyali ve bu sinyalin geçtiği bir filtre sistemini kullanarak konuşma üretiyordu. VODER'ın gösterdiği gibi, yapay konuşma

üretimi gerçekten mümkündür ve bu, iletişim teknolojileri ve yapay zekâ alanında büyük potansiyellere sahip bir alandır.

1951 yılında, Haskins Laboratuvarları'nda Franklin Cooper ve iş arkadaşları tarafından geliştirilen Desen Geri Çalma (Pattern Playback) sentezleyici, ses sentezi teknolojisinde önemli bir adımdı. Bu sentezleyici, kaydedilmiş spektrogram desenlerini seslere dönüştürebilen bir mekanizmaya sahipti. Şekil 7'de Franklin Cooper'ın desen oynatma makinesinin diyagramı gösterilmektedir. Spektrogram desenleri, optik olarak şeffaf bir bant üzerine kaydedilmişti ve bu desenler, ses sinyallerini temsil ediyordu. Bu yöntem, sesin spektral yapısını temsil etmek için kullanılıyordu ve kaydedilen desenler, ardından sentezleyici tarafından yeniden oluşturuluyordu. Sentetik olarak üretilen sesler ya orijinal kaydedilen formda ya da değiştirilmiş formda çalınabiliyordu. Bu gelişme, konuşma sentezi teknolojisinin ilerlemesine büyük katkı sağladı ve yapay ses üretimi konusundaki araştırmaları hızlandırdı. Franklin Cooper ve ekibinin çalışması, ses sentezi teknolojisinin daha da rafine edilmesine ve ileriye taşınmasına yol açtı. Bu tür sentezleyiciler, iletişim teknolojilerinde ve dil araştırmalarında önemli bir rol oynamaktadır, çünkü farklı dil yapılarını ve seslerini incelemek için kullanılabilirler.

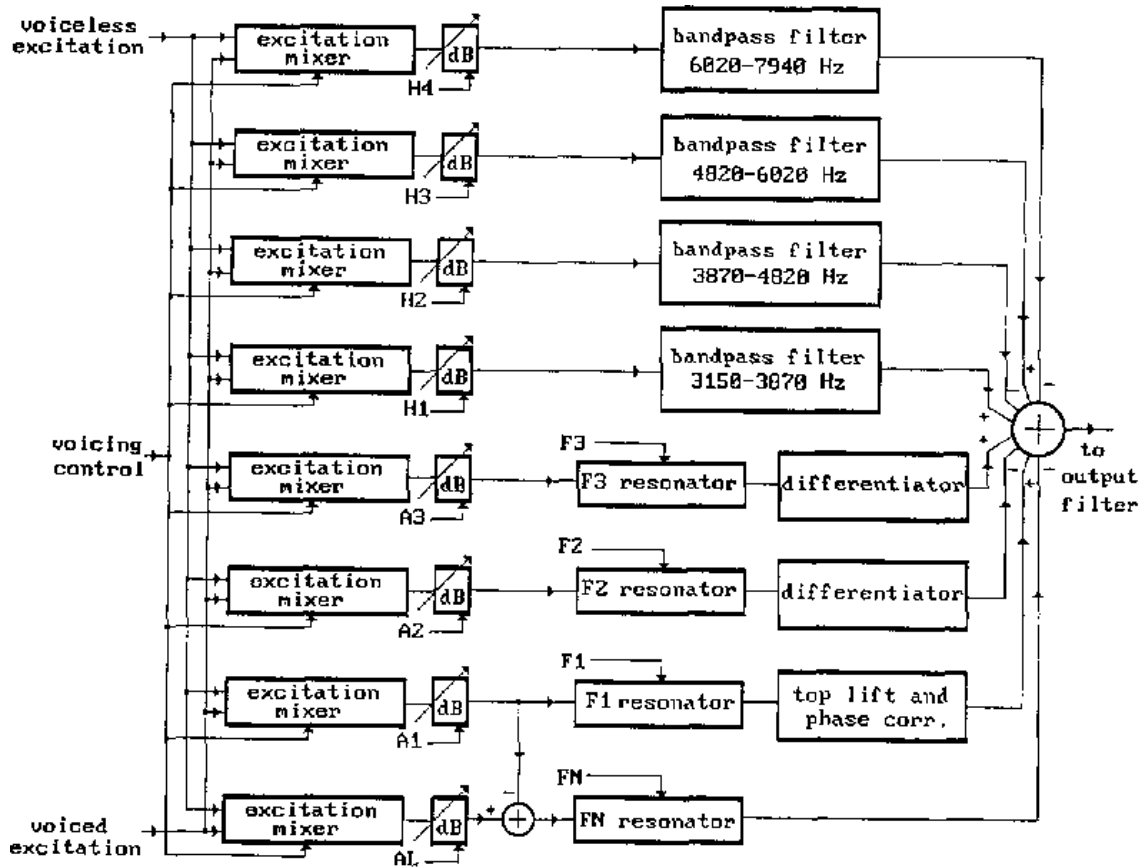


Şekil 7. Franklin Cooper'ın Desen Oynatma Makinesini Diyagramı

1953 yılında, Walter Lawrence tarafından geliştirilen "Parametrik Yapay Konuşmacı" (Parametric Artificial Talker - PAT) adlı cihaz, ses sentezi alanında önemli bir adım olarak kabul edildi. Bu cihaz, elektronik devreler kullanarak sesi yapay olarak üretmek için tasarlandı. PAT, üç elektronik formant rezonatöründen oluşuyordu ve genellikle bir vızıltı veya gürültü gibi bir giriş sinyali ile çalışıyordu. Lawrence'ın bu yeniliği, sesin belirli özelliklerini doğrudan kontrol edebilme yeteneği ile dikkat çekti. Sesin temel frekansı, şiddeti, gürültü seviyesi ve üç formant frekansı gibi özellikler, PAT'ın hareketli bir cam slaytı tarafından kontrol edilen altı zaman işleviyle yönetiliyordu. Aynı dönemde, Gunnar Fant, kendi geliştirdiği "OVE I" adını verdiği kademeli formant sentezleyicisi ile ses sentezi alanında önemli bir rol oynadı. Fant, daha sonra sesli, sessiz ve engelleyici ünsüzler için ayrı parçalardan oluşan gelişmiş bir "OVE II" sentezleyici tanıttı. Bu cihazlar, sesin belirli özelliklerini doğrudan kontrol ederek, formant sentezlemesi olarak bilinen yapay konuşma yöntemini başlattılar. Bu teknolojik ilerlemeler, ses

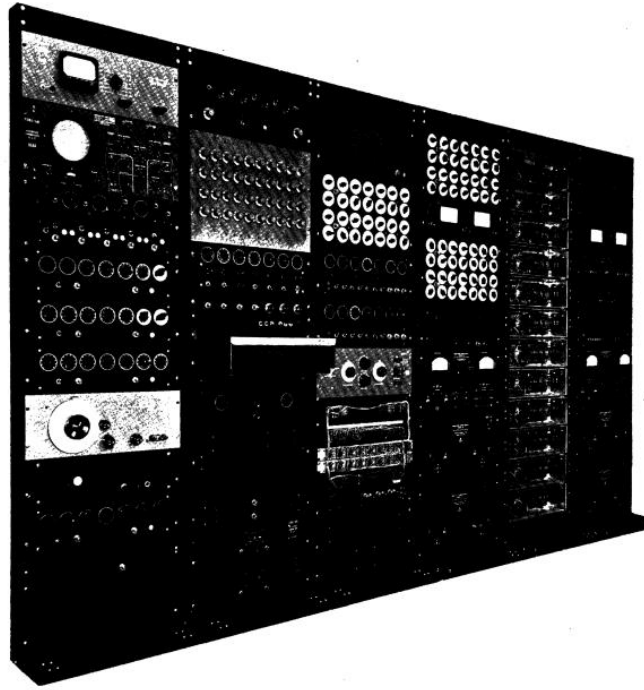
sentezi ve iletişim teknolojilerinde devrim niteliğinde değişikliklere yol açarak, insan-makine etkileşiminde önemli bir dönüm noktası oluşturdu.

PAT ve OVE sentezleyicileri, akustik tüpün aktarım fonksiyonunun paralel mi yoksa kademeli mi modellenmesi gerektiği konusunda bir tartışma başlattı. Bu tartışma, yapay konuşma sentezi alanında önemli bir sorunu ele alıyordu: Konuşma seslerinin oluşturulması için akustik tüpün matematiksel modellenmesi nasıl olmalıydı? Bir yandan, paralel formant sentezleyicileri, her formant frekansının bağımsız olarak işlenmesini sağlar, böylece daha esnek bir kontrol imkânı sunar. Öte yandan, kademeli formant sentezleyicileri, formant frekanslarının birbirini izleyen bir şekilde işlenmesini sağlar, böylece daha doğal bir ses üretimi sağlar. John Holmes, bu tartışmayı inceledikten sonra, 1972'de paralel formant sentezleyicisini tanıttı. Bu sentezleyici, her bir formant frekansının bağımsız olarak işlenmesini sağlayarak, daha detaylı ve esnek bir ses sentezi sağladı. Holmes, bu yeni sentezleyiciyi test etmek için "Basit bir yaşamı seviyorum" cümlesini sentezledi ve sonuçlar inanılmazdı: El ile ayarlanan sentez, doğal konuşmayı o kadar iyi taklit etmişti ki, ortalama bir dinleyici arasındaki farkı ayırt etmekte zorlandı. Yaklaşık bir yıl sonra, Holmes, JSRU (Joint Speech Research Unit) ile iş birliği yaparak paralel formant sentezleyicisini daha da geliştirdi. Şekil 8'de paralel formant sentezleyicinin şeması verilmiştir. Bu iş birliği, sentezleyicinin performansını ve doğruluğunu artırdı ve daha geniş bir ses yelpazesini kapsayacak şekilde tasarlandı. Bu yeni gelişmeler, yapay konuşma sentezi teknolojisinde önemli bir ilerleme sağladı ve sentezlenmiş seslerin daha gerçekçi ve doğal olmasını sağladı.



Şekil 8. Paralel Formant Sentezleyici Şeması

1958 yılında Massachusetts Teknoloji Enstitüsü'nde (M.I.T.) George Rosen tarafından geliştirilen ilk artikülatör sentezleyici, yapay konuşma teknolojisinin önemli bir dönüm noktası olarak kabul edilir. Bu sentezleyici, Vokal kanalın dinamik analoğu (DAVO) olarak adlandırılan bir cihazdı ve el ile oluşturulan kontrol sinyallerinin kaset kaydı yoluyla kontrol edilmesini sağlıyordu. DAVO, insan konuşmasını taklit etmek için akustik tüpün dinamik hareketlerini modellemek üzere tasarlanmıştı. Şekil 9'da DAVO sentezleyicisinin fotoğrafı bulunmaktadır. Ancak, o dönemdeki teknolojik kısıtlamalar nedeniyle, sentezlenen seslerin kalitesi ve doğruluğu sınırlıydı. 1960'ların ortalarında, Lineer Tahmin Kodlaması (LPC) ile ilk deneyler yapılmaya başlandı. Bu teknik, konuşma sinyallerini analiz etmek ve sentezlemek için kullanılan önemli bir yöntemdir. Ancak, o zamanlar LPC'nin kullanılabilirliği ve performansı, günümüz sistemlerine kıyasla oldukça düşüktü. Örneğin, 1980'de TI Speak'n'Spell gibi düşük maliyetli sistemlerde ilk kez kullanıldığında, lineer tahminin ses kalitesi oldukça zayıftı. Ancak, zamanla yapılan bazı temel model değişiklikleri ve geliştirmelerle, LPC yöntemi oldukça yararlı hale geldi ve günümüzde birçok yapay konuşma sistemine entegre edildi. Bu değişiklikler, ses sentezleme sürecini daha doğru ve etkili hale getirdi ve kullanıcılar arasında daha gerçekçi ve anlaşılır bir yapay konuşma deneyimi sağladı. Bu gelişmeler, yapay konuşma teknolojisinin evriminde önemli bir rol oynamıştır, çünkü daha gelişmiş ve verimli sistemlerin geliştirilmesine olanak tanımıştır.



Şekil 9. DAVO Sentezleyicisinin Fotoğrafi

İngilizce için ilk tam metin okuma-ses sistemini Japonya Elektroteknik Laboratuvarı'nda 1968'de Noriko Umeda ve arkadaşları geliştirdi. Bu sistem, bir artikülasyon modeline dayanıyordu ve karmaşık sezgisel kurallara sahip bir sentaks analiz modülünü içeriyordu. Konuşma oldukça anlaşılır olsa da monoton ve günümüz sistemlerinin kalitesinden oldukça uzaktı. Bu gelişme, yapay konuşma teknolojisinin ilerlemesinde önemli bir adımdı ve metin tabanlı iletişimi desteklemek için oldukça değerliydi. Ancak, o dönemdeki teknolojik sınırlamalar nedeniyle, sesin doğallığı ve çeşitliliği açısından hala iyileştirilmesi gereken birçok alan vardı. Umeda ve ekibinin çalışmaları, yapay konuşma sistemlerinin karmaşıklığını ve

performansını artırmak için temel bir zemin oluřturdu. Sentaks analizinin entegrasyonu, kullanıcıların metin tabanlı bilgiye daha etkili bir řekilde erişmesine ve anlamasına olanak tanıdı. Ancak, sesin monotonluğu ve kalite eksikliği, kullanıcıların tamamen doğal bir konuşma deneyimi yaşamalarını engelledi. Bu tür erken sistemler, günümüzdeki gelişmiş TTS sistemlerinin temelini oluřturdu ve yapay zekâ, derin öğrenme ve ses sentezi teknolojisindeki ilerlemelerle birlikte, bugün daha doğal, akıcı ve duygusal olarak zengin konuşma sentezi sağlayan sistemlerin geliştirilmesine yol açtı.

1976'da düzenlenen bir fuar, görme engelliler için önemli bir dönüm noktası oldu. Kurzweil'in okuma makinesi, bu fuarda tanıtıldı ve görme engelli bireyler için büyük bir yenilik olarak kabul edildi. Bu makine, o dönemde mevcut olan yaygın yazı tiplerini ve puntoları, fotokopi makinelerinde olduğu gibi optik tarayıcıyla tarayarak seslendiriyordu. Şekil 10'da Kurzweil'in okuma makinesinin görüntüsü bulunmaktadır. Kurzweil'in okuma makinesi, sadece bir tarayıcı olarak değil, aynı zamanda metinleri sesli olarak okuma yeteneğiyle de öne çıkıyordu. Makinenin özelliği, yaklaşık olarak dakikada 150 kelime hızında metinleri seslendirebilmesiydi. Bu işlemi gerçekleştirirken, 1000'den fazla fonetik kuralı kullanarak metinleri doğru bir řekilde çözümleyebiliyordu. Bu, kullanıcıya yazılı materyalleri dinleme yoluyla erişme imkânı sağlayarak, görme engelli bireyler için büyük bir kolaylık sağlıyordu. Ancak, bu teknolojik yenilik getiren cihazın fiyatı, orta halli müşteriler için oldukça yüksekti. O yıllardaki fiyatı bile 30.000 doların üzerindeydi. Bu durum, cihazın geniş çapta yaygınlaşmasını engelledi ve daha çok kütüphaneler ve görme engelli bireylere hizmet veren merkezlerde tercih edilmesine neden oldu. Bununla birlikte, Kurzweil'in okuma makineleri, görme engelli bireyler için yazılı materyallere erişimi kolaylaştırmak adına önemli bir adım olarak değerlendirildi. Bu cihazlar, görme engelli bireylerin yaşamlarını daha bağımsız hale getirmelerine ve bilgiye daha rahat bir řekilde erişmelerine yardımcı oldu.



Şekil 10. Kurzweil'in Okuma Makinesi

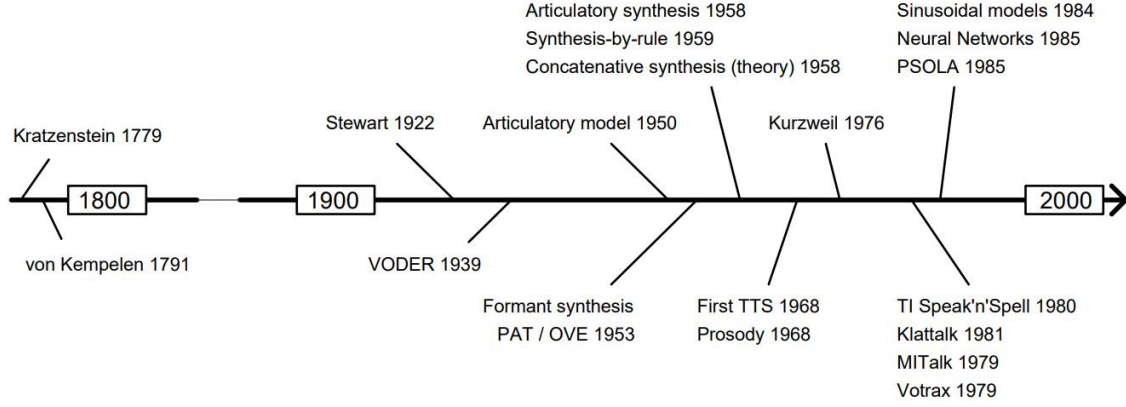
1979'da Allen, Hunnicutt ve Klatt, M.I.T'de geliştirilen MITalk laboratuvar metin-okuma-ses sisteminin bir gösterimini gerçekleştirdi. Bu gösteri, metin tabanlı bilgiye erişimi desteklemenin ve kullanıcıların bilgisayar aracılığıyla doğal bir şekilde konuşma üretmesini sağlamanın önemli bir adımıydı. MITalk sistemi, metin girişini sesli bir çıktıya dönüştürmek için karmaşık algoritmalar ve sentezleme teknikleri kullanıyordu. Özellikle, ses sentezi sürecindeki doğallığı ve anlaşılabilirliği artırmak için bir dizi dilbilgisel ve akustik kuralı içeriyordu. Bu teknolojik gelişme, ticari kullanıma yönelik olarak Telesensory Systems Inc. (TSI) tarafından yapılan bazı değişikliklerle birlikte uygulandı. TSI'nin ticari TTS sistemi, MITalk'ın başarılı sonuçlarından ilham alarak, görsel olarak engelli bireylerin metin tabanlı içeriğe erişimini kolaylaştırmak için kullanılmıştır. Bu şekilde, metin tabanlı bilgiye erişimde engelleri aşmak için geliştirilen bu teknoloji, geniş bir kullanıcı kitlesine ulaşma potansiyeline sahipti. İki yıl sonra, 1981'de Dennis Klatt, ünlü Klattalk sistemini tanıttı. Klattalk, önceki MITalk sisteminde kullanılan temel prensipleri kullanarak, yeni ve daha sofistike bir ses üretim kaynağına dayanıyordu. Bu ses üretim kaynağı, sesin daha doğal ve insan benzeri olmasını sağlamak için geliştirilmiş akustik modeller ve algoritmalar içeriyordu. Klattalk sistemi, ses sentezi teknolojisindeki bir sonraki evrim aşamasını temsil ediyordu ve birçok modern sentezleme sisteminin gelişiminde önemli bir rol oynadı. MITalk ve Klattalk sistemlerinin başarısı, daha sonra birçok farklı sentezleme sisteminin geliştirilmesinde ilham kaynağı oldu. Örneğin, DECtalk ve Prose-2000 gibi ticari sentezleme sistemleri, MITalk ve Klattalk'ın temel algoritmalarını ve tekniklerini kullanarak, daha gelişmiş ses sentezi özellikleri sunuyordu.

1970'lerin sonu ve 1980'lerin başları, konuşma sentezi ve metin-okuma ürünlerinde önemli bir gelişme dönemi idi. Bu dönemde, bir dizi ticari ürün piyasaya sürüldü ve bu ürünler ses sentezi teknolojisinin evriminde büyük bir rol oynadı. Örneğin, konuşma sentezi için kullanılan ilk entegre devre olan Votrax çipi, bu dönemin önemli bir buluşuydu. Bu çip, kademeli formant sentezleyici ve basit düşük geçiren düzleştirme devrelerinden oluşuyordu. Bu teknoloji, ses sentezi alanında önemli bir adımdı ve daha sonraki sentezleme sistemlerinin temelini oluşturdu.

1978'de Richard Gagnon'un uygun fiyatlı Votrax tabanlı "Type-n-Talk" sistemi piyasaya sürüldü. Bu sistem, ses sentezi teknolojisinin daha geniş kitlelere ulaşmasını sağladı ve özellikle sesli iletişim araçlarını geliştirmek isteyenler için büyük bir avantaj sağladı. Aynı dönemde, Texas Instruments, düşük maliyetli doğrusal tahmin kodlaması (LPC) temelli "Speak-n-Spell" sentezleyiciyi tanıttı. Bu ürün, çocuklar için eğitici bir oyuncak olarak piyasaya sürüldü ve hızla popülerlik kazandı.

1982'de Street Electronics, Echo adlı düşük maliyetli bir diphone sentezleyici tanıttı. Bu cihaz, bir önceki Speak-n-Spell'den gelen teknolojiye dayanıyordu ve metin-okuma sistemlerinin daha erişilebilir hale gelmesine yardımcı oldu. Aynı dönemde, Speech Plus Inc., Prose-2000 metin-okuma sistemiyle piyasaya çıktı. Bu ürün, kullanıcıların yazılı metinleri sesli olarak duyabilmesini sağlayarak okuma güçlüğü çekenler için büyük bir kolaylık sağladı.

Bu ürünlerin piyasaya sürülmesi, ses sentezi teknolojisinin yaygınlaşmasına ve gelişmesine önemli katkılarda bulundu. Ticari ürünlerin kullanıma sunulmasıyla birlikte, ses sentezi ve metin-okuma teknolojileri daha geniş bir kitleye ulaşarak, insanların günlük yaşamlarında daha fazla kullanım alanı buldu. Şekil 11'de Modern döneme kadarki TTS teknolojisinde yaşanan önemli gelişmelerin tarihsel şeması gösterilmiştir.



Şekil 11. TTS teknolojisindeki önemli gelişmelerin tarihsel şeması.

3.3. Modern Dönem

Erken dönem bilgisayar tabanlı konuşma sentezi yöntemleri arasında, artikülasyon sentezi, formant sentezi ve birleştirici sentez bulunmaktadır. Artikülasyon sentezi, insan konuşma organlarının (dudaklar, dil, ses telleri vb.) hareketlerini modelleyerek konuşma üretir. Bu yöntem, fiziksel olarak konuşmanın üretilmesini simüle eder ve ses üretim sürecini matematiksel olarak açıklar.

Formant sentezi, ses yolunun fiziksel ve akustik özelliklerini modelleyerek konuşma üretir. Kaynak-filtre modeline dayanarak, bir ses kaynağı (örneğin, titreşimli ses telleri) bir filtreden (örneğin, ses yolunun şekli) geçer. Bu yöntem, belirli sesleri üretmek için formant frekanslarını manipüle eder. Birleştirici sentez, önceden kaydedilmiş ses birimlerinin (fonemler, di-fonemler, kelimeler, cümleler vb.) birleştirilmesiyle yapay konuşma üretir. Bu yöntem, geniş bir ses veri tabanına ihtiyaç duyar ve doğal konuşma üretmek için ses birimlerinin akıcı bir şekilde birleştirilmesi gereklidir.

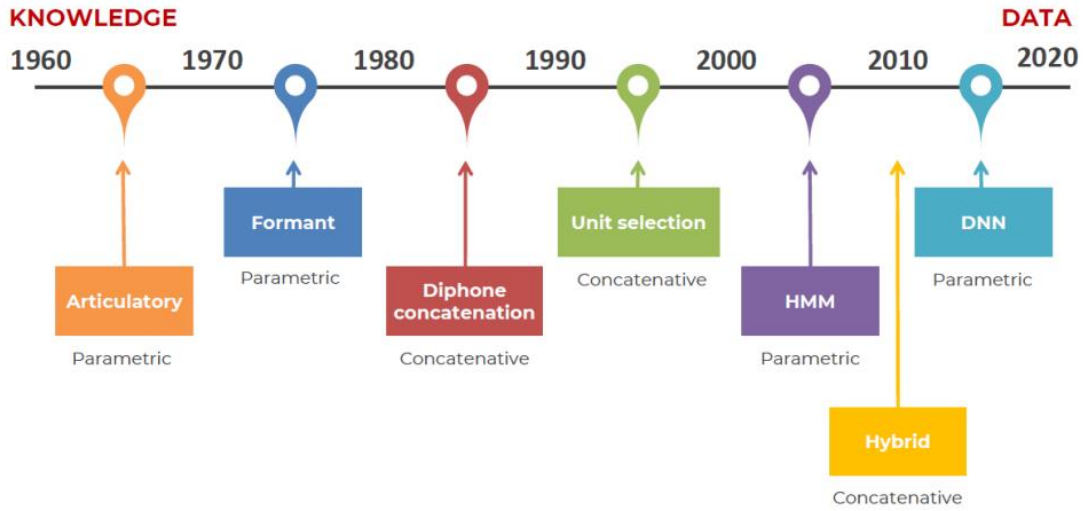
Daha sonra, istatistiksel makine öğrenmesinin gelişimiyle istatistiksel parametrik konuşma sentezi (SPSS) önerilmiştir. SPSS, konuşma sentezi için gerekli olan parametreleri (örneğin, spektrum, temel frekans, süre) tahmin etmek için istatistiksel yöntemler kullanır. Bu yöntem, daha esnek ve adaptif bir yaklaşım sunar ve sentezlenmiş konuşmanın doğallığını artırabilir. 2010'lardan itibaren, sinir ağı tabanlı konuşma sentezi yöntemleri giderek baskın hale gelmiş ve çok daha iyi ses kalitesi elde etmiştir. Derin öğrenme tekniklerinin kullanılması, büyük veri setlerinin işlenmesi ve karmaşık ilişkilerin modellenmesi, sinir ağı tabanlı sentezleme yöntemlerinin başarısını artırmıştır. Bu yöntemler, daha gerçekçi ve doğal sesler üretebilir ve sentezlenmiş konuşmanın kalitesini önemli ölçüde artırabilir.

Modern konuşma sentezi teknolojileri, karmaşık ve sofistike yöntemler ve algoritmalar içerir. Bu teknolojilerin gelişimi, Gizli Markov Modelleri (HMM) ve sinir ağları gibi yeni yaklaşımların kullanımını içerir. Özellikle, HMM'ler 1970'lerin sonlarından itibaren konuşma tanıma alanında başarıyla uygulanmıştır. HMM'ler, bir dizi durum arasında geçişleri modelleyen ve bu geçişlerin olasılıklarını hesaplayan bir matematiksel modeldir. Konuşma tanıma için kullanıldıklarında, bir konuşmanın ardışık seslerini tanımak için kullanılırlar. HMM'ler aynı zamanda konuşma sentezi alanında da kullanılmaktadır. Bir HMM, konuşma sinyalini oluşturmak için gerekli olan parametreleri ve olasılıkları içeren bir yapı sağlar.

Öte yandan, sinir ağları da konuşma sentezi teknolojilerinde giderek daha fazla kullanılmaktadır. Sinir ağları, biyolojik sinir ağlarından esinlenen yapay zekâ algoritmalarıdır.

Bu algoritmalar, büyük miktarda veriyi işleyebilme ve karmaşık ilişkileri öğrenebilme yetenekleriyle dikkat çeker. Konuşma sentezi için kullanıldıklarında, sinir ağları sesleri ve konuşma özelliklerini modellemek ve sentezlemek için kullanılır. Özellikle, derin öğrenme teknikleriyle eğitilen sinir ağları, yüksek kaliteli ve doğal sesler üretebilir.

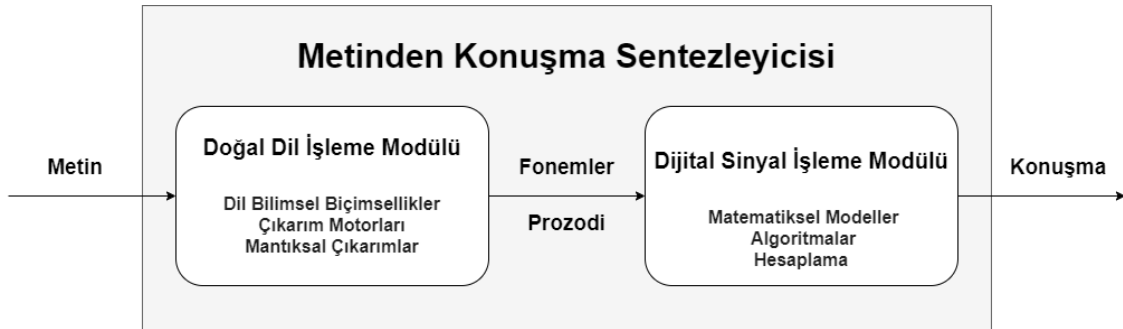
HMM ve sinir ağları, konuşma sentezi teknolojilerindeki başlıca yöntemlerden sadece birkaçıdır. Bu teknolojilerin geliştirilmesi ve iyileştirilmesi, yapay zekâ ve dil işleme alanındaki ilerlemelerle birlikte devam etmektedir. Bu ilerlemeler, daha doğal ve insan benzeri konuşma sentezi sistemlerinin geliştirilmesini sağlarken, kullanım alanlarını da genişletmektedir. Şekil 12’de Modern dönemdeki önemli gelişmelerin tarihsel şeması gösterilmiştir.



Şekil 12. TTS Modern Dönem Gelişmeleri

4. Text To Speech Teknolojisinin Temel İşleyişi

Şekil 13, metinden sese sentezleyicinin (TTS) nasıl çalıştığını gösteren basitleştirilmiş bir diyagramdır. TTS, metni konuşmaya dönüştürmek için iki ana modül kullanır: Doğal Dil İşleme (NLP – Naturel Language Processing) ve Dijital Sinyal İşleme (DSP-Dijital Signal Processing). NLP, metnin anlamını ve tonunu analiz eder ve fonetik transkripsiyonunu oluşturur. DSP ise fonetik transkripsiyonu ses dalgalarına dönüştürür. Geliştiricilerin matematik ve dil bilgisi, bazı işlem adımlarını atlayarak TTS'nin daha hızlı ve daha az bellek kullanarak çalışmasını sağlayabilir. Bu işlem bazen metnin telaffuzunda kısıtlamalara veya sentetik sesin duygusallığının azalmasına neden olabilir, ancak yine de gerçek zamanlı olarak konuşma üretmeye imkân verir.



Şekil 13. Örnek TTS Sentezleyici Modül Diyagramı

Dil Bilimsel Biçimsellikler (Linguistic Formalisms) : Dilin kurallarını ve yapılarını tanımlayan soyut temsillerdir. Gramer kuralları, sözlük ve sentaks gibi öğeleri içerir.

Çıkarım Motorları (Inference Engines) : Dil bilimsel biçimsellikleri ve metni kullanarak fonemleri üretir. Konuşmanın anlamını ve tonunu dikkate alır.

Mantıksal Çıkarımlar (Logical Inferences) : Metnin anlamını ve bağlamını anlamak için kullanılır. Doğru ve tutarlı bir konuşma üretmek için gereklidir.

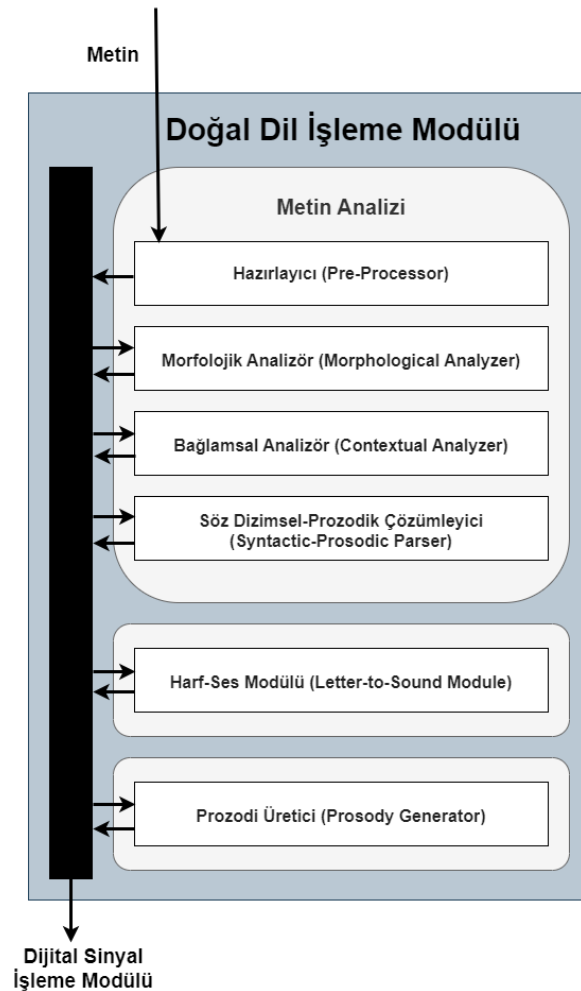
Fonemler (Phonemes) : Konuşmanın temel ses birimleridir. Harfler veya harf kombinasyonları tarafından temsil edilirler.

Prozodi (Prosody) : Konuşmanın ritmini, tonunu ve ses seviyesini kontrol eder. Konuşmayı doğal ve etkileyici hale getirir.

Matematiksel Modeller (Mathematical Models) : Konuşmanın ses dalgalarını oluşturmak için kullanılır. Sesin perdesini, tonunu ve ses seviyesini kontrol eder.

Algoritmalar (Algorithms) : TTS'nin işleyişini tanımlayan bir dizi adımdır. Metni fonemlere ve fonemleri konuşmaya dönüştürmek için kullanılır.

Hesaplama (Computations) : TTS'nin çalışması için gerekli olan işlem gücüdür. Donanım ve yazılım tarafından sağlanır.



Şekil 14. Örnek Doğal Dil İşleme Modülü Diyagramı

Şekil 14’de, TTS amacıyla oluşturulan genel bir doğal dil işleme (NLP) modülünün taslağını sunuyor. Bu modül, harf-ses dönüşümü ve vurgu oluşturma yanı sıra morfolojik ve sözdizimsel analizörü içeriyor. Bu, yüksek kaliteli bir TTS sisteminde sözdizimsel işlemenin önemini vurguluyor. Çünkü doğru fonetik transkripsiyon için kelimelerin sözcük türü ve ardışık kelimeler arasındaki bağlantıların bilinmesi gerekiyor. Ayrıca, doğal vurgu büyük ölçüde sözdizimine dayanır ve bu nedenle TTS sistemleri genellikle sözdizimine odaklanır. Ancak, çok azı tam anlamıyla belirsizlik giderme ve yapılandırma yeteneklerine sahiptir.

4. 1. Metin Analizi

Metin Analizi, TTS teknolojisinin temel işleyişinin ilk aşamasıdır. Bu aşama metni daha küçük ve işlenebilir parçalara böler, metindeki dil bilgisi yapısını anlamayı amaçlar ve özel karakterleri ve sembolleri tanıır. Metin analizin aşamaları aşağıdaki gibidir.

4.1.1. Hazırlayıcı (Pre-Processor)

Girdi olarak alınan doğal dil metnini işlemek ve TTS sisteminin daha etkin ve doğru bir şekilde çalışmasını sağlamak amacıyla kullanılan bir bileşendir. Hazırlayıcı aşamasının temel adımları aşağıdaki gibidir.

A - Metin Temizleme (Text Cleaning)

Metin temizleme, TTS sisteminin daha etkin ve doğru çalışmasını sağlamak için kullanılan önemli bir ön işlem adımındır. Bu adımda, girdi metnindeki gereksiz karakterler, özel işaretler, noktalama işaretleri ve diğer gereksiz bilgiler kaldırılır. Bu işlem, metnin daha basit ve anlaşılabilir bir formda olmasını sağlayarak TTS sisteminin metni daha iyi anlamasına ve daha doğal sesli bir çıktı üretmesine yardımcı olur.

Metin Temizleme Adımında Kaldırılan Bilgiler:

- **Gereksiz Karakterler:** Boşluklar, satır sonları, sekmeler ve diğer kontrol karakterleri metinden kaldırılır.
- **Özel İşaretler:** Parantez, köşeli parantez, ünlem işareti, soru işareti ve diğer özel işaretler metinden kaldırılır.
- **Noktalama İşaretleri:** Nokta, virgöl, iki nokta üst üste ve diğer noktalama işaretleri metinden kaldırılabilir.
- **Diğer Gereksiz Bilgiler:** URL’ler, e-posta adresleri, sayılar ve diğer metin içeriğiyle ilgisi olmayan bilgiler metinden kaldırılabilir.
- Metin Temizleme Adımında Kullanılan Teknikler:
- **Düzenli ifadeler:** Belirli bir kalıba uyan karakterleri veya kelimeleri bulmak ve kaldırmak için kullanılır.
- **Kelime listeleri:** Gereksiz kelimelerin ve kelime öbeklerinin bir listesini oluşturmak ve metinden kaldırmak için kullanılır.
- **Harita tabloları:** Özel karakterleri veya noktalama işaretlerini standart bir formata dönüştürmek için kullanılır.

Metin temizleme, TTS sisteminin performansını ve ses sentezinin kalitesini önemli ölçüde etkileyebilir. Bu adım, metnin daha basit ve net bir hale gelmesini sağlayarak TTS sisteminin metni daha hızlı ve doğru bir şekilde işlemesine yardımcı olur. Ayrıca, metinden gereksiz bilgilerin kaldırılması, ses sentezinin daha doğal ve akıcı olmasını sağlar.

Metin temizleme işlemi, metnin anlamını ve içeriğini değiştirmemelidir. Bazı özel karakterler veya noktalama işaretleri metnin anlamı için önemli olabilir ve bu nedenle korunmalıdır. Metin temizleme işlemi, kullanılan dil ve metnin türüne göre özelleştirilmelidir.

B - Metin Normalleştirme (Text Normalization)

Metin normalleştirme, TTS sisteminin daha etkin ve doğru çalışmasını sağlamak için kullanılan önemli bir ön işlem adımıdır. Bu adımda, girdi metindeki yazım hataları, kısaltmalar, slang ifadeleri ve benzeri durumlar düzeltilir. Bu işlem, metni daha tutarlı ve standart bir formda yapısal hale getirerek TTS sisteminin metni daha iyi anlamasına ve daha doğal sesli bir çıktı üretmesine yardımcı olur.

Metin Normalleştirme Adımında Düzeltilecek Durumlar:

- **Yazım Hataları:** Yanlış yazılmış kelimeler doğru şekilde düzeltilir.
- **Kısaltmalar:** Yaygın kullanılan kısaltmalar tam kelimeler ile değiştirilir.
- **Slang İfadeler:** Slang ifadeler standart kelimeler ile değiştirilir.
- **Büyük/Küçük Harf Kullanımı:** Metin, dilbilgisi kurallarına uygun şekilde büyük/küçük harfe dönüştürülür.
- **Özel Karakterler:** Özel karakterler standart karakterlere dönüştürülür.

Metin normalleştirme, TTS sisteminin performansını ve ses sentezinin kalitesini önemli ölçüde etkileyebilir. Bu adım, metnin daha tutarlı ve standart bir hale gelmesini sağlayarak TTS sisteminin metni daha hızlı ve doğru bir şekilde işlemesine yardımcı olur. Ayrıca, metinden tutarsızlıkların ve hataların giderilmesi, ses sentezinin daha doğal ve akıcı olmasını sağlar.

Metin Normalleştirme Adımında Kullanılan Teknikler:

- **Sözlükler:** Doğru yazılmış kelimelerin bir listesini içeren sözlükler kullanılır.
- **Kısaltma Tabloları:** Yaygın kullanılan kısaltmalar ve tam kelimeler arasındaki ilişkiyi gösteren tablolar kullanılır.
- **Dilbilgisi Kuralları:** Dilbilgisi kurallarına uygun şekilde büyük/küçük harf kullanımı ve noktalama işaretleri düzenlenir.

Metin normalleştirme işlemi, metnin anlamını ve içeriğini değiştirmemelidir. Bazı kısaltmalar veya slang ifadeler metnin anlamı için önemli olabilir ve bu nedenle korunmalıdır. Metin normalleştirme işlemi, kullanılan dil ve metnin türüne göre özelleştirilmelidir.

C - Tokenizasyon

Tokenizasyon, TTS sisteminin metni işlemesine ve anlamasına yardımcı olan önemli bir ön işlem adımıdır. Bu adımda metin, kelimeler, noktalama işaretleri ve diğer semboller gibi küçük parçalara ayrılır. Bu parçalara "token" denir. Tokenizasyon, metnin dilbilgisel yapısını ve anlamını analiz etmek için kullanılır.

Tokenizasyon Çeşitleri:

- **Kelime Tokenizasyonu:** En basit tokenizasyon türüdür. Metni boşluk karakterlerine göre böler ve her kelimeyi bir token olarak ele alır.
- **Noktalama İşareti Tokenizasyonu:** Noktalama işaretlerini de token olarak ele alır.

- **N-gram Tokenizasyonu:** Metni ardışık kelime gruplarına böler. Örneğin, 2-gram tokenizasyonda metin ikişerli kelime gruplarına ayrılır.
- **Karakter Tokenizasyonu:** Metni tek tek karakterlere böler.

Tokenizasyon, TTS sisteminin metni daha hızlı ve doğru bir şekilde işlemesine yardımcı olur. Ayrıca, tokenizasyon metnin dilbilgisel yapısını ve anlamını analiz etmek için kullanılır. Bu analiz, TTS sisteminin metni daha doğal bir şekilde seslendirmesine yardımcı olur. Tokenizasyon işlemi, metnin anlamını ve içeriğini değiştirmemelidir. Bazı kelimeler veya kelime öbekleri birden fazla anlama sahip olabilir ve bu nedenle tokenizasyon işlemi sırasında bu anlamlar dikkate alınmalıdır. Tokenizasyon işlemi, kullanılan dil ve metnin türüne göre özelleştirilmelidir.

Tokenizasyon, sadece TTS sistemlerinde değil, doğal dil işleme (NLP) ile ilgili birçok farklı alanda da kullanılan bir tekniktir. Tokenizasyon işleminin nasıl yapılacağı, kullanılan dil ve metnin türüne göre değişebilir. Tokenizasyon işlemi sırasında, metnin dil bilgilisel yapısını ve anlamını korumak önemlidir.

D – Dil Bilgisi Analizi (Linguistic Analysis)

Dilbilgisi analizi, TTS sisteminin metni daha iyi anlamasına ve daha doğal seslendirmesine yardımcı olan önemli bir ön işlem adımdır. Bu adımda, metindeki kelimelerin ve cümlelerin dilbilgisel yapısı analiz edilir. Bu analiz, kelimelerin anlamlarını ve cümlelerin kurallarını belirlemek için kullanılır.

Dil Bilgisi Analizinde Kullanılan Teknikler:

- **Kelime Biçim Bilimi:** Kelimelerin köklerini, çekim eklerini ve diğer biçim bilimsel özelliklerini analiz eder.
- **Söz Dizimi:** Cümlelerin yapısal kurallarını analiz eder.
- **Anlam Bilimi:** Kelimelerin ve cümlelerin anlamlarını analiz eder.

Dil bilgisi analizi, TTS sisteminin metni daha doğru bir şekilde seslendirmesine yardımcı olur. Örneğin, dil bilgisi analizi ile kelimelerin doğru telaffuzları belirlenebilir ve cümlelerin doğru tonlamayla okunması sağlanabilir. Dilbilgisi analizi, kullanılan dile ve metnin türüne göre özelleştirilmelidir. Dil bilgisi analizi, metnin anlamını ve içeriğini değiştirmemelidir. Dil bilgisi analizi, metnin dilbilgisel hatalarını düzeltebilir, ancak bu her zaman mümkün olmayabilir.

Dil bilgisi analizi, sadece TTS sistemlerinde değil, doğal dil işleme (NLP) ile ilgili birçok farklı alanda da kullanılan bir tekniktir. Dil bilgisi analizi işleminin nasıl yapılacağı, kullanılan dil ve metnin türüne göre değişebilir. Dil bilgisi analizi işlemi sırasında, metnin dil bilgilisel yapısını ve anlamını korumak önemlidir.

E - Sembol Dönüşümü (Symbol Conversion)

Sembol dönüşümü, TTS sisteminin farklı dil alfabeleri ve özel sembolleri içeren metinleri doğru bir şekilde seslendirmesine yardımcı olan önemli bir ön işlem adımdır. Bu adımda, girdi metindeki semboller ve özel karakterler, TTS sistemi tarafından anlaşılacak bir formata dönüştürülür.

Sembol Dönüşümünde Kullanılan Teknikler:

- **Karakter eşleme tabloları:** Farklı dil alfabeleri ve özel semboller için eşdeğer karakterler içeren tablolar kullanılır.

- **Unicode dönüşümleri:** Unicode standardına göre sembollerin kodlanmasını ve dönüştürülmesini sağlayan teknikler kullanılır.
- **Dil özel kurallar:** Belirli bir dil için özel sembollerin ve karakterlerin nasıl dönüştürüleceğini belirleyen kurallar kullanılır.

Sembol dönüşümü, TTS sisteminin farklı dilleri ve metin türlerini desteklemesine yardımcı olur. Örneğin, sembol dönüşümü ile Türkçe metinlerde kullanılan özel karakterler, TTS sistemi tarafından doğru bir şekilde seslendirilebilir. Sembol dönüşümü işlemi, metnin anlamını ve içeriğini değiştirmemelidir. Farklı diller ve metin türleri için özel dönüşüm kuralları ve tabloları kullanılmalıdır. Sembol dönüşümü işlemi sırasında, metnin dil bilgisel yapısı ve anlamını korumak önemlidir.

Sembol dönüşümü, sadece TTS sistemlerinde değil, doğal dil işleme (NLP) ile ilgili birçok farklı alanda da kullanılan bir tekniktir. Sembol dönüşümü işleminin nasıl yapılacağı, kullanılan dil ve metnin türüne göre değişebilir. Sembol dönüşümü işlemi sırasında, metnin dilbilgisel yapısını ve anlamını korumak önemlidir.

F - Ön İşaretçi (Pre-Emphasis)

Ön işaretleme, TTS sisteminin ses sentezinin kalitesini ve doğrallığını artırmak için kullanılan önemli bir tekniktir. Bu adımda, ses sinyallerinin spektrum dengesini iyileştirmek ve ardışık kelimeler arasındaki geçişleri daha doğal hale getirmek için çeşitli işlemler uygulanır.

Ön İşaretleme Teknikleri:

- **Basit Filtreleme:** Yüksek frekanslı bileşenleri artırmak için birinci veya ikinci dereceden bir filtre kullanılır.
- **Pre-Emphasis Filtreleri:** Özel olarak tasarlanmış ön-vurgu filtreleri kullanılır.
- **Cepstral Analiz:** Cepstral analiz ve sentez teknikleri kullanılarak ses sinyallerinin spektrum dengesinin ayarlanması.

Ön işaretleme ses sinyallerinin yüksek frekanslı bileşenlerinin amplitüdünü artırarak daha parlak ve net bir ses elde edilmesini sağlar. Ardışık kelimeler arasındaki geçişleri daha yumuşak ve doğal hale getirerek ses sentezinin akıcılığını artırır. Genel olarak ses sentezinin kalitesini ve anlaşılabilirliğini artırır. Ön-vurgu miktarı dikkatli bir şekilde ayarlanmasına dikkat edilmelidir. Aşırı ön-vurgu, sesin bozulmasına ve metalik bir tınıya neden olabilir. Ön-vurgu algoritması, kullanılan dil ve ses sentez yöntemine göre seçilmelidir. Ön-vurgu, diğer ses işleme teknikleri ile birlikte kullanılabilir.

Ön-vurgu, sadece TTS sistemlerinde değil, ses işleme ve sentez ile ilgili birçok farklı alanda da kullanılan bir tekniktir. Ön-vurgu algoritmasının nasıl seçileceği ve ayarlanacağı, kullanılan dil, ses sentez yöntemi ve istenen ses kalitesine göre değişebilir. Ön-vurgu, ses sentezinin karmaşıklığını ve işlem yükünü artırabilir.

4.1.2. Morfolojik Analiz Modülü (Morphological Analyzer)

Morfolojik analiz modülü, kelime köklerini, çekim eklerini, zamanları, çoğul formları ve diğer dilbilgisel özellikleri tanımlayarak, metnin dilbilgisel yapısını derinlemesine analiz eder. Morfolojik analiz, genellikle dilbilgisi kurallarını ve sözlüklerini kullanarak gerçekleştirilir ve metnin anlamını ve yapısal özelliklerini daha iyi anlamak için kullanılır.

- **Kelime Köklerinin Belirlenmesi:** Morfolojik analiz, her kelimenin kökünü belirleyerek başlar. Kelime kökleri, bir kelimenin anlamını temsil eden temel formdur. Örneğin, "okuma" kelimesinin kökü "oku" dur.
- **Çekim Eklerinin Tanımlanması:** Morfolojik analiz, kelime köklerine eklenen çekim eklerini tanımlar. Çekim ekleri, kelimenin zamanı, çoğulu, kişisi ve diğer dilbilgisel özelliklerini belirler. Örneğin, "okuyor" kelimesinin kökü "oku" dur ve "-yor" çekim eki, şu anki zamanı ve 3. tekil kişiyi belirtir.
- **Zamanların ve Diğer Dilbilgisel Özelliklerin Belirlenmesi:** Morfolojik analiz, kelimenin zamanını (geçmiş, şimdi, gelecek), kişisini (birinci, ikinci, üçüncü), çoğulunu, cinsiyetini ve diğer dilbilgisel özelliklerini belirler. Bu adım, cümlenin anlamını ve yapısını daha iyi anlamak için önemlidir.
- **Sözcüklerin Anlamının Belirlenmesi:** Morfolojik analiz modülü, her kelimenin anlamını belirlemek için bir sözlük veya dilbilgisi kural seti kullanır. Bu adım, kelimenin kullanıldığı bağlama göre doğru anlamın belirlenmesine yardımcı olur.

Morfolojik analiz modülü, doğal dil metninin yapısını ve bileşenlerini anlamak ve dilbilgisel özelliklerini tanımlamak için önemlidir. Bu analiz, genellikle doğal dil işleme ve metin tabanlı uygulamalarda kullanılır ve metnin daha iyi anlaşılmasını ve işlenmesini sağlar.

4.1.3. Bağlamsal Analizör (Contextual Analyzer)

Bağlamsal analizör, bir metnin içeriğini ve anlamını daha iyi anlamak için kullanılır. Bağlamsal analizör, metindeki kelimelerin ve cümlelerin birbirleriyle ilişkisini değerlendirir ve bu ilişkilerden yola çıkarak metnin anlamını daha derinlemesine çözmeye çalışır. Bağlamsal analizörün temel görevleri şunlardır:

- **Anlam Belirleme:** Bağlamsal analizör, metindeki kelimelerin anlamlarını belirler ve bu kelimelerin kullanıldığı bağlamı değerlendirir. Örneğin, bir kelimenin birden fazla anlamı olabilir ve bağlamsal analizör, metnin genel bağlamına göre hangi anlamın daha uygun olduğunu belirler.
- **Kelime İlişkilerini Analiz Etme:** Bağlamsal analizör, metindeki kelimelerin birbirleriyle olan ilişkilerini analiz eder. Bu ilişkiler, eş anlamlılık, zıtlık, bağımlılık veya diğer dilbilgisel ilişkiler olabilir. Örneğin, "anne" ve "çocuk" kelimeleri arasında bir bağlantı olduğu düşünülebilir, bu nedenle bağlamsal analizör bu ilişkiyi tanımlayabilir.
- **Anlam Kaymalarını Belirleme:** Bağlamsal analizör, metindeki anlam kaymalarını belirler. Anlam kayması, bir kelimenin veya ifadenin farklı bağlamlarda farklı anlamlara gelebilmesidir. Örneğin, "banka" kelimesi hem finansal kurumları hem de nehir kenarını ifade edebilir. Bağlamsal analizör, hangi anlamın belirli bir bağlamda kullanıldığını belirleyebilir.
- **Sözdizimi ve Gramer Analizi:** Bağlamsal analizör, metindeki cümle yapılarını ve dilbilgisel kuralları analiz eder. Bu adım, cümlelerin doğru bir şekilde yapılandırıldığından ve dilbilgisel olarak doğru olduğundan emin olmak için önemlidir.

Bağlamsal analizör, metnin daha derinlemesine anlaşılmasına ve daha doğru bir şekilde yorumlanmasına yardımcı olur. Bu, NLP sistemlerinin daha etkili çalışmasını sağlar ve metnin anlamını daha iyi yakalamasını sağlar. Bu nedenle, bağlamsal analizör, doğal dil işleme alanında önemli bir bileşen olarak kabul edilir.

4.1.4. Söz Dizimsel – Prozodik Çözümleyici (Syntactic – Prosodic Parser)

Söz dizimsel-prozodik çözümleyici metnin hem sözdizimsel hem de ses bilgisel yapısını analiz eder. Bu çift yönlü analiz, metnin hem dil bilgisel yapısını hem de tonlama, vurgu ve ritim gibi ses bilgisel özelliklerini anlamamıza yardımcı olur.

Söz dizimsel-prozodik çözümleyici modülünün temel görevleri şunlardır:

- **Söz dizimsel Analiz:** Bu bileşen, metindeki cümle yapılarını, kelime sıralamalarını, dil bilgisel rolleri ve diğer söz dizimsel özellikleri analiz eder. Bu analiz, metnin anlamını ve mantığını anlamamıza yardımcı olur. Örneğin, bir cümlenin öznesini, yüklemi ve nesnesini tanımlayabilir ve bu bileşen aracılığıyla cümlenin yapısını çözebiliriz.
- **Prozodik Analiz:** Bu bileşen, metindeki tonlama, vurgu, ritim ve diğer sesbilgisel özellikleri analiz eder. Prozodik analiz, konuşma akışını, vurgulanmış kelimeleri, cümlenin vurgu desenini ve diğer sesbilgisel özellikleri tanımlar. Örneğin, bir cümlenin vurgulu kelimelerini belirleyebilir ve cümlenin sesbilgisel yapısını analiz edebiliriz.
- **Söz dizimsel ve Prozodik İlişkilerin Tanımlanması:** Bu bileşen, metindeki söz dizimsel yapıların ve prozodik özelliklerin birbiriyle ilişkisini belirler. Örneğin, vurgulanmış bir kelimenin söz dizimsel rolünü belirleyebilir veya belirli bir dil bilgisel yapıdaki bir kelimenin vurgulanmasının nasıl değiştiğini analiz edebiliriz.
- **Metin Akışını ve Anlamını Geliştirme:** Söz dizimsel-prozodik çözümleyici, metnin akışını ve anlamını geliştirmek için söz dizimsel ve ses bilgisel özellikleri birleştirir. Bu analiz, metnin daha doğal, akıcı ve anlaşılır olmasını sağlar.

Söz dizimsel-prozodik çözümleyici, metnin hem dilbilgisel hem de ses bilgisel özelliklerini aynı anda analiz ederek daha kapsamlı bir anlayış sağlar. Bu, konuşma sentezi, dil anlama, metin tabanlı uygulamalar ve diğer NLP alanlarında daha etkili ve doğru sonuçlar elde etmemizi sağlar. Bu nedenle, söz dizimsel-prozodik çözümleyici, doğal dil işleme alanında önemli bir bileşen olarak kabul edilir.

4.2. Otomatik Fonetikleştirme

Otomatik fonetikleştirme veya "Harf-Ses Modülü" (LTS), bir yazılı metindeki harflerin veya grafemlerin seslere dönüştürülmesi işlemidir. Bu işlem, bir kelimenin nasıl telaffuz edileceğini belirlemek için kullanılır. Özellikle konuşma sentezi sistemlerinde, bir metni sesli olarak çıkarmak için otomatik fonetikleme önemlidir.

4.2.1. Giriş Metninin Hazırlanması:

Giriş metninin hazırlanması, otomatik fonetikleştirmenin ilk ve önemli bir adımıdır. Bu adımda, fonetikleme işlemine tabi tutulacak olan metin çeşitli işlemlerden geçirilerek fonetikleme için uygun hale getirilir.

Giriş Metninin Hazırlanmasında Yapılan İşlemler:

A - Metin Temizleme: Metindeki noktalama işaretleri, özel karakterler ve sayılar gibi fonetikleme için gerekli olmayan unsurlar temizlenir.

B - Büyük/Küçük Harf Dönüştürme: Tüm harfler büyük veya küçük harfe dönüştürülür. Bu işlem, dilin ses sistemine göre değişebilir. Örneğin, Türkçede tüm harfler küçük harfe dönüştürülür.

C - Kelime Ayırıştırma: Metin, kelimelerine ayrıştırılır. Bu işlem, kelimelerin telaffuzlarının doğru bir şekilde belirlenmesi için gereklidir.

D - Homofoni Çözümleme: Farklı anlamlara sahip olan ancak aynı şekilde telaffuz edilen kelimeler (homofonlar) için çözümleme yapılır. Bu işlem, fonetikleme işleminin doğru bir şekilde yapılması için önemlidir.

E - Morfem Ayırıştırma: Kelimeler, morfemlerine (en küçük anlamlı dil birimleri) ayrıştırılabilir. Bu işlem, bazı dillerde kelimelerin telaffuzlarının doğru bir şekilde belirlenmesi için gereklidir.

Metin temizleme işlemi dikkatli bir şekilde yapılmalı ve metnin anlamını değiştirecek unsurlar silinmemelidir. Büyük/küçük harf dönüştürme işlemi, dilin ses sistemine göre doğru bir şekilde yapılmalıdır. Kelime ayırıştırma işlemi, kelimelerin doğru bir şekilde telaffuz edilebilmesi için gereklidir. Homofoni çözümleme işlemi, fonetikleme işleminin doğru bir şekilde yapılması için önemlidir. Morfem ayırıştırma işlemi, bazı dillerde kelimelerin telaffuzlarının doğru bir şekilde belirlenmesi için gereklidir.

Giriş metninin hazırlanması, otomatik fonetikleştirmenin en temel adımlarından biridir. Giriş metninin doğru bir şekilde hazırlanması, fonetikleme işleminin doğruluğunu ve başarısını doğrudan etkiler. Giriş metninin hazırlanması için farklı teknikler ve araçlar kullanılabilir.

4.2.2. Grafemleri Soslere Dönüştürme:

Grafemleri soslere dönüştürme, otomatik fonetikleştirmenin en önemli adımlarından biridir. Bu adımda, metindeki her harf veya harf grubu, dilin ses sistemi ve telaffuz kurallarına göre uygun bir şekilde soslere dönüştürülür.

Grafemleri Soslere Dönüştürme İşleminde Kullanılan Teknikler:

Grafem-Ses Kuralları: Belirli bir dildeki harflerin veya harf gruplarının nasıl telaffuz edildiğini belirleyen kurallardır. Bu kurallar, dilbilgisi kitaplarında veya fonetik kaynaklarda bulunabilir.

Sözlükler: Kelimelerin telaffuzlarının bir sözlükte saklanması ve metin işlenirken bu sözlükten yararlanılmasıdır.

İstatistiksel Modeller: Geçmiş verilerden öğrenerek yeni kelimelerin telaffuzlarını tahmin eden modellerdir.

Grafemleri soslere dönüştürme işleminde dilin ses sistemi ve telaffuz kuralları iyi bilinmesine dikkat edilmelidir. Kullanılan teknik, dilin özelliklerine uygun olmalıdır. İşlem sırasında dil bilgisi ve dil bilgisel kurallar dikkate alınmalıdır. Grafemleri soslere dönüştürme, otomatik fonetikleştirmenin en karmaşık adımlarından biridir. Bu adımda kullanılan teknikler ve algoritmalar sürekli olarak geliştirilmektedir. Grafemleri soslere dönüştürme işleminin doğruluğu, fonetikleme işleminin genel doğruluğunu etkilemektedir.

Ses dönüştürme işlemine örnek vermek gerekirse İngilizce de "th" harfi, kelimelerin başında veya ortasında "θ" (sessiz "s") veya "ð" (sessiz "z") seslerine dönüştürülebilir. Örneğin, "think" kelimesinde "th" harfi "θ" sesine ve "that" kelimesinde "ð" sesine dönüştürülür.

4.2.3. Ses Birimlerini Birleştirme

Ses birimlerini birleştirme, otomatik fonetikleştirmenin önemli bir adımdır. Bu adımda, her kelimenin grafemleri (harfler) soslere dönüştürüldükten sonra, bu ses birimleri kelimenin telaffuzunu oluşturmak için birleştirilir.

Ses Birimlerini Birleştirme İşleminde Yapılan İşlemler:

A - Hece Oluşturma: Ses birimleri, heceleri oluşturmak için bir araya getirilir. Heceler, bir kelimenin telaffuzunun temel birimleridir. Türkçe'de heceler genellikle bir ünlü ve bir veya birden fazla ünsüzden oluşur.

B - Tonlama ve Vurgu: Kelimelerin tonlama ve vurgulaması, ses birimlerinin birleştirilmesi sırasında belirlenir. Tonlama, sesin frekansındaki değişimdir ve kelimelerin anlamını ve duygusal tonunu belirler. Vurgu ise, bir hecenin diğer hecelerden daha güçlü bir şekilde telaffuz edilmesidir.

C - Sesli ve Ünsüzlerin Birleştirilmesi: Sesli ve ünsüz harflerin telaffuzlarının birleştirilmesi, kelimenin doğru bir şekilde seslendirilmesi için önemlidir. Sesli harfler, kendi başlarına telaffuz edilebilen harflerdir. Ünsüz harfler ise, sesli harfler ile birlikte telaffuz edilebilen harflerdir.

Ses birimlerini birleştirme işleminde dilin ses sistemi ve telaffuz kuralları iyi bilinmelidir. Kelimelerin heceleri ve tonlaması doğru bir şekilde belirlenmelidir. Sesli ve ünsüz harflerin telaffuzlarının birleştirilmesi sırasında dil bilgisi ve dil bilgisel kurallar dikkate alınmalıdır.

Ses birimlerini birleştirme, otomatik fonetikleştirmenin en karmaşık adımlarından biridir. Bu adımda kullanılan teknikler ve algoritmalar sürekli olarak geliştirilmektedir. Ses birimlerini birleştirme işleminin doğruluğu, fonetikleme işleminin genel doğruluğunu etkilemektedir.

4.2.4. Dil Bilgisi ve Dil Bilgisel Kuralların Uygulanması

Dil bilgisi ve dil bilgisel kuralların uygulanması, otomatik fonetikleştirmenin önemli bir adımıdır. Bu adımda, dil bilgisi ve dil bilgisel kurallar, ses birimlerini birleştirme ve kelimenin telaffuzunu oluşturma işleminde kullanılır. Dil bilgisi ve dil bilgisel kurallar iyi bilinmelidir. Kuralların istisnaları da dikkate alınmalıdır. Farklı dillerde dil bilgisi ve dil bilgisel kurallar farklı şekilde uygulanır. Dil bilgisi ve dil bilgisel kuralların uygulanması, otomatik fonetikleştirmenin en karmaşık adımlarından biridir. Bu adımda kullanılan teknikler ve algoritmalar sürekli olarak geliştirilmektedir. Dil bilgisi ve dil bilgisel kuralların uygulanmasının doğruluğu, fonetikleme işleminin genel doğruluğunu etkilemektedir. Dil bilgisi ve dil bilgisel kurallar karmaşık olabilir ve istisnalar içerebilir. Farklı dillerde dil bilgisi ve dil bilgisel kurallar farklı şekilde uygulanmaktadır. Dil bilgisi ve dil bilgisel kuralların uygulanmasında kullanılan teknikler ve algoritmalar her zaman tam doğru sonuç vermeyebilir.

4.2.5. Çıktı Metninin Oluşturulması

Çıktı metninin oluşturulması, otomatik fonetikleştirmenin son ve önemli bir adımıdır. Bu adımda, önceki adımlarda elde edilen bilgiler kullanılarak kelimenin tam telaffuzu oluşturulur ve metnin sesli bir biçimde çıkarılması için kullanılır.

Otomatik fonetikleme veya Letter-To-Sound, metin tabanlı verilerin sesli biçimlendirilmesini sağlar ve bu da konuşma sentezi sistemlerinin veya sesli kullanıcı arayüzlerinin çalışması için önemlidir. Ayrıca, dil bilgisi ve telaffuz açısından doğru bir temsil sağlamak için kullanılan bir araçtır.

4.3. Prozodi Üretici (Prosody Generator)

Prozodi üretici, bir metnin veya cümlenin sesbilgisel özelliklerini oluşturan ve belirleyen bir bileşendir. Bu bileşen, metindeki vurgu, tonlama, ritim ve diğer ses özelliklerini belirler ve

metnin doğal ve akıcı bir şekilde telaffuz edilmesini sağlar. Prozodi, konuşmanın melodik yapısı olarak düşünülebilir ve konuşmanın duygusal ifadesini, vurgusunu ve akıcılığını belirler.

Prozodi üretici genellikle şu temel görevleri yerine getirir:

Vurgu Deseni Belirleme: Prozodi üretici, metindeki önemli kelimeleri veya kelime gruplarını vurgular. Bu, cümlelerin veya metnin anlamını ve vurgusunu belirleyen önemli bir özelliktir.

Tonlama ve Yükseklik Belirleme: Konuşmanın tonu, yükseklik ve melodik yapısı, prozodi üreticisi tarafından belirlenir. Bu, konuşmanın genel melodisini ve duygusal ifadesini belirler.

Ritim ve Akıcılık Oluşturma: Prozodi üretici, konuşmanın ritmi ve akıcılığı üzerinde etkilidir. Kelimelerin ve cümlelerin doğru bir ritimle telaffuz edilmesini sağlar ve konuşmanın akıcı olmasını sağlar.

Pozitif ve Negatif İfadeleri Belirleme: Prozodi üretici, metnin anlamına göre pozitif veya negatif ifadeleri belirler. Bu, konuşmanın duygusal ifadesini ve tonunu belirleyen önemli bir özelliktir.

Prozodi üreticisi, genellikle konuşma sentezi sistemlerinde veya sesli kullanıcı arayüzlerinde kullanılır. Bu, metni doğal ve akıcı bir şekilde telaffuz etmek için önemlidir ve insanlarla etkili bir iletişim kurmayı sağlar. Prozodi üreticisi, metnin doğru bir şekilde telaffuz edilmesini sağlayarak, konuşma sentezi sistemlerinin veya diğer konuşma tabanlı uygulamaların kullanıcı deneyimini iyileştirir.

4.4. Ses Sentezi

Ses Sentezi, TTS teknolojisinin önemli bir aşamasını oluşturur ve metin analizi sonuçlarına dayanarak yazılı metinleri sesli olarak çıkarmak için kullanılır. Bu aşama, metinde tanımlanan metin birimlerini, kelimeleri, heceleri veya cümleleri seslendirmek için gerekli verileri kullanır. Ses sentezi sırasında aşağıdaki aşamalar aşağıdaki gibidir.

4.4.1. Metin Birimleri Seslendirme

Metin analizi aşamasının sonuçlarına dayanarak, ses sentezi aşamasında metinde tanımlanan metin birimleri seslendirilir. Bu birimler, genellikle kelimeler, heceler veya cümleler olabilir. Özel dikkat, metindeki dil bilgisi yapısı ve vurgu değişikliklerine verilir, böylece sesli çıktı doğal ve akıcı olur.

4.4.2. Dil ve Aksan Uyumu

Ses sentezi aşamasında, metin belirli bir dil ve aksanla uyumlu bir şekilde seslendirilir. Bu, metin içeriğinin hedef kitlesi veya kullanıcısı için daha anlaşılır ve erişilebilir olmasını sağlar. Özellikle çok dilli TTS sistemlerinde, metin hangi dilde ve hangi aksanla seslendirilmesi gerektiği konusunda dikkate alınır.

4.4.3. Doğal Ses Modülasyonu

Ses sentezi, metindeki vurgu, tonlama ve duygusal ifadeleri yansıtmak için sesin yüksekliği, hızı ve vurgusu gibi özelliklerin ayarlanmasıyla gerçekleştirilir. Örneğin, bir mutlu ifade daha yüksek bir ses tonu ve pozitif bir vurgu ile seslendirilirken, üzgün bir ifade daha düşük bir ses tonu ve daha sakin bir vurgu ile seslendirilir.

4.4.4. Ses Akışı ve Akıcılık

Ses sentezi aşamasında, metin birimleri arasındaki geçişler düzgün bir akıcılık sağlayacak şekilde gerçekleştirilir. Bu, sesli çıktının daha doğal ve kolay dinlenir olmasını sağlar.

4.4.5. Sesli Özellikler ve Hız

Ses sentezi sırasında sesli çıktının özellikleri de belirlenir. Ses hızı, ses yüksekliği ve vurgu gibi özellikler, metnin anlamını ve ifadesini daha iyi yansıtmak için ayarlanır.

4.4.6. Ses Kalitesi ve Yükseklik

TTS teknolojisinin ses kalitesi, kullanılan ses motoruna, ses veri tabanına ve algoritmalara bağlı olarak değişebilir. Ses sentezi aşamasında, yüksek kaliteli seslerin kullanılması ve doğal sesin yakalanması için özel önem verilir.

Ses sentezi aşaması, TTS teknolojisinin metinleri sesli olarak çıkarmak için kullanılan en karmaşık ve önemli aşamalardan biridir. Bu aşama, metni insan sesini taklit ederek sesli bir çıktıya dönüştürür. Başarılı bir ses sentezi, metin tabanlı içeriğin daha geniş bir kullanıcı kitlesi için erişilebilir hale gelmesini sağlar ve birçok uygulama alanında kullanılır. Ses sentezi aşamasının kalitesi, TTS teknolojisinin genel etkinliği ve kullanılabilirliği üzerinde büyük bir etkiye sahiptir. Bu adım gerçekleştirilirken uygulanan adımlar 5.Sentezleme Yöntemleri başlığı altında incelenmiştir.

4.5. Ses Çıkışı Oluşturma

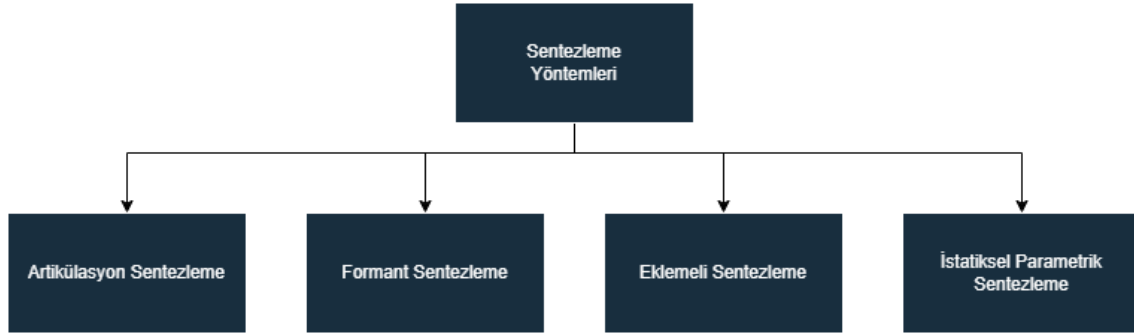
Ses sentezlemesi tamamlandığında, ses çıkışı oluşturma aşamasına geçilir. Bu aşamada, sentezlenmiş ses birimleri birleştirilir ve son kullanıcıya sunulacak sesli çıktı oluşturulur. Bu aşamada ses tonlaması, vurgu ve hız gibi özellikler de dikkate alınır. Ayrıca, sesli çıktının dosya formatı ve sıkıştırma gibi teknik detayları da burada belirlenir. Bu işleyiş TTS teknolojisinin temelini oluşturur. Her bir adım metin tabanlı verilerin sesli çıktıya dönüştürülmesinde önemlidir ve sonuç metni duyan veya kullanıcının metni sesli olarak dinlediği doğal bir sesli çıktıdır. TTS teknolojisi, her geçen gün daha da gelişmekte ve birçok uygulama alanında kullanılmaktadır. Bu işleyişin anlaşılması, TTS teknolojisinin nasıl çalıştığını ve neden bu kadar önemli olduğunu anlamak için önemlidir.

5. Sentezleme Yöntemleri

Yapay konuşma üretimi için birçok sentezleme yöntemi bulunmaktadır. Dünya üzerindeki bütün dillerde işe yarayacak evrensel bir sentezleme türü bulunmadığı için ve konuşma kalitesinin yükseltilmesi amaçlandığından farklı yöntemler ortaya çıkmıştır. Bu makalede temel sentezleme yöntemlerinin işleyişi, avantajları ve dezavantajları anlatılmaktadır. Şekil 15’de temel sınıfların şeması gösterilmiştir. Sentezleme yöntemleri 4 temel sınıfa ayrılarak anlatılacaktır. Artikülasyon Sentezleme (Articulatory Synthesis)

- Formant Sentezleme (Formant Synthesis)
- Eklemeli Sentezleme (Concatenative Synthesis)
 - Birim Seçme Sentezleme (Unit Selection Synthesis)
 - İkili Ses Sentezleme (Diphone Synthesis)
 - Uygulamaya Özgü Sentezleme (Domain-Specific Synthesis)
 - Hece Tabanlı Sentezleme (Syllable-Based Synthesis)
 - Dalga Formu Birleştirme Tekniği (Waveform Concatenation Technique)
- İstatiksel Parametrik Sentezleme (Statistical Parametric Synthesis)

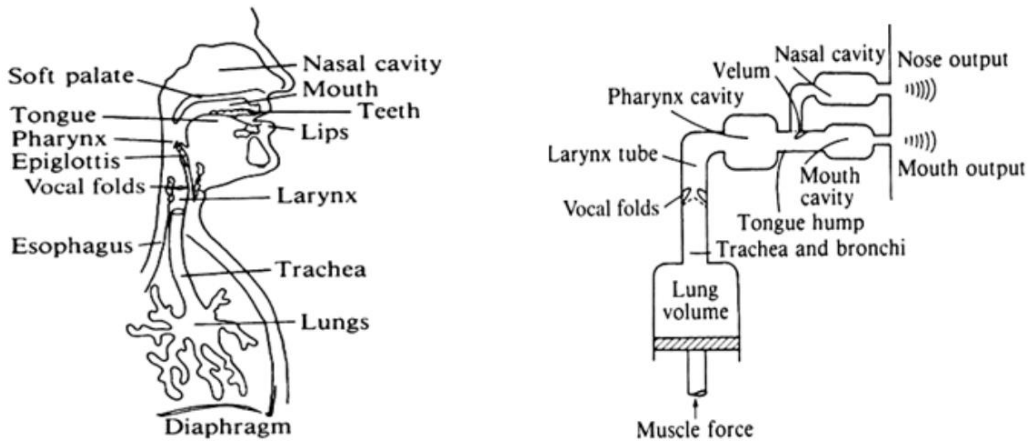
- Hidden Markov Model Sentezleme (HMM)
- Derin Sinir Ağı Sentezleme (DNN)



Şekil 15. Sentezleme Yöntemleri Şeması

5.1. Artikülasyon Sentezleme (Articulatory Synthesis)

Artikülasyon sentezleme yönteminde insanların doğal konuşma üretim sürecini modeller ve bu süreçte hava akışı dijital olarak simüle edilerek yapay konuşma üretmeyi amaçlamaktadır. Artikülasyon sentezleme ile üretilen seslerde anlaşılabilirlik oranı yüksektir fakat artikülerin hassas bir şekilde modellenmesi zordur bu sebeple konuşma kalitesi genellikle diğer sentez türlerine göre daha düşüktür. Bu yöntemle ile üretilen yapay konuşmalar kalite olarak diğer sentez yöntemlerinden düşük kalitede olsa da konuşma üretim sürecini incelemek için araştırmacılar tarafından kullanılan en uygun yöntemdir. Artikülasyon konuşma sentezi sürecini anlamak için, öncelikle insanların konuşurken kullandıkları organların (dil, çene ve dudaklar gibi) anlaşılması gerekmektedir (Şekil 16’da insan konuşma organlarının modellenmesi bulunmaktadır). Kural tabanlı bir sentezleme türü olan artikülasyon sentezlemede kontrol parametreleri dudak açıklığı, dudak çıkıntısı, dil ucu yüksekliği, dil ucu konumu, dil yüksekliği, dil konumu vb. parametreler kullanılır. Uyarılma parametrelerinde ise ses telleri açıklığı, tellerin gerilimi ve akciğer basıncı gibi parametreler kullanılır. Konuşurken solunum kasları bir enerji kaynağı olarak görev yapar ve akciğerler basınçlı hava depolar. Ses yolundaki kaslar, artikulatorleri hareket ettirir ses yolunun şeklini değiştirerek farklı sesleri üretebilmeyi mümkün hale getirir.



Şekil 16. Konuşma sırasında kullanılan organlar ve Organların modellenmesi

Artikülasyon konuşma sentezi, otomatik yüz animasyonu ve konuşma bozukluklarının tedavisi gibi uygulamalarda başarılı sonuçlar verir aynı zamanda sesin fonetik özelliklerini,

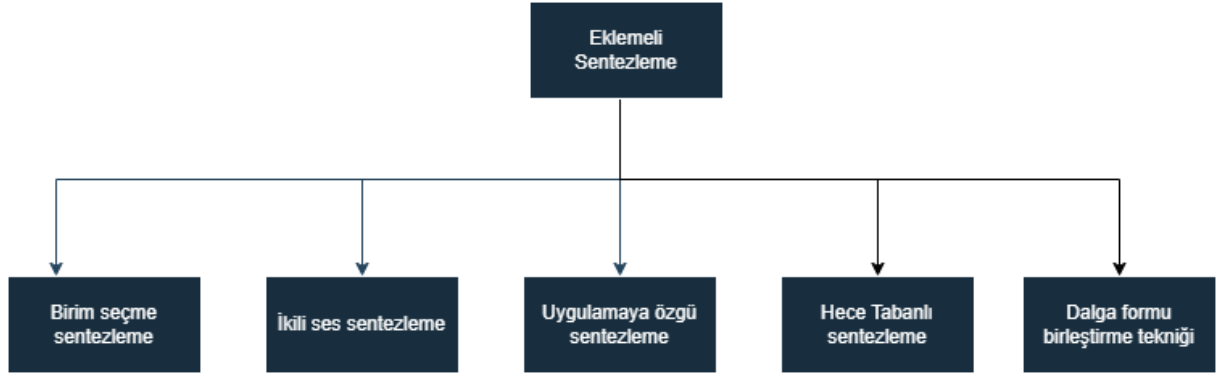
konusmacının fizyolojik özelliklerini (cinsiyet, yaş, duygusal durum) ve konuşma tarzını tanımlamak için kullanılabilir.

5.2. Formant Sentezleme (Formant Synthesis)

Formant sentezleme, sinir ağlarının ortaya çıkmasından önce konuşma sentezi için en iyi sentezleme yöntemi olarak kullanılıyordu. Sinir ağlarının otomatik olarak konuşmada desenleri ve yapıları bulma yeteneğine sahip olmadığı dönemde, formant sentezi kullanılarak karmaşık ve matematiksel yöntemler ile sağlam modeller oluşturuluyordu. Formant sentezi tekniği, kurallara dayalı bir sentezleme tekniği olup artikülasyon sentezi ile yakından ilişkilidir. Bu teknik, doğal konuşmanın formant yapısını mümkün olduğunca yakından taklit eden belirli kurallara dayalı yapay sinyaller üreterek konuşma oluşturur. Konuşma sentezlemek için temel frekans, ses genliği ve gürültü seviyesi gibi farklı parametrelere sahip bir akustik model kullanır. Konuşmayı üretmek için bir konuşma veri tabanına bağlı olmadıkları için sesleri birleştiren tekniklere göre daha küçük programlardır. Bu nedenle, formant sentezi, bellek ve mikro işlemci gücünün sınırlı olduğu gömülü sistemler için uygun bir konuşma sentezi tekniği olabilir. Formant sentezinin başlıca dezavantajı, sistemin insan tarafından konuşulan doğal konuşmadan oldukça farklı olan yapay, robotik bir konuşma üretmesidir. Ayrıca, kaynak ve tüm filtre parametrelerinin zamanlamasını belirleyen kuralları tasarlamak basit kelimeler için bile oldukça zor olabilir.

5.3. Eklemeli Sentezleme (Concatenative Synthesis)

Birleştirici konuşma sentezi tekniği, veri tabanına kaydedilmiş konuşma örneklerini (kelimeler, heceler, yarım heceler, fonemler vb.) kullanarak girdi metin ifadelerine göre uygun birimleri birleştirerek konuşma üretmek için kullanılan sentezleme yöntemidir. Artikülasyon ve Formant sentezi gibi kurallı sentezdeki zorlukları ortadan kaldırmak amacıyla geliştirilmiştir. Ses sentezi kalitesi, veri tabanındaki birimlerin uzunluğundan etkilenir daha uzun birimler kullanarak sentezlenen konuşmanın doğallığı artar, ancak daha uzun birimler kullanıldığında birleştirme noktaları azalır ve bu da birleştirme noktalarında doğal olmayan bölümlerin oluşma olasılığını azaltır. Bununla birlikte, daha fazla belleğe ihtiyaç duyulur ve veri tabanındaki birim sayısı artar. Öte yandan, daha kısa birimler kullanıldığında bellek gereksinimi azalır, ancak örnek toplama ve etiketleme tekniklerinin karmaşıklığı artar. Eklemeli sentezlemenin ürettiği yapay konuşma genel olarak yüksek anlaşılabilirlik ve insan sesine yakın bir sonuç verir. Ancak veri tabanında yüksek yer kaplar ve vurgu, duygu, tonlama gibi konularda eksiklikler yaşar. Bu eksiklerin nedeniyle üretilen sesler doğal ve duygusal değildir. Eklemeli sentezleme başlığı altında birçok sentezleme yöntemi bulunmaktadır. Bu makalede eklemeli sentezlemenin yaygın olarak kullanılan 5 türü anlatılacaktır. Şekil 17’de eklemeli sentezleme için en yaygın kullanılan 5 sentezleme yöntemi gösterilmektedir



Şekil 17. Eklemeli Sentezleme Yöntemleri Şeması

5.3.1. Birim Seçme Sentezleme (Unit Selection Synthesis)

Birim seçimi sentezi, büyük veri tabanlarını kullanır. Bu veri tabanlarını oluştururken, her kaydedilmiş konuşma örneği, kelimeler, heceler gibi küçük parçalara ayrılır. Genellikle bu ayrımı yaparken özel bir konuşma tanıma sistemi kullanılır ve ardından el ile düzeltmeler yapılır. Her bir konuşma örneğinin, temel frekanstan, süreye kadar çeşitli özelliklerle indekslendiği bir veri tabanı oluşturulur. Sentez işlemi yapılırken oluşturulacak konuşma veri tabanındaki en uygun ifadelerden seçilir. Uygun ifadeleri seçmek için karar ağaçları kullanılır. Birim seçme sentezi, hedef maliyeti ölçmek için ses birimlerinin birbirine olan benzerliğini ve uzaklığını ölçerek en iyi eşleşmeyi bulmaya çalışır. Bu sentezleme yönteminde veri tabanındaki konuşmalara az miktarda müdahale edildiği için doğal konuşma oluşturulabilmektedir. Yüksek doğallık seviyelerine ulaşabilmek için kaydedilmiş seslerin büyük bir veri tabanı gereklidir ve bunlar onlarca saatlik konuşma ve terabaytlarca bellek gerektirebilir. Birçok metin okuma sistemleri için bu kadar büyük miktarda veri ve gereken hesaplama kaynakları uygulanabilir değildir. Başka bir dezavantajı da birim seçimi algoritmalarının, en iyi sentezin gerçekleşmesi için daha iyi bir seçenek olsa bile bazı durumlarda daha kötü alternatifleri seçmektedir.

5.3.2. İkili Ses Sentezleme (Diphone Synthesis)

İkili Ses Sentezleme bir konuşmanın seslerinin geçişlerini bir araya getirerek ses üretme yöntemidir. Bu yöntem, bir dilin temel seslerini temsil eden parçaları kullanır. Bir sesin bir diğerine geçişi sırasındaki ses değişimlerini yakalamaktır. Farklı dillerde bu ses değişimlerine ilişkin belirli kurallar olduğundan, her dilin bir sınırlı sayıda ses geçiş parçası vardır. Yapay konuşma üretebilmek için, metin önce temel seslere dönüştürülür ve sonra bu sesler veri tabanındaki uygun ses geçişleri ile eşleştirilir. Sonra, ses geçişleri arasındaki geçişi düzeltmek ve vurgu, süre ve tonlamayı ayarlamak için dijital işlemler yapılır. İkili ses sentezlemenin avantajı, veri tabanının sınırlı olmasıdır, ancak sentezlenen ses monoton ve doğal olmayan sesler üretilmesine yol açabilmektedir.

5.3.3. Uygulamaya Özgü Sentezleme (Domain-Specific Synthesis)

Uygulamaya özgü sentezlemede ifadeleri oluşturmak için genellikle veri tabanından sınırlı kelime ve cümlelerin birimlerini seçerek çalışan bir ses sentezleme yöntemidir. Çoğunlukla belirli bir konu veya uygulama alanına yöneliktir, örneğin konuşan saatler, ulaşım tarifleri, hava raporları, tren sorgular gibi alanlarda kullanılır. Basit ve uygulanması kolay olması nedeniyle uzun süredir ticari kullanımdadır. Uygulamaya özgü sentezlemenin en büyük avantajı, doğallık seviyesidir. Cümle türlerinin çeşitliliği kısıtlı olduğu için çok yüksek bir doğallık seviyesine sahip olabileceği gibi kaydedilen orijinal konuşmanın vurgusu ve tonlamasına da çok yakındır. Ancak sistemin kullanımı sınırlıdır çünkü sadece veri tabanındaki kelimeleri ve ifadeleri

kullanabilir. Yani, özelleştirme veya daha genel konuşmalar için uygun değildir. Bu nedenle, genel kullanım amaçları için uygun bir seçenek değildir.

5.3.4. Hece Tabanlı Sentezleme (Syllable-Based Synthesis)

Hece tabanlı sentezleme yönteminde heceler, bir kelimenin fonetik yapı taşları olarak kabul edilir. Hecelere dayalı konuşma sentezi tekniği, herhangi bir dilde ortak bir hece birimi seti için kaydedilmiş konuşma örneklerini saklar. Genellikle bir hece oluşturulurken başlangıç ve bitişlerine ünsüz harfler, ünsüz harflerin arasına ise ünlü harf ekleyerek 3 harften oluşan heceler oluşturulur. Hecelere dayalı konuşma sentezi sistemi oluşturmak, sistemin oluşturulacağı dildeki hece birimlerini iyi tanımlamak gerekmektedir. Her dildeki hece birimlerinin toplam sayısı, dilin özelliklerine bağlı değişmektedir. Kelimeleri hecelerden oluşturan dillerde başarılı sonuçlar verebilir ancak veri tabanını tasarlamak zor bir görevdir.

5.3.5. Dalga Formu Birleştirme Tekniği (Wavform Concatenation Technique)

Dalga formu birleştirme tekniği (WCT), diğer yöntemlere göre daha az depolama alanı kullanarak konuşma sentezi oluşturur. Bu teknik, temel ses birimlerini depolar ve bunları belirli bir metin için konuşma sesine dönüştürmek için belirli kuralları kullanır. Örneğin, bir dildeki metinden konuşma sentezi oluştururken, belirli seslerin kombinasyonlarını depolar ve metnin sesli bir şekilde çıkarılmasını sağlar. Bu şekilde, daha küçük cihazlarda bile bu teknik kullanılabilir, çünkü daha az depolama alanı ve hesaplama gücü gerektirir. Bu yöntemin en büyük avantajı, küçük cihazlarda kullanıma uygun olmasıdır, çünkü az depolama alanı ve hesaplama gücü gerektirir. Bu model, belirli Hint dillerinde anlaşılabilir konuşma üretebilir ve yeni bir dil eklenmek istendiğinde sadece dilin işleme kurallarının eklenmesi yeterlidir. Ancak başlangıçta bazı ses geçişlerinin belirli sürelerle yapıldığı için, bazen doğal olmayan sesler üretebilir. Bu durumu düzeltmek için, ses geçişlerinde bazı düzeltmeler yapılabilir.

5.4. İstatistiksel Parametrik Sentezleme (Statistical Parametric Synthesis)

İstatistiksel Parametrik Konuşma Sentezi (SPSS), Metin Konuşma Sentezi (TTS) sistemlerinin bir türüdür ve son zamanlarda oldukça popüler hale gelmiştir. Bu yöntem konuşma sentezi için gereken parametreleri çıkarmak için öğrenme verileri üzerinde istatistiksel analizler kullanır. Özellikle konuşma kayıtlarındaki ses özellikleri ve diğer parametreler üzerinde çalışarak, temel frekans, spektral özellikler gibi ana parametreleri belirler. Bu belirlenen parametreler daha sonra sentezleme anında kullanılarak, konuşma dalgasını oluşturmak için vocoder'lar aracılığıyla işlenir. SPSS'nin temel yaklaşımı, eğitim veri tabanındaki geniş bir konuşma veri setini kullanarak öğrenme işlemi gerçekleştirmesidir. Bu süreçte, konuşma kayıtlarındaki çeşitli özelliklerin istatistiksel analizleri yapılır ve konuşma için temel parametreler bu analizler sonucunda belirlenir. Örneğin, konuşmacının temel frekansı, sesin yüksekliği veya alçaklığı gibi temel ses özellikleri bu analizlerle tespit edilir. SPSS'nin en büyük avantajlarından biri, öğrenme verileri üzerinde çalışırken dil bağımsız olmasıdır. Yani, farklı dillerde konuşma sentezi için kullanılabilir ve genellikle çok dilli konuşma sentezinde de etkilidir. Bununla birlikte, SPSS'nin başlıca zorluklarından biri, eğitim veri setinin kalitesi ve çeşitliliğidir. Veri setinin yeterli olmaması veya çeşitlilik eksikliği, sentezlenen konuşmanın kalitesini etkileyebilir. Ayrıca, sentezleme sürecinde gerçekçi ve doğal bir konuşma üretmek için daha gelişmiş algoritmalar ve veri setleri gerekebilir. Genel olarak, SPSS, konuşma sentezi teknolojilerinin ileri düzeydeki bir örneği olarak kabul edilir ve geniş bir kullanım alanına sahiptir. Ancak, sürekli olarak geliştirilmekte olan bir alandır ve daha fazla veri çeşitliliği ve daha sofistike analiz yöntemleriyle gelecekte daha doğal ve etkileyici konuşma sentezi sistemleri sağlamayı hedefler.

5.4.1. Hidden Markov Model Tabanlı Sentezleme (HMM)

HMM tabanlı sentez, konuşma sentezi üretmek için geliştirilmiş bir tekniktir. Bu teknik, sembolik parametrelerle çalışarak doğal dil işleme biriminden gelen bilgiyi kullanır ve konuşma sentezi üretir. HMM adını verdiğimiz istatistiksel bir modelleme yöntemi kullanılır. Bu sentezleme türünün birim seçimi sentezine göre bazı avantajları vardır. İlk olarak, işitsel hatalardan bağımsız olma özelliği vardır. Sembolik parametreler üzerinden sentezleme işlemi yapıldığı için, işitsel hataların etkisi daha azdır. Bu, daha doğal ve anlaşılabilir bir konuşma sentezi elde edilmesini sağlar. Bir diğer önemli avantajı ise depolama alanının boyutudur. Birim seçimi sentezi, büyük bir ses veri tabanına dayanır ve bu nedenle genellikle çok büyük bir depolama alanı gerektirir. Ancak HMM tabanlı sentez, sembolik parametrelerle çalıştığı için depolama alanı gereksinimi daha küçüktür. Bu, özellikle taşınabilir cihazlarda kullanımı için idealdir çünkü daha az depolama alanı kullanarak bu teknolojinin cihazlara entegre edilmesini mümkün kılar. Sonuç olarak, HMM tabanlı sentez, işitsel hatalardan daha az etkilenen ve daha küçük depolama alanı gerektiren bir sentezleme tekniğidir. Bu özellikleri, daha doğal konuşma sentezleri üretmek ve taşınabilir cihazlarda kullanılabilirlik sağlamak açısından önemli bir avantaj sunar.

5.4.2. Derin Sinir Ağı Tabanlı Sentezleme (DNN)

Derin Sinir Ağları makine öğrenmesi ve yapay zekâ alanında kullanılan bir tür sinir ağıdır. Bu teknoloji, biyolojik sinir ağlarını taklit eden yapay bir modelleme biçimine dayanır. DNN'ler, çok katmanlı yapılara sahip olduğundan "derin" olarak adlandırılır. Her katman, önceki katmanın çıktılarından öğrenilen özellikleri içerir, bu da sistemde karmaşıklığı artırır ve daha yüksek düzeyde öğrenme sağlar. Konuşma işleme alanında, DNN'ler genellikle ses ve konuşma verilerinin karmaşıklığını anlamak, özelliklerini öğrenmek ve bu verileri sentezleme veya tanıma amacıyla kullanmak için kullanılır. DNN'ler özellikle HMM tabanlı konuşma sentezi sistemlerinin kısıtlamalarını aşmak ve daha iyi ses kalitesi elde etmek amacıyla önerilmiştir. DNN tabanlı konuşma sentezi, genellikle daha önceki tekniklere kıyasla daha fazla esneklik sunar. Örneğin, HMM tabanlı sistemler genellikle belirli bir konuşma tarzında kısıtlanırken, DNN'ler çeşitli konuşma tarzları üretebilir. Bu, ses dönüştürme, duygu sentezi ve lehçe gibi uygulamalarda çok daha çeşitli ve ifade dolu konuşmaların elde edilmesine olanak tanır. Ayrıca, uzun-kısa dönemli bellekli tekrarlayan sinir ağları (LSTM-RNN'ler) gibi DNN türleri, özellikle konuşma işleme alanında önemli başarılar elde etmiştir. Bu tür ağlar, önceki bilgileri hatırlayabilme yetenekleri sayesinde ses sentezi ve tanıma süreçlerinde daha etkili ve doğru sonuçlar elde etme potansiyeline sahiptir. Dolayısıyla, DNN tabanlı konuşma sentezi, konuşma işleme alanında gelişmiş ve esnek bir yaklaşım sunarak, önceki tekniklerin sınırlamalarını aşma ve daha kaliteli, çeşitli ve ifade dolu sesler elde etme potansiyeli taşımaktadır.

6. Text To Speech Teknolojisinin Uygulamaları

TTS (Text-to-Speech) teknolojisi günümüzde metin tabanlı verilerin sesli olarak sentezlendiği çok çeşitli uygulama alanlarında kullanılmaktadır. Bu bölümde, TTS teknolojinin farklı sektörlerdeki bazı örnek uygulamalarını ve önemini inceleyeceğiz.

6.1. Eğitim Materyalleri

Text to Speech (TTS) sistemleri okuma problemleri çeken öğrencilere destek sağlama ya da akıllı öğretim sistemleri için yardımcı teknoloji olarak birçok farklı amaçla eğitimde kullanılmıştır. Text to Speech ile verilen eğitimlerin öğrenmeyi kolaylaştırması ya da

zorlaştırması hakkındaki bulgular tutarsız olsa da kolay erişebilirlik ve düşük maliyetli olması nedeniyle eğitim materyallerinde kullanılmaya devam edilmektedir. Bern Üniversitesi Tıp Fakültesi bazı eğitim içerikleri için seslendirme kullanmaktadır. Seslendirme maliyetini düşürmek amacıyla profesyonel seslendirmeleri, TTS ile üretilmiş sesler ile değiştirmeye karar verir. 100.000'den fazla kelimeyi içeren yirmi e-öğrenme modülü, TTS teknolojisi ile seslendirilir ve 20 saatlik ses üretir. Üretilen seslerin etkisi üzerine yapılan çalışmada tıbbi bilgi öğrenimi için TTS teknolojisinin etkili bir yöntem olması ve 48 bin dolar tasarruf edildiği gözlemlenir. TTS sistemleri yabancı dil öğreniminde de birçok fayda sağlamaktadır. Google Translate gibi çeviri yapan uygulamaların TTS sistemleri sayesinde kelimelerin telaffuz ve tonlama gibi özelliklerinin öğrenilmesine olanak tanır. Yapılan araştırmalarda öğrenme sonucu açısından TTS sesiyle öğrenen öğrencilerle insan sesiyle öğrenen öğrenciler arasında anlamlı bir fark bulunmadığı gözlemlenmiştir. Ayrıca TTS teknolojisi görme engelli öğrenciler içinde öğrenme fırsatı sağlamaktadır. Görme engelli öğrenciler eğitim içeriklerinin sesli versiyonlarını birilerinin okumasına ihtiyaç duymadan öğrenebilmelerini sağlar. Metinleri seslendirmek için hazırlanan birçok web sitesi ve tarayıcı eklentisi bulunmaktadır. Ayrıca Microsoft Word gibi bazı uygulamaların kendi seslendirme sistemleri bulunmaktadır.

6.2. Sesli Asistanlar

İnsanlar neredeyse bilgisayarların icat edildiği günden itibaren bilgisayarlarla konuşmak istemişlerdir. Bu istek doğrultusunda sesli asistanlar geliştirilerek TTS teknolojisinin dikkat çeken uygulama alanlarından birisi olmuştur. Metinden konuşma üretme (Text To Speech) teknolojisi ile konuşmadan metin üretme (Speech to Text) teknolojisi birleşerek sesli asistanlar oluşturulmuştur. Sesli asistan uygulamaları Speech To Text teknolojisi ile sürekli olarak ortam sesini dinleyerek anahtar kelime veya kelimelerin çağırılmasını bekler. Anahtar kelime seslendirildiğinde (Örneğin : Hey Siri) aktif hale gelerek kullanıcının komutlarını dinler ve yerine getirir. Kullanıcı ile etkileşim kurmak ve komutlarına cevap vermek için ise TTS teknolojisi kullanılır. Sesli asistanlara örnek olarak Siri, Google Asistan, Amazon Alexa ve Microsoft Cortana verilebilir. Apple'ın geliştirdiği Siri sesli asistan uygulaması en uzun süredir var kullanılan sesli asistandır. 2010 yılında bağımsız bir uygulama olarak piyasaya çıkmış ve 2011'de iOS'a sistemine dahil edilmiştir. Microsoft Siri'den kısa bir süre sonra 2013'te Cortana'yı piyasaya sürmüştür. Amazon, 2014 yılında akıllı ev hoparlörü ile tanıtılan Alexa'yı piyasaya sürdü. Google 2016 yılında akıllı ev hoparlörü ile birlikte duyurduğu Google Assistant'ı tanıttı aynı zamanda Android tabanlı akıllı telefonlardaki Google uygulamasına entegre etti.

6.3. Sesli Kitaplar

Sesli kitaplar, kitapların seslendirmenler ya da TTS teknolojisi ile sesli olarak kaydedilmiş versiyonlarıdır. Son yıllarda büyük popülerlik kazanan sesli kitaplar 1933 yılından itibaren on binlerce sayıda üretilmiştir. Sesli kitaplar genellikle profesyonel aktörler tarafından kaydedilir ve okuyucunun duyguları, diyaloglar ve anlatıdaki çevre ve karakter tanımlamalarını vurgulamak için okuyucu tarafından dramatik seslendirmeleri ve tonlamaları içerir. Modern TTS teknolojileri insan konuşmasına yakın sesler üretse de hikâye anlatımı gibi alanlarda önemli bir performans farkları bulunmaktadır. Performans farklarına rağmen sesli versiyonları olmayan kitap ve e-kitapların TTS teknolojisi ile seslendirilmesi çocuklar, görme engelliler, disleksi ve yeni dil öğrenenler için içeriği erişilebilir kılar. TTS teknolojisini kullanarak e-kitapları seslendiren popüler TTS uygulamaları;

- Google Play Kitaplar

- Amazon Kindle
- Apple Books
- Libby
- Speechify

7. Text To Speech Teknolojisinin Geleceği

Günümüzde kullanılan bazı TTS uygulamalarının gelecekte gelişen teknoloji ile birlikte daha etkin kullanılması beklenmektedir. Örneğin reklamcılık sektöründe basit reklamlar için TTS teknolojisi kullanılmaya başlanmıştır. TTS teknolojisi gelecekte ise reklamcılık alanında kişiselleştirilmiş içeriklerin oluşturulmasını kolaylaştırır. Reklamcılar potansiyel müşterilere özelleştirilmiş sesli reklamlar sunarak daha iyi bir kullanıcı deneyimi ve etkili bir pazarlama stratejisi oluşturabilirler. Yüksek kaliteli bir ses ile kullanıcının adını içeren bir sesli reklam müşterileri etkileyecektir. Ayrıca TTS teknolojisi reklamların çok dilli olmasına da olanak tanır. Her dil için ayrı ayrı dublaj yapmanın oluşturduğu maliyeti ortadan kaldırır. Şu anda gelişmekte olan ve IVR sistemlerinde kullanılarak müşteri hizmetleri ve çağrı merkezi uygulamalarının geliştirilmesine TTS sistemlerinin katkıları bulunmaktadır. Müşteriler, sesli yanıtlar ve talimatlar aracılığıyla etkileşime girebilirler. Günümüzde ilk örnekleri görülen sistemlerin ilerleyen süreçte insanlara gerek kalmayacak şekilde geliştirilmesi muhtemeldir. TTS teknolojisinin gelecekte değiştireceği alanlardan bir diğeri ise oyun sektörüdür. Günümüzde kayıtlar kullanılan karakter seslendirmelerinin yerini ilerleyen süreçte TTS teknolojisinin alabileceği öngörülmektedir. Özelleştirilmiş karaktere verilen isimleri oyunda duymak daha fazla etkileşim ve katılım sağlama potansiyeli sağlar. Bu alanın geleceği, daha fazla oyun geliştiricisinin ve içerik üreticisinin TTS teknolojisinin potansiyelini keşfetmesi ve bu alanda yenilikçi uygulamalar geliştirmesi ile şekillenmeye devam edecek gibi görünmektedir.

TTS teknolojisinin gelişimi, kitapları sesli hale getirirken her karakter için özelleştirilmiş ses tonları, aksanlar ve duygu ifadeleri sunma potansiyeli taşıyor. Bu teknoloji, bir kitaptaki farklı karakterlere benzersiz sesler atayarak, dinleyicilere karakterler arasında net bir ayrım sağlayabilir. Örneğin, bir romanda geçen her karaktere farklı bir ses tonu ve hatta özel bir aksan verilebilir. Böylece, dinleyiciler karakterlerin duygusal durumlarına göre ses tonlarının değişimini deneyimleyebilir, bir filmdeki gibi her karakter için farklı kişiler tarafından seslendirilmiş gibi bir etki oluşturulabilir. Bu, okuyucuların metnin derinliklerine daha fazla dalmasına ve karakterlerle daha yakından bağ kurmasına olanak tanırken, sesli kitap deneyimini daha zengin ve kişiselleştirilmiş hale getirebilir.

Deep Fake ve TTS, yapay zekâ teknolojileri kullanılarak insanların yüz, ses ve hareketlerini taklit eden veya değiştiren uygulamalardır. Bu uygulamalar, eğlence, sanat, eğitim, sağlık ve güvenlik gibi pek çok alanda faydalı olabilirler. Örneğin, Deep Fake ile bir filmdeki oyuncunun yüzünü başka bir ünlüye veya tarihi bir kişiye dönüştürmek mümkündür. TTS ile ise, insan sesini taklit ederek metinleri sesli olarak okutmak veya farklı dillerde konuşmak mümkündür. Bu uygulamalar, gelecekte daha da gelişerek, insanların gerçeklik algısını değiştirebilir ve yeni sanal deneyimler sunabilirler. Ancak, Deep Fake ve TTS uygulamalarının kötüye kullanılması da büyük bir tehlike oluşturmaktadır. Bu uygulamalar, insanların kimliklerini çalmak, yalan haberler yaymak, siyasi veya ideolojik amaçlarla manipüle etmek, şantaj veya taciz yapmak gibi suçlara yol açabilirler. Bu nedenle, bu uygulamaların etik ve yasal sınırlarını belirlemek ve denetlemek çok önemlidir. Ayrıca, insanların bu uygulamaları nasıl ayırt edebileceklerine dair bilinç ve becerilerini geliştirmek de gereklidir. Deep Fake ve TTS uygulamaları, yapay zekâ

teknolojisinin hem faydalarını hem de risklerini gösteren örneklerdir. Bu uygulamaların gelecekte nasıl olacağı, insanların bu teknolojiyi nasıl kullandığına ve yönettiğine bağlıdır.

8. Text To Speech Uygulaması Geliştirme

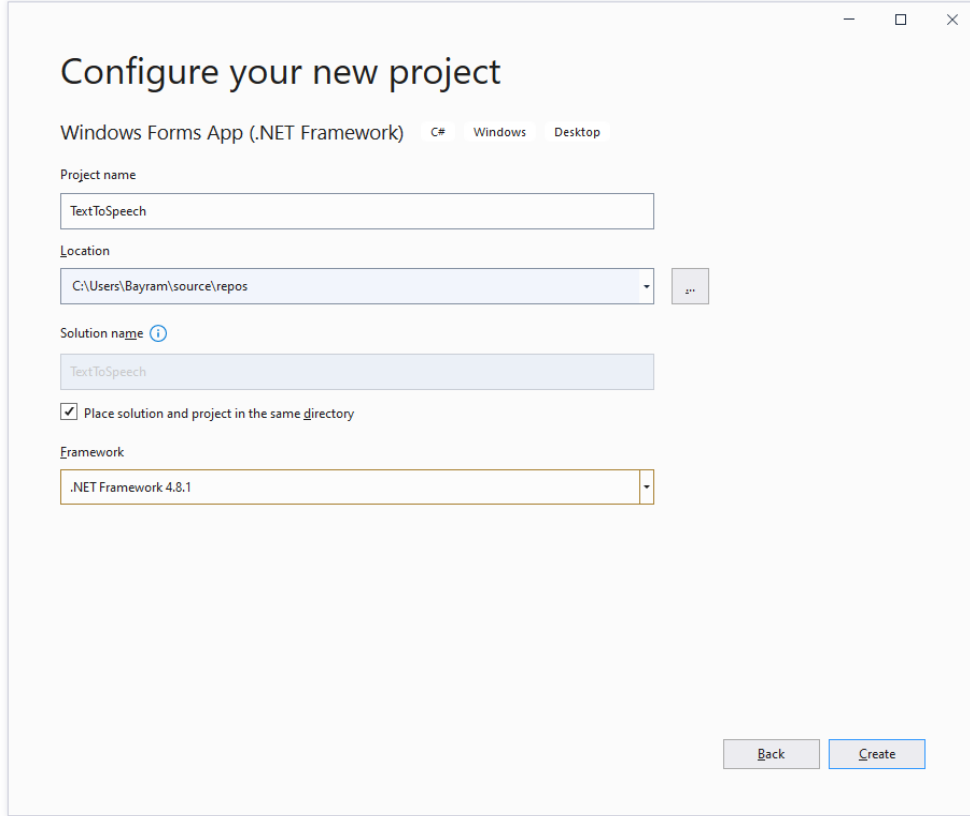
TTS yazılımı yapmak için birçok dil, kütüphane ve SDK kullanılabilir. Örneğin, Python için pyttsx3, C++ için Festival, C# için System.Speech, JavaScript için ResponsiveVoice.JS, Java için FreeTTS gibi birçok kütüphane ve SDK mevcuttur. Bu kütüphanelerin bazılarında Türkçe dil desteği bulunurken bazılarında bulunmamaktadır.

Tablo 2. Text to Speech için kullanılabilecek bazı araçlar

Teknoloji Adı	Teknoloji Açıklama	Geliştirici	Ücretli/Ücretsiz
Google Text-to-Speech	Google'ın metni sese dönüştürme API'si	Google	Ücretsiz (kota ile)
Amazon Polly	Amazon'un metni sese dönüştürme API'si	Amazon	Ücretli
Microsoft Azure Cognitive Services Text to Speech	Microsoft'un metni sese dönüştürme API'si	Microsoft	Ücretli
eSpeak	Ücretsiz ve açık kaynaklı metni sese dönüştürme programı	Richard King	Ücretsiz
MaryTTS	Ücretsiz ve açık kaynaklı metni sese dönüştürme programı	Carnegie Mellon University	Ücretsiz
Ivona	Metni sese dönüştürme API'si	Amazon	Ücretli
IBM Watson Text to Speech	IBM'in metni sese dönüştürme API'si	IBM	Ücretli
Natural Reader	Metni sese dönüştürme programı	Natural Soft	Ücretli
Balabolka	Ücretsiz metni sese dönüştürme programı	CrossOver	Ücretsiz
Acapela	Metni sese dönüştürme API'si	Acapela Group	Ücretli
Festival	Ücretsiz ve açık kaynaklı metni sese dönüştürme programı	University of Edinburgh	Ücretsiz
Tacotron 2	Google tarafından geliştirilmiş metni sese dönüştürme modeli	Google	Ücretsiz
OpenTTS	Ücretsiz ve açık kaynaklı metni sese dönüştürme API'si	OpenAI	Ücretsiz

8.1. Windows Form Uygulaması ve Speech Kütüphanesi ile Masaüstü Uygulaması Geliştirme

C# ile System.Speech kütüphanesi kullanılarak örnek bir uygulama geliştirmek için Visual Studio 2022 versiyonu kullanılmıştır. Kütüphanenin daha kolay anlaşılması amacıyla Windows Forms App (.NET Framework 4.8.1) üzerinden UI tasarımı yapılarak anlatılmıştır. Ses paketlerinin ilgili bilgisayarda kurulu olması halinde herhangi bir internet bağlantısı gerekmemektedir.



Configure your new project

Windows Forms App (.NET Framework) C# Windows Desktop

Project name
TextToSpeech

Location
C:\Users\Bayram\source\repos

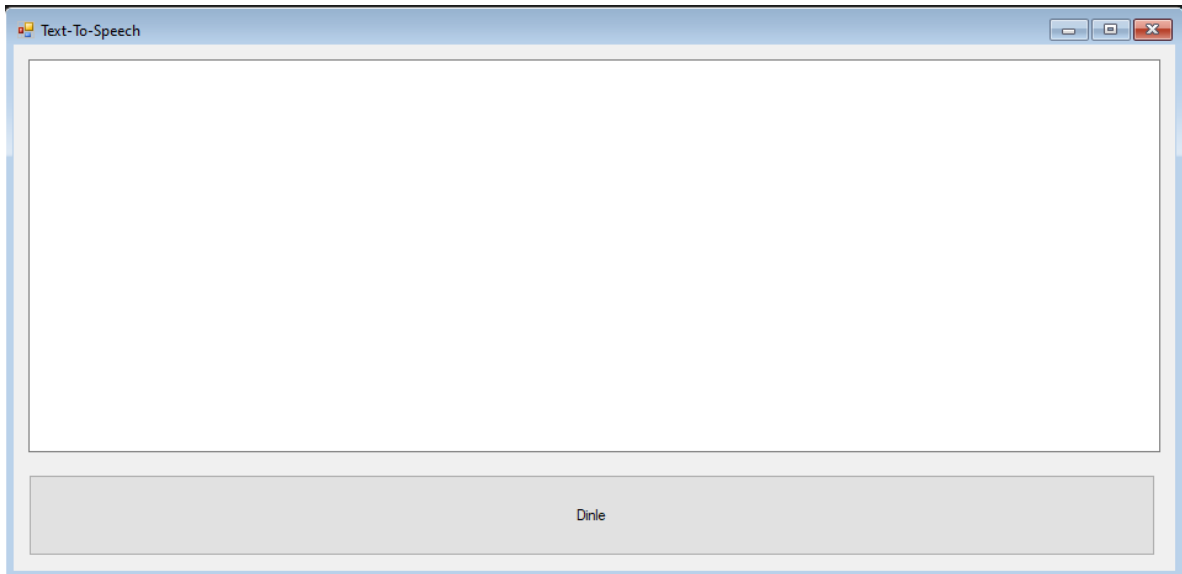
Solution name ⓘ
TextToSpeech

☒ Place solution and project in the same directory

Framework
.NET Framework 4.8.1

Back Create

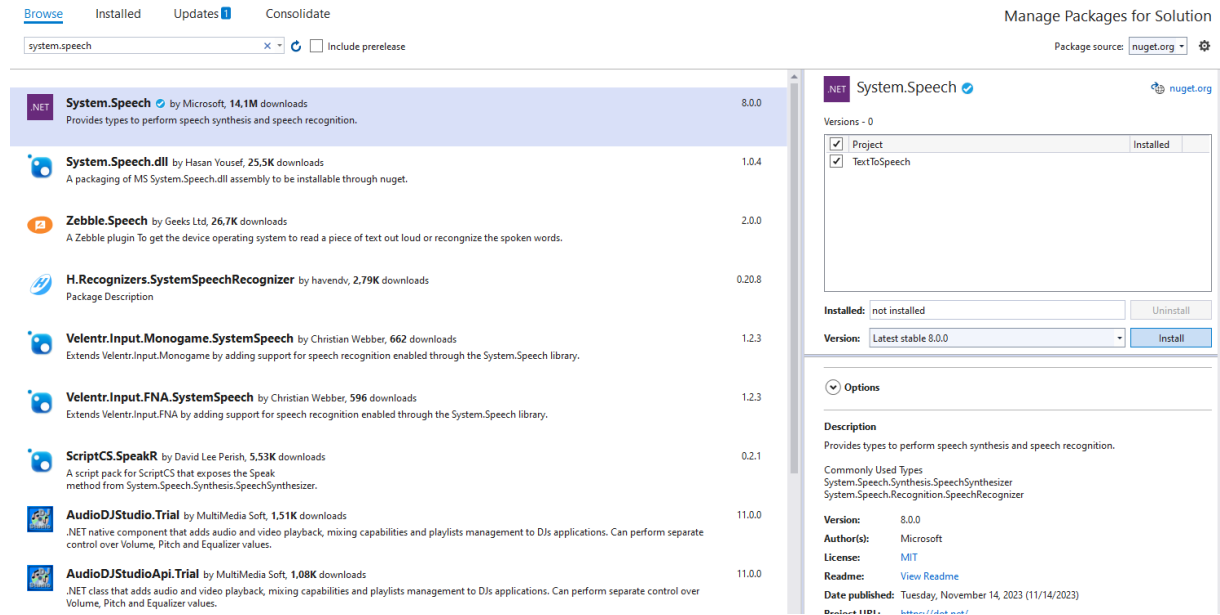
Proje oluşturulduktan sonra ilk adımda forma 1 adet TextBox ve 1 adet Button ekleyerek basit bir UI tasarımı oluşturulmalıdır.



Text-To-Speech

Dinle

Solution Explorer'dan projenizin adına sağ tıklanmalıdır, "Manage NuGet Packages" (NuGet Paket Yöneticisi) seçeneğine tıklanmalıdır. “Browse” (Arama) sekmesine geçerek System.Speech kütüphanesini seçilmelidir.



Açılan sekmeden 8.0.0 versiyonu seçilerek Install butonuna tıklanarak NuGet proje dahil edilmelidir. NuGet ekledikten sonra ilgili formun .cs dosyasına giderek aşağıdaki kütüphaneler eklenmelidir.

```
using System;
using System.Windows.Forms;
using System.Speech.Synthesis;
using System.IO;
```

Kütüphane eklendikten sonra Formun bulunduğu sınıfın içerisine “SpeechSynthesizer” sınıfından türetilmiş bir nesne örneği oluşturulmalıdır.

```
namespace TextToSpeech
{
    3 references
    public partial class Form1 : Form
    {
        SpeechSynthesizer konuşmaSentezleyici = new SpeechSynthesizer();
        1 reference
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

Dinle butonuna tıklandığında TextBox içerisindeki metnin okutulabilmesi için “Dinle” butonuna 2 kere tıklanarak tıklama etkinliği açılır (Click Event). Tıklama etkinliği metodunun içerisinde konuşmaSentezleyici nesnesinin “Speak” metodu çağırılır. Parametre olarak ise formda oluşturduğumuz TextBox’ın içeriği gönderilir.

```

SpeechSynthesizer konuşmaSentezleyici = new SpeechSynthesizer();
1 reference
public Form1()
{
    InitializeComponent();
}

1 reference
private void btn_Dinle_Click(object sender, EventArgs e)
{
    konuşmaSentezleyici.SpeakAsync(txt_Metin.Text);
}

```

Kaynakları kullanması bittiğinde, bellek sızıntısını önlemek ve sistem kaynaklarını serbest bırakmak için “Dispose” metodu kullanılarak kaynaklar serbest bırakılır. Form etkinliklerine (Form Events) gidilerek “FormClosed” etkinliğine iki kere tıklanır. Oluşan metot içerisinde nesne örneğinin “Dispose” metodu çağırılarak kaynaklar serbest bırakılır.

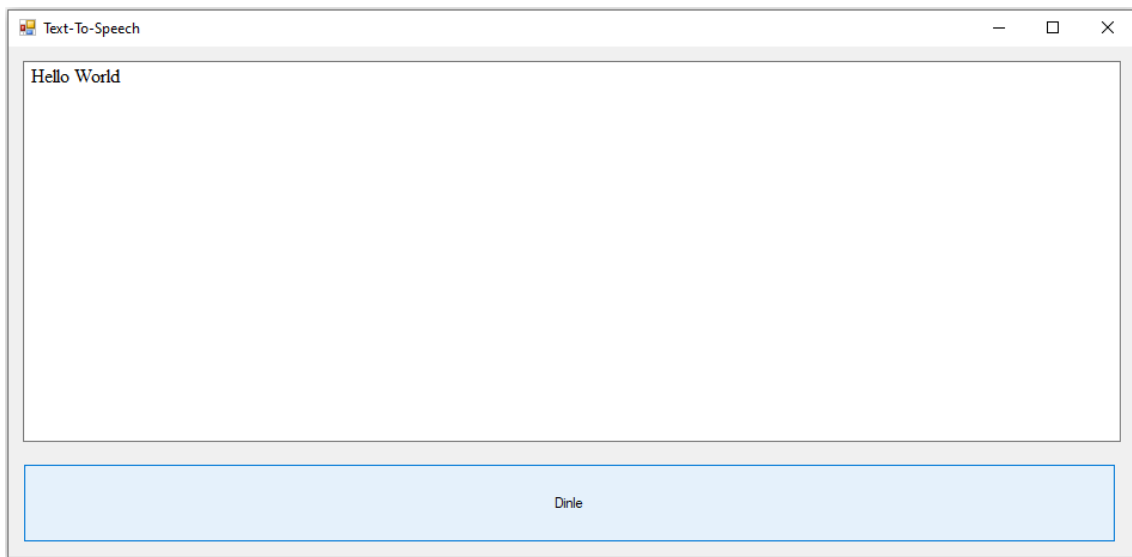
```

1 reference
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
    konuşmaSentezleyici.Dispose();
}

```

1 reference

Uygulamayı çalıştırarak “Dinle” butonuna tıklanıldığında TextBox içerisindeki metinlerin **İngilizce** olarak seslendirilebildiği görülür.



8.1.1. Microsoft Ses Paketleri

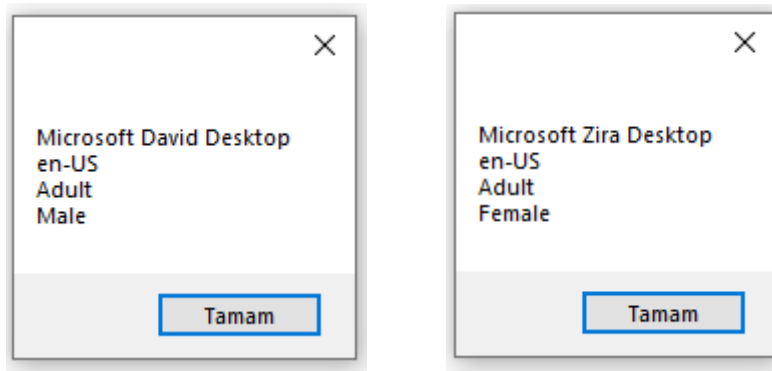
Bilgisayarımızdaki mevcut ses paketlerini inceleyebilmek için formun boş kısmına 2 kere tıklayarak “Form_Load” metodunu getirelim. Metot içerisine aşağıdaki kod parçacığını ekleyelim.

```

public partial class Form1 : Form
{
    SpeechSynthesizer konuşmaSentezleyici = new SpeechSynthesizer();
    1 reference
    public Form1()
    {
        InitializeComponent();
        foreach(InstalledVoice voice in konuşmaSentezleyici.GetInstalledVoices())
        {
            VoiceInfo info = voice.VoiceInfo;
            MessageBox.Show(info.Name + "\n"+info.Culture+"\n"+info.Age+"\n"+info.Gender);
        }
    }
}

```

Foreach döngüsü bilgisayarımızda yüklü olan ses paketleri içerisinde gezerek ses paketlerinin isimlerini, Dil ve aksanlarını, yetişkinlik düzeylerini ve cinsiyetlerini tek tek bize göstermeye yarayacaktır.



Mesaj kutularında “Microsoft David Desktop” ve “Microsoft Zira Desktop” gibi ses paketlerin yüklü olduğunu görebilmekteyiz. Bu ses paketleri uygulamanın çalıştığı bilgisayardaki kurulu olan ses paketleridir. Farklı dilleri, aksanları ve cinsiyetleri ekleyebilmek için işletim sistemine ses paketlerine eklememiz gerekmektedir. Windows “Konuşma Ayarları” sekmesine gidilerek yüklü olan ses paketleri görüntülenir. “Ses Ekle” butonuna tıklanarak gelen menüden eklenilmesi istenen ses paketleri seçilir. Daha sonra “Ekle” butonuna tıklanarak ses paketlerinin yüklenmesi sağlanır. Form tekrar çalıştırıldığında yeni ses paketlerinin de listelendiği görülür.

8.1.2. Seslendirmen Seçimi ve Özelliklerini Belirleme

Seslendirmen özelliklerini belirlemek için “SelectVoiceByHints” metodu kullanılır. Bu metot ile istenen özelliklerde olan ses paketini seçmeyi sağlar. SelectVoiceByHints metoduna parametreler olarak yaş, cinsiyet özellikleri gönderilebilmektedir.

```

public partial class Form1 : Form
{
    SpeechSynthesizer konusmaSentezleyici = new SpeechSynthesizer();
    1 reference
    public Form1()
    {
        InitializeComponent();
        konusmaSentezleyici.SelectVoiceByHints(VoiceGender.Male);
    }

    1 reference
    private void btn_Dinle_Click(object sender, EventArgs e)
    {

```

Cinsiyet özelliği “VoiceGender” ile belirtilir ve 4 adet seçeneği bulunur.

- Male – Erkek sesini belirtir.
- Female – Kadın sesini belirtir.
- Neutral - Cinsiyet ayrımı gözetmeyen bir sesi belirtir.
- NotSet - Ses cinsiyeti belirtilmediğini belirtir.

```

public partial class Form1 : Form
{
    SpeechSynthesizer konusmaSentezleyici = new SpeechSynthesizer();
    1 reference
    public Form1()
    {
        InitializeComponent();
        konusmaSentezleyici.SelectVoiceByHints(VoiceGender.Male, VoiceAge.Adult);
    }

    1 reference
    private void btn_Dinle_Click(object sender, EventArgs e)
    {

```

Yaş özelliği “VoiceAge” ile belirtilir ve 5 adet seçeneği bulunur.

- Child - Bir çocuk sesini belirtir (10 yaş).
- Teen - Genç bir sesi belirtir (15 yaşında).
- Adult - Yetişkin sesini belirtir (30 yaş).
- Senior - Yaşlı bir sesi belirtir (65 yaşında).
- NotSet – Herhangi bir sesi belirtmez.

Ancak bu özellik her zaman doğru çalışmaz. Seslendirmen özelliklerini belirtmek için yapılabilecek en doğru yöntem “SelectVoice” metodunu kullanmaktır. Bir önceki adımda seslendirmenlerin isimlerini listelemeyi görmüştük(Microsoft David Desktop, Microsoft Zira Desktop). “SelectVoice” metodunu kullanmak için seslendirmenin ismine ihtiyaç duyarız.

```

public partial class Form1 : Form
{
    SpeechSynthesizer konusmaSentezleyici = new SpeechSynthesizer();
    1 reference
    public Form1()
    {
        InitializeComponent();
        konusmaSentezleyici.SelectVoice("Microsoft Zira Desktop");
    }

    1 reference
    private void btn_Dinle_Click(object sender, EventArgs e)
    {

```

“SelectVoice” metoduna parametre olarak seslendirmenin ismini göndererek ses paketi seçimi yapılmıştır.

8.1.3. Ses Parametrelerinin Ayarlanması

Öncelikle konuşmacıların listeleneceği ve kullanıcının konuşmacı seçebileceği 1 ComboBox ve 1 Label ekleyerek başlayalım. ComboBox’ın özelliklerinden “DropDownStyle” özelliğini “DropDownList” yapalım. Daha sonra “Form_Load” metodu içerisinde ComboBox’a seslendirmen isimlerini ve dil-aksan bilgilerini ekleyelim.

```

    1 reference
    private void Form1_Load(object sender, EventArgs e)
    {
        foreach(InstalledVoice voice in konusmaSentezleyici.GetInstalledVoices())
        {
            VoiceInfo info = voice.VoiceInfo;
            cmb_Seslendirmen.Items.Add(info.Name + " " + info.Culture );
        }
        cmb_Seslendirmen.SelectedIndex = 0;
    }

    1 reference

```

Form çalıştırıldığında seslendirmen isimlerin ComboBox’ta listelendiği görülecektir. Dinle butonunun “Click” metoduna giderek seslendiriciyi ComboBox’ta ismi seçilen seslendirmen olarak düzenleyelim. Dil-Aksan bölümünü ve boşluğu silmek için “Substring” metodundan faydalanabiliriz.

```

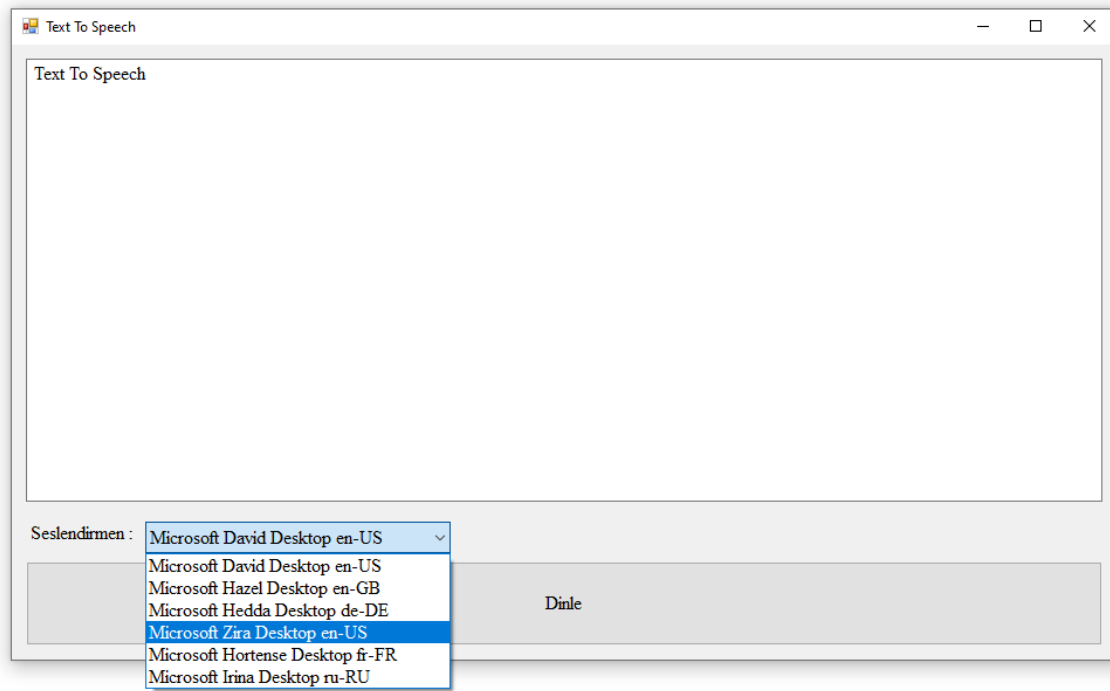
    }
    InitializeComponent();
}

1 reference
private void btn_Dinle_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.SelectVoice(cmb_Seslendirmen.Text.Substring(0, cmb_Seslendirmen.Text.Length - 6));
    konusmaSentezleyici.Speak(txt_Metin.Text);
}

1 reference
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{

```

Artık girilen metnin seçilen seslendirmen tarafından seslendirildiği görülecektir. ComboBox’tan herhangi bir seçim yapılmadığında program hata verecektir. Bu durumu engellemek için “Form_Load” metodu içerisinde ComboBox’ın “SelectedIndex” değerine 0 ataması yapılabilir. Ya da “if” yardımıyla ComboBox’ın değeri kontrol edilerek uyarı mesajı gönderilebilir.



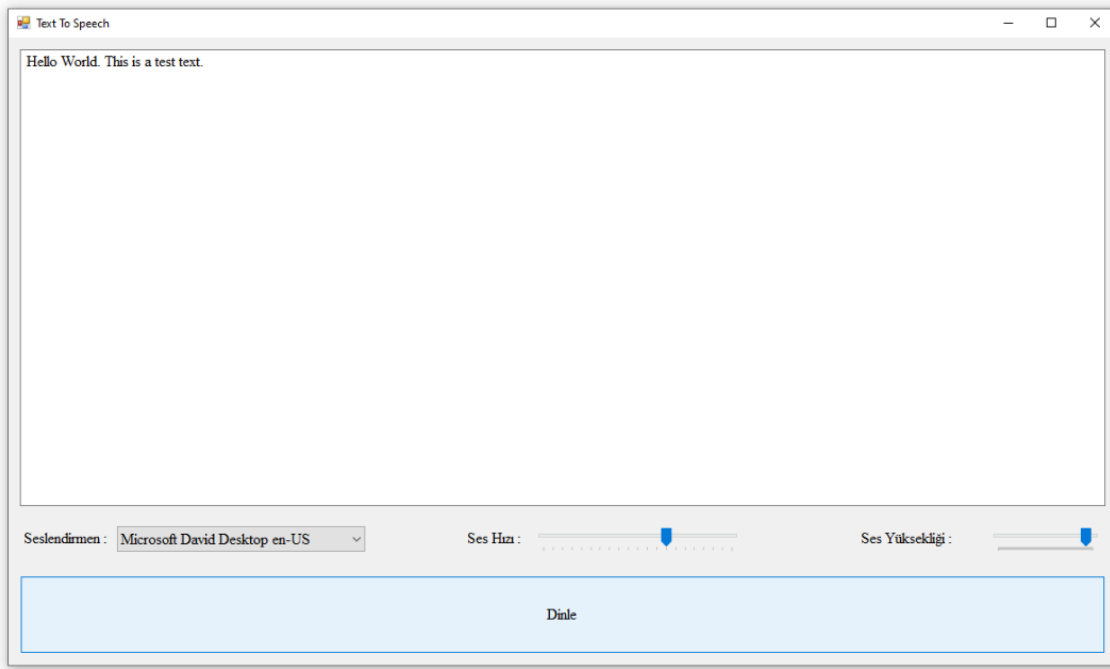
Seslendirme hızını kontrol edebilmek için 1 adet TrackBar ve 1 adet Label ekleyelim. TrackBar'ın "Minimum" özelliğini -10, "Maximum" özelliğini 10 yapalım. Daha sonra ses yüksekliğini ayarlayabilmek için 1 adet TrackBar ve 1 adet Label daha ekleyelim. Bu TrackBar'ın "Maximum" özelliği ise 100 değerinde ve "Minimum" özelliği 0 değerinde, "Value" özelliği 100 olmalıdır.

Dinle butonunun "Click" metodu içerisinde, ses hızını düzenleyebilmek için "Rate" değerini çağırılır ve Ses Hızı TrackBar'ından aldığımız değer atanır. Aynı şekilde ses yüksekliğini ayarlayabilmek için "Volume" değeri çağırılır ve Ses Yüksekliği TrackBar'ından alınan değer atanır. Ses dinleme işleminin asenkron çalışması için "Speak" metodunu "SpeakAsync" metodu ile değiştirin. Ayrıca metodun başına konuşma sentezleyici nesnesine "SpeakAsyncCancelAll" metodunu ekleyerek dinle butonuna tekrar tıklandığında önceki konuşma sonlandırılarak yeni konuşma üretilir.

```
1 reference
private void btn_Dinle_Click(object sender, EventArgs e)
{
    konuşmaSentezleyici.SpeakAsyncCancelAll();
    konuşmaSentezleyici.SelectVoice(cmb_Seslendirme.Text.Substring(0, cmb_Seslendirme.Text.Length - 6));
    konuşmaSentezleyici.Rate = trb_SesHizi.Value;
    konuşmaSentezleyici.Volume = trb_SesYuksekligi.Value;
    konuşmaSentezleyici.SpeakAsync(txt_Metin.Text);
}

1 reference
private void Form1_FormClosing(object sender, FormClosingEventArgs e)
{
}
```

TextBox'a girilen metnin istenen ses yüksekliği ve seslendirme hızında çalıştığı görülecektir.



8.1.4. Sesi Duraklatma, Sesi Devam Ettirme ve Sesi Kaydetme Butonlarının Eklenmesi

Sesi duraklatma ve devam ettirme için 2 adet Button ekleyelim. Duraklatma butonunun “Click” metoduna giderek konuşma sentezleyici nesnesinden “Pause” metodunu çağıralım. Devam et butonunun “Click” metoduna gidelim ve konuşma sentezleyici nesnesinden “Resume” metodunu çağıralım.

```
1 reference
private void btn_Duraklat_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.Pause();
}

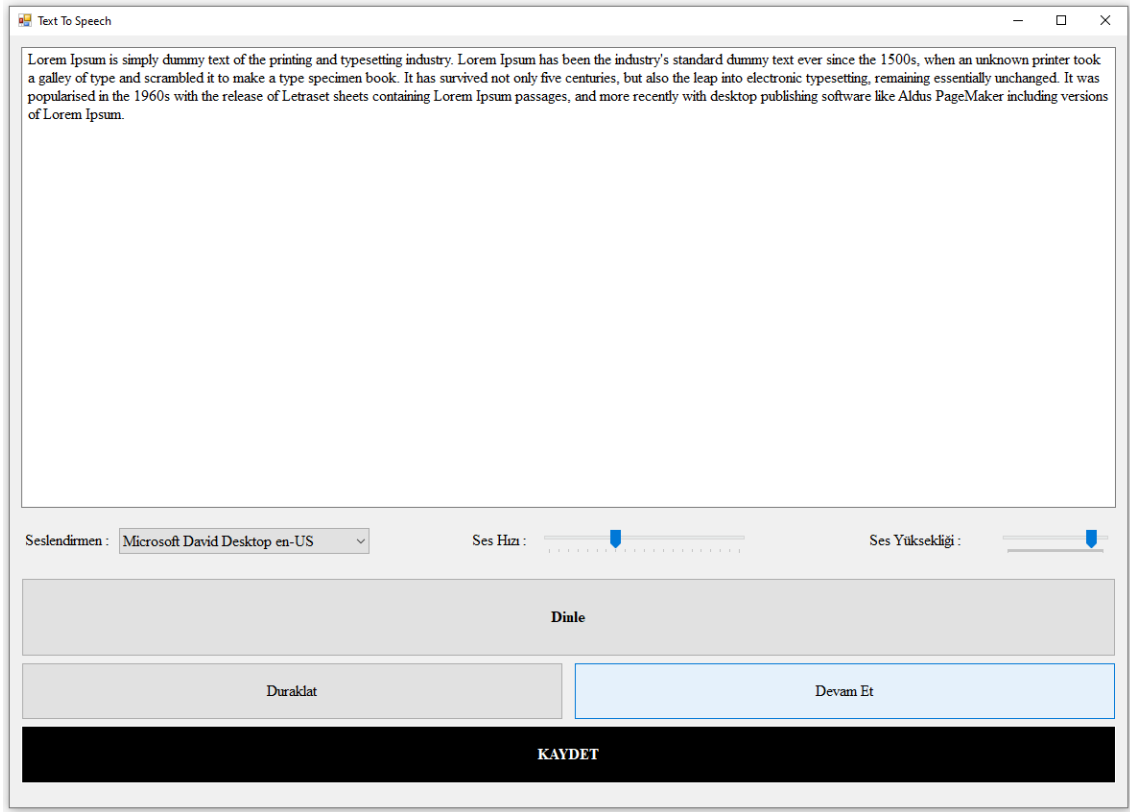
1 reference
private void btn_Devam_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.Resume();
}
```

Yukarıdaki kod parçacıkları eklendiğinde Duraklat ve Devam Et butonlarının çalıştığı görülecektir. Kaydetme seçeneği sunmak için Form’a 1 adet buton ekleyelim. Butona çift tıklayarak “Click” metoduna gidelim. Kaydetme işlemi yapabilmek için öncelikle aktif konuşma kapatılmalıdır. Hiç dinlenmeden kaydedilme ihtimaline karşı “Rate” ve “Volume” değerleri tekrar atanmalıdır. Dosyanın kaydedileceği konum belirlenmelidir. Daha sonra “SetOutputToWaveStream” metodu ile ses çıkış cihazının dosya olduğu belirtilmelidir. Metin dosyanın içerisine okunduktan sonra ses çıkış cihazı tekrar varsayılan hale getirilir.

```

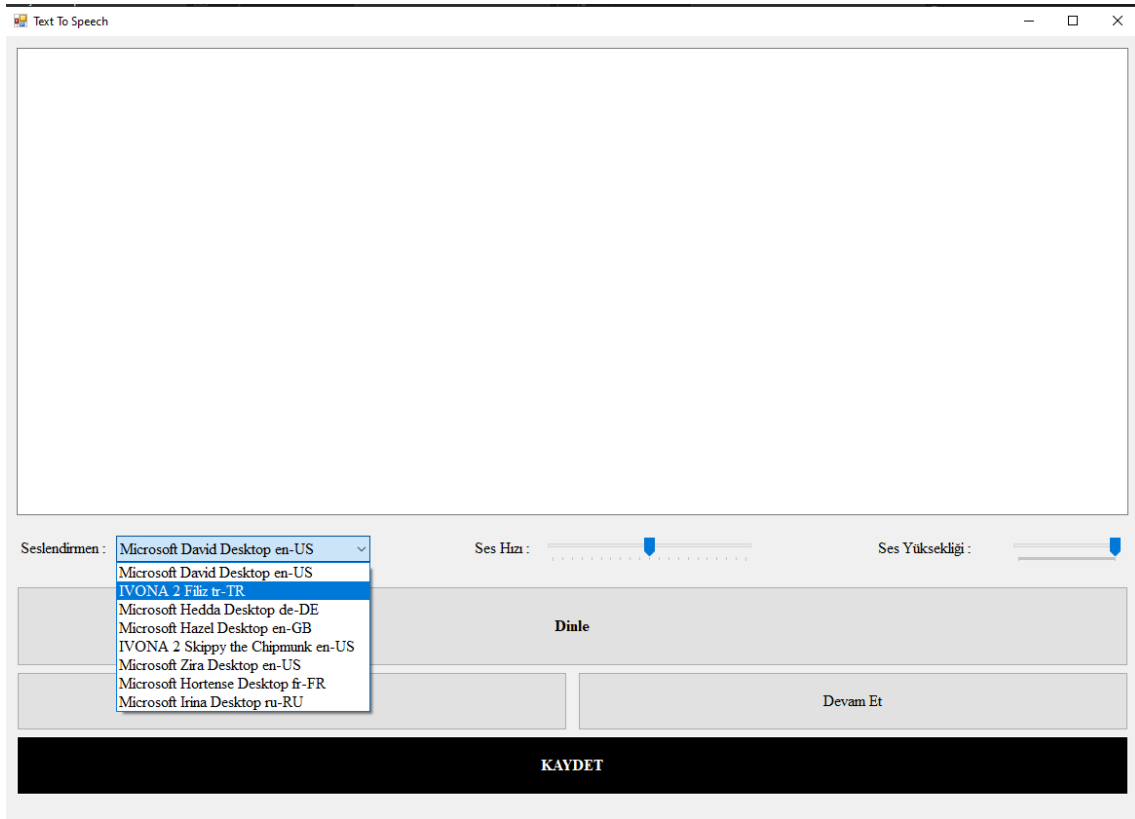
1 reference
private void btn_Kaydet_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.SpeakAsyncCancelAll();
    konusmaSentezleyici.Rate = trb_SesHizi.Value;
    konusmaSentezleyici.Volume = trb_SesYuksekligi.Value;
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "Waveform Audio File Format | *.wav";
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        using (FileStream fileStream = new FileStream(saveFileDialog.FileName, FileMode.Create, FileAccess.Write))
        {
            konusmaSentezleyici.SetOutputToWaveStream(fileStream);
            konusmaSentezleyici.Speak(txt_Metin.Text);
            konusmaSentezleyici.SetOutputToDefaultAudioDevice();
        }
    }
}
}

```



8.1.5. Harici Ses Paketlerinin Eklenmesi

Şimdiye kadar tasarladığımız programda Microsoft'un Türkçe ses paketi (Microsoft Tolga kullanılamıyor) olmamasından dolayı Türkçe seçeneği sunamamıştık. Ancak kendi ses paketlerimizi üretebileceğimiz gibi hali hazırda şirketlerin sunduğu ses paketlerini kullanabiliriz. Türkçe dil seçeneği sunan en popüler uygulamalar arasında IVONA bulunuyor. IVONA, kullanıcılarına yüksek kaliteli ses sentezi sağlayan ve birçok dilde ses paketleri sunan bir platformdur. İlgili bilgisayara, gerekli kurulumlar yapıldığında "Seslendirme" seçeneklerine Türkçe dil paketinin de geldiği görülecektir.



8.1.6. Projenin Kodları

“Form1.cs” dosyasının kodları aşağıdaki gibidir.

```
using System;
using System.Windows.Forms;
using System.Speech.Synthesis;
using System.IO;

namespace TextToSpeech
{
    public partial class Form1 : Form
    {
        SpeechSynthesizer konuşmaSentezleyici = new SpeechSynthesizer();
        public Form1()
        {
            InitializeComponent();
        }

        private void btn_Dinle_Click(object sender, EventArgs e)
        {
            konuşmaSentezleyici.SpeakAsyncCancelAll();
            konuşmaSentezleyici.SelectVoice(cmb_Seslendirme.Text.Substring(0,
            cmb_Seslendirme.Text.Length - 6));
            konuşmaSentezleyici.Rate = trb_SesHizi.Value;
            konuşmaSentezleyici.Volume = trb_SesYuksekligi.Value;
            konuşmaSentezleyici.SpeakAsync(txt_Metin.Text);
        }

        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            konuşmaSentezleyici.Dispose();
        }
    }
}
```

```

private void btn_Duraklat_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.Pause();
}

private void btn_Devam_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.Resume();
}

private void btn_Kaydet_Click(object sender, EventArgs e)
{
    konusmaSentezleyici.SpeakAsyncCancelAll();
    konusmaSentezleyici.Rate = trb_SesHizi.Value;
    konusmaSentezleyici.Volume = trb_SesYuksekligi.Value;
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "Waveform Audio File Format | *.wav";
    if (saveFileDialog.ShowDialog() == DialogResult.OK)
    {
        using (FileStream fileStream = new
FileStream(saveFileDialog.FileName, FileMode.Create, FileAccess.Write))
        {
            konusmaSentezleyici.SetOutputToWaveStream(fileStream);
            konusmaSentezleyici.Speak(txt_Metin.Text);
            konusmaSentezleyici.SetOutputToDefaultAudioDevice();
        }
    }
}
}
}
}
}

```

“Form1.Designer.cs” dosyasının içeriği aşağıdaki gibidir.

```

namespace TextToSpeech
{
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise, false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
        }
    }
}

```

```

        base.Dispose(disposing);
    }

    #region Windows Form Designer generated code
    /// <summary>
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    /// </summary>
    private void InitializeComponent()
    {
        this.txt_Metin = new System.Windows.Forms.TextBox();
        this.btn_Dinle = new System.Windows.Forms.Button();
        this.cmb_Seslendirmen = new System.Windows.Forms.ComboBox();
        this.label1 = new System.Windows.Forms.Label();
        this.trb_SesHizi = new System.Windows.Forms.TrackBar();
        this.label2 = new System.Windows.Forms.Label();
        this.trb_SesYuksekligi = new System.Windows.Forms.TrackBar();
        this.label3 = new System.Windows.Forms.Label();
        this.btn_Duraklat = new System.Windows.Forms.Button();
        this.btn_Devam = new System.Windows.Forms.Button();
        this.btn_Kaydet = new System.Windows.Forms.Button();
        ((System.ComponentModel.ISupportInitialize)(this.trb_SesHizi)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.trb_SesYuksekligi)).BeginInit();
        this.SuspendLayout();
        //
        // txt_Metin
        //
        this.txt_Metin.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
        this.txt_Metin.Location = new System.Drawing.Point(12, 12);
        this.txt_Metin.Multiline = true;
        this.txt_Metin.Name = "txt_Metin";
        this.txt_Metin.Size = new System.Drawing.Size(1141, 480);
        this.txt_Metin.TabIndex = 0;
        //
        // btn_Dinle
        //
        this.btn_Dinle.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
        this.btn_Dinle.Location = new System.Drawing.Point(12, 565);

```

```

this.btn_Dinle.Name = "btn_Dinle";
this.btn_Dinle.Size = new System.Drawing.Size(1141, 82);
this.btn_Dinle.TabIndex = 1;
this.btn_Dinle.Text = "Dinle";
this.btn_Dinle.UseVisualStyleBackColor = true;
this.btn_Dinle.Click += new System.EventHandler(this.btn_Dinle_Click);
//
// cmb_Seslendirmen
//
this.cmb_Seslendirmen.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;
this.cmb_Seslendirmen.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
this.cmb_Seslendirmen.FormattingEnabled = true;
this.cmb_Seslendirmen.Location = new System.Drawing.Point(114, 513);
this.cmb_Seslendirmen.Name = "cmb_Seslendirmen";
this.cmb_Seslendirmen.Size = new System.Drawing.Size(261, 27);
this.cmb_Seslendirmen.TabIndex = 2;
//
// label1
//
this.label1.AutoSize = true;
this.label1.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
this.label1.Location = new System.Drawing.Point(12, 516);
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(96, 19);
this.label1.TabIndex = 3;
this.label1.Text = "Seslendirmen :";
//
// trb_SesHizi
//
this.trb_SesHizi.Location = new System.Drawing.Point(549, 513);
this.trb_SesHizi.Minimum = -10;
this.trb_SesHizi.Name = "trb_SesHizi";
this.trb_SesHizi.Size = new System.Drawing.Size(225, 45);
this.trb_SesHizi.TabIndex = 4;
//
// label2
//

```

```

        this.label2.AutoSize = true;

        this.label2.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));

        this.label2.Location = new System.Drawing.Point(478, 516);

        this.label2.Name = "label2";

        this.label2.Size = new System.Drawing.Size(65, 19);

        this.label2.TabIndex = 5;

        this.label2.Text = "Ses Hızı :";

        //

        // trb_SesYuksekligi

        //

        this.trb_SesYuksekligi.Location = new System.Drawing.Point(1027, 513);

        this.trb_SesYuksekligi.Maximum = 100;

        this.trb_SesYuksekligi.Name = "trb_SesYuksekligi";

        this.trb_SesYuksekligi.RightToLeft = System.Windows.Forms.RightToLeft.No;

        this.trb_SesYuksekligi.Size = new System.Drawing.Size(126, 45);

        this.trb_SesYuksekligi.TabIndex = 6;

        this.trb_SesYuksekligi.Value = 100;

        //

        // label3

        //

        this.label3.AutoSize = true;

        this.label3.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));

        this.label3.Location = new System.Drawing.Point(892, 516);

        this.label3.Name = "label3";

        this.label3.Size = new System.Drawing.Size(104, 19);

        this.label3.TabIndex = 7;

        this.label3.Text = "Ses Yüksekliği :";

        //

        // btn_Duraklat

        //

        this.btn_Duraklat.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));

        this.btn_Duraklat.Location = new System.Drawing.Point(12, 653);

        this.btn_Duraklat.Name = "btn_Duraklat";

        this.btn_Duraklat.Size = new System.Drawing.Size(565, 60);

        this.btn_Duraklat.TabIndex = 8;

        this.btn_Duraklat.Text = "Duraklat";

```

```

this.btn_Duraklat.UseVisualStyleBackColor = true;
this.btn_Duraklat.Click += new System.EventHandler(this.btn_Duraklat_Click);
//
// btn_Devam
//
this.btn_Devam.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
this.btn_Devam.Location = new System.Drawing.Point(588, 653);
this.btn_Devam.Name = "btn_Devam";
this.btn_Devam.Size = new System.Drawing.Size(565, 60);
this.btn_Devam.TabIndex = 9;
this.btn_Devam.Text = "Devam Et";
this.btn_Devam.UseVisualStyleBackColor = true;
this.btn_Devam.Click += new System.EventHandler(this.btn_Devam_Click);
//
// btn_Kaydet
//
this.btn_Kaydet.BackColor = System.Drawing.SystemColors.ActiveCaptionText;
this.btn_Kaydet.FlatStyle = System.Windows.Forms.FlatStyle.Flat;
this.btn_Kaydet.Font = new System.Drawing.Font("Times New Roman", 12F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(162)));
this.btn_Kaydet.ForeColor = System.Drawing.SystemColors.ButtonHighlight;
this.btn_Kaydet.Location = new System.Drawing.Point(12, 719);
this.btn_Kaydet.Name = "btn_Kaydet";
this.btn_Kaydet.Size = new System.Drawing.Size(1141, 60);
this.btn_Kaydet.TabIndex = 10;
this.btn_Kaydet.Text = "KAYDET";
this.btn_Kaydet.UseVisualStyleBackColor = false;
this.btn_Kaydet.Click += new System.EventHandler(this.btn_Kaydet_Click);
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(1165, 804);
this.Controls.Add(this.btn_Kaydet);
this.Controls.Add(this.btn_Devam);
this.Controls.Add(this.btn_Duraklat);
this.Controls.Add(this.label3);

```

```

        this.Controls.Add(this.trb_SesYuksekligi);
        this.Controls.Add(this.label2);
        this.Controls.Add(this.trb_SesHizi);
        this.Controls.Add(this.label1);
        this.Controls.Add(this.cmb_Seslendirmen);
        this.Controls.Add(this.btn_Dinle);
        this.Controls.Add(this.txt_Metin);
        this.Name = "Form1";
        this.Text = "Text To Speech";
        this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Form1_FormClosing);
        ((System.ComponentModel.ISupportInitialize)(this.trb_SesHizi)).EndInit();
        ((System.ComponentModel.ISupportInitialize)(this.trb_SesYuksekligi)).EndInit();
        this.ResumeLayout(false);
        this.PerformLayout();
    }
    #endregion
    private System.Windows.Forms.TextBox txt_Metin;
    private System.Windows.Forms.Button btn_Dinle;
    private System.Windows.Forms.ComboBox cmb_Seslendirmen;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.TrackBar trb_SesHizi;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.TrackBar trb_SesYuksekligi;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.Button btn_Duraklat;
    private System.Windows.Forms.Button btn_Devam;
    private System.Windows.Forms.Button btn_Kaydet;
}
}

```

8.2. ASP.NET Core 6.0 MVC ve Azure Speech Studio ile Web Uygulaması Geliştirme

ASP.NET Core MVC (.NET Core 6.0) çerçevesini kullanarak Azure Speech Studio ile bir web uygulaması geliştirme sürecinde Visual Studio 2022 tercih edilmiştir. Bu uygulama, API kullanımının öncelikli olarak ele alındığı bir yaklaşımla tasarlanmıştır. Bu sebepten ötürü nesne yönelimli programlama (OOP), Model-View-Controller (MVC) mimarisi ve SOLID prensipleri göz ardı edilmiştir.

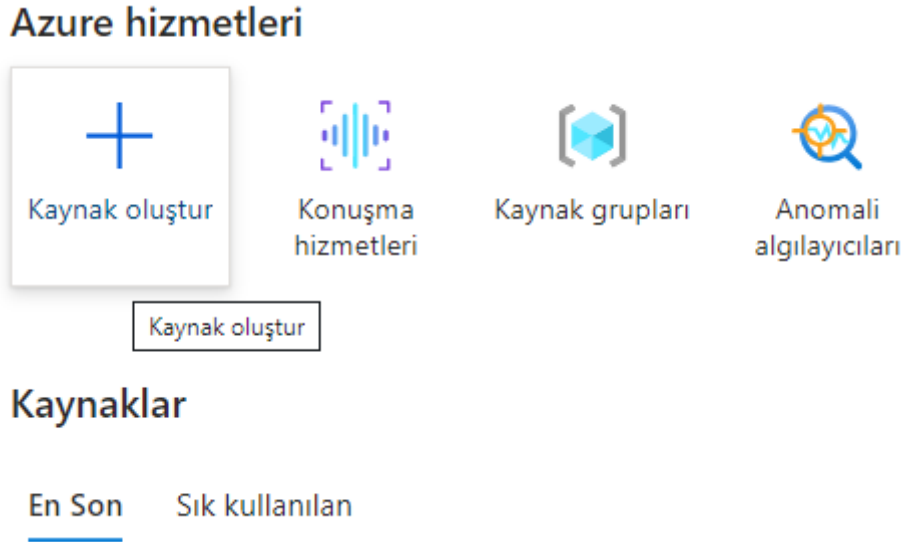
Gerçek dünya projelerinde, iş süreçleri genellikle karmaşıktır ve basit örneklerde bulunmayan çeşitli değişkenler içerebilir. Bu projeler genellikle farklı paydaşların ve departmanların bir araya gelmesini gerektirir, bu da iletişim ve iş birliği gereksinimlerini artırır.

Bu nedenle, gerçek projelerdeki işlemler daha karmaşık hale gelebilir çünkü birden fazla değişken arasında denge sağlama, kaynakları etkin bir şekilde yönetme ve beklenmedik sorunlarla başa çıkma gereksinimleri ortaya çıkar.

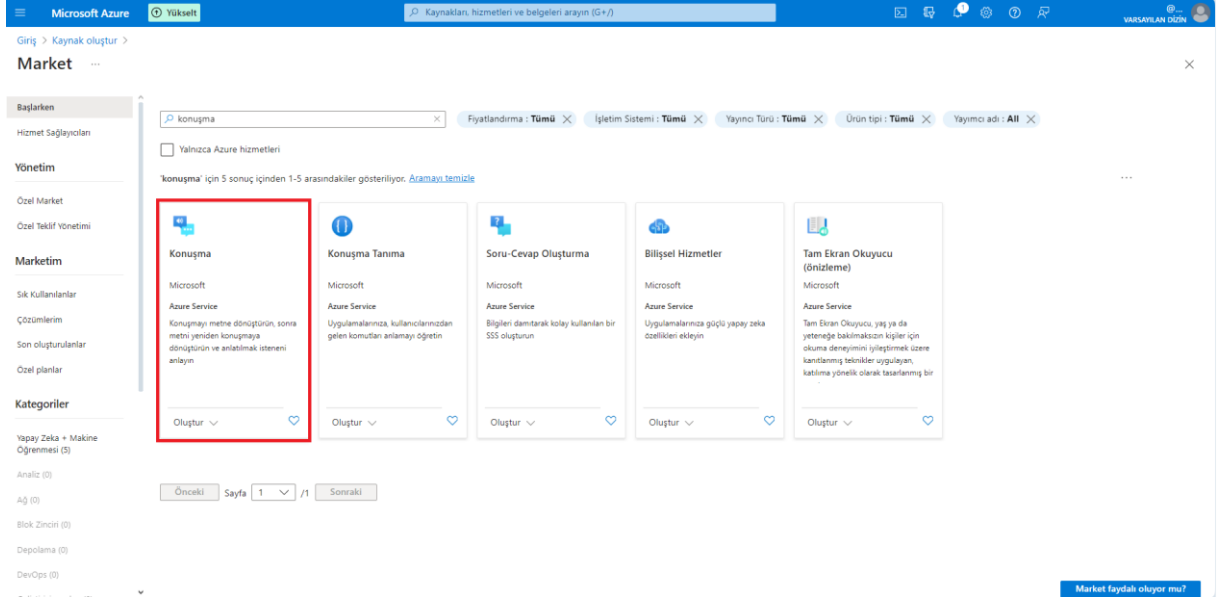
API'lere erişim için internet bağlantısı ve ilgili platforma üyelik gerekmektedir. Ayrıca, platformlar bazı API hizmetlerini ücretsiz sunarken, diğer hizmetler için ücret talep edebilirler. Bu durumda, projenin maliyetleri ve erişim koşulları dikkate alınmalıdır.

8.2.1. Microsoft Azure ile Konuşma Kaynağı Oluşturma

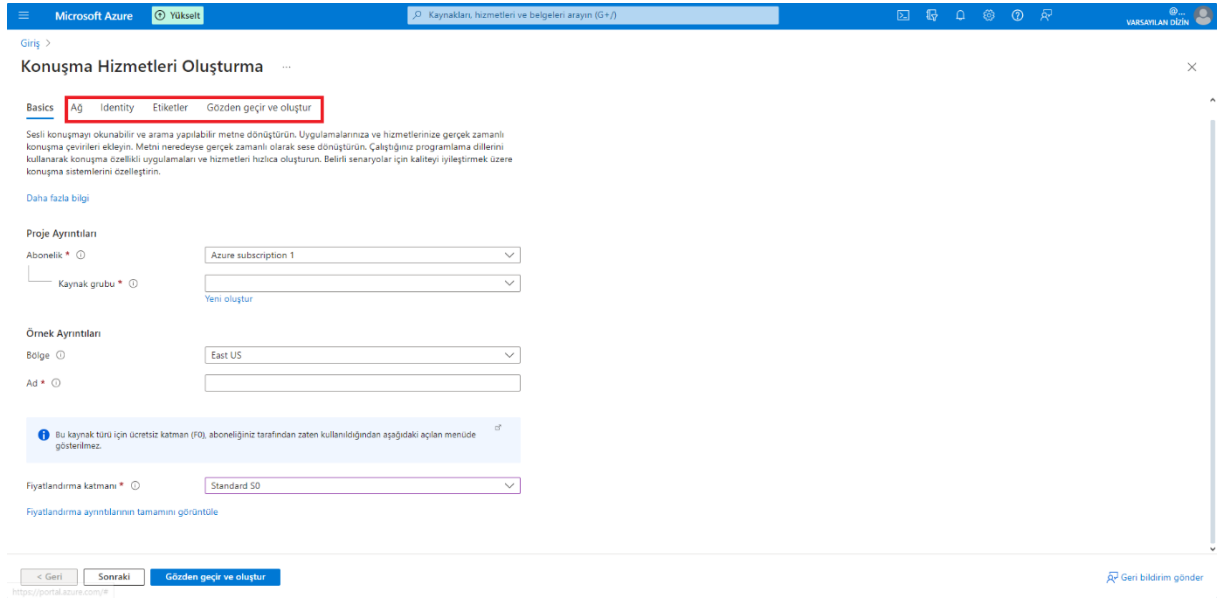
API'ye erişim sağlamak için, başlangıçta Microsoft Azure üzerinde bir konuşma kaynağı oluşturulması gerekmektedir. Bu işlem Azure platformunda bir hesap oluşturulması ve ilgili hizmetin etkinleştirilmesi ile gerçekleştirilir. İlgili kaynağı oluşturmak için, öncelikle <https://azure.microsoft.com/tr-tr/> adresinden bir üyelik oluşturmalıdır. Ardından Azure portalına giriş yaparak "Azure hizmetleri" sekmesine ulaşmalı ve burada "Kaynak oluştur" butonuna tıklamalıdır.



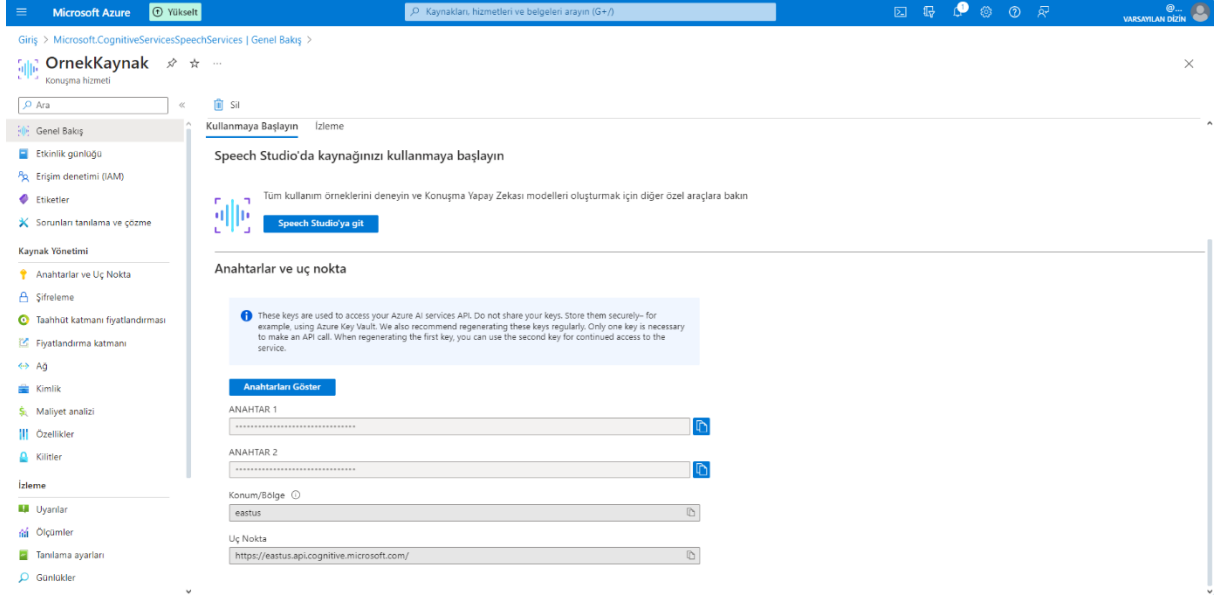
Butona tıklanıldığında, Azure portalında bulunan hizmetlerin listelendiği bir ekran açılacaktır. Kullanıcılar bu ekran aracılığıyla farklı Azure hizmetlerine erişebilir ve bunları etkinleştirebilirler. Arama kutusuna "konuşma" terimi yazılarak "Konuşma" kaynağı görülecektir. Ardından "Oluştur" butonuna tıklayarak kaynak oluşturma sayfasına yönlendirileceklerdir.



Proje ayrıntıları sekmesi altında, kullanıcılar abonelik ve kaynak grubu seçmelidir. Eğer henüz bir kaynak grubu bulunmuyorsa, "Yeni Oluştur" butonuna tıklanarak yeni bir kaynak grubu oluşturulmalıdır. Örnek ayrıntıları sekmesi altında ise kaynak bölgesi seçilmeli ve kaynak ismi belirtilmelidir. Fiyatlandırma Katmanı sekmesinde, projenin gereksinimlerine uygun olarak "Standart" veya "Ücretsiz" seçeneklerinden biri tercih edilmelidir. İhtiyaca bağlı olarak, konuşma kaynağı diğer sekmeler aracılığıyla özelleştirilebilir. Bu adımlar, kullanıcıların projelerinin gereksinimlerine uygun şekilde kaynakları yapılandırma ve anahtar bilgilerine erişim sağlama için gereklidir.



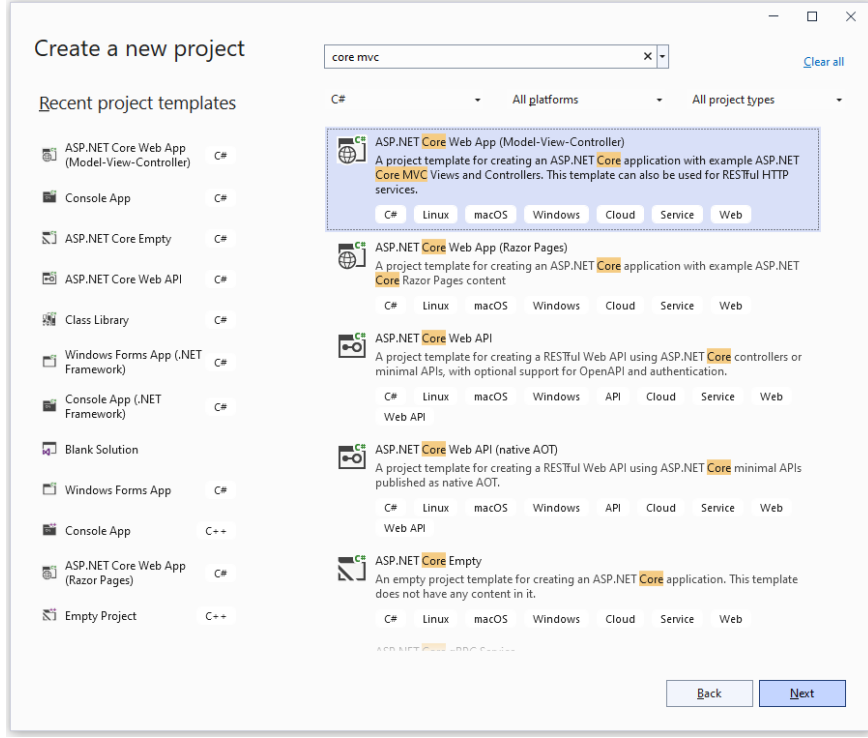
İlgili adımlar tamamlandığında, kaynak oluşturulacaktır. "Kaynağa git" butonuna tıklanarak, Kaynağın Temel Parçaları ve Kaynak Anahtarlarının yer aldığı sayfaya geçiş yapılabilir. Bu geçiş, kullanıcıların oluşturulan kaynağın temel yapılandırma ve anahtar bilgilerine erişim sağlama için gereklidir.



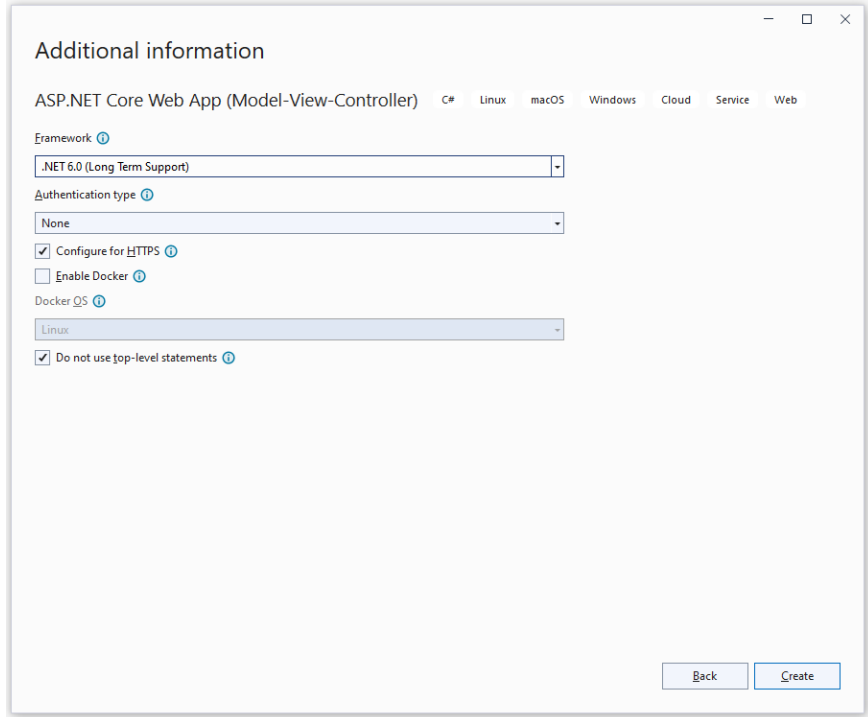
"Speech Studio'ya git" butonuna tıklayarak kaynağın kapsamındaki hizmetlere erişebilirsiniz. Bu hizmetler arasında senaryoya göre konuşma özellikleri, konuşmayı metne dönüştürme, metin okuma ve sesli yardım gibi fonksiyonlar bulunmaktadır. Bu butona tıklama işlemi, kaynakla ilişkili hizmetlere yönlendirilmesini sağlar. Bu hizmetler, metin veya ses tabanlı işlemler gerçekleştirmek için gereken özellikleri ve fonksiyonları içerir. Kullanıcılar bu sayede, kaynağın sunduğu potansiyeli daha yakından inceleyebilir ve projelerinde kullanacakları özellikleri belirleyebilirler.

8.2.2. ASP.NET Core MVC Projesi Oluşturma

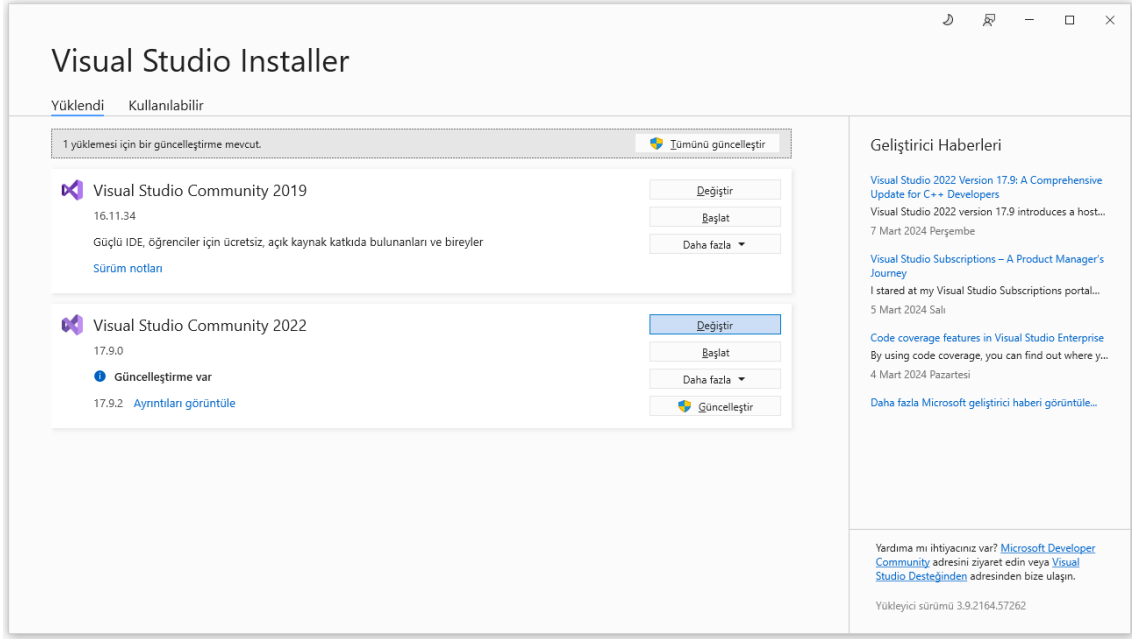
Visual Studio 2022'nin kurulumunun ardından, yeni bir proje oluşturmak için "Create a new project" butonuna tıklanarak proje şablonu seçme bölümüne erişilir. Arama kutusuna "core mvc" yazarak karşınıza çıkan şablonlar arasından "ASP.NET Core Web App (Model-View-Controller)" seçeneğini bulmalı ve bu şablonu seçmelisiniz. Ayrıca dilin C# olarak belirlenmesine dikkat ediniz.



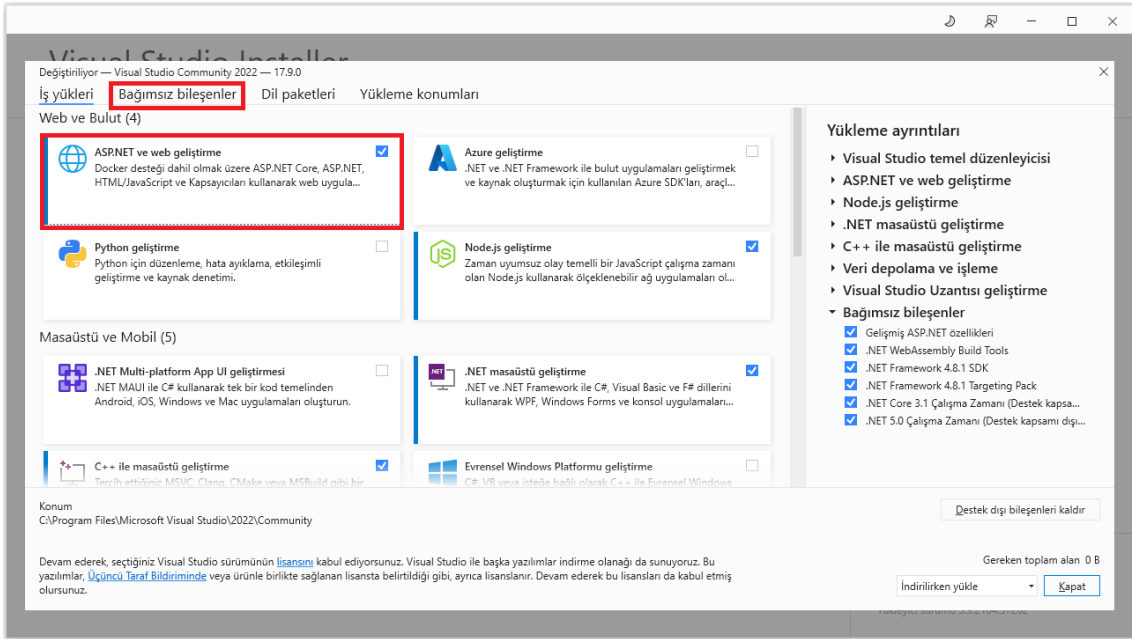
Şablon belirlendikten sonra, “Next” butonuna tıklanılır. Gelen sayfada “Framework” tercihi istenecektir “.NET 6.0” seçildikten sonra “Create” butonuna tıklanarak proje oluşturulur.



Eğer "ASP.NET Core Web App (Model-View-Controller)" seçeneğini bulamazsanız, Visual Studio Installer'ı açmanız gerekmektedir. Bunun için Visual Studio 2022'nin bulunduğu alanda "Değiştir" butonuna tıklamanız gerekmektedir.



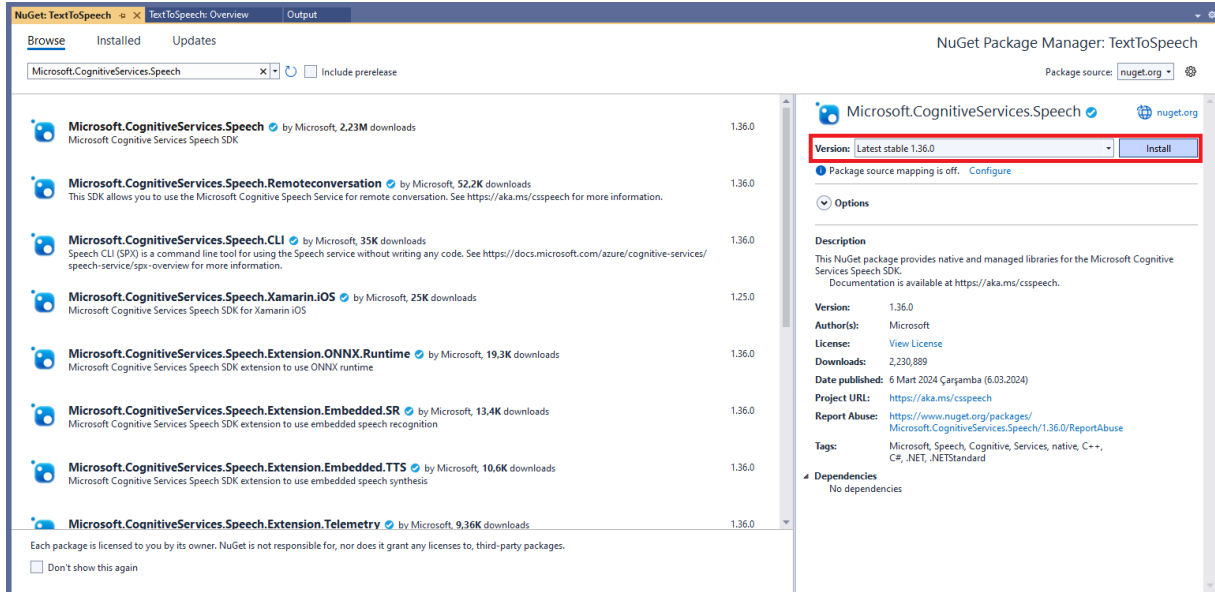
Açılan sayfanın “İş yükleri” sekmesinde “Web ve Bulut” başlığı bulunmaktadır. Bu başlık altındaki “ASP.NET ve web geliştirme” seçeneğinin seçili olması kontrol edilmedir. Ayrıca “Bağımsız bileşenler” sekmesi altındaki “.NET 6.0 Çalıştırılma Zamanı” seçeneğini işaretli olmasını kontrol ediniz. İlgili bölümler seçildikten sonra “Değiştir” butonuna tıklanarak gerekli kurulumlar yapılır.



Açılan sayfanın "İş yükleri" sekmesinde "Web ve Bulut" başlığı bulunmaktadır. Bu başlık altında, "ASP.NET ve web geliştirme" seçeneğinin seçili olup olmadığını kontrol ediniz. Ayrıca, "Bağımsız bileşenler" sekmesi altında ".NET 6.0 Çalıştırılma Zamanı" seçeneğinin işaretli olduğundan emin olunuz. İlgili bölümler seçildikten sonra, gerekli kurulumlar için "Değiştir" butonuna tıklanır.

8.2.3. NuGet Paketinin Eklenmesi ve Metin Okuma Sistemine Giriş

Proje oluşturulduktan sonra, "Solution Explorer" üzerinde projeye sağ tıklayarak "Manage NuGet Packages" seçeneğine tıklanır. Açılan pencerede "Browse" sekmesine geçilir ve arama kutusuna "Microsoft.CognitiveServices.Speech" yazılarak ilgili NuGet paketine erişilir. Örnek kapsamında "1.36.0" sürümü kullanılacağından, ilgili sürüm seçilerek "Install" butonuna tıklanır. Bu adım proje tarafından kullanılacak olan Microsoft Cognitive Services Speech kütüphanesinin projeye eklenmesini sağlar. Bu kütüphane metin tabanlı veya ses tabanlı işlemler gerçekleştirmek için gerekli olan API'leri sağlar. Doğru sürümün seçilmesi, projenin uyumluluğunu ve istikrarını sağlamak için önemlidir.



Paket yüklendikten sonra projenin "Models" klasörü altında "SesSentezleme.cs" adında bir sınıf oluşturulmalıdır.

Sınıf üyeleri aşağıdaki gibi olmalıdır;

- Azure konuşma kaynağı anahtarını private, static ve string türünde bir alan (field) olarak saklamalıdır.
- Azure konuşma kaynağı bölgesini private, static ve string türünde bir alan (field) olarak saklamalıdır.
- Okutulacak metni saklamak için public ve string türünde bir özellik (property) tanımlanmalıdır.
- Public ve asenkron bir fonksiyon tanımlanmalıdır, dönüş değeri Task olmalıdır. Bu fonksiyon içerisinde şu adımlar izlenmelidir:
 - Konuşma yapılandırması için "SpeechConfig.FromSubscription()" fonksiyonu kullanılarak bir değişken oluşturulmalıdır. Bu fonksiyon konuşma kaynağı anahtarını ve bölgesini parametre olarak almalıdır.
 - Using bloğu içinde SpeechSynthesizer() fonksiyonu kullanılarak konuşma sentezlemek için bir değişken oluşturulmalıdır. Bu fonksiyon yapılandırmayı parametre olarak almalıdır.
 - Sentezleyici using bloğu içerisinde using bloğu ile sonuç değişkeni oluşturulmalıdır. Değişken değeri olarak sentezleyici değişkeninden "SpeakTextAsync()" fonksiyonu çağrılarak metin asenkron seslendirilmelidir. Bu

fonksiyon Metin özelliğini (property) parametre olarak almalı ve await anahtar kelimesi ile işaretlenmelidir.

- Sonuç using bloğu içerisinde if blokları ile sonuç değişkeninden “Reason” anahtar kelimesi türetilerek sonuç kontrol edilmelidir.
 - Eğer sentezleme başarılıysa, ResultReason.SynthesizingAudioCompleted durumuyla sonuçlanır. Bu durumda konsola başarılı sonuç metnini yazdırmalıdır.
 - Eğer sentezleme iptal edilmişse, ResultReason.Canceled durumuyla sonuçlanır. Bu durumda iptal nedeni değişken ile belirlenmeli ve konsola yazdırılmalıdır. Değişken değeri “SpeechSynthesisCancellationDetails.FromResult()” fonksiyonu ile atanır. Fonksiyon parametre olarak sonuç değişkenini almalıdır.
 - Eğer sentezleme bir hata ile sonuçlanmışsa, CancellationReason.Error durumuyla sonuçlanır. Bu durumda hata kodu ve detayı konsola yazdırılmalıdır.

Ayrıca “Microsoft.CognitiveServices.Speech” kütüphanesini eklemeyi unutmayınız.

```
using Microsoft.CognitiveServices.Speech;
namespace TextToSpeech.Models
{
    public class SesSentezleme
    {
        private static string abonelikAnahtari = "Kaynak Anahtarınızı Bu Bölgeye Yazın";
        private static string abonelikBolgesi = "Kaynak Bölgenizi Bu Bölgeye Yazın";

        public string Metin { get; set; }

        public async Task MetinOkumaAsync()
        {
            var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari, abonelikBolgesi);
            using (var sentezleyici = new SpeechSynthesizer(yapilandirma))
            {
                var sonuc = await sentezleyici.SpeakTextAsync(Metin);

                if (sonuc.Reason == ResultReason.SynthesizingAudioCompleted)
                {
                    Console.WriteLine($"Metin Seslendirildi");
                }
                else if (sonuc.Reason == ResultReason.Canceled)
                {
                    var iptal = SpeechSynthesisCancellationDetails.FromResult(sonuc);
                    Console.WriteLine($"Seslendirme iptal edildi: Sebep {iptal.Reason}");

                    if (iptal.Reason == CancellationReason.Error)
                    {
                        Console.WriteLine($"Seslendirme sırasında hata oluştu: Hata Kodu - Detay {iptal.ErrorCode}\n{iptal.ErrorDetails}");
                    }
                }
            }
        }
    }
}
```

Model işlemleri tamamlandıktan sonra, Controllers/HomeController.cs dosyasına geçilmeli ve bir "HttpPost" aksiyonu (action) tanımlanmalıdır. Bu aksiyon, Models klasöründe tanımlanan sınıftan bir nesne almalıdır. Action içinde bu nesne üzerinden metin okuma fonksiyonu "Wait()" ile çağrılmalıdır. Bu eylemin geri dönüş değeri yine aynı şekilde bu nesne olmalıdır.

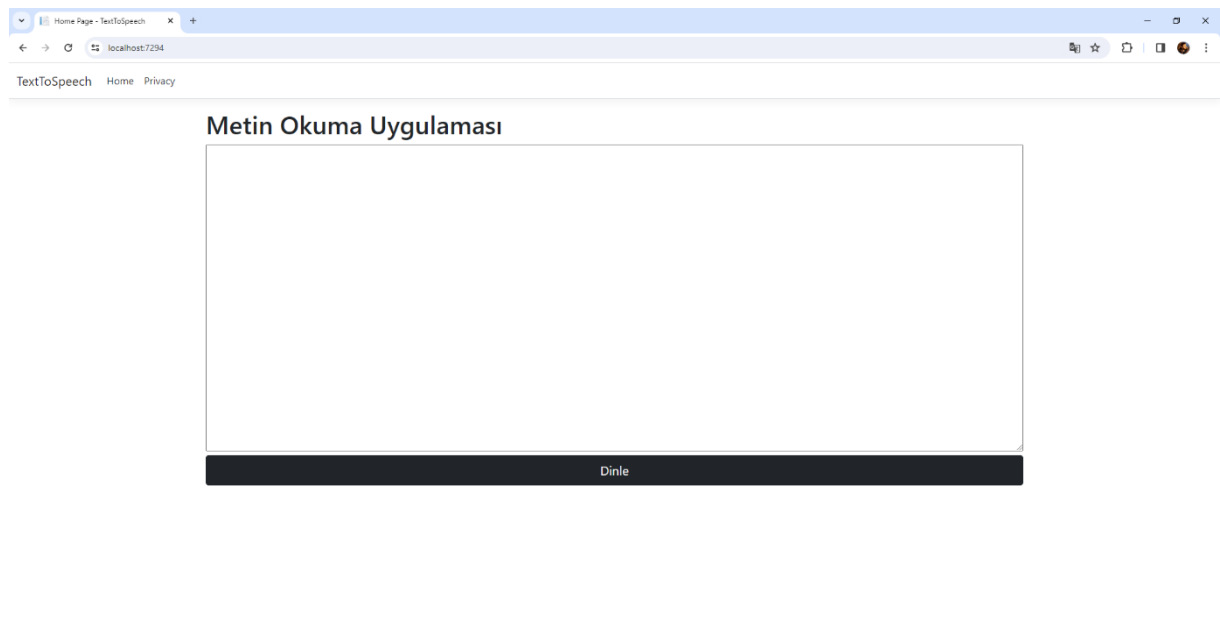
```
public IActionResult Index()
{
    return View(new SesSentezleme());
}

[HttpPost]
public IActionResult Index(SesSentezleme sesSentezleme)
{
    _ = sesSentezleme.MetinOkumaAsync();
    return View(sesSentezleme);
}
```

Controller üzerinde işlemler tamamlandıktan sonra "Views/Home/Index.cshtml" bölümünde basit bir tasarım yapılmalıdır.

```
@{
    ViewData["Title"] = "Home Page";
}
@model TextToSpeech.Models.SesSentezleme
<h1>Metin Okuma Uygulaması</h1>
<form asp-controller="Home" asp-action="Index" method="post">
    <textarea id="MetinKutusu" name="Metin" rows="20" class="w-100" asp-
for="Metin"></textarea><br />
    <button type="submit" class="btn btn-lg btn-dark w-100">Dinle</button>
</form>
```

Sınıf, "@model" ifadesiyle tanımlanmış ve çağırılmıştır. Form etiketi asp-controller ve asp-action öznitelikleri ile ilgili eyleme bağlanmıştır. TextArea etiketi ise asp-for özniteliği ile ilgili özelliğe bağlanmıştır. "Dinle" düğmesinin türü "submit" olarak belirlenerek Post eyleminin tetiklenmesi sağlanmıştır. Tasarımında tamamlanmasıyla uygulama ayağa kaldırıldığıın TextArea içerisindeki İngilizce metnin dinlenebildiği görülecektir.



8.2.4 Durdurma Butonunun ve Dil-Aksan Seçeneklerinin Eklenmesi

Durdurma butonunu oluşturabilmek için öncelikle “Models/SesSentezleme.cs” sınıfında bir alan (field) tanımlanmalıdır. Bu alan static olmalı ve SpeechSynthesizer türünde olmalıdır. Daha sonra “MetinOkumaAsync()” fonksiyonunda sentezleyici bu alana (field) eşitlenir. Ardından public ve void geri dönüş türüne sahip olan “MetinOkumayıDurdur()” fonksiyonu oluşturulur. Bu fonksiyonun içerisinde sentezleyici alanı (field) ile “StopSpeakingAsync()” fonksiyonu çağırılarak seslendirme durdurulur. Dil-Aksan seçeneği için ise public ve string bir özellik (property) tanımlanır. Yapılandırma değişkeninin oluşturulduğu satırdan sonraki satırda, yapılandırma değişkeninin "SpeechSynthesisLanguage" özelliğine (property) "Dil" özelliği atanır.

```
public class SesSentezleme
{
    private static string abonelikAnahtari = "Kaynak Anahtarınızı Bu Bölgeye Yazın";
    private static string abonelikBolgesi = "Kaynak Bölgenizi Bu Bölgeye Yazın";

    static SpeechSynthesizer sentezleyici;

    public string Metin { get; set; }
    public string Dil { get; set; }

    public async Task MetinOkumaAsync()
    {
        var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
abonelikBolgesi);
        yapilandirma.SpeechSynthesisLanguage = Dil;
        using (sentezleyici = new SpeechSynthesizer(yapilandirma))
        {
            var sonuc = await sentezleyici.SpeakTextAsync(Metin);

            if (sonuc.Reason == ResultReason.SynthesizingAudioCompleted)
            {
                Console.WriteLine($"Metin Seslendirildi");
            }
            else if (sonuc.Reason == ResultReason.Canceled)
            {
                var iptal =
SpeechSynthesisCancellationDetails.FromResult(sonuc);
                Console.WriteLine($"Seslendirme iptal edildi: Sebep
={iptal.Reason}");

                if (iptal.Reason == CancellationReason.Error)
                {
                    Console.WriteLine($"Seslendirme sırasında hata oluştu:
Hata Kodu - Detay ={iptal.ErrorCode}\n{iptal.ErrorDetails}");
                }
            }
        }
    }
    public void MetinOkumayıDurdur()
    {
        sentezleyici.StopSpeakingAsync();
    }
}
```


Model işlemlerinin tamamlanmasından ardından “Controllers/HomeController.cs” dosyasına gidilerek durdurma butonunun “HttpPost” aksiyonu (action) eklenmelidir. Aksiyon içerisinde “Models/SesSentezleme.cs” içerisinde tanımlanan “MetinOkumayiDurdur()” fonksiyonu çağırılmalıdır.

```
[HttpPost]
public IActionResult Durdur(SesSentezleme sesSentezleme)
{
    sesSentezleme.MetinOkumayiDurdur();
    return Ok();
}
```

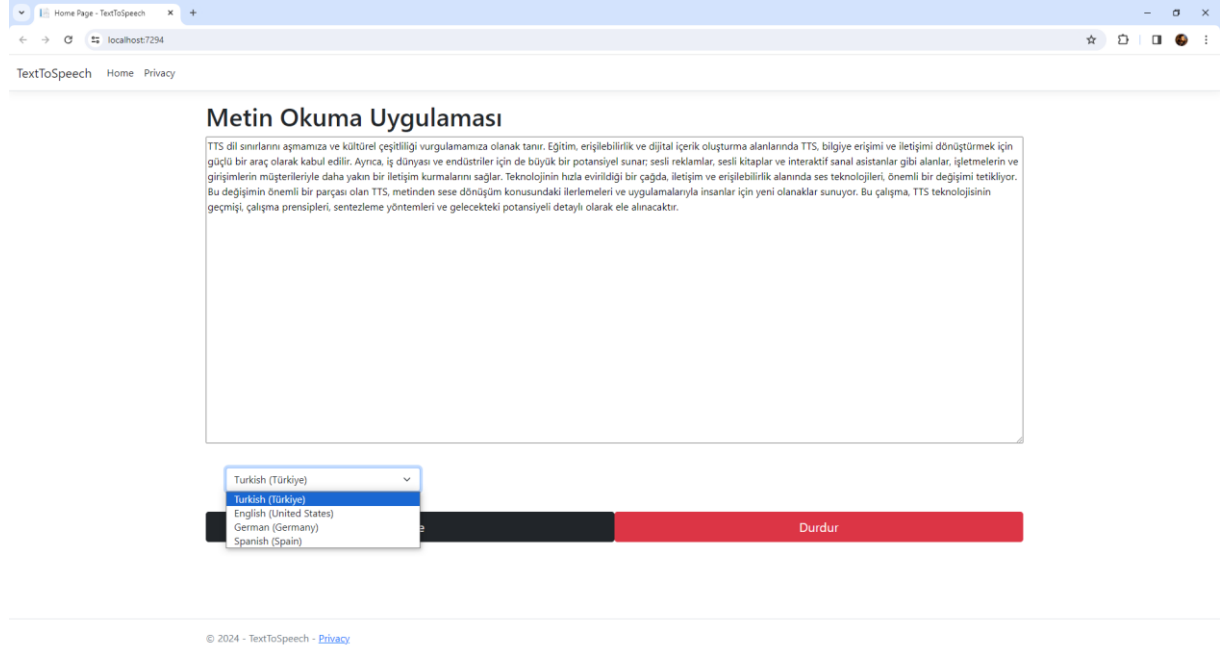
Controller işlemlerinin tamamlanmasının ardından Views/Home/Index.cshtml dosyasına gidilerek Dil-Aksan seçeneklerinin listeleneceği bir select ve bir durdurma butonu eklenmelidir. Ek olarak durdurma butonu ilgili POST aksiyonuna bağlanmalıdır.

```
@{
    ViewData["Title"] = "Home Page";
}
@model TextToSpeech.Models.SesSentezleme
<h1>Metin Okuma Uygulaması</h1>
<form asp-controller="Home" asp-action="Index" method="post">
    <textarea id="MetinKutusu" name="Metin" rows="20" class="w-100" asp-
for="Metin"></textarea><br />
    <div class="m-3 p-3">
        <select asp-for="Dil" class="form-select w-25">
            <option value="tr-TR">Turkish (Türkiye)</option>
            <option value="en-US">English (United States)</option>
            <option value="de-DE">German (Germany)</option>
            <option value="es-ES">Spanish (Spain)</option>
        </select>
    </div>
    <button type="submit" class="btn btn-lg btn-dark w-
50">Dinle</button><button id="durdurButton" type="button" class="btn btn-lg
btn-danger w-50">Durdur</button>
</form>
<script>
    document.getElementById("durdurButton").addEventListener("click", function
() {
        fetch('@Url.Action("Durdur", "Home")', {
            method: 'POST'
        })
    });
</script>
```

Select etiketi, "Dil" özelliğine asp-for ile bağlanmıştır. Select içerisindeki optionlara listelenmesi istenilen diller ve değerleri eklenmelidir. Dinle butonunun genişliği yarıya düşürülerek yanına durdur butonu eklenmiştir. JavaScript kodu ile durdur butonuna tıklandığında çalışması gereken aksiyon yazılmıştır.

Desteklenen dillerin tamamını görüntülemek için bu adresi kullanabilirsiniz: <https://learn.microsoft.com/en-us/azure/ai-services/speech-service/language-support?tabs=stt>

İlgili adımlar tamamlanarak uygulama başlatıldığında, TextArea içerisinde yazılan metnin seçilen dil ve aksanda seslendirildiği, ayrıca durdur butonunun da işlevsel olduğu gözlemlenecektir.



8.2.5. Desteklenen Tüm Ses Seçeneklerini Getirme

Desteklenen tüm dil seçeneklerini alabilmek için, public, static, asenkron bir fonksiyon tanımlanmalıdır. Bu fonksiyon parametresiz olmalı ve "Task<List<VoiceInfo>>" türünde olmalıdır. İlk olarak, yapılandırma değişkeni oluşturulur. Ardından, "VoiceInfo" sınıfını tutan boş bir List oluşturulur. Using bloğu içerisinde sentezleyici oluşturulur. Sentezleyici içerisinde bir using bloğu daha oluşturulmalı ve sentezleyici nesnesinden "GetVoicesAsync()" fonksiyonu çağırılarak sonuç bir değişkene atanmalıdır. Fonksiyon çağırılırken "await" anahtar kelimesi ile işaretlenmelidir. Sonuç if blokları ile kontrol edilmelidir. Eğer gelen sonuç "ResultReason.VoicesListRetrieved" değerine eşitse, başarılı bir sonuç elde edilmiş demektir. Bu durumda konsola çıktı verilir ve sonuç liste tipine dönüştürülerek atanır. Eğer başarısız bir sonuç elde edilirse, hatayı konsola yazdırılır. İşlemler tamamlandıktan sonra liste geri döndürülür.

```
public static async Task<List<VoiceInfo>> SesSecenekleri()
{
    var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
abonelikBolgesi);
    List<VoiceInfo> sesListesi = new List<VoiceInfo>();
    using (var sentezleyici = new SpeechSynthesizer(yapilandirma))
    {
        using (var sonuc = await sentezleyici.GetVoicesAsync())
        {
            if (sonuc.Reason == ResultReason.VoicesListRetrieved)
            {
                Console.WriteLine("Ses seçenekleri getirildi.");
                sesListesi = sonuc.Voices.ToList();
            }
            else if (sonuc.Reason == ResultReason.Canceled)
            {
            }
        }
    }
}
```

```

        Console.WriteLine($"İptal Edildi :
Detay=[{sonuc.ErrorDetails}]");
    }
}
return sesListesi;
}

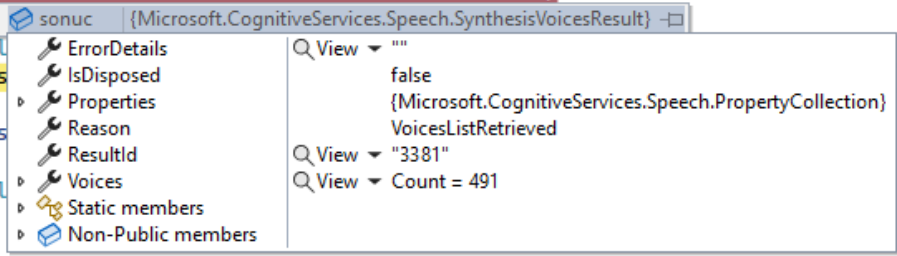
```

1 reference

```

public static async Task<List<VoiceInfo>> SesSecenekleri()
{
    var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari, abonelikBolgesi);
    List<VoiceInfo> sesListesi = new List<VoiceInfo>();
    using (var sentezleyici = new SpeechSynthesizer(yapilandirma))
    {
        using (var sonuc = await sentezleyici.GetVoicesAsync())
        {
            if (sonuc.Reason == ResultReason.VoicesListRetrieved)
            {
                Console.WriteLine(sesListesi);
            }
            else if (sonuc.Reason == ResultReason.NoVoices)
            {
                Console.WriteLine("No voices found");
            }
        }
    }
    return sesListesi;
}

```

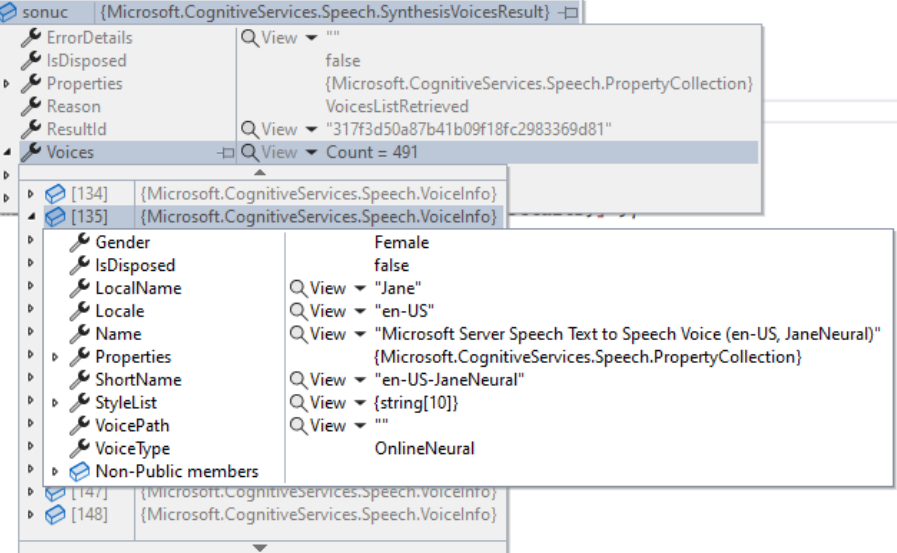


Fonksiyon doğru bir şekilde oluşturulduktan sonra çağırıldığında “Reason” değerinin “ResultReason.VoicesListRetrieved” olarak döndüğü ve kitabın yazıldığı tarih itibariyle 491 adet ses döndüğü görülmüştür.

```

public static async Task<List<VoiceInfo>> SesSecenekleri()
{
    var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari, abonelikBolgesi);
    List<VoiceInfo> sesListesi = new List<VoiceInfo>();
    using (var sentezleyici = new SpeechSynthesizer(yapilandirma))
    {
        using (var sonuc = await sentezleyici.GetVoicesAsync())
        {
            if (sonuc.Reason == ResultReason.VoicesListRetrieved)
            {
                Console.WriteLine(sesListesi);
            }
            else if (sonuc.Reason == ResultReason.NoVoices)
            {
                Console.WriteLine("No voices found");
            }
        }
    }
    return sesListesi;
}

```



"Voices" özelliği, çeşitli özelliklere sahip sesleri döndürmektedir. Bu özellikler, ses dosyalarının daha ayrıntılı ve belirli özniteliklerini tanımlamak için kullanılmaktadır.

Cinsiyet: Konuşmacının cinsiyetini belirtir. Bu, ses dosyasının hangi cinsiyete ait olduğunu belirlemek için kullanılır. Bu özellik, üç farklı değer alabilir: "Unknow", "Female" ve "Male".

LocalName: Konuşmacının adını belirtir. Bu özellik, konuşmacının tanımlanması için kullanılır ve adının yazılım içinde kullanılmasına olanak tanır.

Locale: Ses dosyasının dil ve aksanını belirtir. Bu özellik, hangi dil ve aksanda konuşulduğunu gösterir ve bu bilgi, uygulamaların çok dilli desteği sağlamasına yardımcı olur.

Name: Ses dosyasının tam adını belirtir. Bu, dosyanın adının tam olarak nasıl olduğunu belirler ve tanımlar.

ShortName: Ses dosyasının kısa adını belirtir. Bu özellik, dosyanın adının daha kısa bir versiyonunu sağlar, böylece kullanıcılar ve geliştiriciler için daha kolay erişilebilir hale gelir.

StyleList: konuşmacının duygu durumlarını listeler. Bu özellik, konuşmacının hangi duygusal ifadeleri sunabileceğini ve hangi durumlarda kullanılabileceğini belirlemek için kullanılır. Bu, uygulamaların ses dosyalarını duygusal olarak zenginleştirmesine yardımcı olur ve daha kişiselleştirilmiş bir deneyim sunar. Kitabın yazıldığı tarih itibari ile aşağıdaki değerler mevcuttur.

- **Advertisement – upbeat:** Reklamlarda kullanılan neşeli ve enerjik bir ton.
- **Affectionate:** İlgi ve sevgi dolu bir ton.
- **Angry:** Öfkeli veya sinirli bir ton.
- **Assistant:** Yardımcı veya rehberlikçi bir ton.
- **Calm:** Sakin ve dengeli bir ton.
- **Chat:** Rahat ve sıradan bir sohbet tonu.
- **Chat – casual:** Günlük, gayri resmi sohbet tonu.
- **Cheerful:** Neşeli ve mutlu bir ton.
- **Customer service:** Müşteri hizmetleri veya destek tonu.
- **Depressed:** Üzgün veya hüznü bir ton.
- **Disgruntled:** Memnuniyetsiz veya hayal kırıklığına uğramış bir ton.
- **Documentary – narration:** Belgesel anlatımı için ciddi ve dengeli bir ton.
- **Embarrassed:** Utangaç veya çekingen bir ton.
- **Empathetic:** Duygusal bir bağ kurmaya yönelik empatik bir ton.
- **Envious:** Kıskançlık veya rekabetçilik hissi uyandıran bir ton.
- **Excited:** Heyecanlı ve enerjik bir ton.
- **Fearful:** Korku veya endişe hissettiren bir ton.
- **Friendly:** Dostça ve samimi bir ton.
- **Gentle:** Nazik ve saygılı bir ton.
- **Hopeful:** Umutlu ve iyimser bir ton.
- **Lyrical:** Duygusal ve edebi bir ton.
- **Narration – Professional:** Profesyonel ve resmi bir ton.
- **Narration – relaxed:** Rahat ve hafif bir anlatımcı tonu.
- **Newscast:** Haber sunucusu tonu.
- **Newscast – casual:** Günlük haber sunumu tonu.
- **Newscast - formal:** Resmi ve ciddi bir haber sunumu tonu.

- **Poetry – reading:** Şiir veya edebi eser okuma tonu.
- **Sad:** Üzgün veya hüzünlü bir ton.
- **Serious:** Ciddi ve dikkat çekici bir ton.
- **Shouting:** Yüksek sesle ve dikkat çekici bir ton.
- **Sorry:** Özür dilemek veya özür dilemek için kibar bir ton.
- **Sports commentary:** Spor olaylarını yorumlama tonu.
- **Sports commentary – excited:** Heyecanlı ve enerjik bir spor yorumu tonu.
- **Terrified:** Korkmuş veya endişeli bir ton.
- **Unfriendly:** Soğuk veya mesafeli bir ton.
- **Whisper:** Hafif ve sessiz bir fısıltı tonu.
- **Whispering:** Dikkat çekmeden ve sessizce iletişim tonu.

Bu parametreler ile isteğe göre dil, aksan, duygu durumu, cinsiyet, konuşmacı gibi özellikleri filtreleyerek istenilen sesler oluşturulabilmektedir.

Fonksiyonun yazılmasının ardından, "Views/Home/Index.cshtml" dosyasına gidilerek dil seçeneklerinin listelenmesi amacıyla C# kod parçacığı içerisinde bir HashSet oluşturulmalıdır. HashSet, depolama işlemlerinde tekrar eden değerlerin önlenmesinde kullanılan bir veri yapısıdır. Aynı dilde birden fazla ses dosyası bulunduğu anda, dilin yalnızca bir kez listelenmesi için bütün değerlerin HashSet'e eklenmesi sağlanır. Bu yaklaşım, dil seçeneklerinin benzersiz bir şekilde görüntülenmesini sağlar ve tekrarlanan dil seçeneklerinin önüne geçer.

C# kod parçacığı içerisinde, "Select" etiketi kullanılarak dil seçeneklerinin görüntülediği bir alan oluşturulmuştur. HashSet kullanılarak, tekrar eden dil değerlerinin önlenmesi sağlanmıştır. "foreach" döngüsü ile ses seçenekleri fonksiyonundan dönen sonuçlar gezilmiş ve her bir dil değeri için HashSet içinde var olup olmadığı kontrol edilmiştir. Eğer dil değeri HashSet içinde bulunmuyorsa, dil değeri HashSet'e eklenmiş ve bir "option" elemanı oluşturulmuştur. Bu yöntem, dil seçeneklerinin tutarlı ve tekrarsız bir biçimde görüntülenmesini sağlar.

```
@{
    ViewData["Title"] = "Home Page";
}
@model TextToSpeech.Models.SesSentezleme

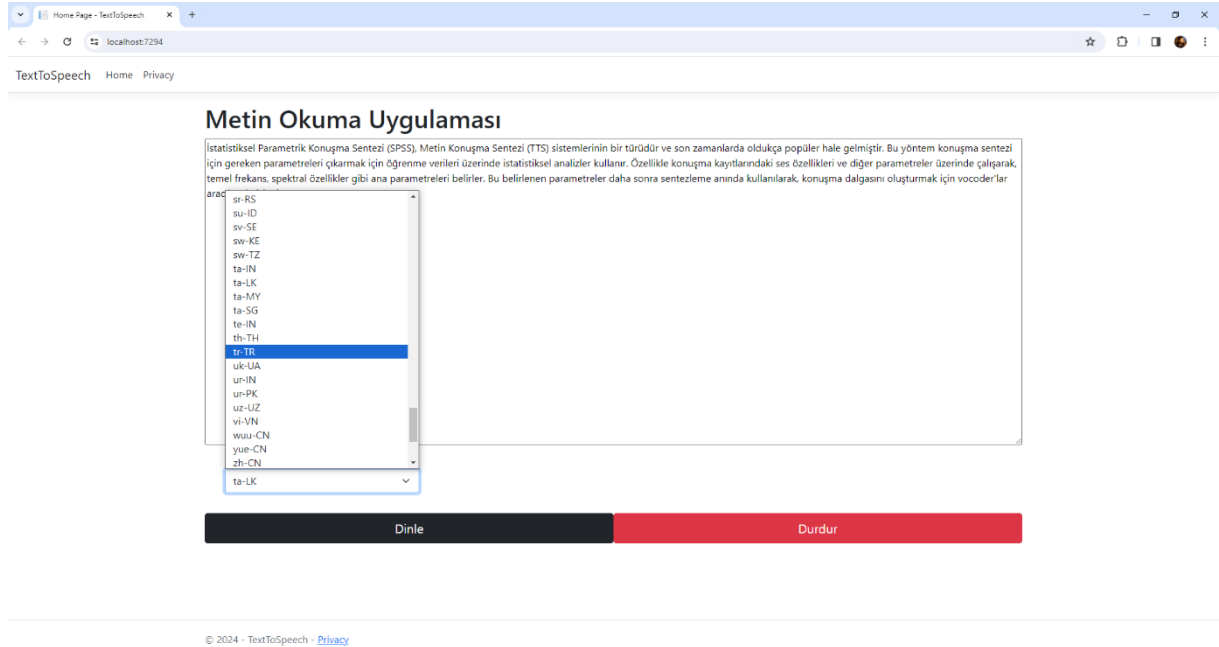
<h1>Metin Okuma Uygulaması</h1>
<form asp-controller="Home" asp-action="Index" method="post">
    <textarea id="MetinKutusu" name="Metin" rows="20" class="w-100" asp-
for="Metin"></textarea><br />
    <div class="m-3 p-3">
        <select asp-for="Dil" class="form-select w-25">
            @{
                HashSet<string> dilSecenekleri = new HashSet<string>();
                foreach (var item in SesSentezleme.SesSecenekleri().Result)
                {
                    if (!dilSecenekleri.Contains(item.Locale))
                    {
                        dilSecenekleri.Add(item.Locale);
                        <option value="@item.Locale">@item.Locale</option>
                    }
                }
            }
        </select>
    </div>
```

```

<button type="submit" class="btn btn-lg btn-dark w-50">Dinle</button><button id="durdurButton" type="button" class="btn btn-lg btn-danger w-50">Durdur</button>
</form>
<script>
    document.getElementById("durdurButton").addEventListener("click", function
    () {
        fetch('@Url.Action("Durdur", "Home")', {
            method: 'POST'
        })
    });
</script>

```

İşlemler tamamlandıktan sonra uygulama çalıştırıldığında, Azure Speech Studio'nun desteklediği tüm dil seçeneklerinin listelendiği gözlemlenecektir.



8.2.6. Dil Seçimine Göre Konuşmacıların Listelenmesi ve Tarayıcı Dilinin Otomatik Seçilmesi

Konuşmacıları listelemek için öncelikle "Models/SesSentezleme.cs" dosyasına gidilmeli ve konuşmacıların bilgilerini depolamak için bir özellik (property) oluşturulmalıdır. Daha sonra bu özellik, MetinOkumaAsync() fonksiyonundaki yapılandırma değişkeninin "SpeechSynthesisVoiceName" özelliğine eşitlenmelidir.

```

public class SesSentezleme
{
    private static string abonelikAnahtari = "Kaynak Anahtarınızı Bu Bölgeye Yazın";
    private static string abonelikBolgesi = "Kaynak Bölgenizi Bu Bölgeye Yazın";
}

```

```

static SpeechSynthesizer sentezleyici;
public string Metin { get; set; }
public string Dil { get; set; }
public string Konusmaci { get; set; }

public async Task MetinOkumaAsync()
{
    var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
abonelikBolgesi);
    yapilandirma.SpeechSynthesisLanguage = Dil;
    yapilandirma.SpeechSynthesisVoiceName = Konusmaci;
    using (sentezleyici = new SpeechSynthesizer(yapilandirma))
    {
        var sonuc = await sentezleyici.SpeakTextAsync(Metin);

        if (sonuc.Reason == ResultReason.SynthesizingAudioCompleted)
        {
            Console.WriteLine($"Metin Seslendirildi");
        }
        else if (sonuc.Reason == ResultReason.Canceled)
        {
            var iptal =
SpeechSynthesisCancellationDetails.FromResult(sonuc);
            Console.WriteLine($"Seslendirme iptal edildi: Sebep
={iptal.Reason}");

            if (iptal.Reason == CancellationReason.Error)
            {
                Console.WriteLine($"Seslendirme sırasında hata oluřtu:
Hata Kodu - Detay ={iptal.ErrorCode}\n{iptal.ErrorDetails}");
            }
        }
    }
}

//Diğer Fonksiyonlar

```

Model dosyasındaki işlemler tamamlandıktan sonra, "Views/Home/Index.cshtml" dosyasına gidilerek, konuşmacıları listelemek için bir seçim alanı daha oluşturulmalıdır. Bu seçim alanını doldurmak için optionlar JavaScript tarafında doldurulacaktır. Bu işlem için, <script> etiketi içerisinde bir fonksiyon tanımlanmalıdır.

Fonksiyon içerisinde aşağıdaki adımlar gerçekleştirilmelidir:

- Dil seçimini tutan bir değişken tanımlanmalıdır.
- Konuşmacıların listeleneceği seçim alanı bir değişkene atanmalıdır.
- Fonksiyon her çağırıldığında, seçim alanındaki değerler sıfırlanmalıdır.
- Ses sentezleme sınıfından ses seçenekleri fonksiyonu çağırılarak JSON formatında bir değişkene atanmalıdır.
- Seslerin bulunduğu değişken içerisinde foreach döngüsü ile gezinilir.
 - If bloğu ile dil değişkeninin seçim alanındaki dile eşit olup olmadığı kontrol edilir.
 - Eğer değişkendeki dil, seçim alanındaki dile eşitse, bir option oluşturularak seçim alanına aktarılır.
- Dil seçimi her değiştirildiğinde bu fonksiyon otomatik olarak çağrılmalıdır.
- Sayfa yüklenirken bu fonksiyonun otomatik olarak çalıştırılması sağlanmalıdır.

```

@{
    ViewData["Title"] = "Home Page";
}
@model TextToSpeech.Models.SesSentezleme

<h1>Metin Okuma Uygulaması</h1>
<form asp-controller="Home" asp-action="Index" method="post">
    <textarea id="MetinKutusu" name="Metin" rows="20" class="w-100" asp-
for="Metin"></textarea><br />
    <div class="m-3 p-3">
        <select asp-for="Dil" class="form-select w-25">
            @{
                HashSet<string> dilSecenekleri = new HashSet<string>();
                foreach (var item in SesSentezleme.SesSecenekleri().Result)
                {
                    if (!dilSecenekleri.Contains(item.Locale))
                    {
                        dilSecenekleri.Add(item.Locale);
                        <option value="@item.Locale">@item.Locale</option>
                    }
                }
            }
        </select> <br />
        <select name="Konusmaci" asp-for="Konusmaci" id="Konusmaci"
class="form-select w-25">
        </select>
    </div>
    <button type="submit" class="btn btn-lg btn-dark w-
50">Dinle</button><button id="durdurButton" type="button" class="btn btn-lg
btn-danger w-50">Durdur</button>
</form>

<script>
    document.getElementById("durdurButton").addEventListener("click", function
() {
        fetch('@Url.Action("Durdur", "Home")', {
            method: 'POST'
        })
    });

    konusmaciDoldur();
    document.getElementById("Dil").addEventListener("change",
konusmaciDoldur);

    function konusmaciDoldur() {
        var seslendirmenDili = document.getElementById("Dil").value;
        var konusmaciSelect = document.getElementById("Konusmaci");
        konusmaciSelect.innerHTML = "";
        var sesler =
@Html.Raw(Json.Serialize(SesSentezleme.SesSecenekleri().Result));

        sesler.forEach(function (sesler) {
            if (sesler.locale === seslendirmenDili) {
                var option = document.createElement("option");
                option.value = sesler.shortName;
                option.text = sesler.localName;
                konusmaciSelect.appendChild(option);
            }
        });
    }
</script>

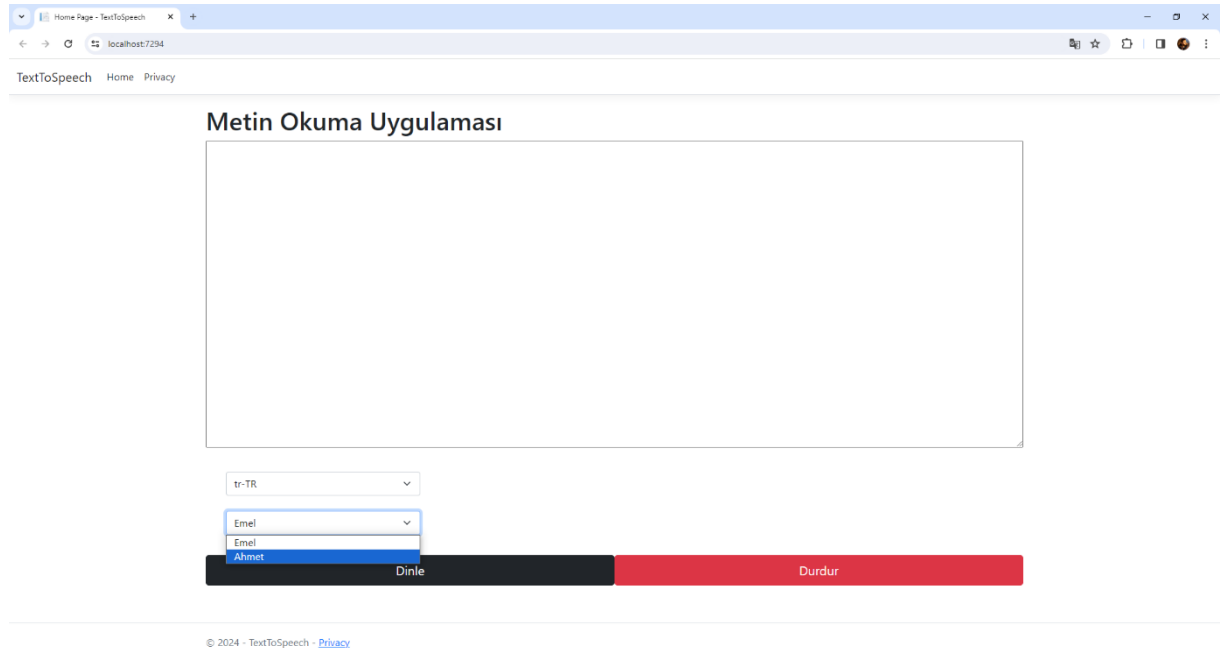
```


Bu adımlar tamamlandığında dil seçeneğine göre konuşmacıların listelendiği görülecektir. Kullanıcının tarayıcıda kullandığı dilin otomatik olarak seçilmesi için “Controllers/HomeController” dosyasına gidilmelidir. Index sayfasının GET aksiyonuna (action) gidilerek aşağıdaki işlemler yapılmalıdır.

- Ses sentezleme sınıfından bir nesne oluşturulur.
- HTTP isteğinin "Accept-Language" başlığından tarayıcının tercih ettiği diller alınır ve virgülle ayrılmış bir dizi olarak ayrıştırılır.
- Tarayıcının tercih ettiği ilk dil, dizinin ilk elemanı olarak alınır ve ses sentezleme sınıfındaki dil özelliğine (property) atanır.
- View dosyasına ses sentezleme nesnesi gönderilir.

```
public IActionResult Index()
{
    SesSentezleme sesSentezleme = new SesSentezleme();
    string[] tarayiciDilleri = Request.Headers["Accept-
Language"].ToString().Split(',');
    sesSentezleme.Dil = tarayiciDilleri[0];
    return View(sesSentezleme);
}
```

Yapılan tüm adımları tamamladığınızda, tarayıcı dilinin otomatik olarak seçilmiş olduğunu ve seçilen dile bağlı olarak konuşmacıların değiştiğini görülecektir.



8.2.7. Ses Dosyalarının MP3 ve WAV Formatlarında İndirilmesi

Ses dosyalarını kaydetmek ilk adım için “Models/SesOzellikleri.cs” sınıfında bir fonksiyon oluşturulmalıdır.

Fonksiyon içerisinde aşağıdaki adımlar gerçekleştirilmelidir:

- Fonksiyonun public ve asenkron olmalıdır ayrıca bir Task döndürmektedir. Parametre değeri string olarak dosya adını almalıdır.
- Fonksiyon içerisinde ilk olarak, SpeechConfig sınıfından bir yapılandırma örneği oluşturulmalıdır.

- Oluşturulan yapılandırmaya, Dil ve Konuşmacı özelliklerinden (property) alınan değer atanır ve böylece dönüştürülen ses dosyasının hangi dilde olacağı ve hangi konuşmacı tarafından seslendirileceği belirlenmelidir.
- Konuşma ses dosyasının WAV formatında kaydedilmesini sağlamak için using bloğu içerisinde AudioConfig.FromWavFileOutput metodu kullanılarak bir dosya çıkışı tutan bir değişken oluşturulur. Bu işlem için dosya adı parametresi kullanılmalıdır.
- SpeechSynthesizer sınıfından bir sentezleyici örneği using bloğu içerisinde yaratılır. Sentezleyici metni ses dosyasına dönüştürmek için kullanılacaktır. Oluşturulan yapılandırma ve dosya çıkışı nesneleri sentezleyiciye parametre olarak iletilmelidir.
- Sentezleyicinin using bloğu içerisinde, Metin “SpeakTextAsync()” metodu ile asenkron olarak ses dosyasına dönüştürülür. Bu işlem using bloğu içerisinde yapılmalıdır ve sonuç bir değişkene atanmalıdır. İşlem sonucu bir Task olarak döner ve işlem tamamlandığında sonuç alınır.
- Sonuç kontrol edilerek işlemin başarılı olup olmadığı belirlenir. Eğer ses dönüştürme işlemi başarıyla tamamlanmışsa (ResultReason.SynthesizingAudioCompleted durumu), bir bilgi mesajı ekrana yazdırılır.
- Eğer işlem iptal edilmişse (ResultReason.Canceled durumu), iptal detayları alınır ve bu detaylar incelenir. İptal sebebi bir hata ise (CancellationReason.Error durumu), hata kodu ve detayı ekrana yazdırılır.

```
// Diğer Fonksiyonlar
public async Task SesDosyasiKaydet(string dosyaAdi)
{
    var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
abonelikBolgesei);
    yapilandirma.SpeechSynthesisLanguage = Dil;
    yapilandirma.SpeechSynthesisVoiceName = Konusmaci;
    using (var dosyaCiktisi = AudioConfig.FromWavFileInput(dosyaAdi))
    using (var sentezleyici = new SpeechSynthesizer(yapilandirma,
dosyaCiktisi))
    {
        using (var sonuc = await sentezleyici.SpeakTextAsync(Metin))
        {
            if (sonuc.Reason == ResultReason.SynthesizingAudioCompleted)
            {
                Console.WriteLine($"Ses dosya gönderim işlemi başarılı");
            }
            else if (sonuc.Reason == ResultReason.Canceled)
            {
                var iptal =
SpeechSynthesisCancellationDetails.FromResult(sonuc);
                Console.WriteLine($"Dosya indirme işlemi iptal edildi:
Sebep={iptal.Reason}");

                if (iptal.Reason == CancellationReason.Error)
                {
                    Console.WriteLine($"Seslendirme iptal edildi: Hata Kodu
- Detay ={iptal.ErrorCode}");
                }
            }
        }
    }
}
```

Model katmanındaki işlemler tamamlandıktan sonra “Controllers/HomeController.cs” dosyasına gidilerek indirme işlemi için “HttpPost” aksiyonu (action) oluşturulmalıdır.

Aksiyon içerisinde aşağıdaki adımlar gerçekleştirilmelidir:

- HTTP Post işlemini işleyen bir metod olduğunu belirten [HttpPost] niteliği ile başlamalıdır.
- İstek parametre olarak ses sentezleme nesnesini ve string olarak dosya türünü almalıdır.
- İlk olarak, gelen dosya türü kontrol edilir. Eğer dosya türü "mp3" veya "wav" değilse, varsayılan olarak "mp3" olarak ayarlanır. Bu işlem istemci tarafından geçersiz bir dosya türü belirtildiğinde, işlemin hataya düşmemesini sağlamaktadır.
- Daha sonra, benzersiz bir dosya adı oluşturulur. Bu, dosyanın isminin çakışmasını önlemek için gereklidir. Rastgele bir Guid oluşturulur ve bu değer bir dizeye dönüştürülerek dosya adına eklenir.
- Ses sentezleme sınıfının “SesDosyasıKaydet()” yöntemi çağrılır. Bu işlem asenkron bir işlemdir bu sebeple “Wait()” metodu kullanılarak, işlem senkron hale getirilir ve tamamlanması beklenir.
- Ardından dosyanın tam yolunu elde etmek için Path.Combine yöntemi kullanılmalıdır. Bu yöntem dosyanın kaydedildiği klasörün ve dosya adının birleştirilmesiyle dosyanın tam yolunu vermektedir.
- Bellek içinde bir akış oluşturulmalıdır. Bu, dosyanın bellekte geçici olarak saklanacağı bir alan sağlamaktadır. Akış oluşturmak için “MemoryStream” sınıfından bir nesne örneği oluşturularak bir değişkende tutulmalıdır.
- Using bloğu içerisinde FileStream sınıfından bir örnek oluşturulmalıdır. Örnek oluşturulurken parametre olarak dosya yolunu ve “FileMode.Open” sabitini almalıdır. Blok içerisinde MemoryStream sınıfının örneği kullanılarak “CopyTo()” metodu ile ses dosyası belleğe kopyalanmalıdır.
- Belleğin başından itibaren okunacağını belirtmek için MemoryStream sınıfının örneğinden “Position” özelliği ile belleğin okuma pozisyonu sıfıra alınmalıdır.
- Dosyanın içeriği bellekte olduğundan, fiziksel olarak diske yazılmasına gerek kalmaz. Bu sebeple dosya silinmelidir çünkü artık dosyanın gereksiz yer kaplayacaktır. Silme işlemini gerçekleştirmek için “System.IO.File.Delete()” fonksiyonu kullanılmalıdır. Fonksiyon parametre olarak dosya yolunu almaktadır.
- Son olarak bellek içeriği dosya olarak istemciye gönderilmelidir. Geri dönüş değeri “File” yapılarak ile bu işlem gerçekleştirilir. Bu yöntem, bellek içeriğini bir dosya olarak istemciye gönderir. Dosyaya parametre olarak bellek değişkeni (bellek içeriğini), "audio/" + dosya türü (içeriğin türünü örneğin, "audio/mp3" veya "audio/wav") ve dosya adı gönderilmelidir.

// Diğer Aksiyonlar

```
[HttpPost]
public IActionResult SesKaydet(SesSentezleme sesSentezleme, string
dosyaTuru)
{
    if( !dosyaTuru.Equals("mp3") && !dosyaTuru.Equals("wav"))
    {
        dosyaTuru = "mp3";
    }
    Guid myuuid = Guid.NewGuid();
    string myuuidAsString = myuuid.ToString();
    var dosyaAdi = myuuidAsString + '.' + dosyaTuru;
```

```

sesSentezleme.SesDosyasiKaydet(dosyaAdi).Wait();
var yol = Path.Combine(Directory.GetCurrentDirectory(), dosyaAdi);
var bellek = new MemoryStream();
using (var stream = new FileStream(yol, FileMode.Open))
{
    stream.CopyTo(bellek);
}
bellek.Position = 0;
System.IO.File.Delete(yol);
return File(bellek, "audio/"+dosyaTuru, dosyaAdi);
}

```

Son olarak “Views/Home/Index.cshtml” dosyasına gidilerek indirme seçeneklerinin olduğu bir dropdown buton eklenmeli ve JavaScript kodları eklenmelidir.

Dinle, Durdur butonlarının altına “Bootstrap dropdown button” eklenmeli ve uygulamaya göre özelleştirilmelidir.

```

//Dinle ve Durdur Butonları
<div class="dropdown w-100 mt-2">
    <button class="btn btn-lg btn-success w-100 dropdown-toggle"
type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-
expanded="false">Ses dosyasını indir</button>
    <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
        <li><a class="dropdown-item" id="mp3indir">MP3 dosyası olarak
indir.</a></li>
        <li><a class="dropdown-item" id="wavindir">Wave dosyası olarak
indir.</a></li>
    </ul>
</div>
</form>

```

Daha sonra bu butonları click etkinlikleri kontrol edilmelidir. Eğer Click eventi çalışıyor ise dosyayı indirmeyi sağlayan bir fonksiyona yönlendirmelidir. Bu fonksiyon her iki dosya türü içinde kullanılacağı için parametre olarak dosya türünü göndermelidir.

```

//Javascript kodları
document.getElementById("mp3indir").addEventListener("click", function ()
{
    SesDosyasiKaydet("mp3");
});
document.getElementById("wavindir").addEventListener("click", function ()
{
    SesDosyasiKaydet("wav");
});

```

Dosyayı kaydetmeyi sağlayan bir fonksiyon oluşturulmalıdır. Fonksiyon içeriği aşağıdaki maddeler gibi olmalıdır.

- Ses özelliklerini tutan bir JavaScript nesnesi oluşturulmalıdır. Bu nesne, kullanıcının girdiği metni, dili ve konuşmacıyı içermelidir. Bu özellikler, HTML sayfasındaki belirli etiketlerden “getElementById” ile alınabilir.

- “\$.ajax” jQuery fonksiyonu kullanılarak bir AJAX isteği yapılır. Bu istek, /Home/SesKaydet adresine HTTP POST yöntemiyle yapılmalıdır. AJAX isteğinin parametreleri şu şekilde olmalıdır
 - **data:** İsteğin gövdesinde taşınacak veridir. Ses özelliklerini ve dosya türünü aksiyona göndermek için kullanılmalıdır.
 - **xhrFields:** Sunucudan alınacak yanıtın türünü belirler. Bu durumda, yanıtın bir dosya (blob) olacağı belirtilmelidir.
 - **success:** AJAX isteği başarılı olduğunda çağrılacak olan geri çağırma fonksiyonu. Bu fonksiyon, sunucudan dönen dosyayı işler ve kullanıcıya indirme işlemini sağlamak için kullanılmalıdır.
 - **error:** AJAX isteği başarısız olduğunda çağrılacak olan geri çağırma fonksiyonu. Bu durumda bir hata ile karşılaşıldığında kullanıcıya bir uyarı mesajı göndermek için kullanılmalıdır.
- Success geri çağırma fonksiyonunda, sunucudan dönen dosya verisi işlenir. Veri, bir Blob nesnesi olarak alınır ve URL'ye dönüştürülür. Bu URL, kullanıcıya dosyayı indirmesi için bir bağlantı sağlar. İndirme bağlantısı (<a> elementi) oluşturulur ve dosya adı belirlenir. Element sayfaya eklenir ve otomatik olarak tıklanır. Bu işlem, dosyanın indirilmesini sağlar. Daha sonra “window.URL.revokeObjectURL” yöntemi kullanılarak oluşturulan URL iptal edilir. Bu, tarayıcı belleğindeki kaynakların serbest bırakılmasını sağlar ve gereksiz bellek tüketimini önler.

```
//Diğer fonksiyonlar
function SesDosyasiKaydet(tur) {
    var sesOzellikleri = {
        Metin: document.getElementById("MetinKutusu").value,
        Dil: document.getElementById("Dil").value,
        Konusmaci: document.getElementById("Konusmaci").value
    };
    $.ajax({
        url: '/Home/SesKaydet',
        type: 'POST',
        data: { sesSentezleme: sesOzellikleri, dosyaTuru: tur },
        xhrFields: {
            responseType: 'blob'
        },
        success: function (data) {
            var url = window.URL.createObjectURL(new Blob([data]));
            var a = document.createElement('a');
            a.href = url;
            a.download = 'sesDosyasi.' + tur;
            document.body.appendChild(a);
            a.click();
            window.URL.revokeObjectURL(url);
        },
        error: function (xhr, status, error) {
            alert("Hata oluştu: " + error);
        }
    });
}
```

8.2.8. Adımların Değerlendirilmesi ve Projenin Kodlarının Tamamı

Web uygulaması başlığı altında şimdiye kadarki adımlarda aşağıdaki maddeler hakkında bilgi sağlanmıştır.

- 1- Microsoft Azure üzerinde TTS için gerekli kaynağın oluşturulması.
- 2- ASP.NET Core MVC projesi oluşturma
- 3- Azure TTS kütüphanesi için gerekli NuGet paketlerinin yüklenmesi
- 4- Asenkron olarak metin okuma işleminin gerçekleştirilmesi.
- 5- Metin okuma için farklı dil destekleri.
- 6- Microsoft Azure TTS'nin desteklediği ses dosyalarının kullanımı.
- 7- Ses dosyalarının indirilmesi.

Bu bilgiler ile örnek proje geliştirilerek yüksek kalitede ses sentezlenmesi için görseldeki gibi bir platform oluşturulmuş.



Projenin çalışması yapısına göre “Dinle” butonuna tıklandığında konuşmacılar her seferinde tekrardan select içerisine option olarak eklenmektedir. Bu sebeple seçili olan konuşmacı, seçili olarak gelmemektedir. Bu durumu düzeltmek için Javascript kodlarının olduğu bölümde “konusmaciDoldur()” fonksiyonu çağırıldıktan sonra aşağıdaki kod bloğu çağırılmalıdır.

```
konusmaciDoldur();
var konusmaci = "@Model.Konusmaci";
if (konusmaci !== null && konusmaci !== "") {
    document.getElementById("Konusmaci").value = konusmaci;
}
```

Projenin başında belirtildiği gibi gerçek dünya projelerinde, iş süreçleri genellikle karmaşıktır. Örneğin bu proje için kaynak anahtarı ve bölgesi daha gizli bir alanda tutulmalıdır. Ayrıca gerekli doğrulama işlemleri yapılmadığı için proje sağlıklı çalışmayacaktır. Kitabımızın konusundan sapmamak için web uygulaması için gerekli düzenlemeler projeye eklenmemiştir. Projenin bu adıma kadarki bölümünde aşağıdaki kodlar kullanılmıştır.

“Views/Home/Index.cshtml” dosyasının içeriği:

```
@{
    ViewData["Title"] = "Home Page";
}
@model TextToSpeech.Models.SesSentezleme

<h1>Metin Okuma Uygulaması</h1>
<form asp-controller="Home" asp-action="Index" method="post">
    <textarea id="MetinKutusu" name="Metin" rows="20" class="w-100" asp-
for="Metin"></textarea><br />
    <div class="m-3 p-3">
        <select asp-for="Dil" class="form-select w-25">
            @{
                HashSet<string> dilSecenekleri = new HashSet<string>();
                foreach (var item in SesSentezleme.SesSecenekleri().Result)
                {
                    if (!dilSecenekleri.Contains(item.Locale))
                    {
                        dilSecenekleri.Add(item.Locale);
                        <option value="@item.Locale">@item.Locale</option>
                    }
                }
            }
        </select><br />
        <select name="Konusmaci" asp-for="Konusmaci" id="Konusmaci"
class="form-select w-25">
        </select>
    </div>
    <button type="submit" class="btn btn-lg btn-dark w-
50">Dinle</button><button id="durdurButton" type="button" class="btn btn-lg
btn-danger w-50">Durdur</button>
    <div class="dropdown w-100 mt-2">
        <button class="btn btn-lg btn-success w-100 dropdown-toggle"
type="button" id="dropdownMenuButton1" data-bs-toggle="dropdown" aria-
expanded="false">Ses dosyasını indir</button>
        <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
            <li><a class="dropdown-item" id="mp3indir">MP3 dosyası olarak
indir.</a></li>
            <li><a class="dropdown-item" id="wavindir">Wave dosyası olarak
indir.</a></li>
        </ul>
    </div>
</form>

<script>
    function konusmaciDoldur() {
        var seslendirmenDili = document.getElementById("Dil").value;
        var konusmaciSelect = document.getElementById("Konusmaci");
        konusmaciSelect.innerHTML = "";
        var sesler =
@Html.Raw(Json.Serialize(SesSentezleme.SesSecenekleri().Result));
        sesler.forEach(function (sesler) {
            if (sesler.locale === seslendirmenDili) {
                var option = document.createElement("option");
                option.value = sesler.shortName;
                option.text = sesler.localName;
                konusmaciSelect.appendChild(option);
            }
        });
    }
    function SesDosyasiKaydet(tur) {
```

```

var sesOzellikleri = {
    Metin: document.getElementById("MetinKutusu").value,
    Dil: document.getElementById("Dil").value,
    Konusmaci: document.getElementById("Konusmaci").value
};
$.ajax({
    url: '/Home/SesKaydet',
    type: 'POST',
    data: { sesSentezleme: sesOzellikleri, dosyaTuru: tur },
    xhrFields: {
        responseType: 'blob'
    },
    success: function (data) {
        var url = window.URL.createObjectURL(new Blob([data]));
        var a = document.createElement('a');
        a.href = url;
        a.download = 'sesDosyasi.' + tur;
        document.body.appendChild(a);
        a.click();
        window.URL.revokeObjectURL(url);
    },
    error: function (xhr, status, error) {
        alert("Hata oluřtu: " + error);
    }
});
}

document.getElementById("durdurButton").addEventListener("click", function
() {
    fetch('@Url.Action("Durdur", "Home")', {
        method: 'POST'
    })
    });
document.getElementById("Dil").addEventListener("change",
konusmaciDoldur);
document.getElementById("mp3indir").addEventListener("click", function ()
{
    SesDosyasiKaydet("mp3");
});
document.getElementById("wavindir").addEventListener("click", function ()
{
    SesDosyasiKaydet("wav");
});

konusmaciDoldur();
var konusmaci = "@Model.Konusmaci";
if (konusmaci !== null && konusmaci !== "") {
    document.getElementById("Konusmaci").value = konusmaci;
}
</script>

```

“Controller/HomeController.cs” dosyasının ierięi :

```

using Microsoft.AspNetCore.Mvc;
using System.Diagnostics;
using TextToSpeech.Models;

```



```

namespace TextToSpeech.Controllers
{
    public class HomeController : Controller
    {
        private readonly ILogger<HomeController> _logger;

        public HomeController(ILogger<HomeController> logger)
        {
            _logger = logger;
        }

        public IActionResult Index()
        {
            SesSentezleme sesSentezleme = new SesSentezleme();
            string[] tarayiciDilleri = Request.Headers["Accept-
Language"].ToString().Split(',');
            sesSentezleme.Dil = tarayiciDilleri[0];
            return View(sesSentezleme);
        }

        [HttpPost]
        public IActionResult Index(SesSentezleme sesSentezleme)
        {
            _ = sesSentezleme.MetinOkumaAsync();
            return View(sesSentezleme);
        }

        [HttpPost]
        public IActionResult Durdur(SesSentezleme sesSentezleme)
        {
            sesSentezleme.MetinOkumayiDurdur();
            return Ok();
        }

        [HttpPost]
        public IActionResult SesKaydet(SesSentezleme sesSentezleme, string
dosyaTuru)
        {
            if( !dosyaTuru.Equals("mp3") && !dosyaTuru.Equals("wav"))
            {
                dosyaTuru = "mp3";
            }
            Guid myuuid = Guid.NewGuid();
            string myuuidAsString = myuuid.ToString();
            var dosyaAdi = myuuidAsString + '.' + dosyaTuru;
            sesSentezleme.SesDosyasiKaydet(dosyaAdi).Wait();
            var yol = Path.Combine(Directory.GetCurrentDirectory(), dosyaAdi);
            var bellek = new MemoryStream();
            using (var stream = new FileStream(yol, FileMode.Open))
            {
                stream.CopyTo(bellek);
            }
            bellek.Position = 0;
            System.IO.File.Delete(yol);
            return File(bellek, "audio/"+dosyaTuru, dosyaAdi);
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None,
NoStore = true)]

```

```

        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id
            ?? HttpContext.TraceIdentifier });
        }
    }
}

```

“Models/SesSentezleme.cs” dosyasının içeriği

```

using Microsoft.CognitiveServices.Speech;
using Microsoft.CognitiveServices.Speech.Audio;
using static System.Net.Mime.MediaTypeNames;
namespace TextToSpeech.Models
{
    public class SesSentezleme
    {
        private static string abonelikAnahtari = "Kaynak_Anahtarini_Girin";
        private static string abonelikBolgesi = "Kaynak_Bolgesini_Girin";
        static SpeechSynthesizer sentezleyici;
        public string Metin { get; set; }
        public string Dil { get; set; }
        public string Konusmaci { get; set; }

        public async Task MetinOkumaAsync()
        {
            var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
            abonelikBolgesi);
            yapilandirma.SpeechSynthesisLanguage = Dil;
            yapilandirma.SpeechSynthesisVoiceName = Konusmaci;
            using (sentezleyici = new SpeechSynthesizer(yapilandirma))
            {
                var sonuc = await sentezleyici.SpeakTextAsync(Metin);

                if (sonuc.Reason == ResultReason.SynthesizingAudioCompleted)
                {
                    Console.WriteLine($"Metin Seslendirildi");
                }
                else if (sonuc.Reason == ResultReason.Canceled)
                {
                    var iptal =
                    SpeechSynthesisCancellationDetails.FromResult(sonuc);
                    Console.WriteLine($"Seslendirme iptal edildi: Sebep
                    ={iptal.Reason}");

                    if (iptal.Reason == CancellationReason.Error)
                    {
                        Console.WriteLine($"Seslendirme iptal edildi: Hata
                        Kodu - Detay ={iptal.ErrorCode}\n{iptal.ErrorDetails}");
                    }
                }
            }
        }
        public void MetinOkumayiDurdur()
        {
            sentezleyici.StopSpeakingAsync();
        }
        public static async Task<List<VoiceInfo>> SesSecenekleri()
        {
            var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
            abonelikBolgesi);

```

```

        List<VoiceInfo> sesListesi = new List<VoiceInfo>();
        using (var sentezleyici = new SpeechSynthesizer(yapilandirma))
        {
            using (var sonuc = await sentezleyici.GetVoicesAsync())
            {
                if (sonuc.Reason == ResultReason.VoicesListRetrieved)
                {
                    Console.WriteLine("Ses seçenekleri getirildi.");
                    sesListesi = sonuc.Voices.ToList();
                }
                else if (sonuc.Reason == ResultReason.Canceled)
                {
                    Console.WriteLine($"İptal Edildi :
Detay=[{sonuc.ErrorDetails}]");
                }
            }
        }
        return sesListesi;
    }
    public async Task SesDosyasiKaydet(string dosyaAdi)
    {
        var yapilandirma = SpeechConfig.FromSubscription(abonelikAnahtari,
abonelikBolgesi);
        yapilandirma.SpeechSynthesisLanguage = Dil;
        yapilandirma.SpeechSynthesisVoiceName = Konusmaci;
        using (var dosyaCiktisi = AudioConfig.FromWavFileInput(dosyaAdi))
        using (var sentezleyici = new SpeechSynthesizer(yapilandirma,
dosyaCiktisi))
        {
            using (var sonuc = await sentezleyici.SpeakTextAsync(Metin))
            {
                if (sonuc.Reason ==
ResultReason.SynthesizingAudioCompleted)
                {
                    Console.WriteLine($"Ses dosya gönderim işlemi
başarılı");
                }
                else if (sonuc.Reason == ResultReason.Canceled)
                {
                    var iptal =
SpeechSynthesisCancellationDetails.FromResult(sonuc);
                    Console.WriteLine($"Dosya indirme işlemi iptal edildi:
Sebep={iptal.Reason}");

                    if (iptal.Reason == CancellationReason.Error)
                    {
                        Console.WriteLine($"Seslendirme iptal edildi: Hata
Kodu - Detay ={iptal.ErrorCode}");
                    }
                }
            }
        }
    }
}

```

9. Referans

- AKAR, F., & GÜNGÖR, B. (2023). Text-to-Speech (TTS) Teknolojisi: Gelişimi, Uygulamaları ve Geleceği. In *Akıllı sistemlerin Endüstriyel Uygulaması II*. BIDGE YAYINLARI.
- Akhmetzhanov, N. (2021). *IMPLEMENTATION OF A MULTISPEAKER TEXT-TO-SPEECH SYNTHESIS WEB APPLICATION*.
- Brackhane, F. (2015). *Kempelen vs. Kratzenstein-Researchers on speech synthesis in times of change*.
- Chettri, B., & Shah, K. B. (2013). Nepali Text to Speech Synthesis System using ESNOLA Method of Concatenation. In *International Journal of Computer Applications* (Vol. 62, Issue 2).
- Dai, L., Kritskaia, V., Van Der Velden, E., Jung, M. M., Postma, M., & Louwerse, M. M. (2022). Evaluating the usage of Text-To-Speech in K12 education. *ACM International Conference Proceeding Series*, 182–188. <https://doi.org/10.1145/3578837.3578864>
- DUTOIT, T. (2004). *High-quality text-to-speech synthesis : an overview*.
- Hoy, M. B. (2018a). Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. *Medical Reference Services Quarterly*, 37(1), 81–88. <https://doi.org/10.1080/02763869.2018.1404391>
- Hoy, M. B. (2018b). Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants. *Medical Reference Services Quarterly*, 37(1), 81–88. <https://doi.org/10.1080/02763869.2018.1404391>
- Huang, G., & Er, M. J. (2012). *An Adaptive Control Scheme for Articulatory Synthesis of Plosive-Vowel Sequences*. IEEE.
- Kaur, N., & Singh, P. (2023a). Conventional and contemporary approaches used in text to speech synthesis: a review. *Artificial Intelligence Review*, 56(7), 5837–5880. <https://doi.org/10.1007/s10462-022-10315-0>
- Kaur, N., & Singh, P. (2023b). Conventional and contemporary approaches used in text to speech synthesis: a review. *Artificial Intelligence Review*, 56(7), 5837–5880. <https://doi.org/10.1007/s10462-022-10315-0>
- Kleiner, A., & Kurzweil, R. C. (1977). *A DESCRIPTION OF THE KURZWEIL READING MACHINE AND A STATUS REPORT ON ITS TESTING AND DISSEMINATION*.
- Kumar, Y., Koul, A., & Singh, C. (2023). A deep learning approaches in text-to-speech system: a systematic review and recent research perspective. *Multimedia Tools and Applications*, 82(10), 15171–15197. <https://doi.org/10.1007/s11042-022-13943-4>
- Lemmetty, S. (1999). *Review of Speech Synthesis Technology*.
- Manero Alvarez Advisors, A., Hernaez Rioja Eva Navas Cordón, I., & Azterketa eta Prozesamendua, H. (2022). *Implementation and evaluation of a Spanish TTS based on FastPitch*.
- Mchargue, J., & Podgurski, A. (2023). *EFFICIENT MULTISPEAKER SPEECH SYNTHESIS AND VOICE CLONING*.

- Moran, M. E. (2007). Evolution of robotic arms. *Journal of Robotic Surgery*, 1(2), 103–111. <https://doi.org/10.1007/s11701-006-0002-x>
- Ohala, J. J. (2011). *CHRISTIAN GOTTLIEB KRATZENSTEIN: PIONEER IN SPEECH SYNTHESIS*.
- Onaolapo, J. ,O., Idachaba, F. , E., Badejo, J., Odu, T., & Adu O, I. (2014). *A Simplified Overview of Text-To-Speech Synthesis*.
- Panda, S. P., Nayak, A. K., & Patnaik, S. (2015). Text to Speech Synthesis with an Indian Language Perspective. In *International Journal of Grid and Utility Computing, Inderscience* (Vol. 6, Issue 4).
- Panda, S. P., Nayak, A. K., & Rai, S. C. (2020). A survey on speech synthesis techniques in Indian languages. *Multimedia Systems*, 26(4), 453–478. <https://doi.org/10.1007/s00530-020-00659-4>
- Sairanen, V. (2023). *Deep learning text-to-speech synthesis with Flowtron and WaveGlow*. www.aalto.fi
- Siri Team. (2017, June). *Deep Learning for Siri's Voice: On-device Deep Mixture Density Networks for Hybrid Unit Selection Synthesis*.
- Story, B. H. (2019). History of speech synthesis. In *The Routledge Handbook of Phonetics* (pp. 9–33). Taylor and Francis. <https://doi.org/10.4324/9780429056253-2>
- Sussman, M. J. (1999). *Performing the Intelligent Machine: Deception and Enchantment in the Life of the Automaton Chess Player*. <https://muse.jhu.edu/article/32960>.
- Tan, X., Qin, T., Soong, F., & Liu, T.-Y. (2021). *A Survey on Neural Speech Synthesis*. <http://arxiv.org/abs/2106.15561>
- Wankhade, M., Rao, A. C. S., & Kulkarni, C. (2022). A survey on sentiment analysis methods, applications, and challenges. *Artificial Intelligence Review*, 55(7), 5731–5780. <https://doi.org/10.1007/s10462-022-10144-1>
- Wikipedia. (2021). *Speech synthesis*. https://en.wikipedia.org/w/index.php?Title=Speech%20synthesis&oldid=1020857981#cite_note-Helsinki-5.
- Wikipedia. (2024). *Mechanical Turk*. https://en.wikipedia.org/wiki/Mechanical_Turk.