
GAZİ ÜNİVERSİTESİ
Gazi
1926

EKLEMELİ SENTEZLEYİCİ YÖNTEMİ KULLANILARAK TÜRKÇE METİNDEN KONUŞMA SENTEZLEME

Fatih UYSAL

YÜKSEK LİSANS TEZİ
ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

OCAK 2017

Fatih UYSAL tarafından hazırlanan "EKLEMELİ SENTEZLEYİCİ YÖNTEMİ KULLANILARAK TÜRKÇE METİNDEN KONUŞMA SENTEZLEME" adlı tez çalışması aşağıdaki jüri tarafından OY BİRLİĞİ ile Gazi Üniversitesi Elektrik Elektronik Mühendisliği Anabilim Dalında YÜKSEK LİSANS TEZİ olarak kabul edilmiştir.

Danışman: Doç. Dr. Fırat HARDALAÇ

Elektrik Elektronik Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ...……

Başkan : Prof. Dr. Sabri KOÇER

Bilgisayar Mühendisliği, Necmettin Erbakan Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……...

Üye : Doç. Dr. Nursel AKÇAM

Elektrik Elektronik Mühendisliği, Gazi Üniversitesi

Bu tezin, kapsam ve kalite olarak Yüksek Lisans Tezi olduğunu onaylıyorum ……...

Tez Savunma Tarihi: 25/01/2017

Jüri tarafından kabul edilen bu tezin Yüksek Lisans Tezi olması için gerekli şartları yerine getirdiğini onaylıyorum.

……..……

Prof. Dr. Hadi GÖKÇEN

Fen Bilimleri Enstitüsü Müdürü

ETİK BEYAN

Gazi Üniversitesi Fen Bilimleri Enstitüsü Tez Yazım Kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- Tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi,
- Tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu,
- Tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi,
- Kullanılan verilerde herhangi bir değişiklik yapmadığımı,
- Bu tezde sunduğum çalışmanın özgün olduğunu,

bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi beyan ederim.

Fatih UYSAL

25/01/2017

EKLEMELİ SENTEZLEYİCİ YÖNTEMİ KULLANILARAK TÜRKÇE METİNDEN KONUŞMA SENTEZLEME

(Yüksek Lisans Tezi)

Fatih UYSAL

GAZİ ÜNİVERSİTESİ

FEN BİLİMLERİ ENSTİTÜSÜ

Ocak 2017

ÖZET

Konuşma sentezleme işlemleri, geçmişten günümüze kadar birçok uygulamada kullanılmıştır. İlk başlarda, mekaniksel olan sentezleme sistemleri, yerini elektriksel sistemlerle değiştirmiştir. Literatüre bakıldığında sentezleme için temel olarak; boğumlama (söyleyiş), formant ve art arda bağlama (eklemeli) sentezleyici yöntemlerinin kullanıldığı görülmektedir. Bu tez kapsamında, konuşma sentezlemede; eklemeli sentezleyici yöntemi ve bu yöntem için gerekli olan konuşma parçası için ise heceler kullanılmıştır. Bu tez çalışmasının amacı, geliştirilen metinden konuşma sentezleme dinamik veritabanı akış diyagramı kullanılarak hem Türkçe hem de hecelere ayrılamayan Türkçedeki yabancı kelimeleri içeren bir ses veritabanıyla çalışabilen bir Türkçe metinden konuşma sentezleme uygulaması gerçekleştirmektir. Sonuç olarak; bu çalışmada literatüre katkı olarak konuşma sentezleme işlemi için gerekli olan konuşma parçaları, sabit veritabanı yerine güncellenen bir ses veritabanı kullanılarak elde edilmiş ve kesintisiz çalışmayı amaçlayan bir sentezleme sistemi geliştirilmeye çalışılmıştır.

Bilim Kodu	:	90537
Anahtar Kelimeler	:	Konuşma işleme, Türkçe metinden konuşma sentezleme, veritabanı, görsel programlama
Sayfa Adedi	:	64
Danışman	:	Doç. Dr. Fırat HARDALAÇ

TURKISH TEXT-TO-SPEECH SYNTHESIS BY USING CONCATENATIVE SYNTHESIZER METHOD

(M. Sc. Thesis)

Fatih UYSAL

GAZİ UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

January 2017

ABSTRACT

The speech synthesizing operations/processes have been used in many applications from past to present. The synthesizing operations which were mechanical initially replaced the place with electrical systems. When the literature is considered, it is seen that basically articulatory, formant and concatenative synthesizing methods have been used. In this thesis, the concatenative synthesizing method is used for speech synthesizing and syllables are used as the speech part required for this method. The objective of this thesis study is to carry out a speech synthesizing application from a created Turkish text which can work with a sound database containing both Turkish words and foreign words that are unable to be syllabified by using a dynamic database flow chart. Lastly, the speech parts required for speech synthesizing were obtained through an updated sound database instead of a stationary database and it was tried to develop a synthesizing system aiming at uninterrupted operation as a contribution to the literature.

Science Code	:	90537
Key Words	:	Speech processing, Turkish text-to-speech synthesis, database, visual programming
Page Number	:	64
Supervisor	:	Assoc. Prof. Dr. Fırat HARDALAÇ

TEŞEKKÜR

Tez çalışmam boyunca bana her türlü desteği sağlayan aileme, tezimle alakalı yapmış olduğum bildirimde katkısı olan arkadaşım Sadık TURGUT'a ve beni çalışmalarım boyunca sürekli olarak yönlendiren ve her türlü konuda çok yardımcı olan danışmanım sayın Doç. Dr. Fırat HARDALAÇ'a çok teşekkür ediyorum.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklamalar**
ms	Mili saniye
Hz	Hertz

Kısaltmalar	**Açıklamalar**
API	Uygulama Programlama Arayüzü
HMM	Saklı Markov Modeli
HTS	Saklı Markov Modeli Sentezleme Aracı
LP-PSOLA	Doğrusal Öngörümlü PSOLA
LPC	Doğrusal Öngörümlü Kodlama
MBROLA	Mons Üniversitesi konuşma sentezleme sistemi
MITalk	Massachusetts Teknoloji Üniversitesi sentezleme sistemi
MKS	Metinden Konuşma Sentezleme
PSOLA	Perde Eşzamanlı Örtüştürerek Ekleme
TD-PSOLA	Zaman Ekseninde PSOLA
TTS	Metinden Konuşma Sentezleme
WAV	Dalgaşekli Ses Dosyası Biçimi

1. GİRİŞ

Türkçe dili dahil olmak üzere birçok dilde yapılan uygulamalarda kullanılan konuşma sentezleme işlemleri birçok akademik ve teknolojik çalışmada kullanılmaktadır. Sistem girdisinin metin, çıktısının ise konuşma olduğu yani yazılı olan bir metnin konuşmaya dönüştürülmesi işlemi olan metinden konuşma sentezlemenin; uygulandığı birçok alan vardır.

Metinden konuşma sentezlemenin uygulandığı alanlara;

- Sağlık sektörü,
- Yabancı dil öğretimi,
- Dilbilim araştırmaları,
- İnsan-makine etkileşimi,
- Sesli yanıt ve uyarı sistemleri,
- Görme engelli kişilerle iletişim,
- Konuşma engelli kişilerle iletişim,

gibi birçok örnek verilebilir.

Bu tez çalışmasında; MKS (metinden konuşma sentezleme) yapılmış ve dinamik veritabanı kullanılmıştır. Bu kullanımın amacı, hem Türkçe kelimeleri hem de Türkçede bulunan ve hecelere ayrılamayan yabancı kelimeleri sentezleyebilmektir. Bu duruma ilişkin olarak bilimsel literatüre katkı yapmak amacıyla konuşma sentezleme işlemi için gerekli olan konuşma parçaları, sabit veritabanı yerine güncellenen bir ses veritabanı kullanılarak elde edilmiş ve kesintisiz çalışmayı amaçlayan bir sentezleme sistemi geliştirilmeye çalışılmıştır.

Bu çalışmada sırasıyla 1.adım girişi ve tezin amacını, 2.adım konuşma sinyalini, Türkçe dilindeki hecelerin yapısını, ses üretimini ve konuşma sentezlemenin tarihçesini, 3.adım bu çalışmada kullanılan materyali ve yöntemleri, 4.adım teze ait uygulama sonuçlarını ve bu sonuçlara ait irdelemeyi, 5.adım ise sonuç bölümünü oluşturacaktır.

Adım 2'de ilk olarak konuşma sinyalinin durağan kabul edildiği aralıktan ve kısımlarından bahsedilmiştir. Bu kısımların enerji değeri büyüklüğüne ve frekans bileşenlerinin durumuna göre hangi tür ünlü ve ünsüzlere ait oldukları belirtilmiştir. Konuşma sinyalinin ardından Türkçe dilindeki hecelerin yapısından bahsedilmiştir. Türkçe dilinde kaç çeşit hece yapısı

olduğu, hecelerin en az ve en çok kaç harften oluşabileceği, kaç ünlü ya da ünsüz harfin yan yana gelemeyeceği anlatılmıştır. Türkçe dilindeki hecelerin yapısından sonra ise ses üretiminden bahsedilmiştir. Ses üretimi kısmında; ses dalgasının ne tür bir dalga olduğu, ses üretim sistemini oluşturan anatomik yapıların ve bu sistemin temel kısımlarının neler olduğu, ses yolunun nasıl bir yapıya sahip olduğu, sesin üretilebilmesi için hangi temel kaynakların gerektiği ve konuşmayı oluşturan organların neler olduğu açıklanmıştır. Ses üretimi bahsedildikten sonrada, adım 2'nin son kısmı olan konuşma sentezlemenin tarihçesi anlatılmıştır. Burada, ilk olarak mekanik olan sistemlerin daha sonra elektriksel sistemlerle yer değiştirdiğinden ve metinden konuşma sentezleme sisteminin başlangıç aşamalarındaki gelişiminden bahsedilmiştir.

Adım 3'te çalışmada kullanılan materyal ve yöntemler belirtilmiştir. Boğumlama (söyleyiş) sentezleyicilerinin, insan ses sistemini modellemeye çalıştıklarından; formant sentezleyicilerin, beyaz gürültünün ya da başka bir sesin, filtre yardımıyla şekillendirilerek istenilen sesin üretimini amaçladıklarından ve son olarak da art arda bağlama (eklemeli) sentezleyicilerin önceden kaydedilmiş konuşma parçaları birleştirilerek sentezleme işlemi yaptıklarından bahsedilmiştir. Sırasıyla; insan ses sisteminin modellenmesi; seri, paralel formant sentezleyiciler ve eklemeli sentezleyicilerin genel akış diyagramı açıklanmıştır.

Adım 4'te teze ait uygulama sonuçları ve bu sonuçlara ait irdeleme belirtilmiştir. Uygulamada kullanılan Türkçe metinden konuşma sentezleme dinamik veritabanı akış diyagramının ne olduğundan ve nasıl bir mantıkla çalıştığından örneklerle birlikte detaylı olarak bahsedilmiştir. Akış diyagramındaki her biri adımı detaylı bir şekilde izah edilmiştir.

Tezin en son bölümü olan adım 5 ise, sonuç bölümünü oluşturmaktadır. Burada, uygulamanın konuşma sentezlemesine dair örnek gösterimler yer almaktadır. Ayrıca, çalışmanın artı ve eksi yönlerinin neler olduğundan, literatüre ne gibi bir katkı sağladığından, ileriki çalışmalarda daha fazla nasıl gelişmeler sağlanabileceğinden bahsedilmiştir.

2. KONUŞMA SİNYALİ, TÜRKÇE DİLİNDEKİ HECELERİN YAPISI, SES ÜRETİMİ VE KONUŞMA SENTEZLEMENİN TARİHÇESİ

2.1. Konuşma Sinyali

Konuşma sinyalinin durağan kabul edildiği aralık, 20-30 ms'lik kısımdır. Konuşma sinyali, ötümlü (sedalı) ve ötümsüz (sedasız) kısımlardan oluşur. Bunlardan ötümlü (sedalı) olan kısım, neredeyse-periyodik özellikte, enerji değeri büyük, alçak frekanslı bileşenler içerirken; ötümsüz (sedasız) olan kısım ise gürültü benzeri özellik gösteren, enerjisi düşük, yüksek frekanslı bileşenler içermektedir. Harflerden bazı ünsüzler (b, d, g, l, m, n, r) ve tüm ünlüler (a, e, ı, i, o, ö, u, ü) ötümlü (sedalı, voiced) özellikteyken; bazı sızıcı ünsüzler (h, j, s, s, z) ve patlamalı ünsüzler (p, t) ötümsüz (sedasız, unvoiced) özelliktedir [1].

Ünlüler

Oluşumları sırasında ses geçidinde belirli hiç bir takıntıya uğramayan, hiç bir engelle karşılaşmayan seslere ünlü denilmektedir. Ünlüler; ses yolunun açıklık derecesine, yani çene, dil ve dudakların durumuna göre çeşitlilik gösterir. Ünlüleri çene açısının durumuna, dudakların biçimine ve dilin devinimine bakılarak üç farklı grupta sınıflandırılabilir [1].

Çene açışının yapısına göre, geniş ve dar ünlüler olmak üzere iki kısıma ayrılırlar. Geniş ve dar ünlüleri içeren çene yapısına göre ünlüler, Çizelge 2.1'de görülmektedir.

Çizelge 2.1. Çene yapısına göre ünlüler

Geniş ünlüler:	a, e, o, ö
Dar ünlüler:	ı, i, u, ü

Dilin devinim yapısına göre, arkadil ve öndil ünlüleri olmak üzere iki kısma ayrılırlar. Arkadil ve öndil ünlülerini içeren dilin devinim yapısına göre ünlüler ise, Çizelge 2.2'de görülmektedir.

Çizelge 2.2. Dilin devinim yapısına göre ünlüler

Arkadil ünlüler:	ı, a, o, u
Öndil ünlüler:	i, e, ö, ü

Dudakların yapısına göre, düz ve yuvarlak ünlüler olmak üzere iki kısıma ayrılırlar. Düz ve yuvarlak ünlüleri içeren dudakların yapısına göre ünlülerde, Çizelge 2.3'de görülmektedir.

Çizelge 2.3. Dudakların yapısına göre ünlüler

Düz ünlüler:	ı, i, a, e
Yuvarlak ünlüler:	o, ö, u, ü

Ünsüzler

Ünsüzlerin oluşumu sırasında ise, ses yolunda belirli bir darlık, gevsek veya gergin bir kapantı olmaktadır. Ötümlülük (sedalık) özelliği açısından ünsüzler, üç farklı gruba ayrılabilir. Ötümsüz karşılıkları olan ötümlü ünsüzler, ötümsüz karşılıkları olmayan ötümlü ünsüzler ve ötümsüz ünsüzleri içeren; ötümlülük (sedalık) özelliği açısından ünsüzler Çizelge 2.4'de görülmektedir.

Çizelge 2.4. Ötümlülük (sedalık) özelliği açısından ünsüzler

Ötümsüz karsılıkları olan ötümlü ünsüzler	b, c, d, g, g, j, v, z
Ötümsüz karsılıkları olmayan ötümlü ünsüzler	l, m, n, r, y
Ötümsüz ünsüzler	ç, f, h, k, s, s, p, t

Oluşumları sırasında ses tellerinin titrediği sesler, ötümlü seslerdir. Ötümsüz seslerde ise, oluşumları sırasında ses telleri titreşmez.

Ünsüzler ayrıca; çıkış yerlerine, çıkış biçimlerine ve ses tellerinin titreşimine göre de sınıflandırılabilmektedirler [1].

Çıkış yerlerine göre; çift dudak ünsüzleri, dudak-diş ünsüzleri, dilucu-dişardı ünsüzleri, dilucu-dişeti ünsüzleri, dil-öndamak ünsüzleri, dilucu-öndamak ünsüzleri, dil-artdamak ünsüzleri ve gırtlak ünsüzleri olmak üzere sekiz kısma ayrılırlar. Çift dudak ünsüzleri, dudak-diş ünsüzleri, dilucu-dişardı ünsüzleri, dilucu-dişeti ünsüzleri, dil-öndamak ünsüzleri, dilucu-öndamak ünsüzleri, dil-artdamak ünsüzleri ve gırtlak ünsüzleri içeren; çıkış yerlerine göre olan ünsüzler, Çizelge 2.5'de görülmektedir.

Çizelge 2.5. Çıkış yerlerine göre ünsüzler

çift dudak ünsüzleri	b, p, m
dudak-diş ünsüzleri	f, v
dilucu-dişardı ünsüzleri	d, t
dilucu-dişeti ünsüzleri	n, r, s, z
dil-öndamak ünsüzleri	c, ç, j, s, y
dilucu-öndamak ünsüzleri	L
dil-artdamak ünsüzleri	g, k
gırtlak ünsüzleri	H

Çıkış biçimlerine göre; patlamalı ünsüzler, geniz ünsüzleri, çarpmalı ünsüzler, yan daralma ünsüzleri ve sürtünücü ünsüzler olmak üzere beş kısma ayrılırlar. Patlamalı ünsüzler, geniz ünsüzleri, çarpmalı ünsüzler, yan daralma ünsüzleri ve sürtünücü ünsüzleri içeren; çıkış biçimlerine göre ünsüzler ise, Çizelge 2.6'de görülmektedir.

Çizelge 2.6. Çıkış biçimlerine göre ünsüzler

Patlamalı ünsüzler	b, d, g, p, t, k
Geniz ünsüzleri	m, n
Çarpmalı ünsüzler	R
Yan daralma ünsüzleri	L
Sürtünücü ünsüzler	c, ç, f, h, j, s, s, v, y, z

Ses tellerinin titreşimine göre; ötümlü ünsüzler ve ötümsüz ünsüzler olmak üzere iki kısma ayrılırlar. Ötümlü ünsüzler ve ötümsüz ünsüzleri içeren; ses tellerinin titreşimine göre ünsüzlerde, Çizelge 2.7'de görülmektedir.

Çizelge 2.7. Ses tellerinin titreşimine ünsüzler

Ötümlü ünsüzler	b, c, d, g, j, l, m, n, r, v, y, z
Ötümsüz ünsüzler	ç, f, h, k, p, s, s, t

2.2. Türkçe Dilindeki Hecelerin Yapısı

Türkçe, diğer Türk dilleriyle birlikte Altay dil ailesinin bir kolunu oluşturmaktadır. Bu ailenin diğer üyeleri Moğolca, Mançu-Tunguzca ve Korecedir. Türkçe, diğer Altay dilleri gibi eklemeli, yani sözcüklerin eklerle yapıldığı ve çekildiği, sondan eklemeli bir dildir. Dilde özleşme çabaları 19. yüzyılın ikinci yarısında Tanzimat dönemi ile başlamıştır. Aydınların Türkçe sözcük kullanma ve Arap alfabesinde yenilik çabalarıyla geçen bir hazırlık döneminden sonra Cumhuriyetle birlikte çağdaş Türkçenin temelleri atılmıştır. Atatürk'ün özel ilgi ve çabalarıyla Latin alfabesine geçilmiş; tarama, derleme ve türetme yoluyla dildeki Türkçe sözcük oranı kısa sürede büyük oranlara ulaşmıştır.

Türkçe dili morfolojik olarak eklemeli bir dil olduğundan, bir sözcükten onlarca sözcük türetilebilmektedir. Türkçe dilinde heceler; en az bir, en çok dört harften oluşmaktadır. Kelime iki ünsüz ile başlamaz. İki ünlü arasında en fazla üç ünsüz olabilir. Kelime üç ünsüz ile bitemez. Türkçede en fazla dört harfli hece vardır. Heceleri u: ünlü, ve s: ünsüz olarak kodlarsak, Türkçede bulunan ve toplamda altı çeşit olan hece yapıları Çizelge 2.8'de verilmektedir.

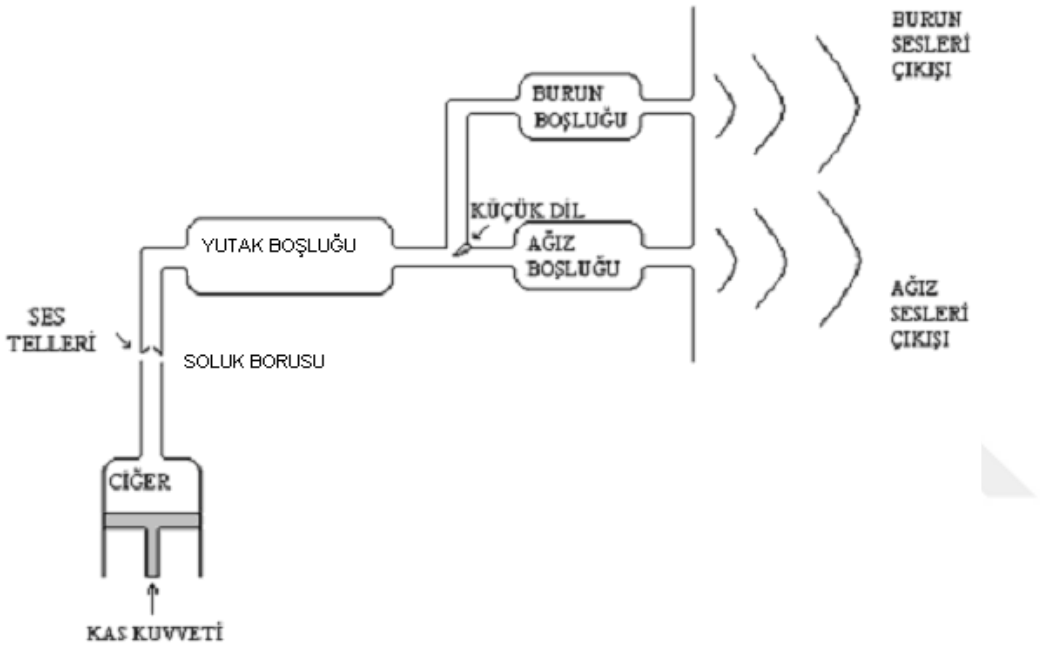
Çizelge 2.8. Türkçe hece yapısı

1.çeşit hece	u: " a, e, ı, i, o, ö, u, ü "
2.çeşit hece	su: " ba, be, bi, … "
3.çeşit hece	us: " ab, ac, aç, … "
4.çeşit hece	uss: " üst, alt, ırk, … "
5.çeşit hece	sus: " bel, gel, tır, … "
6.çeşit hece	suss: " türk, renk, yurt, … "

2.3. Ses Üretimi

Ses dalgası, akustik bir basınç dalgasıdır ve ses üretim sistemini oluşturan anatomik yapıların istemli hareketleri sonucunda meydana gelir. Bu sistemin temel kısımları; burun boşluğu, gırtlak, nefes borusu, boğaz, ağız boşluğu ve ciğerlerdir. Ses üretimi için gerekli olan bu anatomik yapı kısımları, değişik pozisyonlar alarak farklı sesleri meydana getirirler. Şekil 2.1'de ses yolunun nasıl bir yapıda olduğu görülmektedir. İnsan sesi dalga şekline bakıldığında, zamanla bir değişim olduğu gözlemlenmektedir. Konuşma sesleri, ses

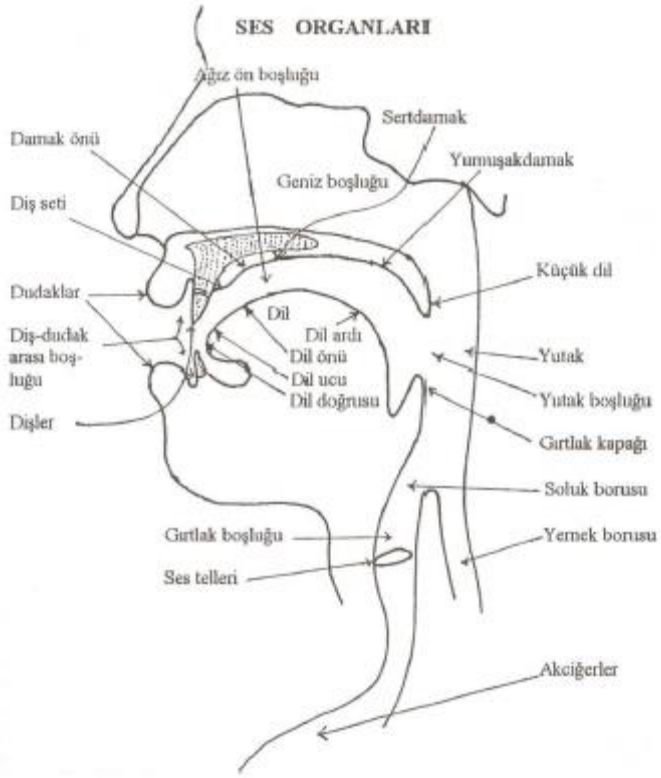
parçalarına ayrılabilir ve bu ses parçaları da kısa süreler boyunca benzer akustik özellikler gösterebilirler. Zamana bağlı dalga şekillerine bakılan ses sinyallerinin; sinyal periyotları, süreleri, yoğunlukları ve her bir ses parçasının sınırları tespit edilebilir. Fakat art arda gelen sesler birbirlerini etkilerse bu sınırlar tam belirlenemeyebilir. İnsanların ses üretme ve sesi algılama sistemlerindeki organların yapılarından kaynaklanan bazı kısıtlamalar nedeniyle, ses dalga seklinde değişime sebep olan yapıların pozisyonlarının kısa zaman aralıklarında değişmediği düşünülebilir [2].



Şekil 2.1. Ses yolunun yapısı [2]

Ses yolunun nasıl yapıya sahip olduğu şekil 2.1‘de görülmektedir. Sesin üretilebilmesi için gerekli olan temel kaynak, diyafram ve akciğerlerdir. Hava akımı, ses telleri ve gırtlaktan geçerek burun ve yutağa boşluğuna ulaşır, ardından burun ve ağızdan dışarı çıkar. Ses sistemindeki en önemli kısım, nefes borusunun bitiminde, ses tellerinin arasında bulunan V seklindeki açıklıktır. Ses tellerinde hava akımı bu açıklıkta modüle edilerek sesli ve nefesli sessiz harfler oluşturulur. Ses tellerindeki temel titreşim frekansları farklılık göstermektedir. Ses tellerinin uzunluğu, gerginliği ve ciğerlerden gelen hava basıncı ayarlayarak titreşim frekansını değiştirebilir. Bu frekanslar çocuklarda 300 Hz, kadınlarda 200 Hz, erkeklerde ise 110 Hz civarlarındadır. Ağız boşluğu ve gırtlak arasında bulunan yutağın boyutu gırtlağın yükselip alçalması ile az miktarda değişmektedir. Ağız boşluğunun akustik yapısı ve boyutu;

dil, damak, dudaklar, yanaklar ve dişlerin hareketi ile değişmektedir. Burun boşluğunun boyutu ise sabit olup buraya giren hava akımı yumuşak damak ile kontrol edilmektedir [3].



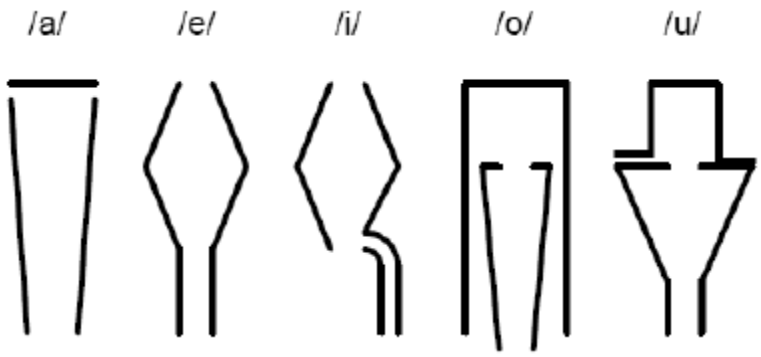
Şekil 2.2. Konuşmayı oluşturan organlar [4]

Konuşmayı oluşturan organlar Şekil 2.2'de görülmektedir. Konuşmanın üretilmesi işlemi, Şekil 2.2'de gösterilen bu organların temel görevi değildir. Bunların temel görevleri, birbirinden farklıdır. Mesela, akciğerlerin temel görevi solunum yoluyla kanın temizlenmesini sağlamak iken, dilin temel görevi ise tat alma ve yenenleri yutmaya yardımcı olmaktır. Ancak bu organların temel görevleri birbirinden farklı olmasına rağmen, ikinci görevlerinin amacı hepsi için aynıdır yani konuşmanın üretilmesine yardımcı olmaktır. Konuşmanın üretilmesinde bu organların hepsinin ayrı bir önemi vardır [4].

2.4. Konuşma Sentezlemenin Tarihçesi

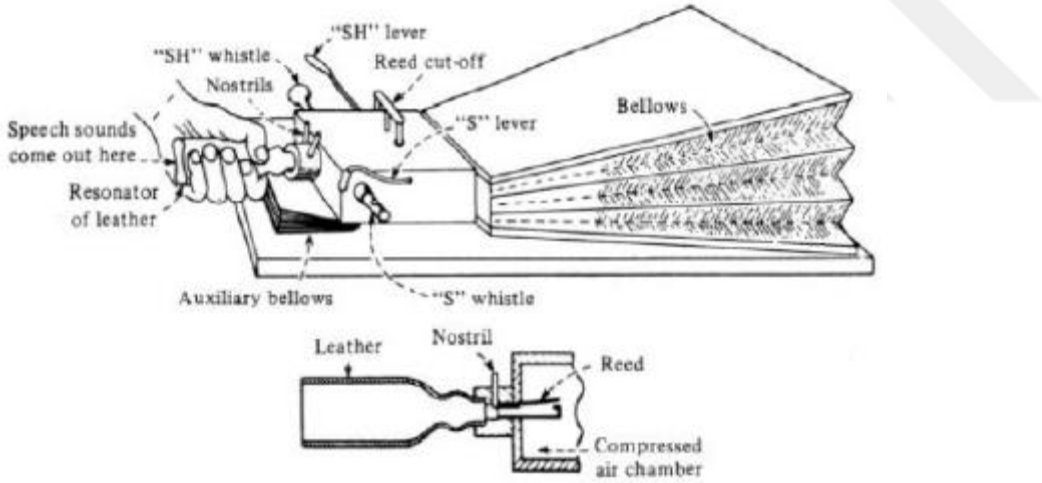
Metinden konuşma sentezleme sistemlerinin gelişimine bakıldığında, ilk başlarında mekanik olan sistemler daha sonra elektriksel sistemlerle yer değiştirmiştir. Mekanik sistem olarak ilk defa 1779 yılında Christian Kratzenstein tarafından 'a, e, i, o, u' ünlü seslerindeki

fizyolojik farklılıklar belirlenmiş ve çınlayıcı yardımıyla yapay olarak sentezlenmiştir [5]. Kratzenstein'in kullandığı çınlayıcılar, Şekil 2.3'de gösterilmektedir.



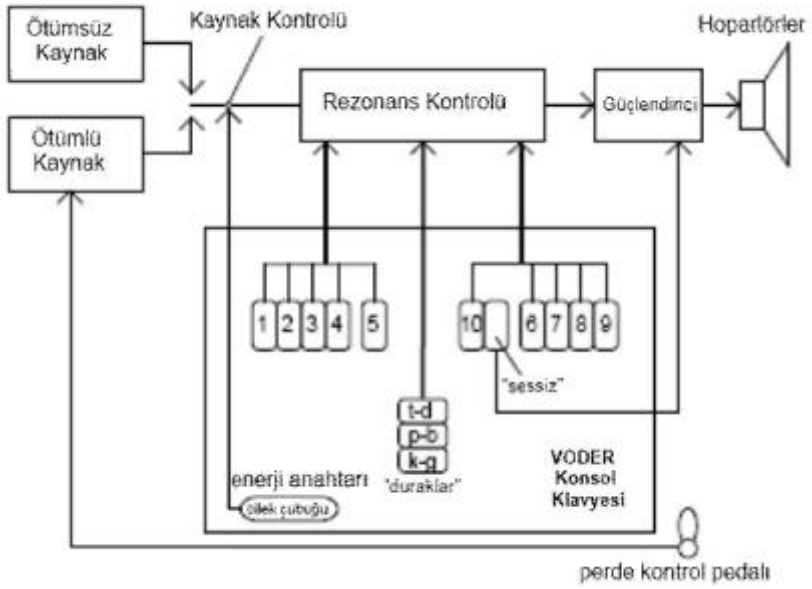
Şekil 2.3. Kratzenstein'in kullandığı çınlayıcılar [5]

Kempelen tarafından 1791'de ses yolunun, akustik telaffuzun esas kaynağı olduğu bulunmuştur. Bu ses cihazı, Şekil – 2.4'de gösterilmektedir.



Şekil 2.4. Kempelen ses cihazı [6]

1838 yılında Willis tarafından, ünlüler ile ses yolunun geometrisi arasındaki bağıntı belirtilmiştir. 1922 yılına gelindiğinde, elektriksel sistem olarak ilk kez Stewart tarafından ünlüleri ilk iki formantlarından sentezleyen iki çınlayıcı makine yapılmıştır. 1939 yılında ise Voice Operating Demonstrator (VODER) ismindeki sistem yapılmış ve bu sistem ilk konuşma sentezleyici olarak kabul görmüştür. Bu konuşma sentezleyici Şekil 2.5'de gösterilmektedir.



Şekil 2.5. İlk konuşma sentezleyici VODER [6]

1953'te Fant tarafından seri formant sentezleyici, Lewrance tarafından ise üç çınlayıcılı paralel formant sentezleyici yapılmıştır. İlk söyleyiş sentezleyicisi olan Dynamic Analog of the Vocal Tract (DAVO) 1958 yılında Rosen tarafından; İngilizce için ilk metinden konuşma sentezleyici 1968 yılında Umeda tarafından ve görme engelliler için tasarlanan ilk okuma sistemi ise 1976 yılında Kurzweil tarafından yapılmıştır. MITalk isimli MKS sistemi Allen, Hunnicutt ve Klatt tarafından 1979 yılında gerçekleştirilmiştir. Sinusoidal modelleme yöntemi MIT'deki McAuley ve Quatieri tarafından 1984 yılında önerilmiştir. 1985'te ise Fransız Telekom tarafından PSOLA yöntemi önerilmiştir [6]. Burada bahsedilen metinden konuşma sentezleme sisteminin başlangıç aşamalarındaki gelişimi, aşağıdaki çizelgede sunulmuştur.

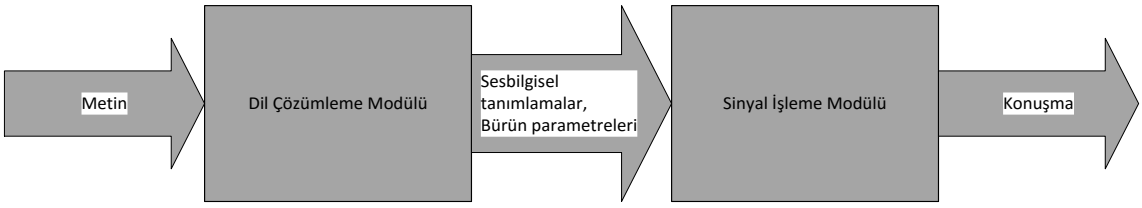
Çizelge 2.9. Metinden konuşma sentezleme sisteminin başlangıç aşamalarındaki gelişimi

Geliştirildiği Yıl	Geliştiren Kişi/Kurum	Yöntem/Kaynak/Cihaz
1779	Kratzenstein	Çınlayıcı yardımıyla yapay sentezleme
1791	Kempelen	Ses yolu
1838	Willis	Ünlüler ile ses yolu geometrisi bağıntısı
1922	Stewart	Ünlüleri ilk iki fortmanlarından sentezleme
1939	Dudley	VODER (ilk konuşma sentezleyici)
1953	Lewrance ve Fant	Formant sentezleyici
1958	Rosen	DAVO (ilk söyleyiş sentezleyicisi)
1968	Umeda	İngilizce için ilk metinden konuşma sentezleyici
1976	Kurzweil	Görme engelliler için tasarlanan ilk okuma sistemi
1979	Allen, Hunnicutt ve Klatt	MITalk isimli MKS sistemi
1984	McAuley ve Quatieri / MIT	Sinüsoidal modelleme yöntemi
1985	Fransız Telekom	PSOLA yöntemi

Literatüre bakıldığında temel olarak; boğumlama (söyleyiş), formant ve art arda bağlama (eklemeli) sentezleyici yöntemlerinin kullanıldığı görülmektedir. Günümüzde yapılan çalışmalara bakıldığında ise, genellikle eklemeli sentezleyicilerin kullanıldığı ve bununla bağlantılı olarak da PSOLA yöntemlerinin tercih edildiği anlaşılmaktadır.

3. MATERYAL VE YÖNTEM

Konuşma sentezleme, yazılı olan bir metnin konuşmaya dönüştürülmesi işlemidir. Konuşmaya dönüştürülen yazılı metin dilleri, yapılan çalışmanın amacına göre Türkçe dilide dahil olmak üzere birçok dilde olabilmektedir.



Şekil 3.1. Metinden konuşma sentezleme sisteminin blok şeması [6]

Genel olarak bir metinden konuşma sentezleme sisteminin blok şeması Şekil 3.1'deki gibi ifade edilmektedir. Bu blok şemaya bakıldığında, dil çözümleme modülü ve sinyal işleme modülü olmak üzere iki ana bölümden oluştuğu görülmektedir. Burada, öncelikle dil çözümleme modülünde, metinden sentezlenecek konuşmayı oluşturan ses bilgilerinin elde edilmesi amacıyla, konuşmaya çevrilecek metin önişlemlerden geçirilir. Daha sonra, sinyal işleme modülünde ise ses bilgisel tanımlamalar ve bürün parametreleri kullanılarak konuşma oluşturulur [6].

Mevcut olan metinden konuşma sistemleri, MITalk, Infovox, AT&T Labs Natural Voices & Bell Labs TTS, ETI Eloquence / Speechworks / Nuance, MBROLA, Festival TTS, GVZ TTS, Multext, Genglish ve HTS'dir [7,8]. Bu sistemlerle ilgili bilgiler Çizelge 3.1'de ve devamında verilmektedir.

Çizelge 3.1. Mevcut olan metinden konuşma sentezleme sistemleri

MTS sistemi	Geliştiren kurum/kişi	Sentezleme sistemi
MITalk	MIT Laboratuvarları	Formant temelli sentezleyici
Infovox	İsveç Royal Institute of Technology	Basamaklı formant sentezleme, birleştirmeli yöntem
AT&T Labs Natural Voices & Bell Labs TTS	Bell Laboratuvarları	Birim seçme yöntemi, eklemeli sentezleme
ETI Eloquence / Speechworks / Nuance	Eloquence Eloquent Teknoloji	Eklemeli ve çok dil destekli sistem
MBROLA	Belçika Mons Polytechnic Üniversitesi	İkili fonem birleştirme
Festival TTS	Edinburg Üniversitesi	İkili fonem birleştirme
GVZ TTS	Setsek firması	Birleştirmeli yöntem
Multext	Provence Üniversitesi	-
Genglish	Mons Polytechnic Üniversitesi	MATLAB'ta toolbox olarak geliştirilmiş
HTS	Nagoya Teknoloji Enstitüsü	HMM tabanlı geliştirilmiş

- MITalk, formant temelli bir sentezleyicidir. MIT laboratuvarlarında 1979 yılında geliştirilmiştir. Allen, Hunnicutt ve Klatt tarafından geliştirilen bu metinden konuşma sentezleme sistemi, günümüzdeki çoğu formant sentezleyici çalışmaları için temel oluşturmaktadır [8].

- Infovox, ticari bir metinden konuşma sentezleme uygulamasıdır. Bu metinden konuşma sentezleme uygulaması, İsveç Royal Institute of Technology'de geliştirilmiştir. Birçok dili destekleyen ve en tanınmış uygulamalardan biridir. 1982 yılında geliştirilen başlangıç sürümünde basamaklı formant sentezleme yöntemi, daha sonraki sürümlerinde ise birleştirmeli yöntem kullanılmıştır [8].

- AT&T Labs Natural Voices & Bell Labs TTS sistemi, birim seçme yöntemini uygulayan eklemeli sentezleme altyapısına sahiptir. Desteklediği diller için iyi sayılabilecek sonuçlar veren bir ticari uygulamadır [8].

- ETI Eloquence / Speechworks / Nuance, eklemeli çok dil desteği sunan bir sistemdir ve Eloquence Eloquent Teknoloji tarafından geliştirilmiştir [8].

- MBROLA sisteminde kullanılan sistem, Belçika Mons Polytechnic Üniversitesi'nde geliştirilmiştir ve ikili fonem birleştirme işlemi yapmaktadır. Bu işlem, PSOLA'ya (Pitch Synchronous Overlap Add) benzeyen bir metod olarak belirtilmektedir. Bu sistem, tam olarak bir metin seslendirme sistemi değildir ve giriş olarak metin yerine gereken fonemleri, frekans ve süre bilgilerini ister [9].

- Festival TTS sistemi, 90'lı yılların sonunda Edinburg Üniversitesinde bulunan Ses Teknolojileri Araştırma Merkezinde geliştirilmiştir. Bu sistem, ikili fonem birleştirme yöntemini kullanır. Hem araştırma hem de bireysel kullanımda kaynak kodu açık olan ücretsiz bir uygulamadır [10].

- GVZ TTS, ticari bir uygulamadır ve birleştirmeli yöntemi kullanır [8].

- Multext, Provence Üniversitesi'nde geliştirilmiştir [11].

- Genglish, Mons Polytechnic Üniversitesi'nde MATLAB'ta toolbox olarak geliştirilmiştir [12].

- HTS, Nagoya Teknoloji Enstitüsü'nde HMM tabanlı geliştirilmiştir [13].

Konuşma sentezleme uygulamaları ile ilgili olarak şu ana kadar yapılmış olan çalışmalara bakıldığında ise, yöntem olarak boğumlama (söyleyiş), formant ve art arda bağlama (eklemeli) sentezleyicilerin kullanıldığı görülmektedir. Ilk olarak insan ses sistemini modellenmeye çalışan boğumlama (söyleyiş) sentezleyicilerden bahsedilmiştir.

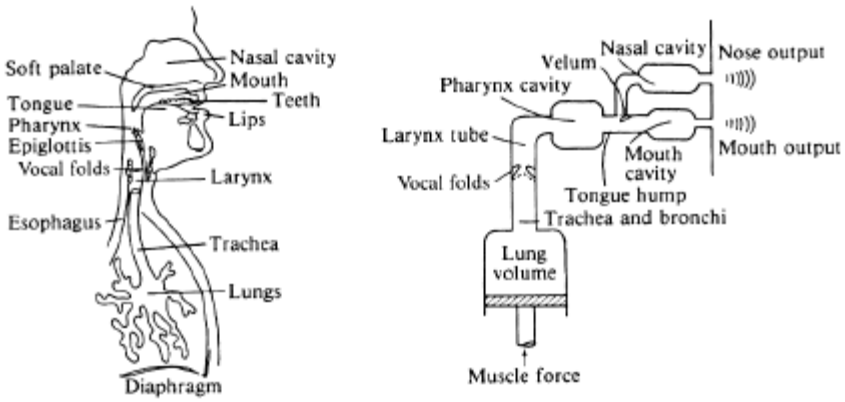
3.1. Boğumlama (Söyleyiş) Sentezleyicisi

Boğumlama (söyleyiş) sentezleyicileri, insan ses sistemini modellemeye çalışırlar. Bu sebeple konuşma organlarımızı modellemeye çalışan bu sentezleyicilerin yapısı karmaşık olmakla birlikte doğal konuşma sentezinde başarılı sonuçlar vermektedirler. Ses tellerinde açığa çıkan titreşimler; dudak, çene, burun ve ağızdan geçerken şekillendirildiğinden dolayı,

boğumlama (söyleyiş) sentezleyicileri bu organlarda oluşan süreçleri modelleyerek konuşma sentezi yapmaya çalışmaktadırlar [14].

Bu sentezleyicilerin amacının, tüm sesbirimlerin, insanın ses üretim mekanizmasında gerçekleştirilme şeklinin en doğru şekilde modellenmesi olmasından dolayı çok fazla parametre içeren bu modelin oluşturabildiği konuşma kalitesi, formant sentezleyiciler ile kıyaslandığında daha iyidir. Ancak işlem ödünleşimi de bir hayli fazladır [14].

Boğumlama (söyleyiş) sentezleyicilerinde modellenmeye çalışılan insan ses sistemi aşağıdaki şekilde görülmektedir.



Şekil 3.2. İnsan ses sistemi ve modellenmesi [14]

Konuşma sentezleme yöntemlerinden ikinci olarak ise, formant sentezleyicilerden bahsedilmiştir.

3.2. Formant Sentezleyici

Son zamanlarda en çok kullanılan konuşma sentezleme yöntemlerinden biri olan formant sentezleyicilerde, beyaz gürültünün ya da başka bir sesin, filtre yardımıyla şekillendirilerek istenilen sesin üretimi amaçlanmaktadır. Formant sentezleyiciler yöntemi, sonsuz sayıda ses üretimine imkan sağladığından dolayı, diğer konuşma sentezleme yöntemleriyle kıyaslandığında bu yöntemin oldukça esnek bir yapıya sahip olduğu görülmektedir [8].
Konuşma sinyalinin doğrusal ön görünümlü kodlaması (linear predictive coding, LPC) temelini esas alan formant sentezleyiciler için aşağıdaki 3.1 nolu denklem; p süzgecin

derecesi olmak üzere konuşma sinyalinin n. örneğinin, önceki p adet örneğin doğrusal kombinasyonu şeklinde ifade edilmektedir [14].

$$s(n) = -\sum_{k=1}^{p} a_k \mathrm{s(n-k)} + e(n) \quad (3.1)$$

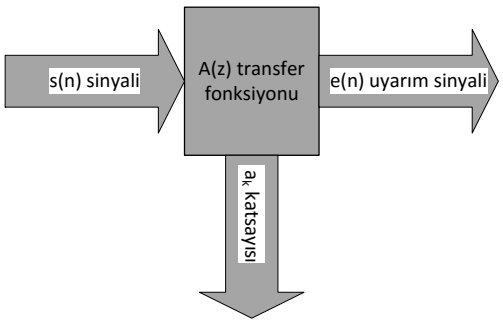
Burada; LPC katsayısı olarak isimlendirilen a_k'lar, konuşmanın çerçeve süresinde yani konuşmanın durağan kabul edildiği 20-30 ms'lik kısa sürelerde sabittir. 3.2 nolu denklemde daha detaylı ifade edilen e(n) ise, sentezleme için uyarım sinyali olarak isimlendirilen ve konuşmanın gerçek değeriyle öngörülen değeri arasındaki fark olan hata sinyalidir. [14]

$$e(n) = s(n) - \hat{s}(n) = s(n) + \sum_{k=1}^{p} a_k \mathrm{s(n-k)}$$
(3.2)

Aşağıda ifade edilen (3.3) nolu denklem ise; z-dönüşümü ile transfer fonksiyonunun hesaplanmasını göstermektedir. [14]

$$\frac{\mathrm{E(z)}}{\mathrm{S(z)}} = A(z) = 1 + \sum_{k=1}^{p} a^k z^{-k} \quad (3.3)$$

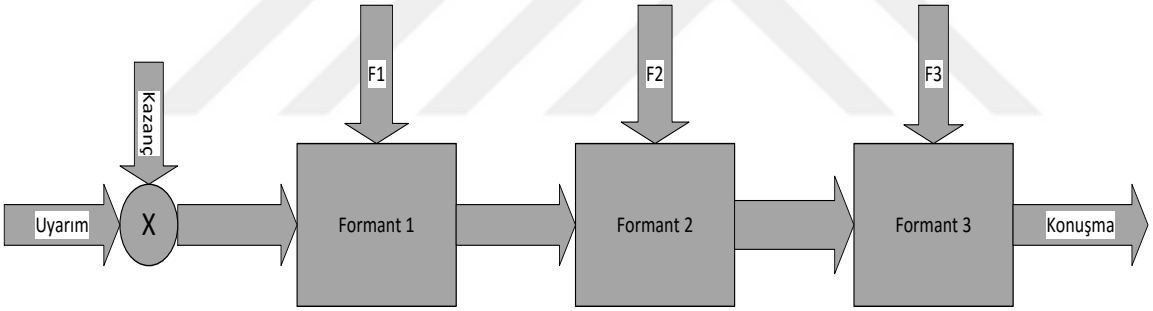
Konuşma sinyalinin, bakıldığı zaman aralığında, e(n) uyarım sinyaline ve a_k süzgeç (LPC) katsayılarına ayrıştırılması, bu işlemlerin sonucunda yani şekil 3.3 'de gösterilen sonlu uzunlukta dürtü tepkili bir süzgeç analiz filtresi yardımıyla gerçekleştirilir [14].



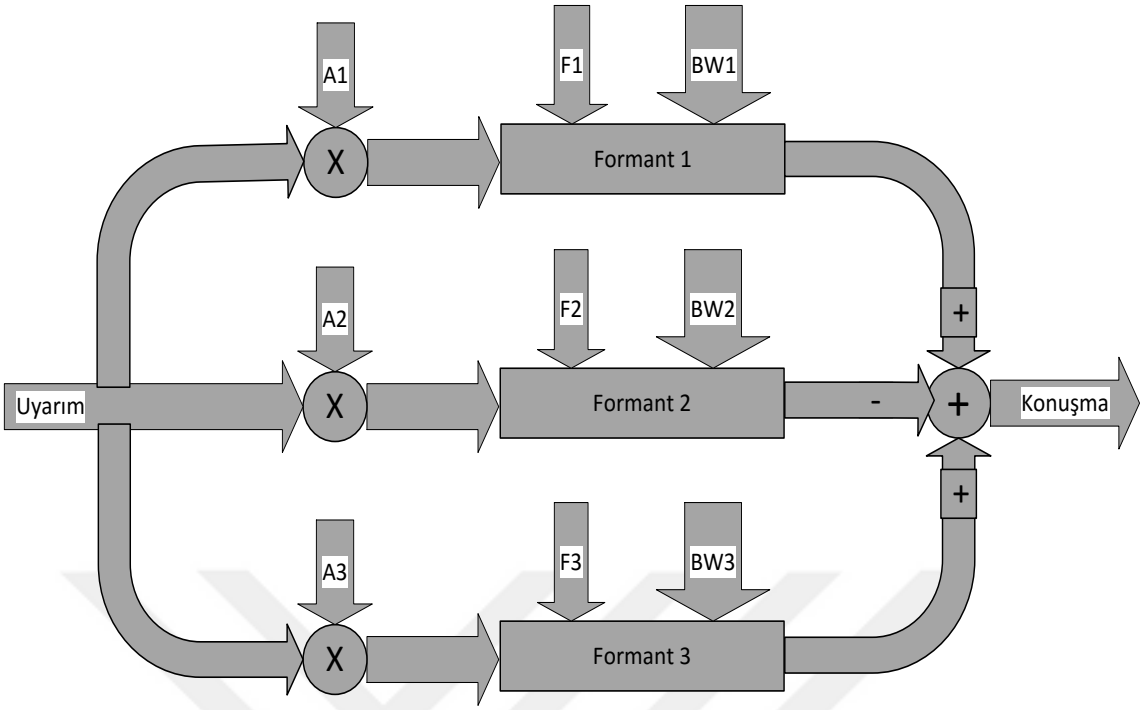
Şekil 3.3. LPC (doğrusal ön görünümlü kodlama) analiz süzgeci [14]

Şekil 3.3'de gösterilmiş olan LPC analiz süzgecinin tersi; LPC sentez filtresi 1/A(z) olarak bilinmektedir ve bu sentez filtresinin genlik tepkisinde bulunan tepeler, ses yolunun rezonans frekanslarına karşılık gelen formant frekansları olarak isimlendirilmektedirler. Böylece formant sentezleyiciler, konuşmayı, formant frekansları ve bant genişlikleri olan bu iki ana bileşenlerine ayarlayarak sentezleme işlemini yapmaya çalışır [14].

Formant sentezleyiciler; denetim bilgisi olarak yalnız formant frekanslarına ihtiyaç duyan seri formant sentezleyiciler ve her bir formant frekansı için ayrı bant genişliği, genlik, frekans parametrelerine ihtiyaç duyan paralel formant sentezleyiciler olmak üzere ikiye ayrılırlar. Seri formant sentezleyicilerin daha az denetim bilgisine sahip olması, paralel formant sentezleyicilere göre daha kolay gerçekleştirilebilir olduğunu göstermektedir. Aynı zamanda paralel formant sentezleyicilerinde ihtiyaç duyduğu parameter sayısının fazla olması, seri formant sentezleyicilerine göre daha karmaşık yapıda olduklarını belirtmektedir [5]. Aşağıda şekil 3.4 ve şekil 3.5 'te sırasıyla seri ve paralel formant sentezleyicilerin yapıları görülmektedir.



Şekil 3.4. Seri formant sentezleyici [5]

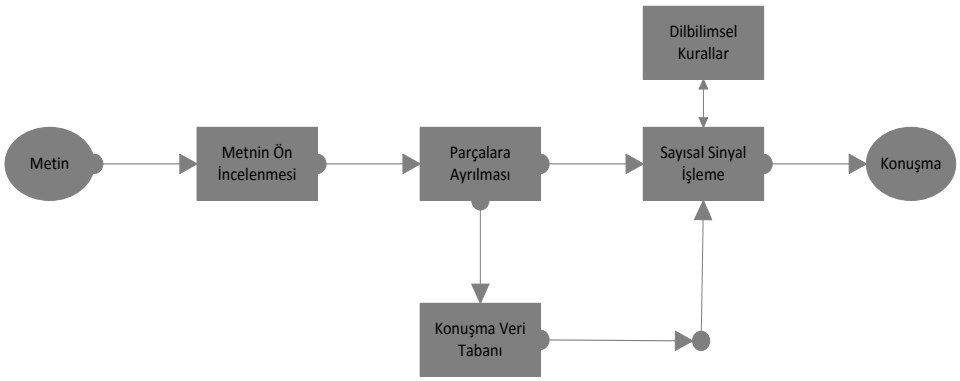


Şekil 3.5. Paralel formant sentezleyici [5]

Son olarakta tez kapsamında kullanılan ve konuşma sentezleme yöntemlerinden biri olan art arda bağlama (eklemeli) sentezleyicilerden bahsedilmiştir.

3.3. Art Arda Bağlama (Eklemeli) Sentezleyici

Art arda bağlama (eklemeli) sentezleyicilerde; önceden kaydedilmiş konuşma parçaları birleştirilerek sentezleme işlemi yapılır. Bu tip sentezleyicilerde birleştirme için kullanılan konuşma parçaları; sözcük, seslem (hece), trifon, difon ya da fonem olabilir. Bunların içerisinden difonlar; fonemler arası ses geçişlerini içermesinden, uygun sayıda olmasından ve de kayıt yüklerinin fazla olmamasından dolayı diğer konuşma parçalarına kıyasla daha çok tercih edilmektedirler [15].



Şekil 3.6. Eklemeli sentezleyiciler için genel akış diyagramı [14]

Eklemeli sentezleyiciler için genel akış diyagramı Şekil 3.6'de görülmektedir. Burada; 'Metnin Ön İncelenmesi' kısmında, metnin içerdiği sayıların, kısaltmaların, noktalama işaretlerinin ön incelemesi yapılır. 'Parçalara ayrılması' kısmında, metni oluşturan ses parçalarının bulunması aşaması gerçekleşir. 'Sayısal Sinyal İşleme' kısmı ise; temel frekans, enerji ve süreleri ayarlanan ses parçalarının örtüştürülüp eklenmesi işleminin yapıldığı kısımdır [14].

Bu sentezleme işlemi için, birleştirilmesi gereken konuşma parçalarının veritabanının hazırlanması gerekmektedir ve bu kayıt aşaması uğraştırıcı bir iştir. Ayrıca sentezlenecek olan metinlerde bulunan kelimelerin ilgili dile ait olmayan veya o dile yeni girmiş olan kelimeler olması durumunda, veritabanında tam karşılıkları bulunamayacağından sentezleme işlemlerinde sorunlara yol açabilme potansiyelleri vardır. Bizim çalışmamızda ise gerçekleştirdiğimiz güncellenen ses veritabanı ile bu ve benzeri şekilde klasik sabit veritabanlarında karşılaşılabilecek olumsuzlukların önüne geçilmesi planlanmaktadır.

Eklemeli sentezleyicilerde kendi içerisinde; birim seçimi sentezleme, difon sentezleme ve alan (domain) temelli sentezleme olmak üzere üç kısma ayrılırlar. Birim seçimi sentezleme, konuşmayı sentezlemek için çok sayıda önkayıtlı kelimelerin kullanıldığı bir yöntemdir ve bu sentezlemede, veritabanı, iyi kalitede sentezlenmiş konuşma yapmak için uygun şekilde etiketlenmiş olmalıdır. Difon sentezlemede ise, istenilen dildeki tüm difonları içeren nispeten küçük boyutlu bir veritabanı kullanır. Burada, sentezlenen konuşmanın kalitesi, birim seçimi sentezlemeye göre kısmen daha düşüktür. Alan temelli sentezleme, belirli bir uygulama ile ilgili cümlelerden ve sözcük gruplarından oluşur. Bu tür sentezleme ise, demiryolu duyuru sistemi, hava raporu vb. gibi birçok uygulamalarda kullanılabilir [16].

Eklemeli sentezleyicilerde işlem yükü; boğumlama ve formant sentezleyicilere göre daha azdır ve konuşmanın doğallığı da daha fazladır. Ayrıca; perde frekansı, enerji seviyesi ve fonem uzunluğu da; konuşma için önemli unsurlar arasında yer alır. Bu sentezleyicilerde, birleştirme işleminde sesler arasında yumuşak geçişler için perde frekansı değiştirilmesi ile Perde Eşzamanlı Örtüştürerek Ekleme (Pitch Synchronous Overlap and Add, PSOLA) yöntemi kullanılabilir [17].

PSOLA yöntemi; özellikle sedalı konuşma sinyalindeki tepe noktaları çevresindeki yerel bölgedeki kısa dönem sinyallerin elde edildikten sonra, perde frekansı degiştirilerek, elde edilmiş olan örneklerin üst üste eklenmesi ve yeni perde frekanslı konuşma sinyalinin elde edilmesi mantığına dayanır [3]. Temel olarak Zaman Ekseninde Perde Eşzamanlı Örtüştürerek Ekleme (Time Domain-PSOLA, TD-PSOLA), Frekans Ekseninde Perde Eşzamanlı Örtüştürerek Ekleme (Frequency Domain-PSOLA, FD-PSOLA) ve Doğrusal Öngörünümlü Perde Eşzamanlı Örtüştürerek Ekleme (Linear-Predictive PSOLA, LP-PSOLA) olmak üzere üç çeşidi vardır [16].

Zaman Ekseninde Perde Eşzamanlı Örtüştürerek Ekleme yönteminde; konuşma sinyalindeki yerel tepe noktaları etrafında pencereleme işlemi yapılıp 3.4 nolu denklemdeki kısa dönem sinyali ve elde edilen kısa dönemli parçalar daha sonra istenilen perde periyodu kadar aralıklarla art arda getirilip toplanarak da 3.5 nolu denklemdeki yeniden oluşturulan konuşma sinyali elde edilir [17].

$$x_m(n) = \mathrm{h}(\mathrm{n} - \mathrm{m}T_0)x(n) \tag{3.4}$$

Yukarıda ifade edilen 3.4 nolu denklemde; m'inci dönem sinyali $x_m(n)$, kullanılan pencere h(n), yerel tepe noktaları arasındaki uzaklık T_0 ve konuşma sinyali ise x(n) olarak gösterilmektedir. Pencerenin orta noktası, yerel tepe noktasına ayarlanır ve pencere uzunluğu, perde periyodunun iki katı olarak seçilir. Aşağıdaki 3.5 nolu denklemde ise; eğer T perde periyodu ile konuşma sinyali yeniden oluşturulmak istenirse, bu $\tilde{x}(n)$ yeniden oluşturulan konuşma sinyalinin nasıl ifade edildiği gösterilmektedir [17].

$$\tilde{x}(n) = \sum_{m=-\infty}^{\infty} x_m(\mathrm{n} - \mathrm{m}(\mathrm{T} - T_0)) \tag{3.5}$$

Tezin uygulama kısmında detaylı olarak bahsedilen Türkçe metinden konuşma sentezleme dinamik veritabanı akış diyagramı mantığı kullanılarak yapılan çalışmada, sentezlenmesi istenen metindeki kelimelerin hecelenip hecelenmediğine dikkat edilerek kullanıldıkça sürekli güncellenen veritabanında sentezleme işlemi için konuşma parçası olarak heceler kullanılmıştır.

4. UYGULAMA

Hem Türkçe hem de hecelere ayrılamayan Türkçedeki yabancı kelimeleri kapsayan ve kullanıldıkça yeni hecelerle birlikte sürekli olarak kendi kendini güncelleyen bir ses veritabanı oluşturmayı amaçlayan Türkçe metinden konuşma sentezleme uygulaması, eklemeli sentezleyicilerin mantığını kullanarak C# programlama dili aracılığıyla gerçekleştirilmiştir.

Türkçe metinden konuşma sentezleme sistemlerinde klasik sabit veritabanlarının kullanılmasının, bu veritabanlarında yeni kullanılmaya başlayan ve hecelere ayrılamayan Türkçedeki yabancı kelimelerin bulunmaması durumunda sentezleme işlemlerinde yetersiz kalıp sorunlara yol açabilme ihtimalleri vardır. Bu sebeple gerçekleştirilmiş olan ve güncellenen veri tabanına sahip olmayı amaçlayan metinden konuşma sentezleme uygulaması; uzun süren, zahmetli bir yöntem olan ve Türkçedeki hecelerin teker teker ses kayıtları alınarak oluşturulan klasik sabit ses veritabanı oluşturulmasına bir alternatif olmayı amaçlamaktadır. Alt başlıkta uygulamanın akış diyagramından bahsedilmiştir.

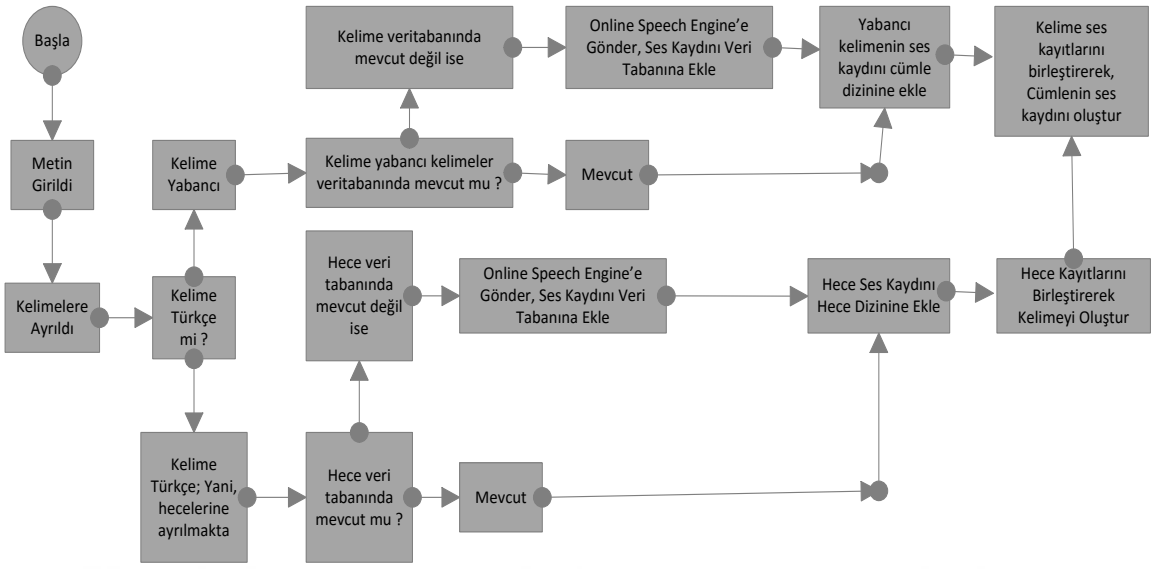
4.1. Türkçe Metinden Konuşma Sentezleme Dinamik Veritabanı Akış Diyagramı

Bu çalışmada uygulanan yöntemde geliştirilen uygulama, ilk defa çalıştırıldığı sistemde aşağıdaki sıralanan adımları listeleyerek kendine bir ses veritabanı oluşturur ve ihtiyaçları doğrultusunda kendini günceller:

1. Metini kelimelere ayır,
2. Kelimenin hecelenip hecelenmediğine göre Türkçe olup olmadığına karar ver,
2.1. Eğer kelime hecelenebiliyorsa, heceleri veritabanında ara,
2.1.1. Heceler veri tabanında mevcut ise ilgili metnin dizinine ekle,
2.1.2. Heceler veri tabanında mevcut değil ise ses motoruna heceyi okutarak Türkçe hece ses veritabanına heceyi ekle ve adım 2.1.1'e git,
2.2. Eğer kelime hecelenemiyorsa yabancı dilden gelen kelimeler veritabanında ara,
2.2.1. Eğer kelime veritabanında mevcut ise ilgili metnin dizinine ekle,
2.2.2. Eğer kelime mevcut değil ise ses motoruna kelimeyi Türkçe olarak okutarak yabancı kelimeler veritabanına ekle ve adım 2.2.1' git,
3. Hece ve kelime ses kayıtlarını birleştirerek metnin sese çevrilmiş halini elde et,

4. Sesi çal ve ses dalgasının görselini arayüze çizdir,
5. Yeni metin girildiğinde 1. adımdan başla.

Uygulamanın çalışma adımları yukarıda maddeler halinde belirtilmiştir. Burada, öncelikle Türkçe metinden konuşma sentezleme uygulamasına girilen metin kelimelere ayrılır. Uygulamanın ilk adımında yapılan bu işlemden sonra, bu girilen metnin ayrılan kelimelerinin hecelenip hecelenmediğine bakılır. Bu adımda yani 2. adımda eğer kelime hecelenebiliyorsa bu kelimenin Türkçe olduğuna; eğer kelime hecelenemiyorsa da hecelere ayrılamayan Türkçedeki yabancı kelimelerden biri olduğuna karar verilir. Bu karar verildikten sonra, eğer kelimeler hecelenebiliyorsa yani kelimeler Türkçe ise, heceleri veri tabanında aranır; bu aramadan sonra eğer heceler veritabanında mevcut ise heceleri ilgili metnin dizinine eklenir; eğer heceler veri tabanında mevcut değil ise ses motoruna heceyi okutarak Türkçe ses veri tabanına heceyi ekler ve ilgili metnin dizinine ekler yani adım 2.1.1'e gider. Karar verildikten sonra, eğer kelimeler hecelenemiyorsa yani kelimeler hecelere ayrılamayan Türkçedeki yabancı kelimelerden biri ise, heceleri yabancı dilden gelen kelimeler veritabanında aranır; bu aramadan sonra eğer kelime veritabanında mevcut ise kelime ilgili metnin dizinine eklenir; eğer kelime mevcut değil ise ses motoruna kelimeyi Türkçe olarak okutarak yabancı kelimeler veritabanına ekler ve ilgili metnin dizinine ekler yani adım 2.2.1'e gider. Veritabanlarında olan ya da güncellenen bu heceler ve kelimeleri ilgili metnin dizinine ekleme işleminden sonra; hece ve kelime ses kayıtları birleştirilerek metnin sese çevrilmiş hali elde edilir. Ses çalınır ve ses dalgasının görseli yani sesin zamana göre nasıl bir değişim gösterdiği arayüze çizilir. Yeni metin girildiğinde ilk adıma geri dönülerek bu işlemler tekrarlanır. Hem maddeler halinde hem de sözel olarak ifade edilen uygulamanın çalışmasının akış diyagramı Şekil 3.1'de gösterilmektedir [18].



Şekil 4.1. Türkçe metinden konuşma sentezleme dinamik veritabanı akış diyagramı

Burada sistemin veri tabanını güncelleme işlemi yukarıda verilen 2. adımda elde edilir ve sistem bu işlemi açık kaynak bir hizmet olan çevrimiçi bir Translate Api Speech Engine'i kullanarak yapar. Sistem veri tabanında bulunmayan heceyi bu API'nin ses motoruna gönderir ve bu motordan gelen ses dosyasını Visual C#'ın bir yapısı olan bir Stream'e kaydeder. Bu Stream'i de veri tabanı olarak kullandığı bir 'array'e kaydeder. Böylece veri tabanı büyümüş olur. Uygulama bir sistemde (PC'de) ilk defa çalıştırıldığında veri tabanını sıfırdan oluşturduğu için internete bağımlılığı yüksek seviyelerdedir. Uygulama ne kadar çok kullanılırsa ses veri tabanı genişleyeceğinden uygulamanın internete bağımlılığı azalır.

Uygulama oluşturduğu toplam ses dosyasını (girilen metnin sese çevrilmiş halini) bilgisayarın ses çıkışından çalarken, uygulamanın kullanıcı arayüzüne de zaman-büyüklük ekseninde 'trial' bir yapıda çizdirir. Bu çizdirme işlemi yine açık kaynak bir proje olan NAudio sayesinde yapılır. Bu kütüphane arayüzlerde ses işlemleri yapmak için kullanılmaktadır.

Türkçe metinden konuşma sentezleme dinamik veritabanı akış diyagramı üzerinden aşağıda aşama aşama açıklanmıştır.

1.Başla → Metin Girildi

Bu adımda uygulama çalıştırılıp, metin kutusunun içerisine sentezlenmesi istenen metin girilip ardından da sentezle butonuna basılarak sentezleme işlemi başlatılır.

2.Metin Girildi → Kelimelere Ayrıldı

Bu adımda sentezlenmesi istenen metin öncelikle kelimelerine ayrılır. Bu işlem girilen metin arasında bırakılan boşluktan yararlanılarak yapılır.

3.Kelimelere Ayrıldı → Kelime Türkçe mi?

Bu adımda kelimelerin Türkçe olup olmadığına bakılır. Bu işlem için ise Türkçenin hece yapısından yararlanılır. Türkçede toplamda 6 çeşit hece tipi bulunmaktadır. Bu tipler "ünlü", "ünlü-ünsüz", "ünsüz-ünlü", "ünlü-ünsüz-ünsüz", "ünsüz-ünlü-ünsüz", "ünsüz-ünlü-ünsüz-ünsüz" şeklindedir.

4.Kelime Türkçe mi? → Kelime Yabancı

Bu adımda; kelimelerin Türkçe olup olmadığına bakıldıktan sonra yani kelimelerin Türkçede bulunan hece yapısına göre uygun olarak hecelenip hecelenemediğine dikkat edilerek, eğer hecelenemiyorsa kelime yabancı olarak kabul edilir.

5.Kelime Yabancı → Kelime yabancı kelimeler veritabanında mevcut mu?

Bu adımda, hecelere ayrılamadığı yani yabancı kelime olduğu tespit edilen kelimenin, yabancı kelimeler veritabanında mevcut olup olmadığına bakılır.

6.Mevcut → Yabancı kelimenin ses kaydını cümle dizinine ekle

Bu adımda, veritabanında mevcut olan yabancı kelimenin ses kaydı cümle dizinine eklenir.

7.Yabancı kelimenin ses kaydını cümle dizinine ekle → Cümle ses kaydı oluştur

Bu adımda, yabancı kelime ses kayıtları birleştirilerek, cümleni ses kaydı oluşturulur.

8.Kelime veritabanında mevcut değil ise → Ses kaydını veritabanına ekle

Bu adımda yabancı kelime veritabanında bulunmayan kelime için, çevrimiçi konuşma motorundan ses kaydı alınarak veritabanına eklenir.

9.Ses kaydını veri tabanına ekle → Yabancı kelimenin ses kaydını cümle dizinine ekle

Bu adımda çevrimiçi konuşma motorundan alınarak veritabanına eklenen ses kaydı, yabancı kelimenin ses kaydını cümle dizinine eklenir.

10. Yabancı kelimenin ses kaydını cümle dizinine ekle → Cümle ses kaydı oluştur
Bu adımda, yabancı kelime ses kayıtları birleştirilerek, cümlenin ses kaydı oluşturulur.

11. Kelime Türkçe → Hece veritabanında mevcut mu?
Bu adımda, Türkçe olduğuna yani hecelerine ayrıldığına karar verilen kelimenin hecelerinin, hece veritabanında mevcut olup olmadığına bakılır.

12. Mevcut → Hece dizinine ekle
Bu adımda, hece veritabanında mevcut olan hecelerin ses kaydı, hece dizinine eklenir.

13. Hece dizinine ekle → Kelimeyi oluştur
Bu adımda, hece kayıtları birleştirilerek kelime oluşturulur.

14. Kelimeyi oluştur → Cümle ses kaydı oluştur
Bu adımda, Türkçe kelime ses kayıtları birleştirilerek, cümlenin ses kaydı oluşturulur.

15. Hece veritabanında mevcut değil ise → Ses kaydını veritabanına ekle
Bu adımda hece veritabanında bulunmayan hece için, çevrimiçi konuşma motorundan ses kaydı alınarak veritabanına eklenir.

16. Ses kaydını veritabanına ekle → Hece dizinine ekle
Bu adımda çevrimiçi konuşma motorundan alınarak veritabanına eklenen ses kaydı, hece ses kaydının hece dizinine eklenir.

17. Hece dizinine ekle → Kelimeyi oluştur
Bu adımda, hece kayıtları birleştirilerek kelime oluşturulur.

18. Kelimeyi oluştur → Cümle ses kaydı oluştur
Bu adımda, Türkçe kelime ses kayıtları birleştirilerek, cümlenin ses kaydı oluşturulur.

4.2. Program Akışı ve Örnek Sentezi

1.Adım: Grafik arayüzünde bulunan metin kutusuna okunması istenen metin küçük harf ve noktalama işareti olmadan girilir. Burada örnek olarak metin kutusuna “ televizyon izlemek “ metninin girildiğini varsayalım.

Girilen metin = “ televizyon izlemek “

2.Adım: Sentezle Butonuna basılarak girilen metin sentezleme işlemine alınır.

Girilen metin = “ televizyon izlemek “
Sentezle tuşuna basılarak metin sentezleme işlemi başlatılır.

3.Adım: Girilen metin başlangıçta; oluşturulmuş olan bir fonksiyon aracılığıyla boşluklarından ayrılarak kelimelere ayrılır ve elde edilen kelimeler bir dizin içerisine doldurulur. Ayrıca, burada elde edilen kelimelerin sayısıda bir değişkenine atanır. Yani girdiğimiz metin örneğini düşündüğümüz zaman;

Girilen metin = “ televizyon izlemek “
Sentezle tuşuna basılarak metin sentezleme işlemi başlatıldı.
Girilen metnin ilk kelimesi = ” televizyon ”
Girilen metnin ikinci kelimesi = ” izlemek ”

4.Adım: Bir önceki adımda boşluklarından ayrılarak bir diziye atanan kelimeler bu adımda filtrelenir. Bu işlem, elde edilen kelimelerin ünlü ünsüz halini elde etmek için yapılmaktadır. Filtreleme işlemi ise kelimelerde bulunan ünlü ve ünsüz harfler kodlanarak yapılır. Bu kodlama işlemi, ünlü harfi için “u”, ünsüz harf için ise “s” olacak şekilde yapılır. Yani girilen örnek için;
Dizinin ilk elemanı olan ”televizyon” kelimesinin filtrelenmiş hali ”susususssus” şeklinde kodlanmış olarak elde edilir. Dizinin ikinci elemanı olan ” izlemek” kelimesinin filtrelenmiş hali ise ”ususus” şeklinde kodlanmış olarak elde edilir.

Girilen metin = “ televizyon izlemek “
Sentezle tuşuna basılarak metin sentezleme işlemi başlatıldı.

Girilen metnin ilk kelimesi = ” televizyon ”
Girilen metnin ikinci kelimesi = ” izlemek ”
Girilen metnin ilk kelimesinin filtrelenmiş hali = “ susususssus “
Girilen metnin ikinci kelimesinin filtrelenmiş hali = “ ussusus “

5.Adım: Kodlanmış hali elde edilen kelimeyi hecelerine ayırmak için Türkçede bulunan hece yapısının çeşitlerinden yararlanılır. Türkçede toplamda 6 farklı hece çeşidi vardır ve bunlar “u”, “su”, “us”, “uss”, “sus”, “suss” şeklindedir. Girilen örnek için 5. Adım şu şekilde edilir.

Girilen metin = “ televizyon izlemek “
Sentezle tuşuna basılarak metin sentezleme işlemi başlatıldı.
Girilen metnin ilk kelimesi = ” televizyon ”
Girilen metnin ikinci kelimesi = ” izlemek ”
Girilen metnin ilk kelimesinin filtrelenmiş hali = “ susususssus “
Girilen metnin ikinci kelimesinin filtrelenmiş hali = “ ussusus “
Girilen metnin ilk kelimesinin ilk hecesinin filtreli hali = “ su “
Girilen metnin ilk kelimesinin ilk hecesi = “ te “
Girilen metnin ilk kelimesinin ikinci hecesinin filtreli hali = “ su “
Girilen metnin ilk kelimesinin ikinci hecesi = “ le “
Girilen metnin ilk kelimesinin üçüncü hecesinin filtreli hali = “ sus “
Girilen metnin ilk kelimesinin üçüncü hecesi = “ viz “
Girilen metnin ilk kelimesinin dördüncü hecesinin filtreli hali = “ sus “
Girilen metnin ilk kelimesinin dördüncü hecesi = “ yon “
Girilen metnin ikinci kelimesinin ilk hecesinin filtreli hali = “ us “
Girilen metnin ikinci kelimesinin ilk hecesi = “ iz “
Girilen metnin ikinci kelimesinin ikinci hecesinin filtreli hali = “ su “
Girilen metnin ikinci kelimesinin ikinci hecesi = “ le “
Girilen metnin ikinci kelimesinin üçüncü hecesinin filtreli hali = “ sus “
Girilen metnin ikinci kelimesinin üçüncü hecesi = “ mek “

6.Adım: Bu adımda; bir önceki adımda elde edilen hecelerin ses kayıtlarının veritabanında olup olmadığında bakılır. Eğer hece ses kayıtları veritabanında mevcut ise ses kayıtları bir sonraki adımdaki birleştirme işlemi için hazır hale getirilir. Eğer hece ses kayıtları veritabanında mevcut değil ise internetten elde edilir. Yani girilen örnek için;

Girilen metin = “ televizyon izlemek “
Sentezle tuşuna basılarak metin sentezleme işlemi başlatıldı.
Girilen metnin ilk kelimesi = ” televizyon ”
Girilen metnin ikinci kelimesi = ” izlemek ”
Girilen metnin ilk kelimesinin filtrelenmiş hali = “ susususssus “
Girilen metnin ikinci kelimesinin filtrelenmiş hali = “ ussusus “
Girilen metnin ilk kelimesinin ilk hecesinin filtreli hali = “ su “
Girilen metnin ilk kelimesinin ilk hecesi = “ te “
Girilen metnin ilk kelimesinin ikinci hecesinin filtreli hali = “ su “
Girilen metnin ilk kelimesinin ikinci hecesi = “ le “
Girilen metnin ilk kelimesinin üçüncü hecesinin filtreli hali = “ sus “
Girilen metnin ilk kelimesinin üçüncü hecesi = “ viz “
Girilen metnin ilk kelimesinin dördüncü hecesinin filtreli hali = “ sus “
Girilen metnin ilk kelimesinin dördüncü hecesi = “ yon “
Girilen metnin ikinci kelimesinin ilk hecesinin filtreli hali = “ us “
Girilen metnin ikinci kelimesinin ilk hecesi = “ iz “
Girilen metnin ikinci kelimesinin ikinci hecesinin filtreli hali = “ su “
Girilen metnin ikinci kelimesinin ikinci hecesi = “ le “
Girilen metnin ikinci kelimesinin üçüncü hecesinin filtreli hali = “ sus “
Girilen metnin ikinci kelimesinin üçüncü hecesi = “ mek “
İlk kelimenin ilk hecesini veritabanına kaydet = te.wav
İlk kelimenin ikinci hecesini veritabanına kaydet = le.wav
İlk kelimenin üçüncü hecesini veritabanına kaydet = viz.wav
İlk kelimenin dördüncü hecesini veritabanına kaydet = yon.wav
İkinci kelimenin ilk hecesini veritabanına kaydet = iz.wav
İkinci kelimenin ikinci hecesini veritabanına kaydet = le.wav
İkinci kelimenin üçüncü hecesini veritabanına kaydet = mek.wav

7.Adım: Bir önceki adımda veritabanında elde edilen ses kayıtları, belli bir oranda baş ve sonlarından kesilerek birbirine eklenir (TDPSOLA).

8.Adım: Ses kayıtlarının birleştirilmesi ile elde edilen ses çalınır. Yani örnek için;

Elde edilen ses = televizyon.wav, izlemek.wav

Yukarıda bahsedilen 8 adımla, programın akışı ve örnek sentezden bahsedilmiştir. Ayrıca programda kullanılan fonksiyonların amaç ve işlevleri ilgili bilgiler aşağıdaki tabloda ve devamında belirtilmiştir.

4.3. Program için Oluşturulan ve Kullanılan Fonksiyonlar

Program için oluşturulan ve kullanılan fonksiyonlar; kelimelere ayırma, kelimeleri filtreleme, hecelere ayırma, veritabanına kaydetme, ses işleme ve sesi çalma amaçlı olarak kullanılan fonksiyonlardır. Bu fonksiyonların amacı, türü ve işlevleri aşağıdaki açıklanmaktadır.

Çizelge 4.1. Program için oluşturulan ve kullanılan fonksiyonlar

Fonksiyon amacı	Tür
Kelimelere ayırma	Fonksiyon
Kelimeleri filtreleme	Sınıf
Hecelere ayırma	Sınıf
Veritabanına kaydetme	Sınıf
Ses işleme	Sınıf
Sesi çalma	Fonksiyon

Kelimelere ayırma amaçlı fonksiyonu

Kelimelere ayırma amaçlı fonksiyonun işlevi ise aşağıda maddeler halinde belirtilmiştir.

- Girilen metni boşluklardan ayırarak bir kelimeler dizisi haline getirir,
- Elde edilen kelime sayısını bir değişkenine atar,
- Elde edilen kelimeler de bir dizide tutulur

Kelimeleri filtreleme amaçlı fonksiyon

Kelimeleri filtreleme amaçlı fonksiyonun, işlevi ise aşağıda maddeler halinde belirtilmiştir.

- Giriş olarak bir kelime alır
- Bu kelimedeki ünlü ve ünsüz harfleri 'u' ve 's' harfleri ile kodlar,
- Örnek olarak 'televizyon' kelimesi bu sınıfa dahil edildiğinde oluşturulan obje içerisinde 'televizyon' kelmesinin bir de kodlanmış hali olan 'sususussus' hali de yer alır.

Hecelere ayırma amaçlı fonksiyon

Hecelere ayırma amaçlı fonksiyonun işlevi ise aşağıda maddeler halinde belirtilmiştir.

- Kelimelerin hecelerine ayrılmasını sağlayan sınıftır,
- İçerisinde bulunan bir fonksiyon kelimeleri hecelere ayırmaya yarar,
- Elde edilen heceler bir diziye konulurken
- Elde edilen hecelerin sayısı bir degişkenine atanır.

Veritabanına kaydetme amaçlı fonksiyon

Veritabanına kaydetme amaçlı fonksiyonun işlevi ise aşağıda maddeler halinde belirtilmiştir.

- Bu sınıf bir ses kütüphanesi oluşturmaya yarar
- Sınıftan bir obje oluşturulacağı zaman kütüphanenin yeri belirtilmelidir,
- Belirtilen yer bilgisayarda bir klasör olup klasör'de kayıtlı hecelerin listelerinin tutulduğu bir text dosyası olmalıdır.
- Girilen metindeki kodlanmış ve hecelerine ayrılmış olan kelimelerin her bir hecesi text dosyasında var ise, yani veritabanında mevcut ise hece ses işleme işlemi için hazır hale getirilir; eğer text dosyasında hece yok ise internetten alınır ve veritabanına kaydedilir.

Ses işleme amaçlı fonksiyon

Ses işleme amaçlı fonksiyonun işlevi ise aşağıda maddeler halinde belirtilmiştir.

- Yerleri belirtilen iki ses kaydını belirtilen uzunlukta birincisinin sonundan ikincisinin başından keserek birleştirip yeni bir ses kaydı oluşturulur.
- Programda kullanımı ise elde edilen ses kayıtlarının birleştirilmesi için kullanılmasıdır.

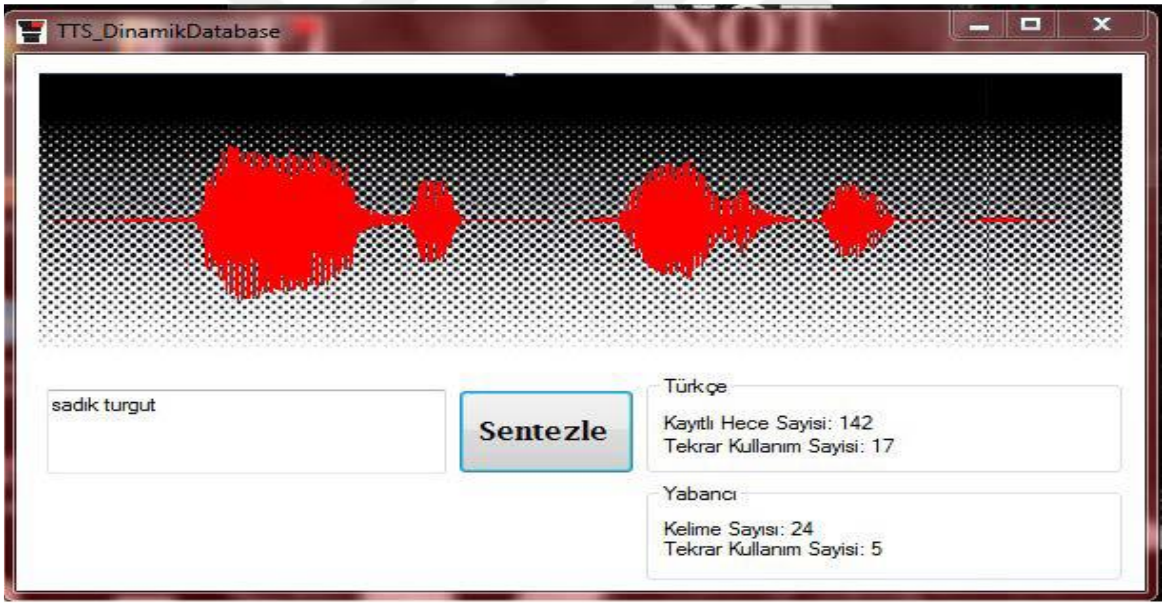
Sesi çalma amaçlı fonksiyon

Sesi çalma amaçlı fonksiyonu işlev olarak; toplamda elde edilen ses kaydının çalınması için kullanılan NAudio kütüphanesinin bileşenlerini içeren ses çalar.

Program için oluşturulan ve yukarıdaki belirtilen amaç ve işlevlerde kullanılan fonksiyonların kullanımları (program kodları) ve kısa açıklamaları ile birlikte tezin ekler kısmında belirtilmiştir.

5. SONUÇ

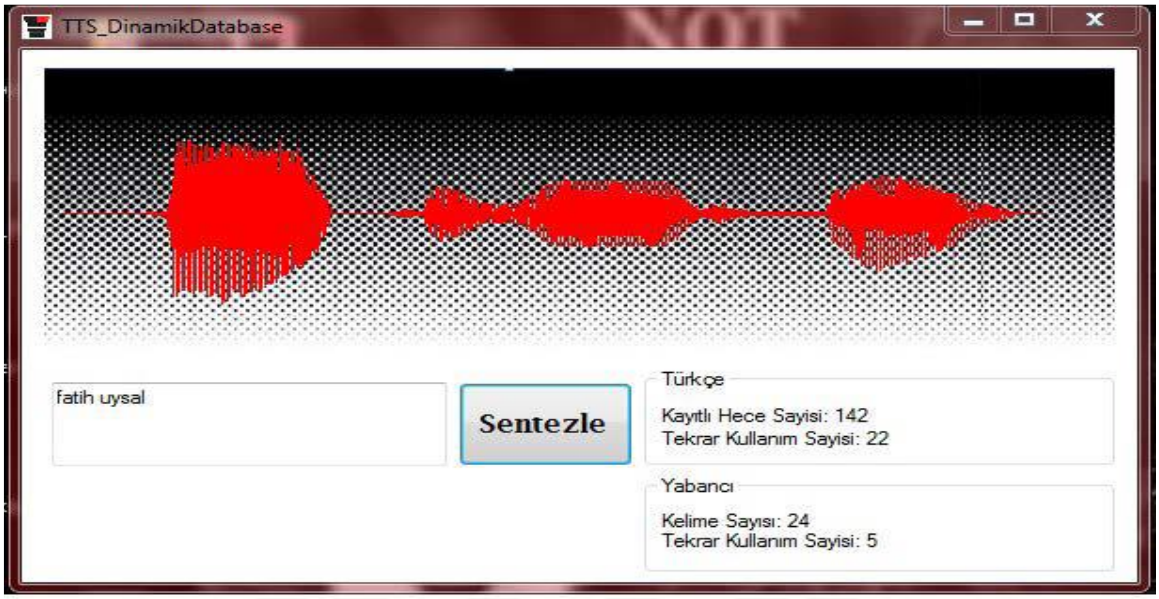
Türkçe metinden konuşma sentezleme işleminde sistemin çalışmasına dair görseller ve sonuçları aşağıdaki örnek gösterimlerde belirtilmiştir. Bu görsellerde; Türkçe konuşma sentezleme dinamik veritabanı elde etmek için yapılan konuşma sentezleme uygulamasının çalıştırılması sonucunda, sentezleme yapılması istenen metindeki hecelerin zamana göre büyüklüklerinin değişimi, görselin en üst kısmında yer almaktadır. Görselin sol alt kısmında yer alan kutucuğa sentezleme yapılması istenen metin girilip, ardından sentezle tuşuna basıldığında; metindeki hecelerin bu zamana göre büyüklüklerinin değişiminin çizilmesinin yanısıra, aynı zamanda Türkçe veritabanı için kayıtlı hece sayısı ve tekrar kullanım sayısı, Yabancı veritabanı içinde kelime sayısı ve tekrar kullanım sayısı, görsellerin sağ alt kısmında görülebilmektedir.



Şekil 5.1. Uygulamanın konuşma sentezlemesine dair örnek gösterim 1

Türkçe metinden konuşma sentezleme dinamik veritabanı akış diyagramını kullanarak, C# programlama dilinde gerçekleştirilen Türkçe metinden konuşma sentezleme uygulaması çalıştırdırıldığında, çıkan uygulama ekranında sol alt köşede bulunan kutucuğa sentezlemek için ‘sadık turgut’ metni girilip, ardından ‘Sentezle’ tuşuna basıldığında; ekranın üst kısmında sentezlenen bu metin için metindeki hecelerin zamana göre büyüklüklerinin değişimi ve ekranın sağ alt kısmında ise bu metinin sentezlenmesi sonucunda Türkçe veritabanındaki kayıtlı hece sayısının 142, tekrar kullanım sayısının 17; Yabancı

veritabanındaki kelime sayısının 24, tekrar kullanım sayısının 5 olduğu Şekil 5.1 ’deki örnek gösterimde görülmektedir.



Şekil 5.2. Uygulamanın konuşma sentezlemesine dair örnek gösterim 2

Şekil 5.2 ’deki örnek gösterime bakıldığında ise; uygulama ekranının sol alt köşesinde bulunan kutucuğa sentezlemek için ‘fatih uysal’ metni girilip, ardından ‘Sentezle’ tuşuna basıldığında, sentezlenen bu metin için ekranın üst kısmında zaman-büyüklük değişimi ve ekranın sağ alt kısmında sentezleme sonucunda Türkçe veri tabanındaki kayıtlı hece sayısının 142, tekrar kullanım sayısının 22; Yabancı veritabanındaki kelime sayısının 24, tekrar kullanım sayısının 5 olduğu görülmektedir.

Sonuç olarak bu tez çalışmasında; konuşma sentezleme yöntemlerinden biri olan art arda bağlama (eklemeli) sentezleyicilerin prensibi kullanılarak ve tezin uygulama kısmında bahsedilen Türkçe metinden konuşma sentezleme dinamik veritabanı akış diyagramından yararlanılarak C# programlama dilinde; hem Türkçe kelimeleri içeren veritabanı, hem de hecelere ayrılamayan Türkçedeki yabancı kelimeleri içeren veritabanı olmak üzere iki farklı veritabanınıda kapsayan ve kullanıldığında sürekli kendi kendini güncelleyen bir ses veritabanına sahip bir Türkçe metinden konuşma sentezleme uygulaması yapılmıştır.

Bu uygulamanın çalıştırılmasıyla sürekli güncellenen bir ses veritabanının elde edilecek olması; klasik sabit bir ses veritabanının Türkçe metinden konuşma sentezleme işleminde yetersiz kaldığı durumlarda çözüm oluşturabilme potansiyeline sahiptir. Ancak, bu veri

tabanı güncellenmesi işleminin internete bağımlı olması uygulamanın tek başına (internet bağlantısı olmadan) kullanımını zorlaştırmaktadır. Yani uygulama internete bağlı iken ne kadar çok kullanılırsa, çevrimdışı olması durumlarında gitgide daha kullanışlı hale gelecektir. Ayrıca bu tekniğin hecelere ayrıştırma işlemlerinde kullanılması gelişim ve geliştirme süreçlerini daha verimli kılacaktır.

KAYNAKLAR

1. Ergenç, İ. (2002). *Konuşma dili ve Türkçenin söyleyiş sözlüğü*. İstanbul: Multilingual Yayınları, 486.

2. Dutoit, T. (1997). *An introduction to text to speech synthesis*. Dordrecht: Kluwer Academic Publishers, 26-32.

3. Flanagan, J. L., Allen, J. B. and Hasegawa-Johnson, M. A. (2008). *Speech analysis synthesis and perception*. Springer, ch6, ch2.

4. Demircan, Ö. (2001). *Türkçenin ses dizimi*. İstanbul: Der Yayınları.

5. Lemmetty, S. (1999), *Review of speech synthesis technology*. Master's Thesis, Helsinki University of Technology, Finland.

6. Uslu, İ. B. (2010). *Metinden konuşma sentezleme 1. bölüm*. TMMOB Elektrik Mühendisleri Odası Ankara Şubesi Haber Bülteni.

7. Bicil, Y. (2010). *Türkçe Metinden Konuşma Sentezleme,* Yüksek Lisans Tezi, Sakarya Üniversitesi Fen Bilimleri Enstitüsü, Sakarya.

8. Güldali, K. (2009). *Türkçe Metin Seslendirme*, Yüksek Lisans Tezi, İstanbul Teknik Üniversitesi Fen Bilimleri Enstitüsü, İstanbul.

9. Dutoit, T., Gosselin, B. (1996). On the use of a hybrid harmonic/stochastic model for TTS synthesis by concatenation. *Speech Communication*, 19, 119-143.

10. Dubuisson, T., Dutoit, T., Gosselin, B., and Remacle, M. (2009). On the use of the correlation between acoustic descriptors for the normal/pathological voices discrimination, *EURASIP Journal on Advances in Signal Processing, Analysis and Signal Processing of Oesophageal and Pathological Voices,* 2009, 1-19.

11. Veronis, J., Hirst, D., Esspesser, R. and Ide, N. (1994). NL and speech in multext project. *AAAI'94 Workshop on Integration of Natural Language and Speech*, 72-78.

12. Dutoit, T., Cernak, M. (2005). TTSBOX: *A matlab toolbox for teaching text-to-speech synthesis*. ICASSP'05, Philadelphia, USA.

13. Yamagishi, J., Zen, H., Toda, T. and Tokuda, K. (2007). *Speaker-independent HMM-based speech synthesis system - HTS-2007 system for the blizzard challenge 2007.* Blizzard Challenge 2007.

14. Uslu, İ. B. (2012). *Konuşma İşleme ve Türkçenin Dilbilimsel Özelliklerini Kullanarak Metinden Doğal Konuşma Sentezleme*, Doktora Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

15. Karli, A. (2005). *Örnek Bir Dizi Cümle için Türkçe Metinden Konuşma Sentezleyici*, Yüksek Lisans Tezi, Ankara Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

16. Waghmare, K., Kayte, S. and Gawali, B. (2016). Analysis of pitch and duration in speech synthesis using PSOLA. *Communications on Applied Electronics*, 4(4), 10-18.

17. Ünaldi, İ. (2007). *Taşınabilir Cihazlar için Türkçe Metinden Konuşma Sentezleme Sistemi,* Yüksek Lisans Tezi, Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, Ankara.

18. Turgut, S., Uysal, F. ve Hardalaç, F, (2016, 26-28 Ekim). *Eklemeli sentezleyici yöntemi kullanılarak Türkçe metinden konuşma sentezleme işleminde güncellenen bir ses veritabanı oluşturulması.* 1. Uluslararası Akdeniz Bilim ve Mühendislik Kongresi *(IMSEC 2016)*, 2375-2383.

EKLER

EK-1. Piecer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace TTS_FDD_CoreFunctions_Dens
{
   class Piecer
   {
      //Kelimenin asıl hali: olarak
      public string NatureWord = null;
      //Kelimenin sesli sessiz filtresinden geçmiş hali: ususus = olarak
      public string VoweledWord = null;
      //Bu array sesli harflerin kelimede nerede olduğu tutulur
      public  int[] pointerOfVowels = {100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100, 100};
      //Vowel sesli harfi ifade demekte, aşağıdaki değişkende kelime kaç sesli harf olduğu
tutulmakta
      public int numberOfVowelInWord;
      //Bu array, ususus'un char array hali
      private char[] VoweledWordCharArray;
      //Kelime'nin içerisindeki harf sayısı array'in uzunluğu kadar
      private int lengthOfVoweledWordCharArray;
      //Kelime yabancı veya Türkçe ise
      private int isItTurkish = 0; //1 ise yabancı kelimedir
      //Global variables
      public Hece[] piecesObtainedFromWord = new Hece[25]; //Bir kelimede 25 hece
olacağını varsayılmış, bu değer değişebilir yani 100'de olabilir 1000'de olabilir
      //Kelimeden elde edilen hece sayısı
      public int pieceCounter = 0;
      //kelimenin asıl halin char array hali
      private char[] NatureWordCharArray;
      //Kelimenin asıl halinde kac harf olduğu
```

EK-1. (devam) Piecer.cs

```
    private int lengthOFNatureWordCharArray;
    //Constructor
    public Piecer(string NewComerNatureWord, string NewComerVoweledWord)
    {
        //Kelimenin asıl halini sınıfın natureword'üne atandı
        NatureWord = NewComerNatureWord;
        //Kelimenin sesli-sessiz halini de voweled değişkenine atandı
        VoweledWord = NewComerVoweledWord;
        //Sesli sessiz halini sınıfın char array'ine atandı
        VoweledWordCharArray = VoweledWord.ToCharArray();
        //Bu arrayin uzunluğunu sınıfın counter değişkenine atandı
        lengthOfVoweledWordCharArray = VoweledWordCharArray.GetLength(0);
        //Kelimenin asıl halini de sınıfın asıl kelime değişkenine atandı
        NatureWordCharArray = NatureWord.ToCharArray();
        //Sonrasında ise kelimede ünlülerin nerede olacağını belirliyen fonksiyonu çağırıldı
        whereIsTheVowels();
        Maker();
        int internalCounter = 0;
        while (internalCounter > 25) {
            piecesObtainedFromWord[internalCounter] = new Hece(' ');
        }
    }
    public void whereIsTheVowels()
    {
        int internalCounter = 0;
        numberOfVowelInWord = 0;
        while (internalCounter < (lengthOfVoweledWordCharArray)) {
            if (VoweledWordCharArray[internalCounter] == 'u')
            {
                pointerOfVowels[numberOfVowelInWord] = internalCounter;
                numberOfVowelInWord++;
            }
            internalCounter++;
```

EK-1. (devam) Piecer.cs

```
        }
    }
    public void Maker() {
        //İki ünlü arasında maksimum üç ünsüz olabilir
        //Kelime iki ünsüz ile başlamaz
        //Kelime üç ünsüz ile bitemez
        int c = 0; //internalCounter
        //Türkçede maksimum 4 harfli hece olur
        /*
        Altı çeşit hece var:
        u: ünlü
        s: ünsüz
        Type 0: unspecified
        Type 1: ü
        Type 2: sü
        Type 3: üs
        Type 4: üss
        Type 5: süs
        Type 6: süss
        */
        int L = lengthOfVoweledWordCharArray; //Kelimedeki harf sayısı
        char[] k = NatureWordCharArray; //Kelimenin asıl halinin array hali
        char[] v = VoweledWordCharArray; //Kelimenin sesli-sessiz halinin char arrayi
        int hc = 0; //pieceCounter'a eşitlenecektir, elde edilen hece sayısı pieceCounter'dır
        //Kelime Türkçe ise ve Counter kelimedeki harf sayısından az ise
        while ((c < lengthOfVoweledWordCharArray)&&(isItTurkish == 0)) {
            //Tek harfli kelimeler
            if ((lengthOfVoweledWordCharArray == 1) && (VoweledWordCharArray[0]
== 'u'))
            {
                piecesObtainedFromWord[0] = new Hece(NatureWordCharArray[0]);
                hc++;
            }
```

EK-1. (devam) Piecer.cs

```
            //Tek harfli kelime fakat harf sessiz ise
            if ((lengthOfVoweledWordCharArray == 1) && (VoweledWordCharArray[0]
== 's'))
            {
               isItTurkish = 1;
            }
            //Kelime'de ki sesli harfin tek başına hece olması durumu
            if (v[c] == 'u')
            {
                  //Kelime başında ünlü olursa ve ondan sonra ünsüz + ünlü gelirse bu ünlü
tek başına hece olur
                  if ((c == 0) && (L > 2))
                  {
                     if ((v[c + 1] == 's') && (v[c + 2] == 'u'))
                     {
                        if (hc == 0)
                        {
                           piecesObtainedFromWord[hc] = new Hece(k[c]);
                           hc++;
                        }
                     }
                  }
                  //
                  if ((c > 1) && (L > (c + 2)))
                  {
                     if ((v[c - 1] == 'u') && (v[c + 1] == 's') && (v[c + 2] == 'u'))
                     {
                        piecesObtainedFromWord[hc] = new Hece(k[c]);
                        hc++;
                     }
                  }
            }
            //İşlenen harf sessiz olacak
```

EK-1. (devam) Piecer.cs

```
            if (v[c] == 's') {
                //Baştaki sessiz harfler için
                if ((L >= 1) && (c == 0)) //ilk karakter sessiz fakat kelimede birden fazla harf var
                {
                    if (v[c + 1] == 'u') //ilk harf sessiz ikinci harf sesli;İdeal Hece
                    {
                        piecesObtainedFromWord[hc] =new Hece(k[c], k[c + 1]);
                        hc++;
                    }
                    if (v[c + 1] == 's') //Türkçe'de ilk iki harf sessiz olmaz
                    {
                        isItTurkish = 1;
                    }
                }
                //Ortadaki sessiz harf
                if ((L > (c + 1)) && (c != 0)) //Buradaki (c!=0) ifadesini ilk if'e giren buraya'da girmesin diye koyuldu
                {
                    if (v[c + 1] == 'u') //İdeal Hece: su
                    {
                        piecesObtainedFromWord[hc] = new Hece(k[c], k[c + 1]);
                        hc++;
                    }
                    else //Yani sessizden sonra sesli yok ise
                    {
                        if((v[c-1] == 'u') && (c !=1)) // Olusturulan hece: us
                        {
                            if (hc == 0)
                            {
                                piecesObtainedFromWord[hc] = new Hece(k[c - 1], k[c]);
                            }
                            else
```

EK-1. (devam) Piecer.cs

```
                {
                    //önceki hece değişir
                    piecesObtainedFromWord[hc - 1] = new Hece(k[c - 1], k[c]);
                }
            }
            if ((v[c - 1] == 'u') && (c == 1))// Olusturulan hece: us
            {
                if (hc == 0)
                {
                    piecesObtainedFromWord[hc] = new Hece(k[c - 1], k[c]);
                    hc++;
                }
                else
                {
                    //önceki hece değişir
                    piecesObtainedFromWord[hc - 1] = new Hece(k[c - 1], k[c]);
                }
            }
            //Yani gerideki sesliyi almış bir sessiz olabilir, burada 'sus' hecesindeki
ikinci s'yi işlemekteyiz
            if (c > 1)
            {
                if((v[c-1] == 'u') && (v[c - 2] == 's'))
                {
                    if(hc == 0)
                    {
                        piecesObtainedFromWord[hc] = new Hece(k[c - 2], k[c-1], k[c]);
                    }
                    else
                    {
                        //önceki hece değişir
                        piecesObtainedFromWord[hc - 1] = new Hece(k[c - 2], k[c - 1],
k[c]);
```

EK-1. (devam) Piecer.cs

```
                    }
                }
            }
            //Yani gerideki sesliyi almış bir sessiz olabilir, burada 'suss' hecesindeki
üçüncü s'yi işlemekteyiz
            if (c > 2)
            {
                if ((v[c - 1] == 's') && (v[c - 2] == 'u') && (v[c - 3] == 's'))
                {
                    if (hc == 0)
                    {
                        piecesObtainedFromWord[hc] = new Hece(k[c-3], k[c - 2], k[c -
1], k[c]);

                    }
                    else
                    {
                        //önceki hece değişir
                        piecesObtainedFromWord[hc - 1] = new Hece(k[c - 3], k[c - 2],
k[c - 1], k[c]);
                    }
                }
            }
        }
    }
    //Sondaki sessiz harf
    if ((c+1) == L)
    {
        if ((v[c - 1] == 'u'))// Olusturulan hece: us
        {
            if (hc == 0)
            {
                piecesObtainedFromWord[hc] = new Hece(k[c - 1], k[c]);
```

EK-1. (devam) Piecer.cs

```
                    hc++;
                }
                else
                {
                    if (c > 1)
                    {
                        //Kendisinden önceki sesliyi alan sessiz olmadığı için aslında önceki heceye eklenmemiş olur
                        if ((v[c - 1] == 'u') && (v[c - 2] == 'u'))
                        {
                            if (hc == 0)
                            {
                                piecesObtainedFromWord[hc] = new Hece(k[c - 1], k[c]);
                                hc++;
                            }
                            else
                            {
                                piecesObtainedFromWord[hc] = new Hece(k[c - 1], k[c]);
                                hc++;
                            }
                        }
                    }
                }
                //Yani gerideki sesliyi almış bir sessiz olabilir, burada 'sus' hecesindeki ikinci s'yi işlemekteyiz
                if (c > 1)
                {
                    if ((v[c - 1] == 'u') && (v[c - 2] == 's'))
                    {
                        if (hc == 0)
                        {
                            piecesObtainedFromWord[hc] = new Hece(k[c - 2], k[c - 1], k[c]);
```

EK-1. (devam) Piecer.cs

```
                    }
                    else
                    {
                        //önceki hece değişir
                        piecesObtainedFromWord[hc - 1] = new Hece(k[c - 2], k[c - 1],
k[c]);
                    }
                }
            }
            //Yani gerideki sesliyi almış bir sessiz olabilir, burada 'suss' hecesindeki
üçüncü s'yi işlemekteyiz
            if (c > 2)
            {
                if ((v[c - 1] == 's') && (v[c - 2] == 'u') && (v[c - 3] == 's'))
                {
                    if (hc == 0)
                    {
                        piecesObtainedFromWord[hc] = new Hece(k[c - 3], k[c - 2], k[c -
1], k[c]);

                    }
                    else
                    {
                        //önceki hece değişir
                        piecesObtainedFromWord[hc - 1] = new Hece(k[c - 3], k[c - 2], k[c
- 1], k[c]);
                    }
                }
            }

        }

    }
```

EK-1. (devam) Piecer.cs

```
                c++;
            }
            pieceCounter = hc;
        }
    }
}
```

EK-2. Hece.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace TTS_FDD_CoreFunctions_Dens
{
   class Hece
   {
      public string hece = null;
      //Hecenin char array hali
      public char[] charsOfHece;
      //Hecede kac harf bulundugu
      public int LengthOfHece = 0;
      //Türkçede en uzun hece 4 harfli olduğu için bu sayıda 4'dür
      private int MaximumLengthOfHece = 4;
      //Türkçede maksimum 4 harfli hece olur
                              /*
                        Altı çeşit hece var:
                        u: ünlü
                        s: ünsüz
                        Type 0: unspecified
                        Type 1: ü
                        Type 2: sü
                        Type 3: üs
                        Type 4: üss
                        Type 5: süs
                        Type 6: süss
                        */
      public int TypeOfHece = 0;
      //uzunluğuna göre hece oluşturucuları
      private char charTypeDetector(char anyChar)
      {
```

EK-2. (devam) Hece.cs

```
        if ((anyChar == 'a') || (anyChar == 'e') || (anyChar == 'ı') || (anyChar == 'i') ||
(anyChar == 'o') || (anyChar == 'ö') || (anyChar == 'u') || (anyChar == 'ü'))
        {
            return 'u';
        }
        else
        {
            return 's';
        }
    }
    //Constructors
    public Hece(char birinci)
    {
        char[] internalArray = { birinci };
        hece = new string(internalArray);
        charsOfHece = hece.ToCharArray();
        LengthOfHece = charsOfHece.GetLength(0);
        if (charTypeDetector(birinci) == 'u')
        {
            TypeOfHece = 1;
        }
    }
    public Hece(char birinci, char ikinci)
    {
        char[] internalArray = { birinci, ikinci };
        hece = new string(internalArray);
        charsOfHece = hece.ToCharArray();
        LengthOfHece = charsOfHece.GetLength(0);
        //İki harfli bir hece de aşağıdakilerden başka dizilim olamaz
        if (((charTypeDetector(birinci) == 'u') && (charTypeDetector(birinci) == 's')) ||
((charTypeDetector(birinci) == 's') && (charTypeDetector(birinci) == 'u')))
        {
            if ((charTypeDetector(birinci) == 'u') && (charTypeDetector(birinci) == 's'))
```

EK-2. (devam) Hece.cs

```
            {
                TypeOfHece = 3;
            }
            if ((charTypeDetector(birinci) == 's') && (charTypeDetector(birinci) == 'u'))
            {
                TypeOfHece = 2;
            }
        }
    }
    public Hece(char birinci, char ikinci, char ucuncu)
    {
        char[] internalArray = { birinci, ikinci, ucuncu };
        hece = new string(internalArray);
        charsOfHece = hece.ToCharArray();
        LengthOfHece = charsOfHece.GetLength(0);
        //Uc harfli bir hece de aşağıdakilerden başka dizilim olamaz
        if (((charTypeDetector(birinci) == 'u') && (charTypeDetector(birinci) == 's') &&
(charTypeDetector(ucuncu) == 's')) || ((charTypeDetector(birinci) == 's') &&
(charTypeDetector(birinci) == 'u') && (charTypeDetector(ucuncu) == 's')))
        {
            if ((charTypeDetector(birinci) == 'u') && (charTypeDetector(birinci) == 's') &&
(charTypeDetector(ucuncu) == 's'))
            {
                TypeOfHece = 4;
            }
            if ((charTypeDetector(birinci) == 's') && (charTypeDetector(birinci) == 'u') &&
(charTypeDetector(ucuncu) == 's'))
            {
                TypeOfHece = 5;
            }
        }
    }
    public Hece(char birinci, char ikinci, char ucuncu, char dorduncu)
```

EK-2. (devam) Hece.cs

```
        {
            char[] internalArray = { birinci, ikinci, ucuncu, dorduncu };
            hece = new string(internalArray);
            charsOfHece = hece.ToCharArray();
            LengthOfHece = charsOfHece.GetLength(0);
            //Dört harfli bir hece de aşağıdaki halinden başka dizilim olamaz
            if ((charTypeDetector(birinci) == 's') && (charTypeDetector(ikinci) == 'u') &&
(charTypeDetector(ucuncu) == 's') && (charTypeDetector(dorduncu) == 's'))
            {
                TypeOfHece = 6;
            }
        }
    }
}
```

EK-3. DatabaseRusher.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
//Bu iki kütüphane sayesinde text dosyası işlemleri yapılmakta
using System.IO;
using System.Text;
namespace TTS_FDD_CoreFunctions_Dens
{
   class DatabaseRusher
   {
      public string Place = null;
      //Okuma için
      private StreamReader TurkishDatabaseIndexFile;
      //Yazma için
      private StreamWriter TurkishDatabaseIndexWriter;
      //Aynı hece metninin birden fazla kullanımında
      public int reUsageCounter = 0;
      //Database'de tutulan bilgilerin(hece veya kelime) listesi
      public string[] ParticlesInDatabase = new string[6000];
      //Database'de tutulan bilgilerin(hece veya kelime) sayısı
      public int ParticleCountInDatabase = 0;
      public DatabaseRusher(string NewComerPlace) {
         Place = NewComerPlace;
      }
      private bool searchInDatabase(string NewComerMetni) {
         //database'deki kayıtlı hece sayısı
         ParticleCountInDatabase = 0;
         //Database'deki heceleri index file'dan okuyoruz
         TurkishDatabaseIndexFile = File.OpenText(Place);
         string siradaki_satir = TurkishDatabaseIndexFile.ReadLine();// ilk satırı okutuyoruz
ve alt satıra geçiyoruz.
```

EK-3. (devam) DatabaseRusher.cs

```
        ParticlesInDatabase[ParticleCountInDatabase] = siradaki_satir;
        ParticleCountInDatabase++;
        while (siradaki_satir != null) //satır boş olana kadar okutuyoruz
        {
            //lstListem.Items.Add(siradaki_satir);//okunan veriyi listboxa eklettik.
            siradaki_satir = TurkishDatabaseIndexFile.ReadLine(); //satırı bidaha okuttuk.
            //Okudugumuz heceleri hece arrayine atıyoruz
            ParticlesInDatabase[ParticleCountInDatabase] = siradaki_satir;
            //Toplamdaki hece sayısının buradan buluyoruz
            ParticleCountInDatabase++;
        }
        //Text file okumaya yarayan sınıf işi bittikten sonra kapatılması aynı text dosyasını başka fonksiyon kullanamaz
        TurkishDatabaseIndexFile.Close();
        //Okuduktan sonra istenen heceyi database'de arıyoruz varsa true yoksa false dönüyoruz
        int internalCounter = 0;
        bool FoundInDatabase = false;
        while (internalCounter < ParticleCountInDatabase)
        {
            if(ParticlesInDatabase[internalCounter] == NewComerMetni)
            {
                FoundInDatabase = true;
            }
            internalCounter++;
        }
        return FoundInDatabase;
    }
    //bu Fonksiyon dışardan istenilen bir sesin database olup olmadığını sorgular var ise şimdilik
    //int bir değer döner(Normal'de wav dosyası dönecek)
    //dönülen değer bir ise database'den bulundu
    //dönülen değer iki ise metin oluşturuldu
```

EK-3. (devam) DatabaseRusher.cs

```
        public int BringItToMe(string NewComerMetin)
        {
            int ReturningValue = 0;

            if (searchInDatabase(NewComerMetin))
            {
                ReturningValue = 1;
                //ReusageCounter Reusage count diye bir dosyada tutulmalı
                reUsageCounter++;
            }
            else
            {
                DatabaseAdder(NewComerMetin);
                ReturningValue = 2;
            }
            /*
            Burada database'de oluşturulan wav dosyalarının listesi dönülecek
            */
            return ReturningValue;
        }
        public void DatabaseAdder(string NewComerMetin)
        {
            TurkishDatabaseIndexWriter = File.AppendText(Place);
            TurkishDatabaseIndexWriter.WriteLine(NewComerMetin);
            TurkishDatabaseIndexWriter.Close();
        }
    }
}
```

EK-4. WavProcessor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using NAudio.Wave;
namespace TTS_FDD_CoreFunctions_Dens
{
   class WavProcessor
   {
      public string PlaceOfWav1 = null;
      public string PlaceOfWav2 = null;
      //Yeni üretilecek sesin kaydedileceği yer
      public string subProcessDirectory = null;
      //Yeni üretilen sesin ismi
      public string ProductName = null;
      public WavProcessor(string Place1, string Place2, string Directory, string
nameOfProduct)
      {
         PlaceOfWav1 = Place1;
         PlaceOfWav2 = Place2;
         ProductName = nameOfProduct;
         /*
         Örneğin:
         Directory c:\dekstop\yeniKlasör
         ProductName: Gazi
         c:\dekstop\yeniKlasör\Gazi.wav
         */
         subProcessDirectory = Directory + "\\" + ProductName + ".wav";
      }
      //Bu fonksiyon integer bir değer döner
      //Eğer 1 döner ise wav birleştirme işlemi yapılmıştır
      //Eğer 2 döner ise wav birleştirme işlemi yapılamamıştır
```

EK-4. (devam) WavProcessor.cs

```
        public int WavAdder()
        {
            int returningValue = 2;
            //Verilen iki wav dosyasını okur
            WaveStream waveStream1 = new WaveFileReader(PlaceOfWav1);
            WaveStream waveStream2 = new WaveFileReader(PlaceOfWav2);
            //İki wav dosyasının uzunluğu kadar(sample sayısı kadar) bir array oluşturalım
            byte[] byter = new byte[waveStream1.Length + waveStream2.Length + 1];
            //Birinci wav
            waveStream1.Position = 0;//Wav'ı okumaya başladığı yer
            int internalCounter = ((int) waveStream1.Length);
            waveStream1.Read(byter, 0, internalCounter);//Buffer'a yazacağı yerin başlangıcı
ve kaç tane yazacağı
            //İkinci wav
            waveStream2.Position = 0;//Wav'ı okumaya başladığı yer
            int internalCounter2 = (int)waveStream2.Length;
            waveStream2.Read(byter, internalCounter, internalCounter2);//buffer yazacağı
yerin başlangıcı ve kaç tane yazacağı
            //WaveFileWriter bir sayı dizisini bir wav dosyası haline dönüştürmeyi sağlar
            using (WaveFileWriter waveFileWriter = new
WaveFileWriter(subProcessDirectory, waveStream1.WaveFormat))
            {
                int internalCounter3 = (int)waveStream2.Length + (int)waveStream1.Length;
                waveFileWriter.Write(byter, 0, internalCounter3); //Byter array'in neresinden
başlayarak ne kadar'ını wav dosyasına yazayım
                waveFileWriter.Flush();
                returningValue = 1;
            }

            return returningValue;
        }
    }
}
```

EK–5. VowelFilteredWord.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace TTS_FDD_CoreFunctions_Dens
{
   class VowelFilteredWord
   {
      /*
      ünlü harfler: u
      ünsüz harlef: s ile işaretlenecek
      Örneğin: "olmak" elemanı "ussus"
            "sevmek" elemanı "sussus" olacak
      input: void,
      output: bir kelime stringi
      */
      public char[] charDirectory = new char[73];
      public string NativeWord = null;
      public int charDirectoryCounter = 0;
      public string VoweledWordForDebug = null;
      public int IsItTurkish = 2; //non processed
      public VowelFilteredWord(string NewComer) {
         //Newcomer'e bir char array'e dönüştür
         NativeWord = NewComer;
         char[] internalArray = NewComer.ToCharArray();
         int internalCounter = 0;
         IsItTurkish = 0; // 1 olursa yabancı karakter var mı?
         //İlk Turkce olmayan karakter var mı?
         while (internalCounter < internalArray.GetLength(0)) {
            if ((internalArray[internalCounter] == 'a') || (internalArray[internalCounter] ==
'b')
```

EK–5. (devam) VowelFilteredWord.cs

```
            || (internalArray[internalCounter] == 'c') || (internalArray[internalCounter] ==
'ç')
            || (internalArray[internalCounter] == 'd') || (internalArray[internalCounter] ==
'e')
            || (internalArray[internalCounter] == 'f') || (internalArray[internalCounter] ==
'g')
            || (internalArray[internalCounter] == 'ğ') || (internalArray[internalCounter] ==
'h')
            || (internalArray[internalCounter] == 'ı') || (internalArray[internalCounter] ==
'i')
            || (internalArray[internalCounter] == 'j') || (internalArray[internalCounter] ==
'k')
            || (internalArray[internalCounter] == 'l') || (internalArray[internalCounter] ==
'm')
            || (internalArray[internalCounter] == 'n') || (internalArray[internalCounter] ==
'o')
            || (internalArray[internalCounter] == 'ö') || (internalArray[internalCounter] ==
'p')
            || (internalArray[internalCounter] == 'r') || (internalArray[internalCounter] ==
's')
            || (internalArray[internalCounter] == 'ş') || (internalArray[internalCounter] ==
't')
            || (internalArray[internalCounter] == 'u') || (internalArray[internalCounter] ==
'ü')
            || (internalArray[internalCounter] == 'v') || (internalArray[internalCounter] ==
'y')
            || (internalArray[internalCounter] == 'z'))
        {
        }
        else
        {
           IsItTurkish = 1; //Bu karakter bu kelimenin Türkçe olmadığını belirtir
        }
```

EK–5. (devam) VowelFilteredWord.cs

```
            internalCounter++;


        }
        internalCounter = 0;
        //yani sadece Turkcedeki harfler var
        if (IsItTurkish == 0) {
            while (internalCounter < internalArray.GetLength(0))
            {
                if ((internalArray[internalCounter] == 'a') || (internalArray[internalCounter] ==
'e')
                    || (internalArray[internalCounter] == 'ı') || (internalArray[internalCounter]
== 'i')
                    || (internalArray[internalCounter] == 'o') || (internalArray[internalCounter]
== 'ö')

                    || (internalArray[internalCounter] == 'u') || (internalArray[internalCounter]
== 'ü'))
                {
                    charDirectory[internalCounter] = 'u';
                    internalCounter++;
                    charDirectoryCounter++;
                }
                else
                {
                    charDirectory[internalCounter] = 's';
                    internalCounter++;
                    charDirectoryCounter++;
                }
            }
        }
        //For debug
        VoweledWordForDebug = new string(charDirectory, 0, charDirectoryCounter);
    }
```

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : UYSAL, Fatih
Uyruğu : T.C.
Doğum tarihi ve yeri : 19.10.1988, İskenderun
Medeni hali : Bekar
Telefon : +90 534 022 61 28
e-mail : uysal@gazi.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Yüksek lisans	Gazi Üniversitesi /Elektrik-Elektronik Müh.	Devam Ediyor
Lisans	Kırıkkale Üniversitesi /Elektrik-Elektronik Müh.	2010
Lise	Etimesgut Mehmetçik Lisesi	2005

İş Deneyimi

Yıl	Yer	Görev
2014-Halen	Gazi Üniversitesi	Araştırma Görevlisi
2012-2014	Kafkas Üniversitesi	Araştırma Görevlisi

Yabancı Dil

İngilizce

Yayınlar

Turgut, S., Uysal, F. ve Hardalaç, F, (2016, 26-28 Ekim). *Eklemeli sentezleyici yöntemi kullanılarak Türkçe metinden konuşma sentezleme işleminde güncellenen bir ses veritabanı oluşturulması.* 1. Uluslararası Akdeniz Bilim ve Mühendislik Kongresi *(IMSEC 2016)*, 2375-2383.

Hobiler

Müzik, Yürüyüş

GAZİ GELECEKTİR...