T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNGİLİZCE'DEN TÜRKÇE'YE MAKİNE ÇEVİRİSİ MODÜLÜ

Cengizhan TEKİN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Konya, 2008

T.C.
SELÇUK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNGİLİZCE'DEN TÜRKÇE'YE MAKİNE ÇEVİRİSİ MODÜLÜ

Cengizhan TEKİN

YÜKSEK LİSANS TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 08.01.2008 tarihinde aşağıdaki jüri tarafından oybirliği / ~~oyçokluğu~~ ile kabul edilmiştir.

Prof.Dr.Ahmet ARSLAN	Prof.Dr.Şirzat Kahramanlı	Yrd.Doç.Dr.Mesut Gündüz
(Danışman)	(Üye)	(Üye)

ÖZET

Yüksek Lisans Tezi

İNGİLİZCE'DEN TÜRKÇE'YE MAKİNE ÇEVİRİSİ MODÜLÜ

Cengizhan TEKİN

Selçuk Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Prof. Dr. Ahmet ARSLAN

2008, 75 Sayfa

Jüri: Prof. Dr. Ahmet ARSLAN
Prof. Dr. Şirzat KAHRAMANLI
Yrd. Doç. Dr. Mesut Gündüz

Bu çalışmada, çok yeni fakat giderek yaygınlaşan ve zamanla her sektörde ihtiyaç haline gelecek olan Makine Çevirisi konu edilmiştir. Makine Çevirisi, uygulama alanları, kullanılan yöntemler araştırılmıştır. Makine Çevirisi'nin somut uygulaması olarak İngilizce'den Türkçe'ye çeviri modülünün altyapısı kurulmuş ve bir başlangıç projesi hazırlanmıştır. Proje boyunca karşılaşılan güçlüklere çözümler aranmış, değerlendirme ve analizler yapılmıştır. Projede, en geçerli yöntemlerden Öge Dizilişi (Kural Tabanlı Çeviri) yöntemi kullanılmıştır. Bu yöntem, insanın, dili öğrenme şekline en çok benzeyen ve en gelişmiş olandır.

Anahtar Kelimeler: Makine Çevirisi, Bilgisayar Destekli Çeviri, Otomatik Dil Çevirisi, Anında Çeviri, Öge Dizilişi Yöntemi, Kural Tabanlı Makine Çevirisi, Sözlük Tabanlı Makine Çevirisi, İstatistiksel Makine Çevirisi, Örnek Tabanlı Makine Çevirisi, Çeviri Hafızası, Ara Dil Tabanlı Makine Çevirisi

ABSTRACT

MS Thesis

MACHINE TRANSLATION MODULE FOR ENGLISH TO TURKISH

Cengizhan TEKİN

Selcuk University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering

Supervisor : Prof. Dr. Ahmet ARSLAN

2008, 75 Pages

Jury: Prof. Dr. Ahmet ARSLAN
Prof. Dr. Şirzat KAHRAMANLI
Assoc. Assoc. Prof. Dr. Mesut Gündüz

In this study, Machine Translation (MT), a very new but rapidly improving subject which will be needed in every branch in the future, was issued. MT, its application fields and methods used, were researched. The infra-structure of translation module of English to Turkish, as a pattern of MT, was established and a startup project was developed. Solutions to the challanges, faced during the study, were researched, the evaluations and analysis were performed. Pattern-based translation (Rule-based), as one of the most valid methods, was used in the project. This method is the most advanced and similar to learning way of human.
Keywords: Machine Translation, Computer Aided Translation, Automatic Language Translation, Instant Translation, Pattern-Based MT, Rule-Based MT, Dictionary-Based MT, Statistical MT, Example-Based MT, Translation Memory, Interlingua-Based MT

ÖNSÖZ

Bilindiği üzere yaşadığımız dünya giderek küreselleşmektedir. Buna bağlı olarak, uluslararası ilişkiler, hayatın her alanında vazgeçilmez hale gelmektedir. Dolayısıyla, farklı dilleri kullanan insanların iletişim kurma gereksinimi artmıştır.

Yine herkesin takdir ettiği üzere, teknoloji, özellikle bilgisayar ve internet hayatımızı oldukça kolaylaştırmıştır. Teknolojinin, bahsettiğimiz iletişimde de kullanılabilmesi olağandır.

Bu temel düşüncelerle, Makine Çevirisi ile ilgili yapılan bu çalışmanın diğer araştırmacılara faydalı olmasını dilerim.

TEŞEKKÜR

Bu çalışmayı gerçekleştirmemde yol gösterici ve manevi destekleyici olan Sayın Danışmanım Prof. Dr. Ahmet Arslan'a ve benden hiçbir desteğini esirgemeyen sevgili aileme teşekkür ederim.

İÇİNDEKİLER

KISALTMALAR

AAMT	: Asia-Pacific Association for Machine Translation
ALPAC	: Automatic Language Processing Advisory Committee
ALT	: Automatic Language Translation
AMTA	: Association for Machine Translation in the Americas
ASCII	: American Standard Code for Information Interchange
ASP	: Active Server Pages
BLEU	: Bilingual Evaluation Understudy
CAT	: Computer Aided Translation
DBMT	: Dictionary-Based Machine Translation
DTD	: Document Type Definition
EAMT	: European Association for Machine Translation
EBMT	: Example-Based Machine Translation
FAHQMT	: Fully Automatic High Quality Machine Translation
GILT	: Globalization, Internationalization, Localization, Translation
GMX	: GILT Metrics
GUI	: Graphical User Interface
IAMT	: International Association for Machine Translation
IBMT	: Interlingua-Based Machine Translation
IDE	: Integrated Development Environment
ISO	: International Organization for Standardization
LISA	: Localization Industry Standards Association
MT	: Machine Translation
NYP	: Nesne Yönelimli Programlama
OLIF	: Open Lexicon Interchange Format
OOP	: Object Oriented Programming
OSCAR	: Open Standards for Container/Content Allowing Re-use
RBMT	: Rule-Based Machine Translation
RED	: Ranker based on Edit Distances
S2SMT	: Speech-to-Speech Machine Translation
SMT	: Statistical Machine Translation

SQL	: Structured Query Language
SRX	: Segmentation Rules Exchange
TBX	: Termbase Exchange
TMBMT	: Translation Memory-Based Machine Translation
TMX	: Translation Memory Exchange
TransWS	: Translation Web Services
UTF	: Unicode Transformation Format
XLIFF	: XML Localisation Interchange File Format
XML	: Extensible Markup Language

ŞEKİLLER VE TABLOLAR ÇİZELGESİ

1 GİRİŞ

1.1 Dil

Çok geniş anlamıyla dil, **düşünce, duygu ve güdüleri, doğrudan doğruya ya da dolaylı olarak bildirmeye yarayan herhangi bir anlatım aracıdır**. Bu tanım bütün canlıların kendi aralarındaki bildirişimlerle ilgili işaret sistemlerini olduğu kadar, insanlar tarafından doğanın ve eşyanın ortak kalıplar halinde manalandırılması olgularını da kapsamaktadır.

İnsanlar ve hayvanlar bir takım sesler ve işaretlerle düşünce, duygu ve güdülerini anlatmaktadırlar. Bunlar birer "dil"dir. Yaprakları solmaya başlayan bir bitki de "susadım" veya "hastayım" demektedir. O halde bitkilerin bile doğaya dönük dilleri vardır. Demek ki tüm canlıların, kendilerini ve hallerini anlatabilme olanakları vardır. Buna dolaysız (doğrudan doğruya) bildirişim denir

Bir de insanların, uzun bir yaşantı sonunda, ortak sembollerle, ortak kalıplarla, evrende, doğada ve eşyada manalandırdıkları, özel anlamlar aşıladıkları, dolaylı birer bildirişim aracı olarak kullandıkları işaretler ve sesler var ki bunlardan da sembolik, artistik bir dil oluşabiliyor; yağmurun dili, denizin dili, göklerin dili, güllerin dili gibi.

Bizim konumuz insan dilidir. Bunun için dilin, dar, bilimsel bir tanımını yapacağız. İnsanların aralarında anlaşmaya, kendilerini ifade etmelerine araç olan **dil, bir gramer sistemi içinde örgütlenmiş, düşünce ve duyguları bildirmeye yarayan ses, işaret ya da hareketlerin bütünüdür.**

İnsan anlatım ve bildirişim için ya hareket eder (jest), ya da ses çıkarır (konuşma) ya da belirli işaretler çizer (yazı). Konuşma dili, yazı dili, hareket dili, insan dilinin üç ayrı görüntüsüdür.

(**http://www.dusunenadam.com.tr/edebiyat14.htm**)

1.2 Özelliklerine Göre Dil Türleri

I- DOĞAL BAKIMDAN	II- TEKNİK BAKIMDAN
1- Doğa dili 2- Hayvan dili 3- İnsan dili	1- Hareket dili 2- Konuşma dili 3- Yazı dili
III- COĞRAFYA BAKIMINDAN	**IV- TARİH BAKIMINDAN**
1- Yabancı dil 2- Milli dil	1- Ölü dil 2- Canlı dil 3- Uygarlık dili
V- ANLATIM DÜZEYİ BAKIMINDAN	**VI- ANLATIM BİÇİMİ BAKIMINDAN**
1- Günlük dil 2- Halk dili 3- Elit dili	1- Bilim dili 2- Sanat dili 3- Teknik dili 4- Kitlesel haberleşme dili a) Basın dili b) Radyo-Televizyon dili 5- Edebiyat Şiir dili 6- Müzik dili 7- Mekanik dil
VII- DİL BİLİM BAKIMINDAN	
1- Benzer dil 2- Devrik dil 3- Analitik dil 4- Sentetik dil.	

Tablo-1.1: Dil Türleri (**Kaynak:http://www.dusunenadam.com.tr/edebiyat14.htm**)

1.3 Dilin Tarihçesi

Bütün canlılar birbirleriyle iletişim kurar. Ancak sadece insanlar bir dil geliştirmiştir. Öyle ki, bu, önceden düzenlenmiş işaretler kümesinden daha fazlasıdır.

Konuşmamız bile, diğer canlıların iletişim kurmasından fiziksel anlamda farklılık göstermektedir. İçgüdüsel olarak çalışmayan bir merkezden gelir ve akılcı bir temele dayalı olarak ses ve anlam olarak düzenlenir. Beynin bu bölümü insanlara hastır.

Bu özel yeteneğin ne zaman ve nasıl geliştiğini söylemek imkansızdır. Fakat, genellikle dil gelişiminin çok uzun bir süreç olduğu varsayılmaktadır. Muhtemelen binlerce yıl önce insanlar konuşuyordu, fakat daha yavaş, daha kısıtlı bir sözlükle ve hepsinden öte alıştığımızdan daha basit bir gramerle.

İnsan dilinin kökeni belki de sonsuza kadar bilinmez şekilde kalacak.

Halen dünyada konuşulan yaklaşık 5000 dil mevcut (üçte biri Afrika'da), fakat bilim adamları ilişkili aileler şeklinde 20'den daha az gruba ayırmaktadır. Diller, ortak kelimeler, sesler ve gramatik yapılarla birbirine bağlıdır. Şöyle bir teori vardır; her dilsel yapı genel bir ata dilden türemiştir.

(**http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ab13**)

1.4 Dünyada En Çok Konuşulan Diller

Sıra	Dil	Alfabe	Sayı (Mil.)	En Çok Konuşulduğu Yerler
1	**Çince**	Çin	1051	Çin, Malezya, Tayvan
2	**İngilizce**	Latin	510	ABD, İngiltere, Avustralya, Kanada
3	**Hintçe**	Sanskrit	490	Kuzey ve Orta Hindistan
4	**İspanyolca**	Latin	425	Kuzey ve Güney Amerika, İspanya
5	**Arapça**	Arap	255	Orta Doğu, Arabistan, Kuzey Afrika
6	**Rusça**	Kiril	254	Rusya, Orta Asya
7	**Portekizce**	Latin	218	Brezilya, Portekiz, Güney Afrika
8	**Bengalce**	Bengal	215	Bangladeş, Doğu Hindistan
9	**Malayca-End. Dili**	Latin	175	Endonezya, Malezya, Singapur
10	**Fransızca**	Latin	130	Fransa, Kanada, Batı ve Orta Afrika
20	**Türkçe**	Latin	75	Türkiye, Orta Asya

Kaynak: http://www.krysstal.com/spoken.html

Tablo-1.2: Dünyada En Çok Konuşulan Diller

1.5 Makine Çevirisi (MT)

Literatürde geçen bazı MT tanımları şöyledir:

EAMT: Makine Çevirisi (*Machine Translation* - MT), bir bilgisayarın, verilen metni, bir doğal dilden diğerine çevirmesini sağlayan uygulamadır.

Wikipedia: Bilgisayar yazılımının metin ya da konuşmayı doğal diller arasında çevirmek amacıyla kullanılmasını araştıran bilgisayar dilbiliminde bir alt daldır.

University of Essex: Bir insan dilinden diğerine gerçekleştirilecek olan çevirme sürecinin bir kısmının ya da tamamının otomatikleştirilmesi girişimidir.

1.6 Makine Çevirisinin Tarihçesi

İnsan dilleri arasında otomatik çeviri düşüncesinin ilk kimden çıktığı tam olarak bilinmemektedir. Ancak, bir MT geliştirme fikri ilk olarak, İngiliz kristalograf Andrew D. Booth ve Rockefeller Foundation'da çalışan Warren Weaver arasında 1947 yılında gerçekleşen bir diyalogta ortaya çıkmıştır. Sonrasında 1949 yılında şirket için, iki cümleden oluşan bir muhtıranın çevirisinde kullanılan bir sistem kurulmuştur.

Bu muhtıra ABD ve Avrupa'da 1950'li yıllarda çok ilgi çekmiş ve o dönemde yapılan çalışmalarda yaklaşık £20M yatırım yapılmıştır. Fakat, bazı başarılı sonuçlar ortaya çıksa da, yatırımların geri dönüşü kısmında yaygın bir hayal kırıklığı söz konusuydu ve genel olarak ya da en azından mevcut bilgilerle çevirinin otomatik hale getirilmesi konusunda şüpheler baş göstermiştir.

Teorik şüpheler Bar-Hillel tarafından 1959'da açık olarak bir raporla somutlaştırıldı. Ona göre; sadece o periyotta değil prensip olarak da tam otomatik, yüksek kalitede bir MT (FAHQMT) geliştirmek imkansızdı. Evet, belki bu imkansız, fakat, MT oluşturmak imkansız ya da faydasız olarak yorumlanmamalıdır.

1964'te yayınlanan US National Academy of Sciences raporunda da konuyla ilgili yatırım konusundaki şüpheler dile getirildi. Bu rapor, o dönemde, Akademi tarafından oluşturulan, Automatic Language Processing Advisory Committee (ALPAC) kuruluşuna aitti. Bu kuruluşun amacı, MT ile ilgili olarak kalite, maliyet, öngörüler, beklentiler ve ihtiyaçlar konusunda çalışmalar yapmaktı.

ALPAC raporu, insan çevirmenler konusunda bir yetersizlik olmadığını, genel bilimsel metinlerin çevirisinde kullanılabilecek bir MT'nin öngörülmediğini belirtmişti. Bu rapor, ABD hükümetinin MT için sağladığı fonun sanal olarak sonu oldu. Daha da kötüsü, sektörde genel bir moral kaybına neden oldu.

Daha yakın zamanlarda baktığımızda, tarihin daha belirsiz ve subjektif olduğunu görüyoruz. 1970'lerde, üzerinde çalışılan MT sistemleri, bilimsel veya teknik nedenlerden daha çok ticaret ve pazarlama kaygıları ile tasarlanmıştır. 1970 ve 1980'li yıllar arasında gerçekleştirilen araştırma projeleri ticari olarak kullanılır hale gelmiştir.

Kişisel bilgisayarlar için 1980'li yıllarda (Weidner MicroCAT) kullanılmaya başlamış olsa da, satışlarındaki yükseliş 1990'lı yıllarda olmuştur. Günümüzde tahminen 1000'den fazla çeşit MT paketi satılmaktadır.

Bu bilgiler şu anlama gelmelidir; MT kesin olarak kurulmuştur, makul bir araştırma alanı ve teknolojinin faydalı bir uygulaması haline gelmiştir. Ancak, MT Ar-Ge çalışmaları, hem teknik olarak, hem de yönetim, organizasyon ve altyapı açısından zordur, ayrıca, zaman, iş gücü ve parasal açıdan yüksek maliyete sahiptir. (**Douglas ve ark. 1994**)

1.7 Makine Çevirisinin Önemi

Yapılması planlanan bir Makine Çevirisi çalışmasında, proje geliştiricinin, ilgili konuyu daha iyi kavraması için, ne kadar önemli olduğu konusunda bir ön araştırma yapması önerilir.

Makine çevirisini genel olarak gerekli kılan en önemli özellikleri şunlardır:

- Diğer Dillerle İletişim Kurmak
- Pazar Payını Artırmak
- Çeviri Sürecini Hızlandırmak

Ayrıca, makine çevirisi kullanılabilirliği sağlandığı sürece, sosyal, siyasi, ticari, bilimsel ve felsefi anlamda büyük öneme sahiptir.

1.7.1 Sosyal veya siyasi açıdan önemi

Bu alandaki önemi, genellikle, birden fazla dilin konuşulduğu topluluklarda, çevirinin öneminden gelmektedir. Böyle yerlerde, çeviri yerine uygulanabilecek tek alternatif tek bir dil seçmek olur ki, bu da seçilmeyen dili konuşanların ikinci sınıf topluluk durumunda düşmesine ve ayrıca, dilin zamanla yok olmasına neden olacaktır. Dilin kaybolması kültürün de zayıflaması anlamına geleceğinden bu alternatif çok geçerli bir çözüm değildir. Dolayısıyla çeviri iletişim için gereklidir. Bunun otomatikleştirilmesi de hayatı kolaylaştıracaktır.

1.7.2 Ticari açıdan önemi

MT'nin ticari önemi ilgili faktörlerin sonucudur. Öncelikle çevirinin kendisi ticari olarak önemlidir, çünkü bir ürün üretildiğinde kullanım kılavuzu hazırlanır ve bu ürünün diğer ülkelerde satılması için kullanım kılavuzunun çevrilmesi gerekir. İkinci olarak, çeviri işlemi pahalıdır. Yüksek yetenek isteyen bir iştir. Bu nedenle bazı ülkelerde çevirmenler iyi gelir elde etmektedirler. Bunun ötesinde, çevirme işleminde yaşanan gecikmeler maliyetin artmasına neden olmaktadır. Tahmini bir hesap yapacak olursak, profesyonel bir çevirmen, günde ortalama 4-6 sayfadan fazla çeviri yapamaz (yaklaşık 2000 kelime) ve bu ortalamayla, ürünün, pazara tanıtımı çok kolay bir şekilde gecikebilir. Avrupa Topluluğu kuruluşları, çeviri için yapılan masrafların tüm giderlerin %40-45'i olduğunu öngörmektedir.

1.7.3 Bilimsel açıdan önemi

Bilimsel olarak MT ilginçtir, çünkü Bilgisayar Bilimi, Yapay Zeka ve Bilgisayar Dilbilimi dallarındaki birçok fikir için açık bir uygulama ve test alanıdır ve bu dallardaki bazı önemli geliştirmeler MT'de başlamıştır.

1.7.4 Felsefi açıdan önemi

Felsefi olarak MT ilginçtir, çünkü, çok geniş insan bilgisi gerektiren bir aktiviteyi otomatize etme girişimidir. Örneğin, *negatif yüklü elektronlar ve protonlar* ifadesini çevirirken protonların pozitif yüklü olduğunu bilmek ifadenin doğru yorumlanmasını sağlayacak ve *negatif yüklü elektronlar ve negatif yüklü protonlar* şeklinde çevrilmesini engelleyecektir.

(Douglas ve ark. 1994)

1.8 Makine Çevirisi İle İlgili Yanlış Düşünceler

- MT, sadece zaman kaybıdır, çünkü Shakespeare'e ait yazıları çevirebilecek bir makine hazırlanamaz.
- Tamamiyle hatalı çeviri yapan sistemler mevcut, MT faydasızdır.
- Genel olarak, bir MT sistemden alacağınız çeviri kalitesi çok düşüktür. Bu, onu, pratikte kullanışsız yapar.
- MT, çevirmenlerin mesleklerini tehdit etmektedir.
- Japonlar, Sesten-Sese (*Speech-to-Speech*) MT yapmışlar.
- Bir Güney Amerika-Hint dili vardır ve sahip olduğu mantıksal yapı MT sistemi tasarlama problemini çözer.
- MT sistemleri makinelerdir ve satın almak bir araba almak gibidir.

1.9 Makine Çevirisi İle İlgili Doğru Düşünceler

- MT faydalıdır. METEO sistemi 1977'den beri günlük hayatta kullanılmaktadır. 1990'da günlük olarak yaklaşık 45000 kelime çeviriyordu. 1980'lerde, dizel motor fabrikası Perkins Engines 4000 pound para ve 15 hafta süre kazanmaktaydı.
- MT sistemleri bazen gaf yapsa da, birçok durumda güvenilir ve hızlı sonuç üretmektedirler.
- Bazı durumlarda MT sistemleri iyi kalitede çıkış üretebilir. METEO çevirilerinin %4'ten daha azı insan çevirmenler tarafından düzeltmeye ihtiyaç duymaktadır.
- MT, çevirmenleri mesleklerini tehdit etmez. Çeviri gereksinimi çok fazladır ve bu ihtiyacın azalması olası değildir. Bununla birlikte, MT sistemleri, tekrarlayan sıkıcı çevirileri hallederek, çevirmenlerin, gerçek yetenek gerektiren işlere konsantre olmasını sağlar.
- Sesten-Sese (*Speech-to-Speech*) MT, hala bir araştırma konusudur.
- MT'de sadece açık araştırma problemleri yoktur, ayrıca, bir MT sistemi kurmak, çok çaba ve zaman isteyen bir iştir. Pratikte, bir MT sisteminin gerçekten faydalı hale gelmesi için, kullanıcının önemli miktarda efor harcayarak, onu özelleştirmesi gerekir. (**Douglas ve ark. 1994**)

2 KAYNAK ARAŞTIRMASI

2.1 Makine Çevirisi Uygulamaları

Giderek yaygınlaşan bu teknolojinin kullanım alanları da artmaktadır. Halen ve gelecekte de en yaygın kullanım sağlanan uygulama alanları şunlardır:

- Düz Metin
- Döküman
- E-Posta
- Web Sayfası
- Yazılım Arayüzleri
- Mobil Uygulamalar
- Ses

2.1.1 Dünya'daki örnekleri

World Lingo – http://www.worldlingo.com

- Düz Metin (+OCR), Döküman, Web Sayfası, Eposta, Yazılım, Chat, Gazete gibi uygulamaları içermektedir.
- 15 Dil, tam karşılıklı olarak çevirmektedir. Bu diller: Almanca, Arapça, Çince (Temel), Çince (Geleneksel), Flemenkçe, Fransızca, İngilizce, İspanyolca, İsveççe, İtalyanca, Japonca, Korece, Portekizce, Rusça, Yunanca

Google - http://www.google.com/language_tools?hl=en

- Düz Metin, Web Sayfası
- 12 Dil, kısmi karşılıklı olarak çevirmektedir. Bu diller: Almanca, Arapça, Çince (Temel), Çince (Geleneksel), Fransızca, İngilizce, İspanyolca, İtalyanca, Japonca, Korece, Portekizce, Rusça.

Altavista – http://babelfish.altavista.digital.com

- Düz Metin, Web Sayfası
- 13 Dil, kısmi karşılıklı olarak çevirmektedir. Bu diller: Almanca, Çince (Temel), Çince (Geleneksel), Flemenkçe, Fransızca, İngilizce, İspanyolca, İtalyanca, Japonca, Korece, Portekizce, Rusça, Yunanca.

Reverso – http://www.reverso.net

- Düz Metin, Web Sayfası
- 5 Dil, sadece İngilizce ile çevirmektedir. Bu diller: Almanca, Fransızca, İngilizce, İtalyanca, İspanyolca

2.1.2 Türkiye'deki örnekleri

Çevirmen – http://www.itc.com.tr

- Düz Metin, Döküman
- İngilizce => Türkçe
- 45.000 kelime genel sözlük
- 35.000 kelime özel sözlük (kullanıcılar geliştirmiş)
- İnteraktif sözlük güncelleme
- Çeviri esnasında bilinmeyen kelimeleri öğrenme
- Bilinmeyen ya da yanlış gramatik yapı bildirme

Bilsag (Web) Çevirim – http://www.bilsag.com.tr

- Düz Metin, Döküman
- İngilizce => Türkçe
- 130.000 kelime sözlük
- Etkileşimli çeviri yapma
- MS Word entegrasyonu

2.2 Makine Çevirisi Yöntemleri

Makine Çevirisi oluşumunda kullanılan yöntemler giderek artmaktadır. En çok bilinen yöntemler:

- Sözlük Tabanlı Makine Çevirisi
- İstatistiksel Makine Çevirisi
- Örnek Tabanlı Makine Çevirisi
- Çeviri Hafızası Makine Çevirisi
- Ara Dil Makine Çevirisi
- Kural Tabanlı Makine Çevirisi

2.2.1 Sözlük Tabanlı Makine Çevirisi (DBMT)

Makine çevirisinde sözlük bilgisine göre uygulanan bir çeviri yöntemidir. Çeviri, anlam karşılıklarına önem vermeksizin, kelime kelime yapılır. Bu işlem sırasında morfolojik analiz yapımı seçime bağlıdır. Bu yaklaşım, en az gelişmiş olan yöntem olsa da, uzun kelime veya ifade listelerinin, envanterlerin veya basit ürün/hizmet kataloglarının çevirisinde faydalı olabilir.

(**http://en.wikipedia.org/wiki/Dictionary-based_machine_translation**)

2.2.2 İstatistiksel Makine Çevirisi (SMT)

Bu yöntem, çevirilerin, parametreleri iki dilli metinlerin analizinden türetilen istatistiksel modeller temeline göre üretildiği bir makine çevirisi örneğidir. İstatistiksel yaklaşım, kural-tabanlı ve örnek-tabanlı makine çevirilerine zıt bir yöntemdir.

Bu yaklaşımın avantajları; genel olarak, herhangi bir dil çiftine özel olarak hazırlanmamaları ve kural tabanlı çeviri sistemlerindeki gibi dil kurallarının manuel olarak geliştirilmesinin gerekmemesi.

Dezavantajları ise; dillere ait bileşik kelimeler, deyimler, özel morfoloji, farklı kelime sıraları, sentaks, sözlük dışı kelimelerin olması.

SMT'nin iki uygulama çeşidi vardır:

a. Kelime Tabanlı Çeviri

Bu yöntemde elemanlar kelimelerdir. Dilin sahip olduğu bileşik kelimeler, morfoloji ve deyimlerden dolayı, çevrilen cümlelerin kelime sayıları farklıdır. Çevrilen kelime dizileri uzunlukları oranı verimlilik (*fertility*) olarak adlandırılır. Basit kelime tabanlı çeviri, verimlilik oranı 1'den farklı olan dil çiftlerinin çevirisini yapamaz. Kelime tabanlı çeviri sistemlerinin bunu başarması için, örneğin yüksek verimlilik oranı varsa, sistem, bir kelimenin birçok karşılığına hakim olmalı ancak bunun tersi olmamalı.

b. İfade Tabanlı Çeviri

Bu yöntemde, farklı uzunluktaki kelime dizilerinin çevrilmesi ile Kelime Tabanlı Çevirideki kısıtlamalar azaltılmaya çalışılmıştır. Kelime dizileri *blok* veya *ifade* olarak adlandırılır ancak, bunlar dil ifadeleri değil, istatistiksel metotlar kullanılarak metinden çıkarsanan ifadelerdir.

(**http://en.wikipedia.org/wiki/Statistical_machine_translation**)

2.2.3 Örnek Tabanlı Makine Çevirisi (EBMT)

Ana bilgi tabanındaki paralel metinlerin kullanımıyla karakterize edilen bir yöntemdir. Bu tip çevirinin kuruluşundaki fikir, insanların, cümleleri çevirirken derin dil analizinden kaçındıkları düşüncesi olmuştur. Bu analiz yerine, insanların cümleleri belli ifadelere ayırıp, çevirdikten sonra tekrar doğru bir şekilde birleştirdikleri düşünülmüştür.

İngilizce	Türkçe
How much is this **red umbrella**?	Bu **kırmızı şemsiye** kaç paradır?
How much is this **small camera**?	Bu **küçük kamera** kaç paradır?

1. How much is that X => Bu X kaç paradır
2. red umbrella => kırmızı şemsiye
3. small camera => küçük kamera

(**http://en.wikipedia.org/wiki/Example-based_machine_translation**)

2.2.4 Çeviri Hafızası Destekli Makine Çevirisi (TMBMT)

Çeviri hafızası, makine çevirisi yazılımlarında kullanılan bir çeşit veritabanıdır. Bu veritabanı, bir kaynak dildeki metin sekmanlarının, diğer hedef dillerdeki karşılıklarını tutar. Bu sekmanlar, blok, paragraf, cümle veya ifade olabilir.

Bu yöntemin en önemli avantajı, metnin tamamen çevrilmesini sağlamasıdır. Çünkü hedef sekmanın boş olmasına izin vermez.

Bu yaklaşımın kullanıldığı bazı standartlar şunlardır:

TMX, TBX, SRX, GMX, OLIF, XLIFF, TransWS (**bkz. KISALTMALAR**)

(**http://en.wikipedia.org/wiki/Translation_memory**)

Bu standartlardan TMX oldukça yaygın ve gelişmiştir:

TMX – Çeviri Hafıza Takası

TMX formatı, yazılım araçları ve çeviri satıcıları arasında takas edilen hafıza verilerini tanımlamak için kullanılan ve çeviride gerekli önemli bilgilerde kayba neden olmayan standart bir metot sağlamaya yarar.

TMX iki kısım ile tanımlanır:

- İçerik formatının tanımlanması (tüm dosya ve girişler ile ilgili bilgi veren yüksek seviye elemanlar). TMX'te, iki veya daha fazla dilde metin parçalarından oluşan bir giriş Çeviri Birimi (*Translation Unit*) (<tu>) olarak adlandırılır.

- Çeviri hafıza metnindeki bir parçanın içeriği için, düşük seviye meta işaretleme formatının tanımlanması. TMX'te, bu parçalar Sekman (*Segment)* (<seg>) olarak adlandırılır.

TMX iki seviye implementasyon sunar:

1. Seviye (Sadece Düz Metin) : Sadece içeren için destek sağlar. Her <seg> elemanı (*element*) içindeki veri, İçerik İşaretleme (*Content Markup*) olmaksızın, düz metin içerir. Verinin içinde kodlama olmadığı zaman bu seviye yeterlidir; örneğin, yazılım mesajları. Doküman tipli formatlar için yeterli değildir.

2. Seviye (İçerik İşaretleme) : Hem içerik hemde içeren ile ilgili destek sağlar. Uygulama, TMX 2. Seviyeyi destekleyen diğer araçların, çevrilmiş versiyonu kullanarak orijinal dokümanı yeniden oluşturmasına izin vermek için TMX İçerik İşaretlemesini kullanır.

XML uyumu:

TMX, XML uyumludur. Ayrıca, tarih/zaman, dil kodları ve ülke kodları için çeşitli ISO standartları kullanır.

TMX dosyaları, export rutinleri tarafında otomatik olarak yaratılır ve import rutinleri tarafından otomatik olarak işlenir. Bu dosyalar, TMX DTD ile explicit referans olmaksızın işlenebilen düzenli hazırlanmış XML dokümanlarıdır. Bununla birlikte, geçerli bir TMX dosya TMX DTD standardına uymalıdır ve şüpheli dosyalar bir XML parser ile doğrulanmalıdır.

XML syntax büyük/küçük harf duyarlı olduğu için, TMX'e ait bütün eleman (*element*) ve nitelik (*attribute*) ifadeleri küçük harfle tanımlanmıştır.

TMX namespace olarak http://www.lisa.org/tmx14 ifadesiyle tanımlanır.

```
<myformat>
 <data>
  <tmx xmlns="http://www.lisa.org/tmx14"
       version="1.4">
   ... TMX data ...
  </tmx>
 </data>
</myformat>
```

Karakter kodlaması:

TMX dosyaları her zaman Unicode ile hazırlanır. Şu üç kodlamadan herhangi biri kullanılabilir:

- UTF-16 (16-bit dosyalar)
- UTF-8 (8-bit dosyalar)
- ISO-646 [ASCII] (7-bit dosyalar)

Her kodlamada, HTML'nin aksine, sadece aşağıdaki beş karaktere izin verilir:

- & (&)
- < (<)
- > (>)
- ' (‘)
- " (“)

7-bit dosyalar için, ilave karakterler daima sayısal referanslar ile temsil edilir. Örneğin &#x0396, Ζ gibi.

Genel Yapısı:

Bir TMX doküman bir <tmx> kök elemanı içinde barındırılır. Bu eleman, <header> ve <body> elemanlarını içerir.

*** Başlık (*Header*)**

Bu bölüm, dokümanla ilgili meta-data içerir. Niteliklere (*Attribute*) ilaveten, <note> ve <prop> elemanlarında doküman seviyesinde bilgi depolayabilir. Ayrıca, kullanıcı tanımlı karakterlerde bu bölümde <ude> elemanları ile listelenebilir.

*** Gövde (*Body*)**

Bu bölüm <tu> elemanlarını içerir. Bu elemanlar özel bir sırada değildir. Her eleman, en az bir Çeviri Birimi (*Translation Unit*) değişkeni (<tuv>) içerir . Her <tuv> verilen bir dil için kullanılan sekman ve bilgiyi tutar.

Metnin kendisi <seg> elemanında tutulur. <note> ve <prop> elemanları ise her bir <tuv> ile ilgili bilgiyi saklamak için kullanılabilir.

Bir sekman, işaretleme içerik elemanları içerebilir: <bpt>, <ept>, <it> ve <ph> elemanları orijinal iç kodları kapsamak için kullanılır. Ve <sub> elemanı ilişkili metinleri ayırmak için kullanılır.

TMX elemanları:

Yapısal Elemanlar	`<body>`, `<header>`, `<map>`, `<note>`, `<prop>`, `<seg>`, `<tmx>`, `<tu>`, `<tuv>`, `<ude>`.
İç Elemanlar	`<bpt>`, `<ept>`, `<hi>`, `<it>`, `<ph>`, `<sub>`, `<ut>`.

TMX nitelikleri:

TMX Nitelikler,	`adminlang`, `assoc`, `base`, `changedate`, `changeid`, `code`, `creationdate`, `creationid`, `creationtool`, `creationtoolversion`, `datatype`, `ent`, `i` `lastusagedate`, `name`, `o-encoding`, `o-tmf`, `pos`, `segtype`, `srclang`, `subst`, `tuid`, `type`, `unicode`, `usagecount`, `version`, `x`.
XML AdAlanı Nitelikleri	`xml:lang`.

(**http://www.lisa.org/standards/tmx/tmx.html**)

2.2.5 Ara Dil Makine Çevirisi (IBMT)

Bu yöntem kural tabanlı makine çevirisinin farklı bir yaklaşımıdır. Buna göre, önce kaynak dildeki metin, hedef veya kaynak dilden bağımsız başka bir ara dile çevrilir, daha sonra bu ara dilden hedef dile çeviri yapılır.

Avantajı : Her dil ikilisi için ayrı bir çeviri bileşeni oluşturmak gerekmez.

Dezavantajı: Bu tip bir ara dili tanımlamak zordur. Hatta çok geniş dil sayısını destekleyecek bir ara dil oluşturmak mümkün olmayabilir.

(**http://en.wikipedia.org/wiki/Interlingual_machine_translation**)

2.2.6 Kural Tabanlı Makine Çevirisi (RBMT)

Diğer bir adı Öge Dizilişi yöntemidir. Bu yöntemde öncelikle geniş kapsamlı bir sözlük gereklidir. Bu sözlüğün tutulacağı veritabanı, kelimelerin kaynak dil değerleri, hedef dil karşılıkları ve kelime türlerini (isim, sıfat, fiil,...) mutlaka içermelidir.

Kaynak_Dil	**Hedef_Dil**	**Kelime_Tipi**
Book	Kitap	Isim
Do	Yapmak	Fiil
Blue	Mavi	Sıfat

Tablo-2.1: Sözlük Veri Tablosuna Ait Basit Bir Yapı

İkinci olarak, bir kural tabanı hazırlanmalıdır. Kural tabanını oluşturan kurallar, kaynak dildeki cümle dizilişinin, hedef dildeki karşılıklarını belirlemektedir.

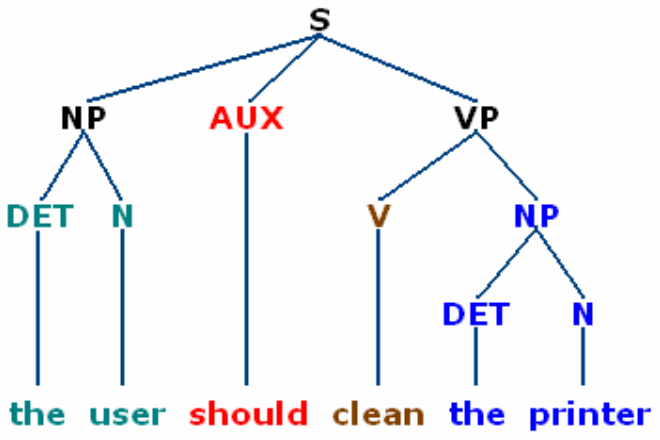
Kaynak_Dil	Özne – Yüklem – Nesne
Hedef_Dil	Özne – Nesne - Yüklem

Tablo-2.2: Kural Tabanı Örneği

Son olarak bir araç (parser) aracılığıyla, kaynak dildeki cümle ayrıştırılır, sözlükten kelime karşılıkları bulunur, kural tabanından hedef dilin yapısı belirlenip, bu yapıya göre kelimeler uygun biçimlerde birleştirilerek sonuca gidilir.

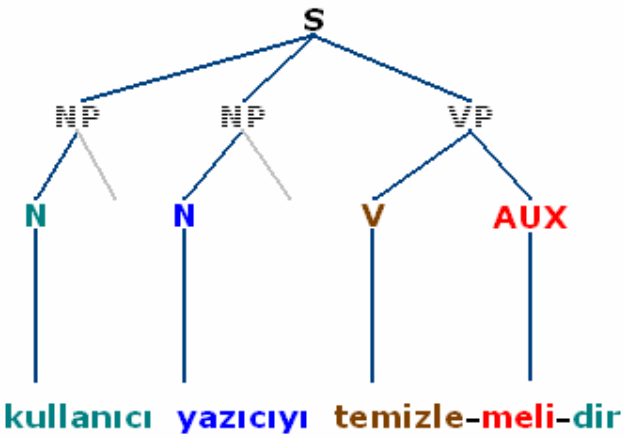
Öge Dizilişi (*Pattern-Based*) Çeviri Örneği

İNGİLİZCE :



Özne + Yardımcı Fiil + Yüklem + Nesne

TÜRKÇE :



Özne + Nesne + Yüklem

Şekil-2.1: Öge Dizilişi Çeviri Örneği

2.3 Makine Çevirisi'nin Değerlendirilmesi

Makine Çevirisi alanında yapılacak herhangi bir çalışmanın, daha önceki araştırmalara göre oluşturulmuş bazı testlerle değerlendirilmesi, projenin başarısı ve geleceği ile ilgili oldukça önemli miktarda bilgi verici olacaktır. Ancak, bu metotların uygulanması için çalışmanın ve gerekli verilerin, özellikle oluşturulan sözlüğün, belli bir safhaya taşınmış olması gerekir.

Halen uygulanmakta olan değerlendirme yöntemleri şunlardır:

2.3.1 ATR standardı

MT sistemlerin değerlendirilmesi için kabul edilmiş (Sumita 1999) bir standarttır. Bu standardın 4 seviyesi vardır:

A – Mükemmel:

Hem bilgide hem de gramerde hiç bir problem yoktur, gayet anlaşılır bir sonuç üretir.

B – Vasat:

Küçük bilgi eksikliği veya gramer kusurları vardır, ancak anlaması kolaydır.

C – Kabul Edilebilir:

Bozuk bir sonuç üretir ama biraz çaba gösterilirse anlaşılır.

D – Saçma:

Önemli miktarda bilginin çevirisi yanlış yapılmıştır ve anlaşılamaz.

(**Douglas ve ark. 1994**)

2.3.2 MT test yöntemleri

En çok bilinen iki otomatik MT değerlendirici eş zamanlı olarak hazırlanmıştır: BLEU (Bilingual Evaluation Understudy – Papineni 2001) ve RED (Ranker Based On Edit Distances - Akiba 2001) . Bunlar hızlı çalışan, MT değerlendirme metodudur. BLEU, MT sistemlerim performansını değerlendirmek üzere test maiyetli olarak tasarlanmıştır. Diğer taraftan, RED, çeviri parçlarını değerlendirmek için tasarlanmıştır. Bununla birlikte, RED, çeviri değerlendirme sonuçlarının tamamı toplanarak elde edilen sonuca gore MT sistemlerin performans değerlendirmesinde de kullanılabilir.

Her iki otomation değerlendiricinin de insan değerlendirmesine yakın bir değerlendirme yapabileceği bildirilmiştir. BLEU, çeviride insan değerlendirmesinin iki ölçüsü olan yeterlilik ve akıcılık faktörlerine uyumlu çalışan bir yöntem olarak gösterilmiştir. Bu faktörler 1 (Çok Kötü)'den 5 (Çok İyi)'e derecelendirilmiştir. RED ise bir MT sisteminin performansını çok yaklaşık olarak değerlendirir ve ATR standardına göre ölçümlendirme yapar.

Birçok araştırmacı, BLEU değerlendirme sonuçları ile MT sistemleri geliştirmek için düşünülen yeni yaklaşımların kalitesi konusunda ısrarcı olmalarına rağmen, otomatik değerlendiricilerin mükemmel olarak veya yeterince güvenli bir şekilde çalıştığı konusunda şüpheler mevcuttur. Örneğin, regresyon sonuçlarına göre, BLEU ve insan değerlendirmesi arasında küçük fakat önemli bir boşluk vardır. Bu boşluk, BLEU'nun yanlış sonuç üretmesine neden olabilir.

RED Yöntemi:

RED, referans çevirilerle olan farka dayanan otomatik bir ölçüm metodudur. Bir öğrenme ve bir de değerlendirme fazından oluşur. Öğrenme fazında, rank tahmini yapmak için, RED ölçülendirme için bir Karar Ağacı (Ranker) öğrenir. Değerlendirme fazında ise bu ağacı kullanarak her MT çıkışı için bir rank atar.

Her ölçümleme örneği, farklılıklara göre kodlanır ve üç ya da daha fazla insan tarafından tespit edilen rank ler arasından bir medyan rank atanır.

BLEU Yöntemi:

BLEU, birçok referans çevirisi ile n-gram eşleştirmesine dayalı otomatik skorlama yöntemidir. Unigram, bigram, trigram ve 4-gram ların doğruluğunu ölçer. Çok kısa cümleler için ceza puanı verir. BLEU skoru yüksek olursa sonuç daha iyidir.

FARKLAR	RED	BLEU
Değerlendirme Birimi	Bir ifade	Bir bölüm
Değerlendirme Hedefi	Bir ifade	Bir doküman
Değerlendirme Sonuçları	Rank	Skor
Öğrenme Stratejisi	Denetimli	Öğrenmiyor
Yaklaşım	Farklılık	N-gram Eşleştirmesi
Kelimeleri Değiştirme Güçlülüğü	Nazaran Zayıf	Güçlü

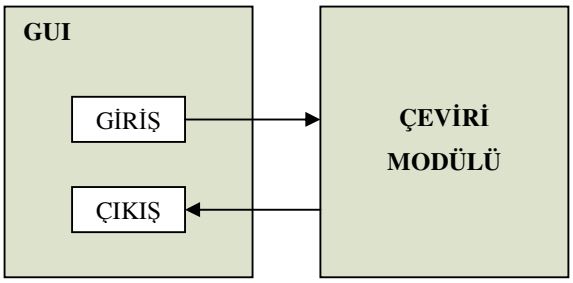
Tablo-2.3: RED ve BLEU Yöntemleri Farkları

(**Akibay ve ark.**)

3 MATERYAL VE METOT

3.1 Kullanılan Teknolojiler

Proje, halen Micrsoft'un son teknolojileri olan Visual Studio.NET 2005 platformu ve SQL Server 2005 Veritabanı sunucusu kullanılarak hazırlanmıştır. Kodlama, üç-katmanlı mimari ve tamamen nesne yönelimli programlama (OOP) mantığı ile yapılmıştır. Bu yöntem kullanıldığı için, modüler bir yapı sağlanmış ve çeviri programının birçok sisteme kolaylıkla entegre edilebilmesi hedeflenmiştir. Ayrıca, kullanıcı arayüzünde (GUI) olabilecek kod yükü engellenmiştir.



Şekil-3.1: Kullanıcı Arayüzü Blok Şeması

Bu modüler yapı, dil ve uygulama anlamında da esneklik sağlamıştır, yani uygulama, Windows, Web, Console vb. ile gerçekleştirilebilen bir şekilde hazırlanmıştır. Bu modülü hazırlamak için C# programlama dili tercih edilmiş, uygulama testi ASP.NET web formları ile yapılmıştır.

3.1.1 Visual Studio.NET

Microsoft'un geliştirme platformunda tüm .NET dilleri için tek bir Tümleşik Geliştirme Ortamı (IDE) sağlamaktadır. Bu, Visual Basic, Visual C++ ve Visual C# geliştiricilerinin tümünün, aynı ortamdaki farklı diller üzerinde hata ayıklama ve hata işleme becerileri de dahil, aynı IDE'yi paylaşmaları anlamına gelir.

Visual Studio.NET önemli ölçüde yeni özellikler sağlar:

- Tüm .NET dilleri ile hem Windows hem de Web uygulamaları için tek ve birleşik bir programlama modeli,
- Server Explorer'ı kullanan sunucu için sürükle ve bırak geliştirmesi,

- Devingen Yardım (Dynamic Help),
- IDE için sağlam bir özelleştirme ve genişletme modeli,
- Güçlü XML desteği,
- Çok daha kolay platformlar arası uygulama tümleştirme özelliğine sahip web hizmetleri. (**Duthie 2005**)

3.1.2 C#

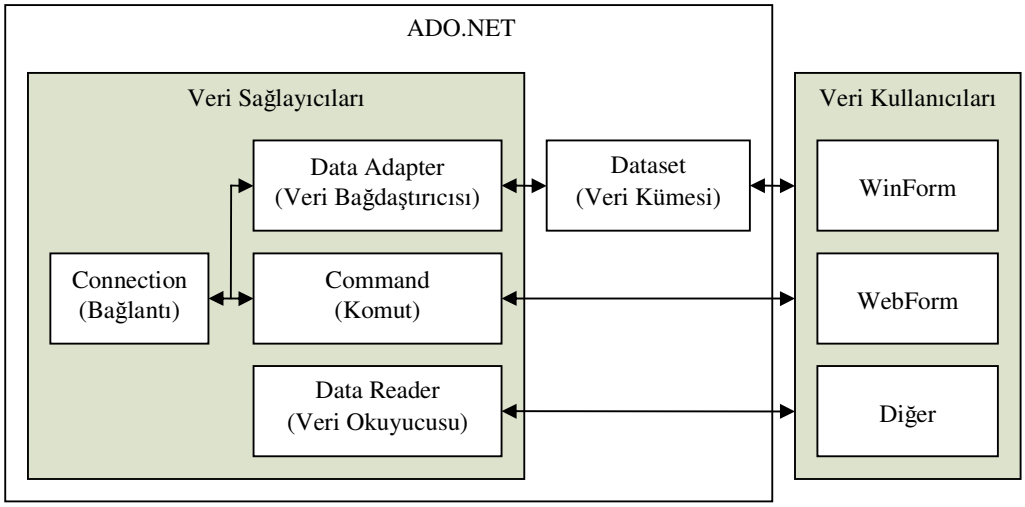
Visual Studio ailesinin yeni bir üyesi olan C#, C ailesinden gelir. C++'a çok benzerdir ama çok daha basit olması ve kolay kullanılması düşünülerek tasarlanmıştır. C#, bellek ayırma ve boşaltma yönetimi gerektirmez. (**Archer 2004**)

3.1.3 ASP.NET

ASP.NET yani Active Server Pages .NET Microsoft'un yeni vizyonu .NET ile duyurmuş olduğu internet uygulamaları, web servisleri ve mobil uygulamalar için sunucu taraflı yazılım geliştirmeyi kolay, güvenli ve genişleyebilir yapıda sağlayan uygulama geliştirme teknolojidir. ASP .NET ile sunucu tarafında çalışacak uygulamayı hızlı bir biçimde geliştirirken aynı zamanda istemci tarafındaki kontrolleri hazırlayabilirsiniz. Bunların yanında uygulamayı geliştirirken .NET Framework'un sağladığı nesnelerin gücünden faydalanabilirsiniz. (**http://www.yazgelistir.com/Makaleler/makaleler.aspx?KatId=1000000000&Kat=ASP.NET**)

3.1.4 ADO.NET

ADO, Microsoft´un tüm Windows tabanlı programlama dillerinde programcıların veritabanı işlemlerini kolay, hızlı ve güvenli bir şekilde yapması için oluşturduğu bir arayüzdür. ADO.NET ise ADO'nun Visual Studio.NET platformunda kullanılan gelişmiş yeni versiyonudur. (**Riordan 2003**)



Şekil-3.2: ADO.NET Bileşenleri

3.1.5 Nesne yönelimli programlama (OOP)

1960'lı yılların sonuna doğru ortaya çıkan bu yaklaşım, o dönemin yazılım dünyasında beliren bir bunalımın sonucudur. Yazılımların karmaşıklığı ve boyutları sürekli artıyor, ancak belli bir nitelik düzeyi korumak için gereken bakımın maliyeti (zaman ve çaba olarak) daha da hızlı artıyordu. NYP'yi bu soruna karşı bir çözüm haline getiren başlıca özelliği, yazılımda birimselliği (*modularity*) benimsemesidir.

NYP'nın altında yatan birimselliğin ana fikri, her bilgisayar programının (izlence), etkileşim içerisinde olan birimler veya nesneler kümesinden oluştuğu varsayımıdır. Bu nesnelerin her biri, kendi içerisinde veri işleyebilir, ve diğer nesneler ile çift yönlü veri alışverişinde bulunabilir. Halbuki NYP'dan önce var olan tek yaklaşımda (Yordamsal programlama), programlar sadece bir komut dizisi veya birer işlev (fonksiyon) kümesi olarak görülmektediler.

Bilimsel çevreler tarafından NYP'nın geçmişe göre daha yüksek esneklik ve bakım kolaylığı sunduğu iddia edilmektedir. Bu sebepten dolayı, günümüzün geniş çaplı yazılım projelerinde yaygınca kullanılmaktadır.

NYP'nin sağladığı avantajlar şunlardır:

- Programlara modüler bir yapı kazandırır,
- Mevcut kodların bakım ve güncellemesini kolaylaştırır,
- Programcının mevcut yazılım bileşenlerini kolaylıkla adapte edebileceği kod kütüphaneleri için iyi bir çatı kurar.

NYP'nın bilgisayar bilimlerine girişinin üzerinden 40 yılı aşkın bir süre geçmiş olmasına rağmen, henüz yaygın olarak kabul görmüş bir tanımı yoktur. Ancak önerilen sayısız tanımların çoğunluğunda yer alan bazı ortak kavramlar sayılabilir:

Sınıf (*Class*) — bir 'şeyin' soyut özellik ve davranışlarını tanımlar. Örneğin 'Köpek' sınıfının, tüm köpeklerde ortak olarak görülen özellikleri (tür, renk, vb), ve davranışları (havlamak) içermesi beklenir. "Özellikler" (*attributes*) ile "davranışlar"ı (*events*) tanımlayan "yöntemler"e (*methods*) topluca "sınıf üyeleri" (*class members*) denir. Sınıflar NYP'nın birimselliğini sağlayan en önemli öğeler konumundadırlar ve tasarım ilkesi olarak, tanımladıkları 'şey' ile aşina kişilerce kolayca anlaşılmaları beklenmektedir.

Nesne (*Object*) — bir sınıfın örneği veya olumudur (*instance*). Örneğin 'Köpek' tüm köpekleri tanımlayan bir sınıf, yani soyut bir yapı ise, 'Karabaş' bu sınıfa ait bir bireyi, yani somut bir nesneyi tanımlar. Sınıf üyelerinin renk özelliği ve havlamak yönteminden oluştuğunu var sayarsak, Karabaş da pekala beyaz renk ve havlama yeteneğine sahip bir köpek olabilir. Tanım itibari ile, bir nesnenin özelliklerinin değer kümesine "nesnenin durumu" (*object state*) denir.

Yöntem (*Method*) — bir nesnenin yeteneklerine denir. Herbir yöntem, nesnenin yapabileceği bir davranışı simgeler. Örneğin 'Karabaş', 'Köpek' sınıfına ait bir nesne olarak, o sınıfta tanımlı davranışları sergiler, yani havlar. Diğer bir deyişle, 'havlamak' 'Karabaş'ın bir yöntemidir.

Kalıtım (*Inheritance*) — sınıflararası sıradüzensel (hiyerarşik) bir ilişkiyi tanımlar. Ortalama bir yazılım tasarısında sınıf sayısının onlar hatta yüzler ile ifade edildiği düşünülürse, yapı esnekliğini sağlamak açısından bu tür ilişkiler hayati önem taşır. Kısaca kalıtım 'altsınıflar', yani belli bir sınıf veya sınıflardan türeyen, uzmanlaşmış/ayrıntılandırılmış sınıflar yaratılmasına olanak tanır. Örneğimize dönecek olursak, genel 'Köpek' sınıfının olası ayrıntılandırılmış altsınıfları, 'Kangal',

‘Dalmaçyalı’ ve ‘Sibirya kurdu’ olabilir. Bu altsınıflara, 'Köpek' sınıfının üyeleri kalıtım ilkesine göre “miras” kalır. Böylece, tekrar yazmaya gerek kalmadan her üç alt sınıf da 'Köpek' sınıfında yer alan renk özelliğine ve havlamak yöntemine sahip olurlar. Kalıtımın bu özelliği sayesinde yazılım geliştirme süresinde ciddi oranda tasarruf edilmektedir. Üst sınıftan miras kalan sınıf üyelerinin yanısıra, her altsınıf kendi özellik ve yöntemlerini de tanımlayabilir. Örneğin Kangal altsınıfına ona özel bir özellik olan ‘kurt_boğ’ yöntemi eklenebilir. Böylece, eğer 'Karabaş' ‘Kangal’ sınıfına ait ise, üstsınıf olan 'Köpek' sınıfının renk ve havlamak üyelerinin yanısıra ‘kurt_boğ’ yöntemine de sahip olur. Bir sınıfın birden fazla üstsınıfı olduğu takdirde “çoklu kalıtım” dan söz edilmektedir.

Sarma (*Encapsulation*) — bir sınıfın içeriğinin, onun üyelerini kullananlara, ancak gerektiği kadar erişim hakkı tanınmasından. Bu şekilde, dış dünyaya açık olan sınıf üyelerinin imzası değiştirilmedikçe, o sınıfın nesnelerini kullananlara sorun yaratmadan, sınıf içerisinde değişikliklere gidilebilir. Örnek vermek gerekirse, 'Köpek' sınıfındaki havlamak yöntemi, havlamayı tam olarak tanımlar (mesela, önce *nefes_al* ve sonra *nefes_ver* yöntemleri ile), ancak bu sınıfı kullanacak olanlara nefes_al ve nefes_ver yöntemlerine erişim hakkı tanımak gereksizdir, zira havlamanın nasıl gerçekleştiği onları ilgilendirmemektedir. Başka bir deyişle, bu iki yöntem “sarmalanmıştır”. Sonuç itibarı ile sınıf üyeleri genelde üçe ayrılır, tüm sınıflarca kullanılanlar (*public*), altsınıflarca kullanılanlar (*protected*) ve sadece üyelerin ait olduğu sınıfça kullanılanlar (*private*).

Çok biçimlilik (*Polymorphism*) — çok biçimlilik, belli bir davranışın, gerçekleştiren sınıfa bağlı olmasını tanımlar. Örneğin, 'Hayvan' sınıfı 'konuşmak' yöntemine sahip olsun. 'Köpek' ve 'Kedi' onun altsınıfları olmalarına rağmen, köpeğin konuşması havlamak biçiminde, kedininki ise miyavlamak olarak gerçekleşebilir.

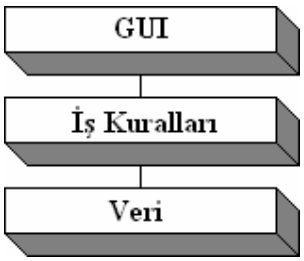
Soyutlama (*Abstraction*) — belli bir kavram yapısını sınıflar ile oluştururken, söz konusu soruna en uygun kalıtım düzeyini belirtmeyi tanımlar. Örneğin, bazı durumlar için 'Köpek' ve 'Kangal' altsınıfı yeterli olurken, daha

karmaşık durumlar Karabaşın 'Hayvan','Memeli','Evcil','Köpek','Kangal' kalıtım dizisine ait bir nesne olmasını gerektirebilir.

(**http://tr.wikipedia.org/wiki/Nesne_Y%C3%B6nelimli_Programlama**)

3.1.6 Üç-Katmanlı Mimari

Üç-katman (*Three-tier*), kullanıcı arayüzü, işlevsel süreç mantığı, veri deposu ve veri erişiminin bağımsız modüller olarak geliştirildiği bir istemci-sunucu mimarisidir. (**http://en.wikipedia.org/wiki/Three_layer_architecture**)



Şekil-3.3: Katmanlı Mimari Blok Şeması (**http://www.stevenblack.com/PTN-Layers.asp**)

Genel olarak katmanlı mimari kullanmanın sağladığı faydalar şöyledir:

- Karmaşık problemler, daha küçük ve yönetilebilir parçalara ayrılabilir.
- Bir katmanın niteliği, uygulaması hakkında bilgi vermediği için, uygulama detayları, diğer katmanlardan gizlenmiş olur (soyutlanır).
- Birçok üst katman, bir alt katmanın hizmetlerini paylaşabilir. Böylelikle işlevselliğin yeniden kullanılması sağlanmış olur.
- Mantıksal gruplandırmanın sayesinde, ekip çalışması gerçekleştirilebilir.
- İleriki bir zamanda, bölümlerin kolaylıkla takas edilebilmesi sağlanır.

3.1.7 SQL Server 2005

Saklı Yordamlar (*Stored Procedures*) — Bir saklı yordam, daha önceden oluşturulan ve sunucu veritabanında saklanan, isimlendirilmiş SQL ifadeleri grubudur. Saklı yordamlar parametre kullanımına uygun olarak hazırlanabilirler, böylece tek bir yordam ağ üzerinden birçok istemci tarafından farklı verilerle kullanılabilir.

Saklı yordamların sağladığı avantajlar şöyledir:

- Çalışma Planını bellekte tutma ve yeniden kullanma,
- Otomatik sorgu parametreleri kullanma,
- İş kural ve politikalarının sarmallaştırılması,
- Uygulama modülerizasyonu,
- Uygulamalar arasında mantık paylaşımı,
- Veritabanına güvenli ve tümleşik erişim,
- Tutarlı ve güvenli veri güncellemesi,
- Ağ band genişliği kullanımında tasarruf,
- Sistem başlangıcında otomatik çalıştırma desteği.

Şemalar (*Schemas*) — SQL Server 2005'te yeni bir özellik olarak sunulan ve tablo, saklı yordam vb bileşenleri mantıksal olarak gruplandırmak ve ortak nitelikler katmak için kullanılabilen yapılardır.

(**Riordan 2004**)

3.2 Projenin Gerçekleştirilmesi

3.2.1 Veritabanı

Veritabanında tablolar ve saklı yordamlar oluşturulurken, tablo işlevselliklerine göre gruplandırma yapılmıştır. Bu gruplandırma Şema denen yapılar ile sağlanmıştır. Projede 5 farklı şema kullanılmıştır: cod, dbo, exc, sfx, tns.

Tablolar

- [cod] – Yazılımda gerekli kodları içeren tablolar bu şemada toplanmıştır.

cod.LetterType : Sesli ve sessiz harf kodlarını içerir.

cod.NounCase : İsim hal ekleri kodlarını içerir.

cod.VerbType : Fiil tipi kodlarını içerir.

cod.WordType : Sözcük türü kodlarını içerir.

- [dbo] – Veritabanın varsayılan şemasıdır ve genel tablolar bu şemadadır.

dbo.Dictionary : Genel sözlük tablosudur. Sözcüğün İngilizce, Türkçe karşılıklarını ve sözcük türünü içerir.

dbo.NounCases : İngilizce hal eklerini içerir.

dbo.Patterns : İngilizce ve Türkçe Patternleri ve Tense bilgilerini içerir.

dbo.QuestionWords : Soru kelimelerini içerir.

dbo.Tenses : Tenseleri içerir.

dbo.Verbs : Fiiller ve fiil tiplerini içerir.

- [exc] – Dillerle ilgili istisnai durumların çözülmesinde kullanılan tabloları içeren şemadır.

exc.SessizYumusamasi : Sessiz yumuşamasına uğramayan kelimeleri içerir.

exc.SimplePresentTense : Geniş zamandaki istisnai kelimeleri içerir.

- [sfx] – Türkçe ekleri içeren tabloları barındıran şemadır.

sfx.Case : Türkçe hal eklerini içerir.

sfx.Personal : Şahıs eklerini içerir.

- [tns] – Tense eklerini içeren tabloları barındıran şemadır.

tns.PresentContinousTense : Şimdiki zaman eklerini içerir.

tns.SimpleFutureTense : Gelecek zaman eklerini içerir.

tns.SimplePresentTense : Geniş zaman eklerini içerir.

Saklı Yordamlar:

- [dbo] – Genel tablolarla bağlantı kurup bilgi sorgulayan yordamları içerir.

dbo.get_NounCaseCode : Hal eki kodunu döndürür.

dbo.get_Pattern : İngilizce patternin Türkçe karşılığını döndürür.

dbo.get_VerbType : İngilizce fiilin tipini döndürür.

dbo.list_TurWords : İngilizce kelimenin Türkçe karşılıklarını ve türlerini listeler.

- [exc] – İstisnai durumlarla ilgili tablolardan bilgi sorgulayan yordamları içeren şemadır.

exc.chk_SessizYumusamasi : Verilen kelimenin, sessiz yumuşaması olayında istisnası olup olmadığını kontrol eder.

exc.chk_SimplePresentTense : Verilen fiilin geniş zamanda istisnası olup olmadığını kontrol eder.

- [sfx] – Ek tablolarından veri alınmasını sağlayan yordamları içerir.

sfx.get_Case : Hal eklerini döndürür.

sfx.get_Personal : Şahıs eklerini döndürür.

- [tns] – Tense tablolarından veri alınmasını sağlayan yordamları içerir.

tns.get_PresentContinuousTense : Şimdiki zaman eklerini döndürür.

tns.get_SimpleFutureTense : Gelecek zaman eklerini döndürür.

tns.get_SimplePresentTense : Geniş zaman eklerini döndürür.

3.2.2 Veri Katmanı

Bu katmanda veritabanı ile direkt bağlantı kurup, gerekli bilgi alışverişini sağlayan sınıflar vardır.

Data.cs:

Bu sınıfta veritabanına bağlantıyı sağlayan ve açma, kapama işlemlerini gerçekleştiren metotlar ile genel dbo şemasına ait işlemleri gerçekleştiren metotlar bulunmaktadır.

- *ToTurkish()* : Parametre olarak verilen İngilizce kelimenin Türkçe karşılıklarını ve sözcük türlerini bir tabloyla döndürür.
- *GetWordType()* : Parametre olarak verilen İngilizce kelimenin sözcük türünü ve sözcük alt türünü döndürür.

- *GetVerbType()* : Parametre olarak verilen İngilizce fiilin, türünü döndürür.
- *GetPattern()* : Parametre olarak verilen İngilizce patternin (sözcük sıralaması) Türkçe karşılığını ve ait olduğu zaman ve çatı yapısını döndürür.
- *GetNounCaseCode()* : Parametre olarak verilen İngilizce isim hal ekinin Türkçe karşılığını döndürür.

DataCode.cs:

Veritabanında [cod] şemasına ait olan objelerle bilgi alışverişini gerçekleştirmek için kullanılan metotları içermek üzere tasarlanmıştır.

DataException.cs:

Veritabanındaki istisnai durumlarla ilgili [exc] şemasına bağlı tablolarla bilgil alışverişi yapan sınıfları içerir.

- *SimplePresentTense()* : Türkçe'de bazı tek heceli fiillerde, geniş zamanda ortaya çıkan istisnai durumun belirlenmesi ile görevlidir.
- *SessizYumusamasi()* : Türkçe'de, kural dışı olarak sessiz yumuşamasına uğramayan bazı kelimelerin tespit edilmesi ile görevlidir.

DataSuffix.cs:

Veritabanında, Türkçe eklerin bulundurulduğu tablolarla bilgi alışverişi yapan sınıfları içerir.

- *Personal()* : Parametre olarak verilen özne, son harf tipi ve son sesli harf değişkenlerine göre, kelimeye eklenmesi gereken şahıs ekini, varsa kaynaştırma harfi ile birleştirerek döndürür.
- *Case()* : Parametre olarak verilen hal kodu, son harf tipi ve son sesli harf değişkenlerine göre, kelimeye eklenmesi gereken hal ekini, varsa kaynaştırma harfi ile birleştirerek döndürür.

DataTense.cs:

Veritabanında cümle zamanları ile ilgili [tns] şemasına bağlı tablolarla bilgi alışverişi yapan metotları içerir.

- *SimplePresentTense()* : Parametre olarak verilen, istisnailik kodu, tek hecelilik kodu, son harf tipi ve son sesli harf değişkenlerine göre, varsa gerekli değişikliklerle birlikte geniş zaman çıktısını döndürür.
- *PresentContinuousTense()* : Parametre olarak verilen, son sesli harf değişkenine göre varsa gerekli değişikliklerle birlikte şimdiki zaman çıktısını döndürür.
- *SimpleFutureTense()* : Parametre olarak verilen, son harf tipi ve son sesli harf değişkenlerine göre varsa gerekli değişikliklerle birlikte gelecek zaman çıktısını döndürür.

3.2.3 İş Katmanı

GUI ve Veri katmanı arasında iletişimi sağlayan, iş kurallarının gerçekleştirildiği ve hiyerarşik bir yapıda birbirine bağlanarak veri alışverişi sağlayan sınıfları içerir.

Machine.cs:

GUI'nin doğrudan kullandığı tek sınıftır. Kullanımı çok kolay olarak hazırlanan ve bir ara sınıf olarak kullanılan bu yapıyla, Makine Çevirisi Modülünü bir çok platforma entegre etmek çok kolay hale getirilmiştir. Sadece iki satır kod yazarak kaynak bilgi, modüle iletilir ve hedef bilgiye ulaşılır.

- *Translate()* : Bu metotla, alınan giriş bilgisi bir alt sınıfa aktarılır ve çıktı beklenir.

Text.cs:

GUI ile aracılık eden ve aslında tüm modülün ana sınıfı sayılabilecek olan Machine sınıfından aldığı bilgiyi metin olarak kabul eder, içerdiği bazı metotlarla alt sınıfa aktarır.

- *Paragraphs()* : Gelen metni, satır sonu karakterine ('\n') göre paragraflara ayırıp, bir string paragraf dizisi oluşturur.

- *Translate()* : Metnin ayrıştırılması ile ortaya çıkan paragrafları bir alt sınıfa aktarır ve geri dönen çıktıları yine satır sonu karakteriyle birleştirerek bir üst sınıfa döndürür.

Paragraph.cs:

Bir üst sınıf olan Text sınıfından döngüyle gelen paragrafları cümlelere ayırır ve içerdiği metotlarla bir alt sınıfa aktarır.

- *Sentences()* : Gelen paragrafı, uygun noktalama işaretlerine (‘.’, ‘?’, ‘!’, ‘:’) göre cümlelere ayırıp bir string cümle dizisi oluşturur.
- *Translate()* : Paragrafın ayrıştırılması ile ortaya çıkan cümleleri bir alt sınıfa aktarır ve geri dönen çıktıları noktalama işaretleriyle birleştirerek bir üst sınıfa döndürür.

Sentence.cs:

Genel olarak makine çevirisi de gerçek çeviri gibi cümle bazında yapıldığı için modüldeki en önemli sınıftır. Çeviri tam olarak bu sınıfta sonlandırılır ve üst sınıfa aktarılarak önce paragraf sonra tüm metnin oluşturulması sağlanır.

- *Words()* : Gelen cümleyi, boşluk karakteri ile kelimelere ayırıp bir string kelime dizisi oluşturur.
- *Translate()* : Diğer metotları çağırarak, hata yoksa çevrilmiş çıktıyı bir üst sınıfa aktarır.
- *SetPattern()* : Cümledeki kelime sayısına göre bir döngü kurar. Bu döngü içinde kelimelerin Türkçe karşılıklarını ve sözcük türlerini bir alt sınıf aracılığıyla alır. Sözcük türleri ile cümlenin öznesini ve yüklemini set eder. Türkçe karşılıkları bir diziye atar, sözcük türleri ile İngilizce patterni oluşturur. İlgili veri katmanı sınıfını kullanarak Türkçe patterni ve cümlenin zaman bilgisini alır.
- *ToTurkish()* : İngilizce ve Türkçe patternlere dayalı ikinci dereceden bir döngü aracılığıyla, diller arası sözcük türü eşlemesi yaparak ve cümlenin zaman bilgisini kullanarak çeviriyi gerçekleştirir.
- *ErrorMessage()* : Çeviri esnasında oluşan hatalarla ilgili mesajları birleştirir ve çeviri yapılamadığı için, bilgilendirme amaçlı olarak bir üst sınıfa iletir.

- *SetPronoun()* : Cümlede metinsel olarak belirlenen özneyi, kod olarak cümleye atar.
- *IsQuestion()* : Cümlenin soru cümlesi olup olmadığını belirler.
- *ConvertToTense()* : Metinsel cümle zamanını koda çevirir.

TurSentence.cs:

Çevirinin yapıldığı esnada oluşturulan Türkçe cümlelerle ilgili bazı verilerin tutulmasında kullanılan yardımcı bir sınıftır ve bazı propertyleri vardır.

Word.cs:

Cümle çevirisinin yapıldığı esnada, cümleyi oluşturan kelimelerle ilgili gerekli bilgi ve işlemlerle alakalı bazı görevleri vardır.

- *Translate()* : Veri katmanındaki ilgili sınıfı kullanarak, cümle içindeki İngilizce kelimelerin Türkçe karşılıklarını ve sözcük türlerini döndürür.
- *IsOneSyllabled()* : Kelimenin tek heceli olup olmadığını kontrol eder.
- *FirstLetter()* : Kelimenin ilk harfini döndürür.
- *LastLetter()* : Kelimenin son harfini döndürür.
- *FirstLetterType()* : Kelimenin ilk harfinin sesli mi sessiz mi olduğunu döndürür.
- *LastLetterType()* : Kelimenin son harfinin sesli mi sessiz mi olduğunu döndürür.
- *LastVowel()* : Kelimenin son sesli harfini döndürür.

Tenses.cs:

Çeviri esnasında belirnenen cümle zamanına göre fiile uygun eklerin getirilmesini sağlayan sınıftır.

- *SimplePresentTense()* : Cümleyle ilgili tüm işlemler bittikten sonra uygun parametreleri alıp, varsa istisnai durumları da değerlendirerek, özne, ses olayları ve yüklemin yapısına göre geniş zaman ekinin kazandırılıp, çevirinin tamamlanmasıyla görevlidir.

- *PresentContinuousTense()* : Cümleyle ilgili tüm işlemler bittikten sonra uygun parametreleri alıp, özne, ses olayları ve yüklemin yapısına göre şimdiki zaman ekinin kazandırılıp, çevirinin tamamlanmasıyla görevlidir.
- *SimpleFutureTense()* : Cümleyle ilgili tüm işlemler bittikten sonra uygun parametreleri alıp, özne, ses olayları ve yüklemin yapısına göre şimdiki zaman ekinin kazandırılıp, çevirinin tamamlanmasıyla görevlidir.

Suffixes.cs:

Eklerle ilgili metotları içeren bir sınıftır.

- *CheckEvents()* : Türkçe kelimelere ait ses olayların kontrol edilmesi ve gerekli değişikliğin yapılması ile görevlidir.
- *Personal()* : Parametre olarak gönderilen Türkçe kelimeye, yine parametre olarak gönderilen özneye göre şahıs ekini hazırlamakla görevlidir.
- *Case()* : Parametre olarak gönderilen Türkçe isim ve hal eki koduna göre, Türkçe hal ekini döndürür.

Types.cs:

İş katmanını oluşturan sınıflarda kullanılan, kod yapılı değişkenleri barındıran bir sınıftır.

- *LetterTypes* : Harflerin sesli veya sessiz olduğunu tutan kod grubudur.
- *WordTypes* : İsim, sıfat, zarf vb gibi sözcük türlerini içeren kod grubudur.
- *PronounTypes* : Özneleri (zamir) içeren kod grubudur.
- *TenseTypes* : Cümlelerin zaman bilgisi için kullanılan kod grubudur.
- *SentenceTypes* : Olumlu, olumsuz, soru vb cümle tipleri için kullanılan kod grubudur.

3.2.4 Kullanıcı Arayüzü – GUI

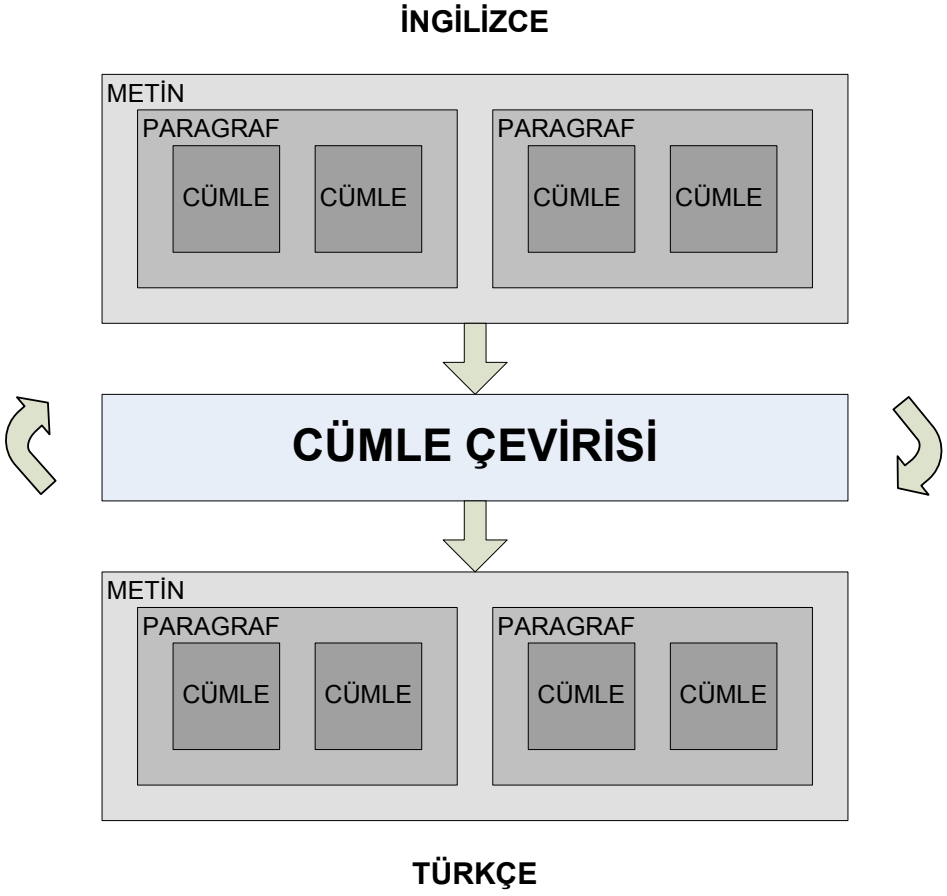
Hazırlan makine çevirisi modülünün uygulaması için sade bir web sayfası hazırlanmıştır. Bu arayüzde, kullanıcıdan İngilizce metin alınıp, çeviri modülüne gönderilmektedir. Çeviri modülünde iş katmanı, veri katmanı aracılığıyla

veritabanına bağlanarak ve gerekli algoritmaları uygulayarak çevir yapmakta ve sonrasında yine kullanıcı arayüzü ile çıktıyı yani Türkçe metni vermektedir.

Çeviri modülü daha önce de bahsedildiği üzere, kullanımı kolay şekilde hazırlanmış ve her platforma basit bir şekilde entegre edilerek çok kısa kodlama ile çalıştırılabilir kılınmıştır. Zira hazırlanan arayüzün kod sayfasında iki satırlık modül referansının dışında, bir satır nesne oluşturma ve bir satır da metnin gönderilmesi ve çevirinin alınması için olmak üzere toplamda dört satır ile çözüme ulaşılmıştır.

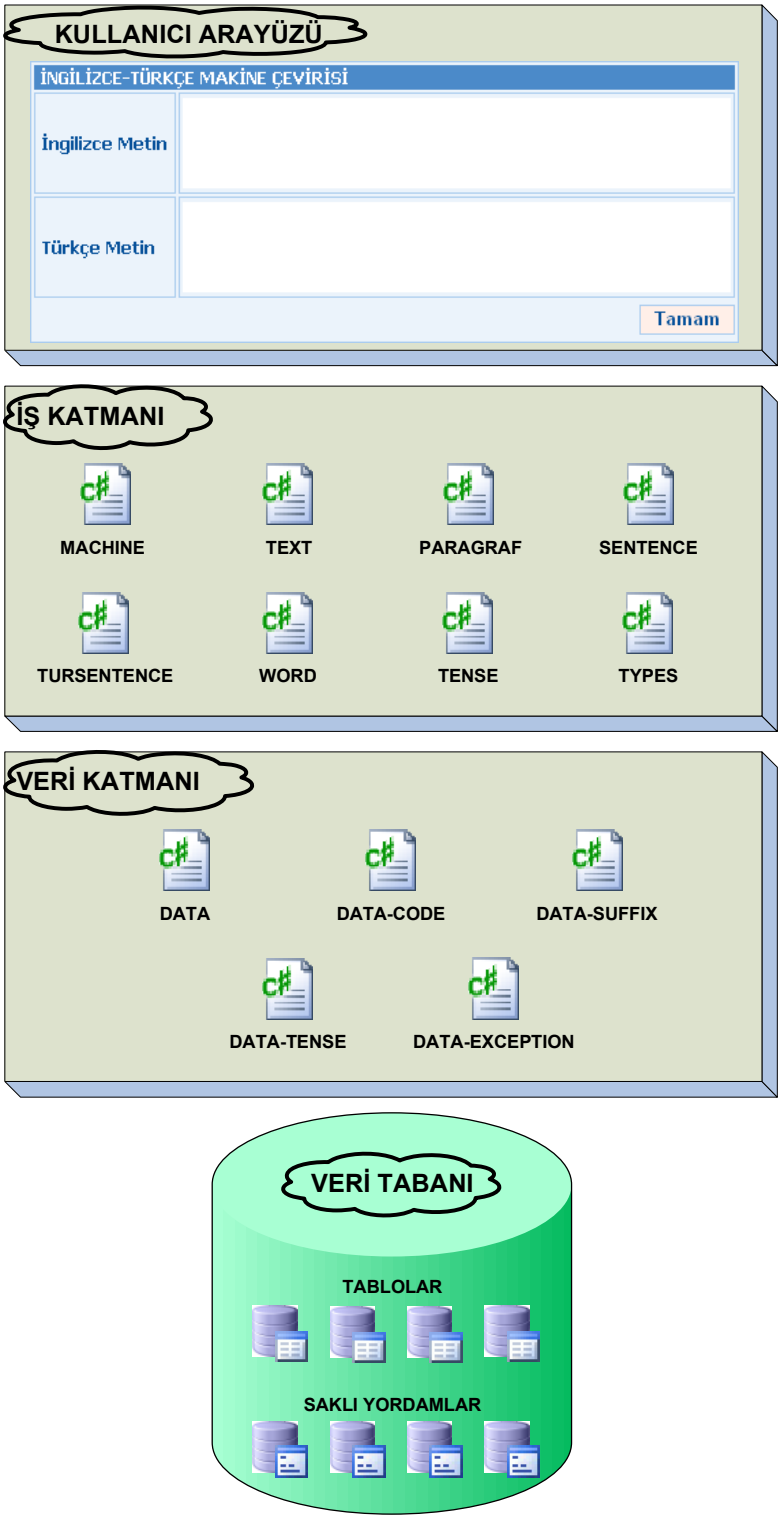
3.2.5 Blok Diyagramı

Kullanıcı arayüzünden girilen İngilizce metin paragraflara, paragraflar da cümlelere ayrıştırılır. Makine Çevirisi cümle bazında gerçekleştirilir ve çevrilmiş cümleler sentezlenerek önce paragrafları ve sonra son Türkçe metni oluşturur.



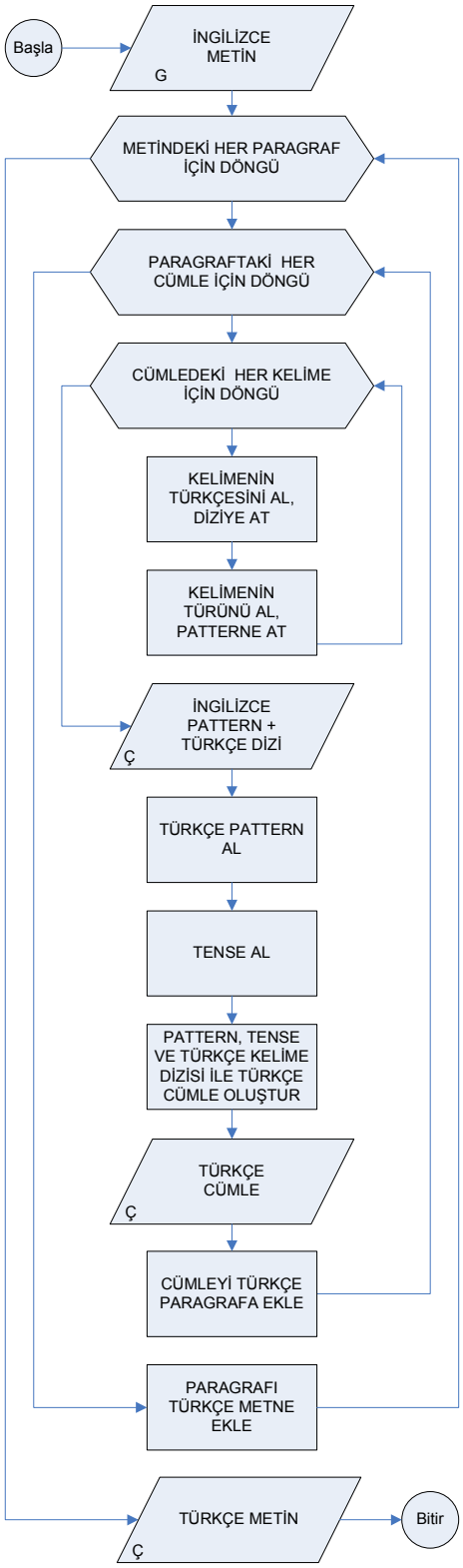
Şekil-3.4: İngilizce-Türkçe Makine Çevirisi Genel Blok Şeması

3.2.6 Teknik Diyagram



Şekil-3.5: Projenin Teknik Diyagramı

3.2.7 Akış Şeması ve Algoritma



Şekil-3.6: Projenin Akış Şeması

Algoritma:

1- İngilizce metni giriş olarak al

2- İngilizce metni paragraflara böl

3- Her paragraf için döngü kur

    3.1- Paragrafı cümlelere böl

    3.2- Her cümle için döngü kur

        3.2.1- Cümleyi kelimelere böl

        3.2.2- Her kelime için döngü kur

            3.2.2.1- Kelimenin Türkçesinin veritabanından al

            3.2.2.2- Kelimenin Türkçesini bir diziye at

            3.2.2.3- Kelimenin türünü veritabanından al

            3.2.2.4- Kelime türünü Patterne ekle

        3.2.3- Patternin Türkçe karşılığını al

        3.2.4- Patternin Tensini al

        3.2.5- Kelime dizisi, Pattern ve Tense ile Türkçe cümle oluştur

        3.2.6- Türkçe cümleyi Türkçe paragrafa ekle

    3.3- Türkçe paragrafı Türkçe metne ekle

4- Türkçe metni çıkış olarak ver

3.2.8 Proje Detayları

Zamanlar (*Tenses*):

Projede gerçekleştirilen cümle zaman yapıları şunlardır:

- Geniş Zaman (*Simple Present Tense*): Bu zamanda fiiller, hece sayısı, son harfin sessli veya sessiz oluşu ve son sesli harfin şekline göre gerekirse bir sesli kaynaştırma harfi ile –r ekini alırlar.
- Şimdiki Zaman (*Present Continuous Tense)*: Bu zamanda fiiller, son sesli harfe göre uygun sesli kaynaştırma harfini ve –yor ekini alırlar. Burada gerektiğinde ses daralması gerçekleştirilmektedir.
- Gelecek Zaman (*Simple Future Tense*): Bu zamanda fiiller, son harfin sesli veya sessiz oluşu ve son sesli harfin şekline göre gerekirse –y sessiz kaynaştırma harfi ile –acak, -ecek eklerinden birini alırlar.

Ekler (*Suffixes*):

- Şahıs Ekleri (*Personal Suffixes*): Cümledeki özneye göre fiile eklenen eklerdir:

Ben: -m	Sen: -sin	O: (Şahıs eki katmayan özne)
Biz: -z	Siz: -siniz	Onlar: -ler

- Hal Ekleri (*Case Suffixes*): İngilizce'de bazı *preposition*ların Türkçe'deki karşılıkları hal ekleridir:

Yönelme Hali (*Dative*): -e, -a

Bulunma Hali (*Locative*): -de, -da

Ayrılma Hali (*Ablative*): -de, -dan

Ses Olayları (*Sound Events*):

- Sessiz Yumuşaması: Türkçe son harfi p, ç, t, k olan kelimeler sesli harfle başlayan ek aldığında, son harf sessiz yumuşamasına uğrayarak b, c, d, g, ğ olur.

İstisnalar (*Exceptions*):

- Sessiz Yumuşaması: Türkçe'deki bazı tek heceli kelimeler (at, ot, saç, vb.), sessiz yumuşamasına uğramazlar.
- Geniş Zaman: Bu zamanda tek heceli fiiller -ar, -er eklerini alırlar. Ancak bazı istisnai fiiller (al, bil, bul, gör, vb.) -ır, -ir, -ur, -ür eklerini alır.

Sözlük (*Dictionary*):

Kaynak metindeki İngilizce kelimelerin Türkçe karşılıklarını, sözcük türlerini ve alt sözcük türlerini barındırır. Kelimeler, türlerine göre gruplanmış, grupların parçalanmasını engellemek için uygun ID aralıkları kullanılmıştır. Sözlükte bulunan türler ve kodları şöyledir:

[PRO] : Özne / Zamir	[NOU] : İsim
[QUE] : Soru kelimeleri	[AUX] : Yardımcı fiil
[VER] : Fiil	

[VE1] : Fiilin *Present* hali [VE2] : Fiilin *Past* hali

[VE3] : Fiilin *Participant* hali [VE4] : Fiilin *–ing* hali

[BE1] : *Be* fiilinin *Present* hali [BE2] : *Be* fiilinin *Past* hali

[BE3] : *Be* fiilinin *Participant* hali [CAS] : Hal eki

[FUT] : *Will*

Söz Dizilimleri (*Patternler*):

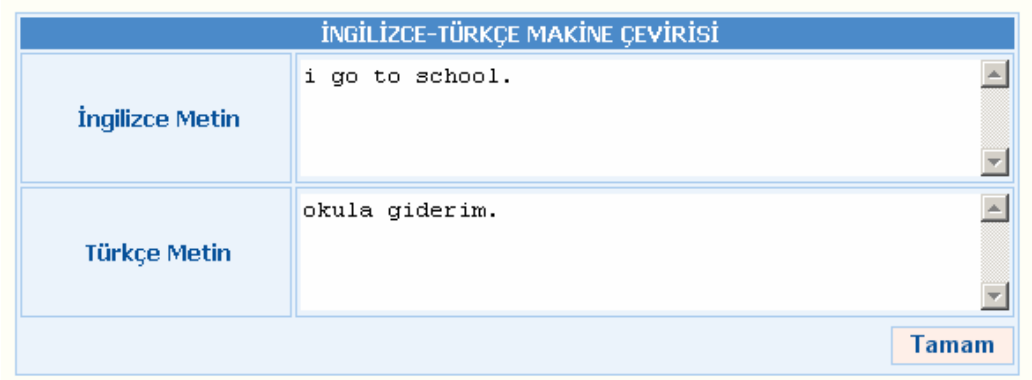
Cümlenin sahip olduğu ögeleri dizilişine *pattern* denmektedir. Projede kullanılan İngilizce patternler ve Türkçe karşılıkları aşağıdaki tabloda verilmiştir.

İngilizce	Türkçe
[PRO]-[VE1]-	[VE1]-
[PRO]-[VE1]-[NOU]-	[NOU]-[VE1]-
[PRO]-[VE1]-[CAS]-[NOU]-	[NOU]-[CAS]-[VE1]-
[PRO]-[BE1]-[VE4]-	[VE4]-
[PRO]-[BE1]-[VE4]-[NOU]-	[NOU]-[VE4]-
[PRO]-[BE1]-[VE4]-[CAS]-[NOU]-	[NOU]-[CAS]-[VE4]-
[PRO]-[FUT]-[VE1]-	[VE1]-
[PRO]-[FUT]-[VE1]-[NOU]-	[NOU]-[VE1]-
[PRO]-[FUT]-[VE1]-[CAS]-[NOU]-	[NOU]-[CAS]-[VE1]-

Tablo-3.1: Projede Kullanılan Numuneler (*Patterns*)

3.2.9 Örnek Uygulamalar

Geniş Zaman Örneği:



Şekil-3.7: Program Ekran Çıktısı – Geniş Zaman

İngilizce	Tür	Alt Tür	Türkçe
i	PRO	PRO	ben
go	VER	VE1	git
to	PRE	CAS	-e, -a
school	NOU	NOU	okul

İngilizce *Pattern* : [PRO]-[VE1]-[CAS]-[NOU]-

Özne (*Pronoun*) : Ben (*I*)

Zaman (*Tense*): Geniş Zaman (*Simple Present Tense*)

Türkçe *Pattern*: [NOU]-[CAS]-[VE1]-

Ham Çeviri: okul + e, a + git + e, a + r + ı, i, u, ü + m

LastVowel(“okul”) : “u” => +a okula

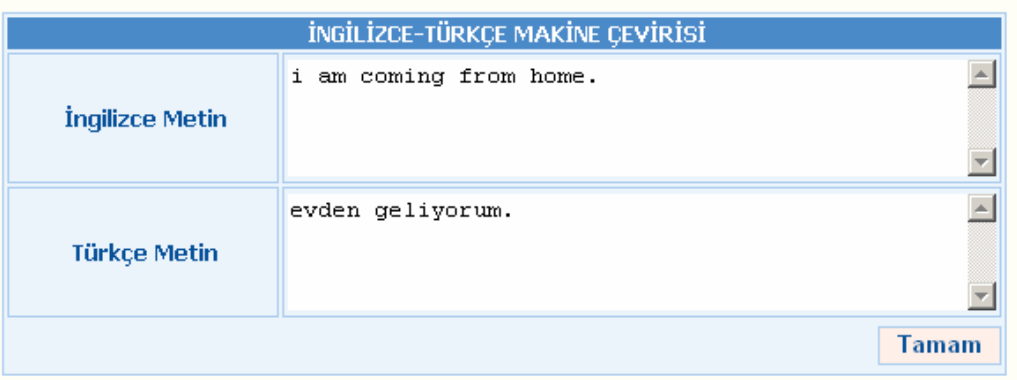
SessizYumusamasi(“git”) : gid gid

LastVowel(“git”) : “i” => +e gider

LastVowel(“gider”) : “e” => +i giderim

Son Çeviri: **okula giderim.**

Şimdiki Zaman Örneği:



Şekil-3.8: Program Ekran Çıktısı – Şimdiki Zaman

İngilizce	Tür	Alt Tür	Türkçe
i	PRO	PRO	ben
am	VER	BE1	-
coming	VER	VE4	gel
from	PRE	CAS	-den, -dan
home	NOU	NOU	ev

İngilizce *Pattern* : [PRO]-[BE1]-[VE1]-[CAS]-[NOU]-

Özne (*Pronoun*) : Ben (*I*)

Zaman (*Tense*): Şimdiki Zaman (*Present Continuous Tense*)

Türkçe *Pattern*: [NOU]-[CAS]-[VE4]-

Ham Çeviri: ev + den, dan + gel + ı, i, u, ü + yor + ı, i, u, ü + m

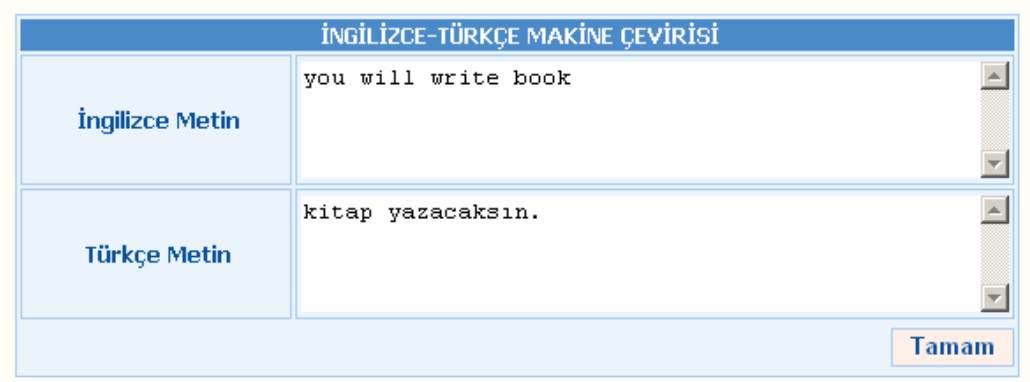
LastVowel(“ev”) : “e” => +den evden

LastVowel(“gel”) : “e” => +i geliyor

LastVowel(“geliyor”) : “o” => +u geliyorum

Son Çeviri: **evden geliyorum.**

Gelecek Zaman Örneği:



Şekil-3.9: Program Ekran Çıktısı – Gelecek Zaman

İngilizce	Tür	Alt Tür	Türkçe
you	PRO	PRO	sen
will	AUX	FUT	-
write	VER	VE1	yaz
book	NOU	NOU	kitap

İngilizce *Pattern* : [PRO]-[FUT]-[VE1]-[NOU]-

Özne (*Pronoun*) : Sen (*You*)

Zaman (*Tense*): Gelecek Zaman (*Simple Future Tense*)

Türkçe *Pattern*: [NOU]-[VE1]-

Ham Çeviri: kitap + yaz + y,'' + acak, ecek + sın, sin, sun, sün

LastLetter("yaz") : CON (Sessiz) => '' yaz

LastVowel("yaz") : "a" => +acak yazacak

LastVowel("yazacak") : "a" => +sın yazacaksın

Son Çeviri: **kitap yazacaksın.**

4 ARAŞTIRMA SONUÇLARI

Bu projeden önce yapılan literatür çalışması ve proje öncesinde ve proje süresince gerçekleştirilen kaynak araştırmasında, Makine Çevirisi, uygulama alanları, mevcut uygulamalar, kullanılan yöntemler incelenmiştir. Araştırmalar boyunca, yapılacak çalışmaların, gerçek hayatta kullanılabilecek bir program modülü ile sonuçlanması hedeflenmiş ve bunu sağlamak adına, en uygun yol ve yöntemler değerlendirilmiştir.

Araştırma konusunun diğer alanlara göre daha yeni olması ve zor bir çalışma gerektirmesi nedenleriyle, çok fazla kaynak ve yapılmış çalışma bulunamamıştır. Bununla birlikte, çalışmalar süresince konuya olan eğilimin arttığı ve böyle bir çalışma alanının gerekliliğinin birçok kişi tarafından farkedildiği gözlemlenmiştir. Bu da; bu alanda çalışmayı düşünenler için ümit vericidir. Ayrıca, konunun insanlar için faydalı olacağı inancı ile söylenebilir ki, kısa süre içinde mevcut uygulamalar daha da geliştirilecek ve yenileri hazırlanacaktır.

Araştırmalarda, konunun anlatımıyla ilgili düzenli bilgi veren bir kaynak bulunamamış ve dolayısıyla, bulunan bilgiler uygun şekilde derlenerek incelenmiştir.

Proje için en önemli konulardan birisi, kullanılacak olan Makine Çevirisi yönteminin seçimi olmuştur. Tezde bahsedilen bazı yöntemlerden, en uygun olarak Kural Tabanlı Makine Çevirisi veya diğer bir adlandırma ile Öge Dizilişi yöntemi görülmüş ve çalışma bu yönde ilerletilmiştir. Bu yöntemin seçilmesinde etkili olan kriterler ve bahsi geçen yöntemlerin detaylı karşılaştırılması Tartışma bölümünde verilmiştir.

Konunun önemi açısından sevindirici olan bir gelişme de, işin daha resmi ortamlar tarafından incelenmesi ve geliştirilmesi anlamına gelen bazı organizasyonların kurulmuş ve aktif bir şekilde çalışmakta olmalarıdır. Bununla ilgili detaylı bilgiler Ek-1'de bulunabilir.

5 TARTIŞMA

Yapılan araştırmalar sonucunda, kaynaklarda çeşitli Makine Çevirisi yöntemleri görülmüştür. Bulunan bilgiler derlenerek, temel bilgiler Kaynak Araştırması bölümünde verilmiştir. Bu bölümde ise bu yöntemlerin bazı kriterlere göre karşılaştırılması ve seçilen yöntemin tercih nedenleri verilecektir.

5.1 Makine Çevirisi Yöntemlerinin Karşılaştırılması

Derlenen yöntemlere ait açıklamalardan yola çıkılarak, aşağıdaki gibi bir sayısal değerlendirmeye varılmıştır:

	Sözlük Tabanlı	İstatiksel	Örnek Tabanlı	Çeviri Hafızası	Ara Dil	Kural Tabanlı
Çeviri Kalitesi	1	2	3	6	4	5
İnsan Yöntemine Yakınlık	3	1	5	4	2	6
Kaynak Tasarrufu	6	5	2	1	4	3
Proje Süresi Uzunluğu	1	2	3	6	5	4
Çeviri Hızı	6	5	4	1	2	3

Tablo-5.1: Makine Çevirisi Yöntemlerinin Karşılaştırılması

Tablo Değerlendirmesi:

Çalışmada bahsi geçen altı makine çevirisi yöntemi olduğu için kriterlerde 1-6 arası değerler kullanılmıştır. 1, en düşük/kötü; 6, en yüksek/iyi anlamına gelmektedir.

Çeviri Kalitesi; bu kriterde en düşük değer Sözlük Tabanlı yöntem içindir, çünkü bu yöntemde, çeviri sadece kelime dizilerinden ibaret olacaktır ve cümle çevirisi istendiğinde anlamsız bir sonuç üretilecektir. Öte yandan en iyi değer Çeviri Hafızası'ndadır. Bunun nedeni ise, yapılan çevirinin zaten daha önce insanlar tarafından yapılmış olmasıdır.

İnsan Yöntemine Yakınlık; çalışmada tercih edilmiş olan Kural Tabanlı yöntem insan çevirisine en yakın olarak görülmüştür. Zira insanlar da çeviri yaparken belli bir kelime haznesine sahiptir ve belirli dilbilgisi kurallarına göre çeviri

yaparlar. Çeviri Hafızası'ndaki gibi ezber yöntemi çok kullanılmaz, veya İstatistiksel yöntemdeki gibi matematiksel işlemler çeviride yer almaz.

Kaynak Tasarrufu; bu kriterde hem bilgisayar kaynakları hem de insan ve zaman gibi faktörler göz önünde bulundurulmuştur. Mesela Sözlük Tabanlı yöntemde sadece bir sözlük vardır ve bunun için verinin girilmesi ve basit bir uygulama ile sorgulanması gerekir. Diğer taraftan, Kural Tabanlı yöntemde bunlara ilaveten dil kurallarının öğretilmesi ve daha gelişmiş bir veritabanı tasarımı ve karmaşık bir yazılım uygulaması gerekir. Çeviri Hafızasında ise birçok insan çevirmen görev almalıdır.

Proje Süresi Uzunluğu; bu kriter, Kaynak Tasarrufuna oldukça paraleldir. Gerekli kaynak miktarı, proje süresinin de uzun olacağı anlamına gelebilir. Mesela İstatistiksel yöntemde, belirli matematiksel kurallar uygulanırken; Ara Dil yönteminde, kullanılacak olan ara dilin tanımlanması gerekecek ve bu zaman alacaktır.

Çeviri Hızı; yöntemlerin kullandığı veri veya kural tabanının büyüklüğü ile çeviri sırasında gerçekleştirilen analiz ve sentez gibi işlemlerin gerekliliği bu kriterle doğrudan ilişkilidir. Bu nedenle, çok büyük veritabanı gerektiren Çeviri Hafızası yöntemi, çeviride ek bir katman içeren Ara Dil yöntemi ve hem geniş bir kural tabanına sahip olan, hem de bu kuralların uygulanmasını sağlayan analiz ve sentez işlemleri gerektiren Kural Tabanlı yöntem, diğerlerine göre daha yavaş sonuç verecektir.

Yapılan araştırmalar ve yukarıda verilmiş olan kriter değerlendirmeleri göz önüne alınarak proje uygulamasında Kural Tabanlı Makine Çevirisi yöntemi kullanılmıştır. Bunun önemli sebeplerini şöyle özetleyebiliriz:

Neden Kural Tabanlı Makine Çevirisi?

- Gerçek çeviriye, yani insan çevirisine çok daha uygun bir yöntemdir.
- Daha yeni, yani araştırmaya açık bir alandır.
- Proje geliştiricinin, daha aktif olarak çalışmasını sağlar.
- Diğerlerine göre daha bilimsel bir yöntemdir.

6 SONUÇ VE ÖNERİLER

6.1 Çalışmada Karşılaşılan Güçlükler

Çalışma boyunca genel çeviri güçlüklerinin yanı sıra Türkçe'nin sahip olduğu komplike yapıdan dolayı farklı güçlüklerle de karşılaşılmıştır.

6.1.1 Ekler

Türkçe'de kullanılan cümlelerde olaylar genellikle eklerle ifade edilir:

Özne : "gel-di – **m**", ...

Zaman : "gel-**di**", ...

Aitlik : "araba-**m**", ...

Olumsuzluk : "gel-**me**", ...

Kipler : "gel-e-**bil-**i-r", ...

Edilgen : "kır-ı-**l**-dı", ...

Ettirgen : "yap-**tır**", ...

Şartlı İfade : "gel-i-r-**se**", ...

Akraba İfade : "gel-e-**n**", ...

Raporlanan İfade : "gel-**diğ**-i-m-i", ...

...

6.1.2 Ses olayları

Türkçe kelimeler ek aldıklarında, fiziksel olarak ses değişiklikleri içerebilir:

Ses Uyumu : "yap**a**cak", ...

Ses Benzeşmesi : "aç**t**ım", ...

Ünlü Daralması : "başl**ı**yor", ...

Ünlü Düşmesi : "boy<u>u</u>nu", ...

Sessiz Yumuşaması : "barda**ğ**ı", ...

Ses Türemesi : "af**f**etmek", ...

Ünsüz Düşmesi : "ufa<u>k</u>cık", ...

Kaynaştırma : "kapı**y**ı", ...

...

6.1.3 Çok anlamlılık

Her dil ikilisinde, kelimelerin birden fazla karşılığı bulunabilmektedir:

Türkçe - Yüz : Face, Hundred, Swim, Front, Float, ...

İngilizce - Book : Kitap, Liste, Senaryo, Rezervasyon Yapmak, ...

6.1.4 İstisnai Durumlar

Birçok dilde olduğu gibi Türkçe'de de bazı istisnai durumlar tespit edilmiştir. Proje kapsamında çözüm geliştirilen istisnai durumlar şunlardır:

Geniş Zaman:

Bazı tek heceli kelimelere eklenen geniş zaman eki, değişime uğramaktadır.

Örneğin; "kalır" kelimesi "*kalar"* şekline geçmemektedir.

Sessiz Yumuşaması:

Bazı tek heceli kelimelerde, normal şartlarda olması beklenen sessiz yumuşaması etkisiz kalmaktadır.

Örneğin; "yapar" kelimesinde sessiz yumuşaması olmamaktadır.

6.2 Projede Alınan Sonuçlar

Yapılan çalışmada, İngilizce'den Türkçe'ye makine çevirisi modülü için çok gelişmiş ve sağlam bir altyapı kurulmuş, yapılacak ek çalışmalarla, çok iyi çalışacak ve iyi sonuçlar verecek bir çeviri modülünün oluşturulmasına zemin hazırlanmıştır.

Projede kurulan yapıyla, bir dil çevirisinde karşılaşılacak birçok duruma uygun çözümler düşünülmüş, genişletilebilir bir çalışma gerçekleştirilmiştir. Bu sayede, oldukça kısa sürede, modülü zenginleştirecek eklentiler yapılabilecektir.

Ayrıca, kullanılan Kural Tabanlı çeviri yönteminin doğru seçim olduğuna kanaat getirilmiş olup, yeni araştırmacılara bu konuda kaynaklık edecek bir çalışma hazırlandığına inanılmaktadır.

7 KAYNAKLAR

7.1 Kitaplar

[1.1] - C#'ı Kavramak, Archer, T., 2004, Ankara

[1.2] - Adım Adım Microsoft ASP.NET, Duthie, G. A., 2005, Ankara

[1.3] - Adım Adım Microsoft ADO.NET, Riordan, M. R., 2003, Ankara

[1.4] - Adım Adım Microsoft SQL Server 2000 Programlama, Rebecca M. Riordan, 2004, Ankara

[1.5] - Machine Translation An Introductory Guide, Douglas, A., Balkan, L., Meijer, S., Humphreys, R.L., Sadler, L. 1994, Londra, İngiltere

7.2 Bilimsel Araştırmalar

[2.1] - Automatic Translation in Cross-Lingual Access to Legislative Databases, Bounsaythip, C., Lehtola, A., Tenni, J.

[2.2] - BuzzTrans: Evaluating a Translating Instant Messenger Client, Nair, R., Shiroor, K.

[2.3] - Machibe Translation: A Brief History, Hutchins, W.J.

[2.4] - Machine Translation and Computer-based Translation Tools: What's Available and How It's Used, Hutchins, W.J., 2003, Valladolid, İspanya

[2.5] - A Machine Translation System from Japanese into English: Another Perspective of MT Systems, Nagao, M., Tsujii, J., Mitamura, K., Hirakawa, H., Kume, M., Kyoto, Japonya

[2.6] - Machine Translation, Hinz, J.

[2.7] - Extending the BLEU MT Evaluation Method with Frequency Weightings, Babych, B. Hartley, A., Leeds, İngiltere

[2.8] - Experimental Comparison of MT Evaluation Methods: RED vs. BLEU, Akibay, Y., Sumitay, E., Nakaiway, H., Yamamotoy, S., Okuno, H.G., Kyoto, Japonya

[2.9] - IBM Research Report, Papineni, K., Roukos, S., Ward, T., Zhu, W.J., 2001, New York, ABD

[2.10] - Statistical Significance Tests for Machine Translation Evaluation, Koehn, P., Cambridge, İngiltere

[2.11] - Pattern-Based Machine Translation, Takeda, K., Tokyo, Japonya

[2.12] - A Pattern-based Machine Translation System Extended by Example-based Processing, Watanabe, H., Takeda, K., Tokyo, Japonya

[2.13] - MetaMorpho: A Pattern-Based Machine Translation System, Proszeky, S., Tihanyi, L., Budapeşte, Macaristan

7.3 Internet Kaynakları

[3.1] - http://eli-project.sourceforge.net/elionline4.4/ptg_toc.html

[3.2] - http://www.trl.ibm.com/projects/langtran/index_e.htm

[3.3] - http://www.nlpu.com/whatnlp.htm

[3.4] - http://www.systransoft.com/IDC/OvercomingMTQualityImpasse-IDC-Aug03.pdf

[3.5] - http://courses.essex.ac.uk/lg/lg619/mt/index.pdf

[3.6] - http://www.historyworld.net/wrldhis/PlainTextHistories.asp?historyid=ab13

[3.7] - http://www.nist.gov/speech/tests/mt/doc/ngram-study.pdf

[3.8] - http://www.cs.rochester.edu/~gildea/Slides/picture.pdf

[3.9] - http://www.dusunenadam.com.tr/edebiyat14.htm

[3.10] - http://www.stevenblack.com/PTN-Layers.asp

[3.11] - http://tr.wikipedia.org/wiki/Nesne_Y%C3%B6nelimli_Programlama

[3.12] - http://en.wikipedia.org/wiki/Three_layer_architecture

[3.13] - http://en.wikipedia.org/wiki/Statistical_machine_translation

[3.14] - http://en.wikipedia.org/wiki/Interlingual_machine_translation

[3.15] - http://en.wikipedia.org/wiki/Dictionary-based_machine_translation

[3.16] - http://en.wikipedia.org/wiki/Example-based_machine_translation

[3.17] - http://en.wikipedia.org/wiki/Translation_memory

[3.18] - http://www.yazgelistir.com/Makaleler/makaleler.aspx?KatId=1000000000&Kat=ASP.NET

EKLER

EK.1 – Makine Çevirisi ile İlgili Kuruluşlar

Bugüne kadar yapılan çalışmalar sonrasında, bu alandaki çalışmaları desteklemek ve yaygınlaştırmak için uluslararası arenada birçok organizasyonlar kurulmuştur. Bu bilim dalında araştırma yapacak kişilerin bu kurumları tanıması ve faydalanmaya çalışması önerilir.

Bahsi geçen kuruluşlardan en önemlileri şunlardır:

EAMT

1991'de İsviçre'de ticari amaç güdülmeksizin kurulan ve, MT ve çeviri araçları ile ilgilenen kullanıcılar, geliştiriciler ve araştırmacılardan oluşan topluluğun büyümesine hizmet eden bir organizasyondur.

Diğer kuruluşlarla birlikte MTNI gazetesini yayınlamaktadırlar. Ayrıca eğitimleri ve konferansları organize ederler. Bunun dışında ürünlerini çok düşük fiyatlara veya ücretsiz olarak sağlayan şirketlerin listesini üyeleri için hazırlarlar. Son olarak mt-list@eamt.org email listesi aracılığıyla çeviri teknolojisi için bir tartışma forumu oluşturmuşlardır.

(**http://www.eamt.org**)

AAMT

1991'de Japan Association for Machine Translation (JAMT) olarak kurulmuştur. Çalışmalarını genişletmek için Asia-Pacific Association for Machine Translation (AAMT) olarak değiştirmiştir.

2005 itibariyle kuruluşun, MT sistem üreticisi, MT-ilişkili yazılım ve araç geliştiriciler, MT sistem distribütörleri, MT araştırma enstitüleri ve diğer organizasyonlardan oluşan 30 kurumsal üyesi, öğrenci, araştırmacı, mühendis, geliştirici, distribütör, çeviri teknisyenleri ve çevirmenlerden oluşan 120'den fazla bireysel üyesi vardır.

(**http://www.aamt.info**)

AMTA

Dillerin çevirisinde, herhangi bir şekilde bilgisayarı kullanma konusunda ilgili herkese açık bir organizasyondur. Yani, çeviri ihtiyacı olan insanlar, ticari sistem geliştiricileri, araştırmacılar, sponsorlar ve MT sistemler üzerine eğitim alan, değerlendirme yapan herkesi kapsar. Ticari geliştiriciler (profesyonel çevirmenler ve MT kullanıcıları) ve bilimsel araştırmacılar için fikir ve ümitlerin paylaşılacağı bir atmosfer sağlar.

Kuruluşun, Kanada, Latin Amerika ve ABD'de üyeleri vardır. Bireysel ve kurumsal üyeleri kabul etmektedir.

(http://www.amtaweb.org)

LISA

Küresel çalışan kuruluşlar için önde gelen uluslararası forumdur. Uluslararası müşterileri, ürünleri ve hizmetleri desteklemenin doğru ve yanlış yollarını ayrıştırmıştır.

(**http://www.lisa.org**)

EK.2 – Projenin Kaynak Kodları

```
public partial class _Default : System.Web.UI.Page
{
   protected void Page_Load(object sender, EventArgs e)
   {
   }

   protected void btnTranslate_Click(object sender, EventArgs e)
   {
      Machine machine = new Machine();
      txtTarget.Text = machine.Translate(txtSource.Text);
   }
}


public class Machine
   {
      public Machine()
      {
      }
      public string Translate(string content)
      {
         Text text = new Text(content);
         return text.Translate();
      }
   }


public class Text
   {
      private string _content;

      public Text()
      {
      }

      public Text(string content)
      {
         this.Content = content;
      }

      public string Content
      {
         get
         {
            return _content;
         }
         set
         {
            _content = value;
         }
      }


      public string[] Paragraphs()
```

```
        {
            string[] content = this.Content.Split('\n');
            for (int i = 0; i < content.Length; i++)
                content[i] = content[i].Replace("\n", "");
            return content;
        }

        public string Translate()
        {
            string sb = "";
            Paragraph paragraph = null;
            for (int i = 0; i < this.Paragraphs().Length; i++)
            {
                paragraph = new Paragraph(this.Paragraphs()[i]);
                sb+=paragraph.Translate();
                sb+="\n";
            }
            return sb.Replace("..",".");
        }
    }


public class Paragraph
    {
        private string _content;

        public Paragraph()
        {
        }

        public Paragraph(string content)
        {
            this.Content = content;
        }

        public string Content
        {
            get
            {
                return _content;
            }
            set
            {
                _content = value;
            }
        }

        public string[] Sentences()
        {
            return this.Content.Split(new Char[] {'.','?','!',':'});
        }

        public string Translate()
        {
            string sb="";
            Sentence sentence = null;
            string[] ss = this.Sentences();
            for (int i = 0; i < ss.Length; i++)
```

```
            {
                try
                {
                    if (ss[i].Substring(0,1) == " ")
                        ss[i]=ss[i].Substring(1, ss[i].Length - 1);
                }
                catch
                { }
                sentence=new Sentence(ss[i]);
                sb += sentence.Translate();
                sb = sb.Substring(0, sb.Length - 1);
                sb += ". ";
            }
            return sb;
        }
    }


public class Sentence
    {
        private string _content;
        private string _verb;
        private string _pattern;
        private PronounTypes _pronoun;
        private TenseTypes _tense;
        private VerbTypes _verbType;
        private SentenceTypes _type;
        private TurSentence _turkish = new TurSentence();
        private byte _numVerbs=0;
        private ArrayList _errors;

        public Sentence()
        {
        }

        public Sentence(string content)
        {
            this.Content = content;
            this.Errors = new ArrayList();
        }

        public string Content
        {
            get
            {
                return _content;
            }
            set
            {
                _content = value;
            }
        }


        public ArrayList Errors
        {
            get
            {
```

```
            return _errors;
        }
        set
        {
            _errors = value;
        }
    }

    public string Pattern
    {
        get
        {
            return _pattern;
        }
        set
        {
            _pattern = value;
        }
    }

    public PronounTypes Pronoun
    {
        get
        {
            return _pronoun;
        }
        set
        {
            _pronoun = value;
        }
    }

    public string Verb
    {
        get
        {
            return _verb;
        }
        set
        {
            _verb = value;
        }
    }
    public TenseTypes Tense
    {
        get
        {
            return _tense;
        }
        set
        {
            _tense = value;
        }
    }
    public VerbTypes VerbType
    {
        get
        {
```

```
            return _verbType;
        }
        set
        {
            _verbType = value;
        }
    }

    public SentenceTypes Type
    {
        get
        {
            return _type;
        }
        set
        {
            _type = value;
        }
    }

    public byte NumVerbs
    {
        get
        {
            return _numVerbs;
        }
        set
        {
            _numVerbs = value;
        }
    }

    public TurSentence Turkish
    {
        get
        {
            return _turkish;
        }
        set
        {
            _turkish = value;
        }
    }

    public string[] Words()
    {
        return this.Content.Split(' ');
    }

    public string Translate()
    {
        this.SetPattern();
        if (this.Errors.Count == 0)
            return this.ToTurkish();
        return this.ErrorMessage();
    }

    public void SetPattern()
```

```
        {
            try
            {
                Word word = null;
                Data data = new Data();
                ArrayList turWords = new ArrayList();
                StringBuilder sb = new StringBuilder();
                string[] engWords = this.Words();
                for (int i = 0; i < engWords.Length; i++)
                {
                    word = new Word(engWords[i]);
                    DataTable dt = word.Translate();
                    switch (dt.Rows[0]["SubType"].ToString())
                    {
                        case "PRO": this.SetPronoun(engWords[i]); break;
                        case "VER": this.Verb = engWords[i]; this.NumVerbs =
Convert.ToByte(this.NumVerbs + 1); break;
                    }
                    turWords.Add(dt.Rows[0]["TurWord"].ToString());
                    sb.Append("[" + dt.Rows[0]["SubType"].ToString() + "]-");
                }
                this.Pattern = sb.ToString();
                DataRow dr = data.GetPattern(this.Pattern);
                this.Turkish.Pattern = dr["TurPattern"].ToString();
                this.Tense = this.ConvertToTense(dr["Tense"].ToString());
                this.Turkish.Words = turWords;
            }
            catch
            {
                this.Errors.Add("=Pattern= oluşturulamadı!");
            }
        }

        public string ToTurkish()
        {
            Tenses tenses = new Tenses();
            Suffixes suffixes = new Suffixes();
            string[] turPattern = this.Turkish.Pattern.Split('-');
            string[] engPattern = this.Pattern.Split('-');
            string result = "";
            for (int i = 0; i < turPattern.Length - 1; i++)
            {
                for (int j = 0; j < engPattern.Length - 1; j++)
                {
                    if (turPattern[i].Equals(engPattern[j]))
                    {
                        if (turPattern[i] == "[VER]" ||
                                                turPattern[i] == "[VE1]" ||
                            turPattern[i] == "[VE2]" ||
                                                turPattern[i] == "[VE3]" ||
                            turPattern[i] == "[VE4]" ||
                                                turPattern[i] == "[FUT]")
                        {
                            switch (this.Tense)
                            {
                                case TenseTypes.PresentContinuousTense:
```

```
                        result +=
suffixes.Personal(tenses.PresentContinuousTense(this.Turkish.Words[j].ToString()),
this.Pronoun.ToString()) + " ";
                        break;
                    case TenseTypes.SimplePresentTense:
                        result +=
suffixes.Personal(tenses.SimplePresentTense(this.Turkish.Words[j].ToString()),
this.Pronoun.ToString()) + " ";
                        break;
                    case TenseTypes.SimpleFutureTense:
                        result +=
suffixes.Personal(tenses.SimpleFutureTense(this.Turkish.Words[j].ToString()),
this.Pronoun.ToString()) + " ";
                        break;
                }
            }
            else
            {
                if (turPattern[i] == "[NOU]" && turPattern[i + 1] == "[CAS]")
                {
                    result += suffixes.Case(this.Turkish.Words[j].ToString(), this.Words()[j - 1]) + "
";
                    i++;
                }
                else
                {
                    result += this.Turkish.Words[j] + " ";
                }
            }
            break;
        }
      }
    }
    return result;
  }

  public string ErrorMessage()
  {
    string msg = "";
    for (int i = 0; i < this.Errors.Count; i++)
    {
      msg += this.Errors[i].ToString() + "\n";
    }
    return msg;
  }

  public void SetPronoun(string engPronoun)
  {
    switch (engPronoun.ToLower())
    {
      case "i": this.Pronoun = PronounTypes.I; break;
      case "you": this.Pronoun = PronounTypes.You; break;
      case "he": this.Pronoun = PronounTypes.He; break;
      case "she": this.Pronoun = PronounTypes.She; break;
      case "it": this.Pronoun = PronounTypes.It; break;
      case "we": this.Pronoun = PronounTypes.We; break;
      case "they": this.Pronoun = PronounTypes.They; break;
    }
```

```
        }


        public bool IsQuestion()
        {
            return false;
        }

        public TenseTypes ConvertToTense(string tenseType)
        {
            switch (tenseType)
            {
                case "SimplePresentTense": return TenseTypes.SimplePresentTense;
                case "PresentContinuousTense": return TenseTypes.PresentContinuousTense;
                case "SimpleFutureTense": return TenseTypes.SimpleFutureTense;
            }
            return TenseTypes.SimplePresentTense;
        }
    }


public class TurSentence
    {
        private string _content;
        private string _verb;
        private string _pattern;
        private ArrayList _words;

        public TurSentence()
        {
        }

        public TurSentence(string content)
        {
            this.Content = content;
        }

        public string Content
        {
            get
            {
                return _content;
            }
            set
            {
                _content = value;
            }
        }

        public string Pattern
        {
            get
            {
                return _pattern;
            }
            set
            {
                _pattern = value;
```

```
            }
        }

        public string Verb
        {
            get
            {
                return _verb;
            }
            set
            {
                _verb = value;
            }
        }

        public ArrayList Words
        {
            get
            {
                return _words;
            }
            set
            {
                _words = value;
            }
        }

        public string Translate()
        {
            return "";
        }
    }


public class Word
    {
        private string _content;
        private WordTypes _type;

        public Word()
        {
        }

        public Word(string content)
        {
            this.Content = content;
        }

        public string Content
        {
            get
            {
                return _content;
            }
            set
            {
                _content = value;
            }
```

```
}

public WordTypes Type
{
   get
   {
      return _type;
   }
   set
   {
      _type = value;
   }
}

public DataTable Translate()
{
   Data data = new Data();
   return data.ToTurkish(this.Content);
}

public bool IsOneSyllabled()
{
   int cnt = 0;
   for (int i = 0; i < this.Content.Length; i++)
   {
      if (this.Content.Substring(i, 1) == "a" ||
         this.Content.Substring(i, 1) == "e" ||
         this.Content.Substring(i, 1) == "ı" ||
         this.Content.Substring(i, 1) == "i" ||
         this.Content.Substring(i, 1) == "o" ||
         this.Content.Substring(i, 1) == "ö" ||
         this.Content.Substring(i, 1) == "u" ||
         this.Content.Substring(i, 1) == "ü")
      {
         cnt++;
         if (cnt > 1) return false;
      }
   }
   if (cnt == 1) return true;
   if (cnt == 0) throw new Exception("Invalid Word!!!");
   return false;
}

public string FirstLetter()
{
   return this.Content.Substring(0, 1);
}

public string LastLetter()
{
   return this.Content.Substring(this.Content.Length - 1, 1);
}

public LetterTypes FirstLetterType()
{
   if (this.Content.Substring(0, 1) == "a" ||
      this.Content.Substring(0, 1) == "e" ||
      this.Content.Substring(0, 1) == "ı" ||
```

```
            this.Content.Substring(0, 1) == "i" ||
            this.Content.Substring(0, 1) == "o" ||
            this.Content.Substring(0, 1) == "ö" ||
            this.Content.Substring(0, 1) == "u" ||
            this.Content.Substring(0, 1) == "ü")
            return (LetterTypes.Vowel);
        else
            return (LetterTypes.Consonant);
    }

    public LetterTypes LastLetterType()
    {
        if (this.Content.Substring(this.Content.Length - 1, 1) == "a" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "e" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "ı" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "i" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "o" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "ö" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "u" ||
            this.Content.Substring(this.Content.Length - 1, 1) == "ü")
            return (LetterTypes.Vowel);
        else
            return (LetterTypes.Consonant);
    }

    public string LastVowel()
    {
        for (int i = this.Content.Length - 1; i >= 0; i--)
        {
            if (this.Content.Substring(i, 1) == "a" ||
                this.Content.Substring(i, 1) == "e" ||
                this.Content.Substring(i, 1) == "ı" ||
                this.Content.Substring(i, 1) == "i" ||
                this.Content.Substring(i, 1) == "o" ||
                this.Content.Substring(i, 1) == "ö" ||
                this.Content.Substring(i, 1) == "u" ||
                this.Content.Substring(i, 1) == "ü")
            {
                return (this.Content.Substring(i, 1));
            }
        }
        return "1";
    }
}


public class Tenses
{
    public Tenses()
    {
    }

    public string SimplePresentTense(string turPhrase)
    {
        Word word = new Word(turPhrase);
        DataTense dataTense = new DataTense();
        DataException dataException = new DataException();
        bool isExceptional = dataException.SimplePresentTense(turPhrase);
```

```
            bool isOneSyllabled = word.IsOneSyllabled();
            string lastVowel = word.LastVowel();
            string lastLetter = word.LastLetterType().ToString().Substring(0, 3).ToUpper();
            string suffix = dataTense.SimplePresentTense(isExceptional, isOneSyllabled, lastVowel,
lastLetter);
            Suffixes suffixes = new Suffixes();
            return suffixes.CheckEvents(turPhrase, suffix);
        }

        public string PresentContinuousTense(string turPhrase)
        {
            Word word = new Word(turPhrase);
            DataTense dataTense = new DataTense();
            Suffixes suffixes = new Suffixes();
            string suffix = dataTense.PresentContinuousTense(word.LastVowel());
            if (word.LastLetterType() == LetterTypes.Consonant)
            {
                return suffixes.CheckEvents(turPhrase, suffix);
            }
            else
            {
                return suffixes.CheckEvents(turPhrase.Substring(0, turPhrase.Length - 1), suffix);
            }
        }

        public string SimpleFutureTense(string turPhrase)
        {
            Word word = new Word(turPhrase);
            DataTense dataTense = new DataTense();
            Suffixes suffixes = new Suffixes();
            string suffix = dataTense.SimpleFutureTense(word.LastLetterType().ToString().Substring(0,
3).ToUpper(), word.LastVowel());
            return suffixes.CheckEvents(turPhrase, suffix);
        }
    }


public class Suffixes
    {
        public Suffixes()
        {
        }

        public string CheckEvents(string turWord, string suffix)
        {
            Word word = new Word(turWord); //Sessiz Yumuşaması
            Word wordSuffix = new Word(suffix);
            if (word.IsOneSyllabled() == false)
            {
                if (wordSuffix.FirstLetterType() == LetterTypes.Vowel)
                {
                    switch (word.LastLetter())
                    {
                        case "p": return word.Content.Substring(0, word.Content.Length - 1) + "b" +
wordSuffix.Content;
                        case "ç": return word.Content.Substring(0, word.Content.Length - 1) + "c" +
wordSuffix.Content;
```

```
                    case "t": return word.Content.Substring(0, word.Content.Length - 1) + "d" +
wordSuffix.Content;
                    case "k": return word.Content.Substring(0, word.Content.Length - 1) + "g" +
wordSuffix.Content;
                }
            }
        }
        else
        {
            DataException dataException = new DataException();
            bool isExceptional = dataException.SessizYumusamasi(turWord);
            if (isExceptional == false)
            {
                if (wordSuffix.FirstLetterType() == LetterTypes.Vowel)
                {
                    switch (word.LastLetter())
                    {
                        case "p": return word.Content.Substring(0, word.Content.Length - 1) + "b" +
wordSuffix.Content;
                        case "ç": return word.Content.Substring(0, word.Content.Length - 1) + "c" +
wordSuffix.Content;
                        case "t": return word.Content.Substring(0, word.Content.Length - 1) + "d" +
wordSuffix.Content;
                        case "k": return word.Content.Substring(0, word.Content.Length - 1) + "g" +
wordSuffix.Content;
                    }
                }
            }
        }
        return turWord + suffix;
    }

    public string Personal(string turWord, string pronoun)
    {
        Word word = new Word(turWord);
        DataSuffix dataSuffix = new DataSuffix();
        Data data = new Data();
        string lastVowel = word.LastVowel();
        string lastLetter = word.LastLetterType().ToString().Substring(0,3).ToUpper();
        string suffix = dataSuffix.Personal(pronoun, lastLetter, lastVowel);
        return this.CheckEvents(turWord, suffix);
    }

    public string Case(string turNoun, string engNounCase)
    {
        Word word = new Word(turNoun);
        DataSuffix dataSuffix = new DataSuffix();
        Data data = new Data();
        string nounCaseCode = data.GetNounCaseCode(engNounCase);
        string lastVowel = word.LastVowel();
        string lastLetter = word.LastLetterType().ToString().Substring(0, 3).ToUpper();
        return turNoun + dataSuffix.Case(nounCaseCode, lastLetter, lastVowel);
    }
}


public enum LetterTypes
    {
```

```
    Consonant,
    Vowel
}

public enum WordTypes
{
    Adjective,
    Adverb,
    AuxilaryVerb,
    Conjuction,
    Interjection,
    Noun,
    Preposition,
    Pronoun,
    Question,
    Verb
}

public enum PronounTypes
{
    I,
    We,
    He,
    She,
    It,
    You,
    They
}

public enum TenseTypes
{
    PresentContinuousTense,
    SimplePresentTense,
    SimpleFutureTense
}

public enum VerbTypes
{
    Infinitive,
    Continuous,
    Participant,
    Past,
    Present
}

public enum SentenceTypes
{
    Affirmative,
    Imperative,
    Negative,
    Question
}
```