Türk Dilleri İçin Açık Kaynaklı Doğal Dil İşleme Kütüphanesi: ZEMBEREK

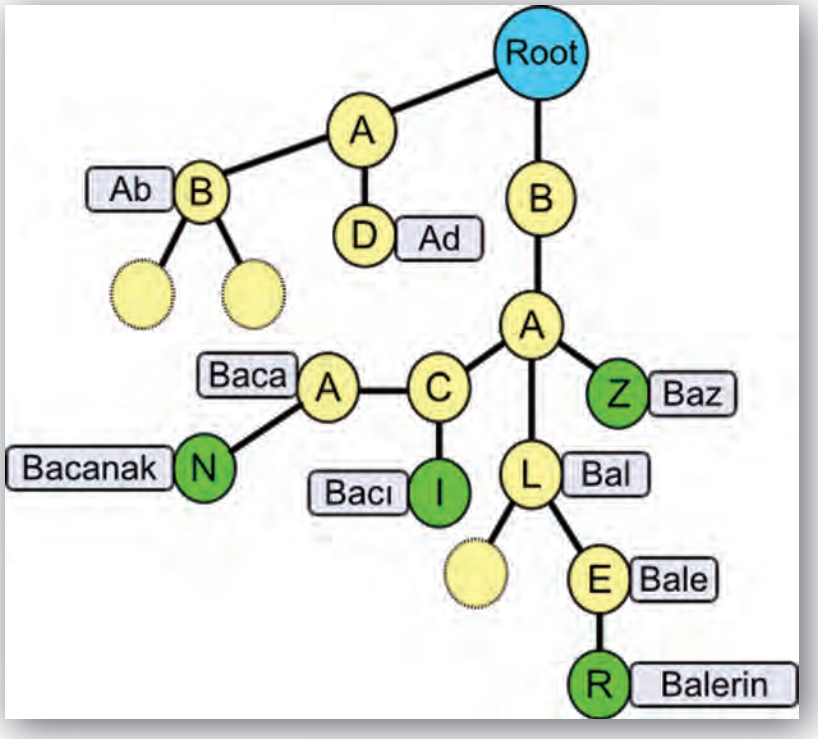
Mehmet Dündar Akın
TÜBİTAK-UEKAE
mdakin@gmail.com

Ahmet Afşin Akın
Softek Inc./Porto Riko
ahmetaa@gmail.com

Doğal Dil İşleme (DDİ), son yıllarda yazılım dünyasının en zorlu ve popüler konularından birisi haline gelmiştir. Ancak bu konuda sayısız araştırma yapılmış olsa da, uygulamaya dönüştürülebilecek kalitedeki çalışmalar henüz son derece kısıtlıdır. Yapısı itibariyle dünyada üzerinde çok çalışma yapılan Hint-Avrupa dillerinden ciddi farklılıkları bulunan eklemeli diller için ise DDİ çalışmaları çok daha az bir seviyede gerçekleştirilmektedir. Türk dilleri için ise durum daha kötüdür. Türk dilleri, Avrupa'dan Sibirya'ya [1] yaklaşık 140 milyon insan tarafından kullanılıyor olsa da, gerek çalışan kişi sayısının azlığı, gerekse açık kaynaklı kütüphanelerinin olmaması nedeniyle, yazım denetimi gibi en temel işlemler bile uygulamalarda yer almamıştır. Konu üzerine yapılan akademik çalışmalarda çoğunlukla sadece belirli bir Türk Dili [2], özellikle Türkçe [3] [4] [5] kullanılmıştır. Daha da önemlisi, Zemberek'in ortaya çıkışına dek, bu alanda hemen hemen hiç kullanılabilir açık kaynak kütüphane yoktu. Zemberek, sadece Türkçe için değil diğer ihmal edilmiş Türk dilleri için de temel bir DDİ sistemi sunmayı amaçlıyor. Mevcut sistem, yazım denetimi, biçimsel analiz, ek ayırma, sözcük türetme, sözcük önerme, Türkçeye özel karakterlerin ASCII benzerlerinin kullanılması ile yazılı sözcüklerin çevrilmesi, heceleme gibi temel NLP işlemlerini sağlamaktadır.

Yapı

Zemberek kütüphanesinin gerçeklemesi, çekirdek ve çekirdeği kullanan özel dil gerçeklemelerini içerir. Merkez kütüphane özel DDİ algoritmalarını ve diğer diller için gerekli araç ve gerekli soyutlamalar sağlar. Çekirdek özellikle Türk dilleri için tasarlanmış olsa da herhangi bir Türk diline özel kod içermez. Bu esnekliği sağlamak için birkaç yardımcı mekanizma ve soyutlamalardan yararlanılmaktadır. Zemberek'e eklenecek her dil, önceden belirlenmiş dil bilgisi koşullarına uyma ve çeşitli dil verisini sağlamadan sorumludur. Çoğu durumda



Şekil 1

eklenecek yeni dil için çekirdek mekanizma üzerinde değişiklik yapılmasına ihtiyaç duyulmaz. Bir dile ilişkin bilgiler yazıldıktan, eklendikten sonra, temel DDİ fonksiyonları kullanımı kolay bir yazılım arayüzü ile kullanıcıya sunulur.

Sisteme yeni bir Türk dilinin eklenmesi kolaydır. Dile özel veriler çoğu yerde kod içerisinde değil dış veri dosyaları üzerinden sağlanır. Bu şekilde kodlama bilgisi olmayan kişilerin de gerçeklemeye katkı yapabilmeleri sağlanır. Fakat bu durum genellikle esnekliği ve performansı olumsuz etkilediğinden özel durumlar ve ek üretme mekanizması gibi bilgiler kod yazılarak gerçekleştirilmiştir. Şekil-1'de dil bilgisi unsurları ve bulunması gereken dile özel veriler gösterilmiştir.

Yeni bir Türk dilinin Zemeberek'e dahil edilmesi için atılması gereken adımlar şunlardır:

- *İçinde alfabe ve ilgili harf verisini içeren bir harf dosyası*
- *Ek verisi içeren ek dosyası*
- *Kök halindeki sözcükleri ve özel durum bilgisini içeren bir metin dosyası*
- *Çoğunlukla yazım denetiminde kullanılan kelimeleri içeren seçimlik cep dosyası.*
- *Seçimlik hece bulucu.*
- *Ek üretim unsurlarını yorumlayacak bir ek üretim sınıfı.*
- *Ek üretimi için herhangi bir özel durum varsa bu durumları içermek amacıyla oluşturulan bir sınıf ve muhtemelen her özel durum için bir sınıf.*
- *Kök sözcüklerde ortaya çıkan özel durumlar için bir sınıf.*
- *Kelime çözümleme öncesi ve sonrasında yapılacak işlemler için bir yardımcı.*
- *Özel dil sınıfları ve kök sözcük dosyasındaki sözcük tür adları, kök sözcüklere getirilen son eklerin adları, kök sözcük dosyalarının adları gibi bilgileri içeren bir sınıf.*

Bir Türk dilinin en temel şekilde sisteme eklenmesi nispeten kolay bir iştir. Ancak, bütün dil bilgisini ve özel durumları eklemek kaçınılmaz olarak zaman alır. Şimdi bu unsurları detaylı bir biçimde anlatacağız.

Harfler ve Alfabe

Her dil alfabe bilgisini belirlemelidir. Bunun için, alfabe bilgisini taşımak için kullanılan basit bir metin temelli dosya kullanılır. Bu dosya harfler, sesliler, sesli çeşitleri (ince, yuvarlak gibi), sert sessizler ve ASCII olmayan harfler gibi bilgileri içerir. Alfabe sistemde harflerin bilgisine ulaşım noktası olarak kullanılır.

Ek Bilgisi

Bütün Türk dillerinin temeli eklerdir. Zemberek içerisinde ekler özel bir XML konfigürasyon dosyasında tanımlanmıştır. Bu dosya iki ana bilgi bölümü içerir, ek kümeleri (ek-kümesi) ve tek ek bilgisi (ekler, ek). Ek kümelerinin tek amacı asıl ek bilgisinin yazılmasını kolaylaştırmaktır. Aşağıda Türkiye Türkçe'si için hazırlanan ek dosyasından bir örnek bulunmaktadır:

```
<ek ad="ISIM_COGUL_LER"
uretim="lAr">
<ardisil-ekler>
<kume>ISIM_HAL</kume>
<kume>IMEK_ZAMAN</kume>
<aek>ISIM_SAHIPLIK_BEN_IM</
aek>
<aek>ISIM_SAHIPLIK_SEN_IN</
aek>
<aek>ISIM_SAHIPLIK_O_I</
aek>
...
</ardisil-ekler>
</ek>
```

Her ek dile göre farklı olabilen tekil bir ada sahiptir. Ek adları için belirli bir standart bulunmadığından nispeten anlaşılır ek adları sistemde kullanıldı, ileride daha yaygın kullanılan bir adlandırma sistemine geçilerek bu konuda gelişme sağlanabilir. Çoğu ek bir üretim sözcüğü içerir. Bu sözcük bir ekin üretimi sırasında kullanılan bileşenleri ifade eder. Bu bilgi ileride kelime çözümleme sırasında eklerin üretiminde kullanılacaktır. Şu an için, tanımlanmış üç tip ek üretim elemanı vardır:

1. Harfler (küçük harflerle temsil edilen harfler): Belirli bir ek oluşturulduğunda doğrudan eke eklenirler.

2. Sesli kuralı elemanları: Bunlar farklı sesli üretim kurallarını ifade ederler. Örneğin Türkçe açısından, A, harfin ekleneceği kelimenin son seslisi kalın (a, ı, o,u) ise , sözcüğe bir 'a' eklenir anlamındadır.

3. İlk harf ekleme veya değiştirme: Belirli durumlarda bir harfi değiştirirler ya da yeni bir harf eklerler, örneğin, '+n', ekli sözcük bir sesli ile biterse 'n'nin eklendiğini ifade eder.

Aşağıda bir örnek gösterilmiştir.

ek üretim sözcüğü:'lAr'

ek üretim elemanları: [harf:l, sesli kuralı:A, harf:r]

ekin getirileceği sözcük: elma

ek üreticisi sonucu:'lar'

Eklerin ortak bir karakteristiği bazı özel koşulların dışında, onların ardından sadece belirli eklerin gelebildiğidir. Zemberek'in ana kelime çözümleme mekanizması bu basit ilkeye dayanır. Dolayısıyla çoğu ek tanımı ardışıl ek bilgisi içerir ("ardışıl ekler" elemanı). Ardışıl ekler bilgisi tek ek adları, ek kümeleri veya diğer bir ekin tam kopyası olabilir.

Ayrıca, ek ağacının başlangıç noktasını belirlemek için kullanılan "başlangıç ekleri" vardır. Bu, bir kök sözcüğe (hiçbir eki olmayan bir ad gibi) yeni bir ek eklemek istediğimiz de gelebilecek ilk eklerin hangilerinin olduğunun belirlenmesi için gereklidir. Sözcüğün türü bize nereden başlamamız gerektiğine ilişkin bir ipucu verir. Bu nedenle, ilk önce, söz konusu kök sözcüğe eklenebilecek ardışıl ek bilgisi listesini taşıyan bir boş başlangıç eki ekleriz. Aşağıda sayı ekleri için tanımlanmış bir başlangıç ek örneği vardır.

```
<ek ad="SAYI_KOK" uretim="">
<ardisil-ekler kopya-ek = "ISIM_
KOK">
<aek>SAYI_ULESTIRME_ER</
aek>
<aek>SAYI_KESIR_DE</aek>
<aek>SAYI_SIRA_INCI</aek>
```

```
<aek>SAYI_TOPLULUK_IZ</aek>
<aek>SAYI_KOSE_GEN</aek>
</ardisil-ekler>
</ek>
```

Ek Özel Durumları

Ekler için özel durumlar bütün Türk dillerinde vardır. Örneğin, Türkiye Türkçesinde şimdiki zaman ekinin son harfi düşer veya edilgenlik eki farklı bir biçimde son sessize bağlıdır.

***ara→ar-ıyor**, 'ara-yor' veya 'ara-ıyor' değil*

***kes→kes-il-mek , gel→gel-in-mek**. gel-il-mek değil*

Yaygın bir özel durum, ek biçiminin bir önceki eke bağlı olarak değişmesidir. Bunlar ek dosyasında ön ek "ON_EK" özel durumu olarak tanımlanırlar.

```
<ek ad="ISIM_YONELME_E" uretim="+yA">
<ozel-durum ad="ON_EK" uretim="nA">
<on-ek ad="ZAMAN_BELIRTME_KI"/>
<on-ek ad="ISIM_BULUNMA_KI"/>
<on-ek ad="ISIM_SAHIPLIK_O_I"/>
<on-ek ad="ISIM_TAMLAMA_I"/>
</ozel-durum>
....
```

Burada, normal ek üretimi "+yA" olarak tanımlanmıştır, bu Türkçe için {a, e, ya, ye}eklerini üretir (Örneğin, "kedi-ye bak"). Ancak, bu ek 3. tekil şahıs iyelik ekinden (bu ek ilgili "ON-EK" elemanında tanımlanmıştır) sonra gelirse, "nA" üretim kelimesini kullanır ("Ahmet'in kedi-si-ne bak" örneğinde olduğu gibi).

Zemberek'teki ek sistemi oldukça ilkel bir yapıya sahiptir. Daha gelişmiş harf ile sürülen bir sonlu durum makinesi kullanılarak yapıdaki bazı özel durumlar yok edilebilir [2], fakat bunlar sık karşılaşılan durumlar olmadığı için Zemberek, aşağıdaki basit ağaç temelli yaklaşımı tercih etmiştir.

Kütüphane içerisinde ek dosyası yüklendiğinde, veri ek nesnelerine aktarılır. Bütün ek verileri bu dosyada temsil edilmemiştir, bir ekin sesliyle başlayıp başlayamayacağı gibi bazı veriler okunduktan sonra hesaba katılmıştır. Ek nesneleri ayrıca olası ardışıl eklerin listesini ve bu eke ilişkin özel durumları içerir. Çekirdek kütüphanede bütün diller tarafından kullanılabilecek bazı yaygın ortak özel durumlar vardır, fakat her dil, varsa kendisinin özel durumlarını sunmalıdır.

Kök Sözcük Sözlüğü

Bir kök sözcük bir ek almaksızın anlamlı olan bir sözcüğü ifade eder. Kök sözcükler metin-temelli bir dosyada tanımlanır. Örneğin, Azerice için kök tanım dosyası şu şekilde olabilir:

```
sağlıq AD YUM
al EY
gel EY
istiot AD
bir RA
dünen ZAMAN
```

Biçimi çok basittir, kök sözcük, sözcüğün türü ve varsa özel durumlar. Dosyalar, elle kolayca değiştirilebilmesi için bu şekilde bir yapıda hazırlanmıştır. Sözcük tür adları dile bağlıdır. Her dil, kendi sözcük türü ve özel durum adlarını tanımlayabilir. Performans kaygıları nedeniyle Zemberek programda metin dosyasını doğrudan değil, işleyip ikili dosyaya dönüşmüş halini kullanır.

Kök Sözcüğe Özel Durumlar

Türk dillerinde, kök sözcükler için özel durumlar vardır. Bunların çoğu, başka dillerden geçen sözcükler nedeniyle ortaya çıkmıştır. Ancak bazı özel durumlar dilin aslında olan özelliklerdir. Zemberek içerisinde algoritma zorlukları nedeniyle "özel durum" olarak adlandırılmıştır. Bunun Türkiye Türkçesinden bazı örnekleri:

***saat→saatler**, saatlar değil. Sesli uyumu bozulur, ikinci 'a' önce incedir. (Arapça kökenlidir).*

***red→reddi**, redi değildir. Son sessiz tekrarlanır (Arapça kökenlidir).*

***burun→burnu**, burun-u değil. Kök halindeki sözcüğün son seslisi düşer. (Türkçe kuralı).*

***su→suyu**, sunu değil. Bu sadece "su" kelimesi için söz konusudur.*

***ben→bana**, bene değil. Sesli bir kurala bağlı olmaksızın değişir, bu sadece birinci "ben" ve ikinci tekil kişi "sen" için geçerlidir.*

Zemberek içerisinde özel durumlar işlemin kökü değiştirip değiştirmediği, seçimlik olup olmadığı ya da gelen ekin bir sesliyle başlayıp başlamaması gibi özelliklere sahip olabilir. Eğer bir köke ilişkin belirli bir özel durum kök halindeki sözcüğü değiştirirse (kitap sözcüğünü kitab sözcüğüne dönüştürmek gibi), bir sözcük değiştirici kullanır. Sözcük değiştiriciler genel özellikleri olan basit nesnelerdir. Örneğin, bir addaki son seslinin düşmesi gibi özel bir durumda, sözcükteki son sesliyi sözcükten silen bir sözcük değiştirici kullanılır.

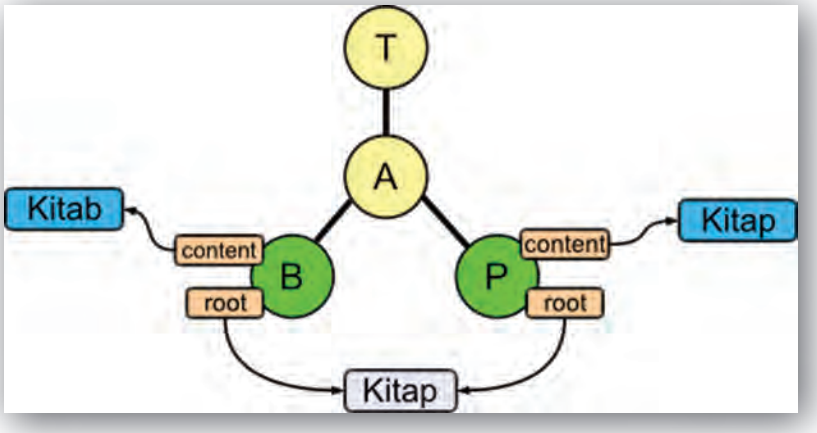
Çoğu kök sözcüğe ilişkin özel durum, sadece konfigürasyon dosyalarına dayanmaktan ziyade kök sözcüğün sonuna belirli ekler geldiğinde ortaya çıkar, kök özel durumları Zemberek'te bir miktar özel kodlama gerektirir.

Kök Ağacı

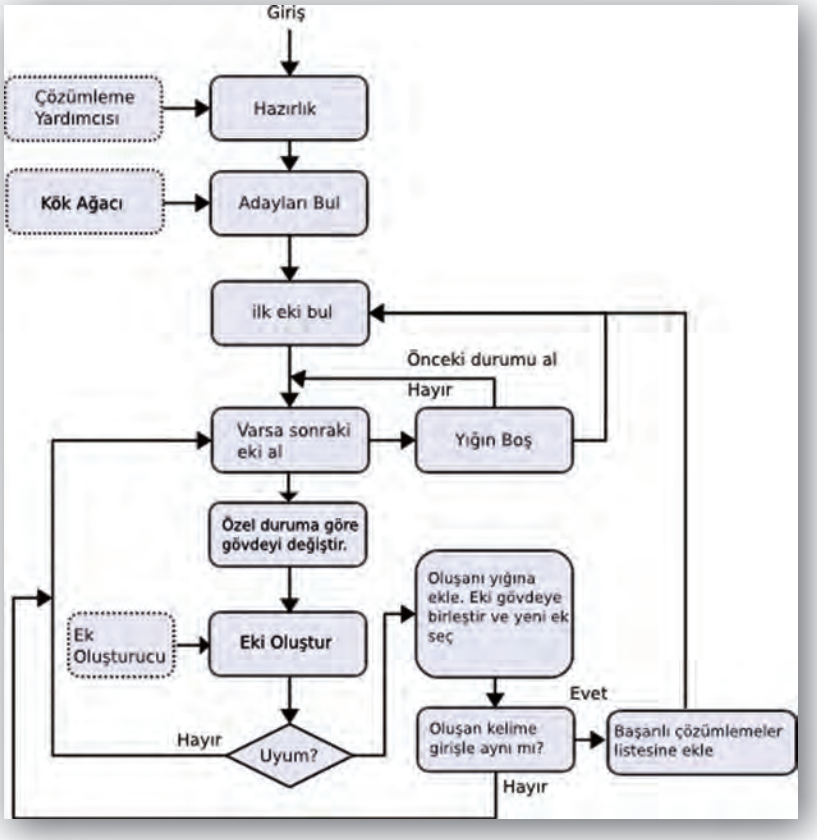
Zemberek kök halinde sözcük sözlüğü temelli ayrıştırıcı kullandığı için bir kök bulucu mekanizmaya ihtiyaç duyar. Ayrıştırıcı, kök halinde olabilecek sözcükleri bularak işlemi ayrıştırmaya başlar.

Kütüphanenin başlangıç aşaması boyunca, Zemberek ilk olarak ikili kök dosyasını yükler. Kök sözcük bir kez okunduğunda, ilgili özel durumlar kök nesnesine eklenir ve sonuç nesnesi, hızlı erişimi ve uzayabilirliği kolaylaştırmak için özel bir ağaç veri yapısında depolanır. Bu tip bir ağacın basitleştirilmiş yapısı, Şekil 2'de gösterilmiştir.

Kök sözcük ağaca eklendikten sonra, sistem ayrıca, sözcüğe bir özel durum uygulandığında ortaya çıkabilecek kök sözcüğün deforme olmuş hallerini de hesaplar. Ayrıştırılacak sözcüğün içinde kök sözcüğün tamamını içermeyebileceği için bunu yapmak zorunludur. Örneğin, Türkçede "kitaba" sözcüğü kök sözcük olan "kitap" sözcüğünü içermez, çünkü son harf sesli harfle başlayan bir ek gelmesinin ardından değişmiştir. Dolayısıyla, hesaplanan deformasyon "kitab" da, orijinal kök nesnesine atıfla ağaca eklenir (Şekil-3).



Şekil 2



Şekil-3

Bu ağaç bir sözlük olarak kullanılır ve farklı türlerde kök seçicilerin kolayca uygulanmasına olanak tanır. Son dönemde, Zemberek'te üç tane kök seçici bulunmaktadır; birincisi verili bir sözcük için normal katı kök seçimi amacıyla kullanılmaktadır. Örneğin, Türkçe "elmaslar" sözcüğü için {el:ad, elma:ad, elmas: ad} kümesini bulur. İkinci seçici/ selektör, dizilim benzerliği algoritmasını kullanarak olası/aday kök sözcükleri seçerken bir tolerans düzeyi uygular. Doğal olarak, bu katı selektörden daha fazla sonuç üretir. Üçüncü seçici, ASCII kodlamasında olmayan harfler için bir toleransa sahiptir. Ağaç ve seçiciler dil uygulamalarından tamamen bağımsızdır.

Standart Morfolojik Ayrıştırıcı

Morfolojik ayrıştırıcı temel olarak verilen bir sözcüğün olası kökünü ve eklerini bulur. Yapısı, basit sözlük temelli yukarıdan-aşağıya ayrıştırıcı olarak tanımlanabilir.

Bir sözcüğün morfolojik analizinde iki temel basamak vardır:

- *Sözcüğü ön işlemden geçirme*
- *Uygun kök seçisini kullanarak girdi sözcük için kök adaylarını bulma,*
- *Her kök adayı için, girişteki sözcük oluşturulana ya da ek alternatifi kalmayana dek olası ekleri köke eklemeye devam etmek, gerekirse özel durumları uygulamak.*
- *Ayrıştırıcı sonuçlarını bildirmek*

Ön işleme, sözcüğü ayrıştırma işlemine hazırlar. Bu, özel işaretleri, tireleri, vs.yi kaldırmayı ve sözcüğü küçük harfe dönüştürmeyi içerir. Sözcük tanımlanan alfabede olmayan karakterler içeriyorsa, işlem sonlanır. Aday kök sözcükler, girdi sözcüğün başlangıç parçası olarak görünebilen köklerdir. İşleme bağlı olarak kök adaylarını bulmak için uygun kök seçicileri kullanılır.

Kök adayları bulunduktan sonra sistem, kök halindeki sözcüğün türüne bağlı

olarak birinci kök eklerini bulur. Yukarıda anlatıldığı gibi, ilk ek (ve hemen hemen tüm ekler) kendisinden sonra gelebilecek eklerin listesini içerir.

Bu nedenle, birinci ekten başlanır ve kök özel durumuna neden olup olmadığı kontrol edilir. Eğer ek gerçekten bir özel durum yaratıyorsa, kök sözcüğü gerekli şekilde değiştirilir. Örneğin, eğer kök "burun"sa ve sözcüğe getirilen ek bir sesli ile başlıyorsa, ayrıştırıcı sözcükteki son sesliyi, kök halindeki sözcükten düşürür, (sözcük "burn" haline gelir).Türkçe için örnek bir ek üretim sözde kodu (pseudo-code) aşağıdaki gibidir:

```
produce( current, input , suffix_production_elements)
last_vowel = current.last_vowel
for each element in suffix_production_elements
switch (element.rule)
case LETTER:
result.append(element.letter)
case COMBINE:
if (current.last_letter is vowel)
result.append(element.letter)
case HARDEN:
if (current words last_letter is unvoiced)
result.append(element.letter.harden)
else
result.append(element.letter)
case WOWEL_TYPE_A:
last_vowel= wowelProducer.typeA(last_vowel)
result.append(last_vowel)
case WOWEL_TYPE_I:
last_vowel= wowelProducer.typeI(last_vowel)
result.append(last_vowel)
return result
```

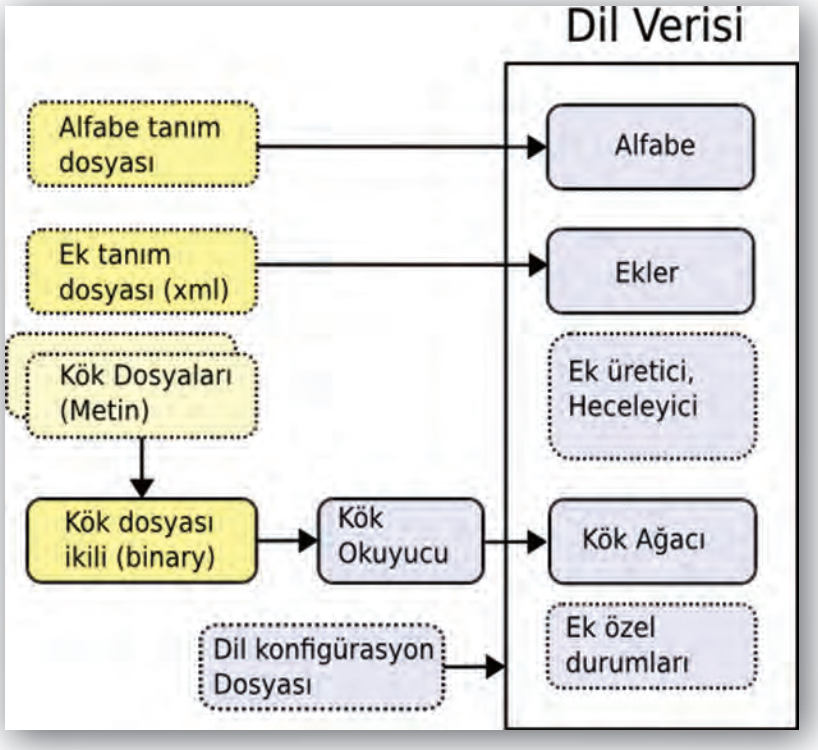
Bir ek üretildikten sonra, doğru olup olmadığını belirlemek için girdi sözcükle karşılaştırılır. İşleme bağlı olarak asıl ASCII toleranslı veya hata toleranslı gibi, farklı karşılaştırıcı türleri kullanılabilir. Eğer başarılı bir ek oluşturulduysa, yeni oluşturulan sözcük ve ek durumu bir bellekte depolanır ve işlem sonraki ek bilgisiyle devam eder. Başarılı bir ek serisiyle birlikte girdi sözcük ve kök tam olarak eşleştirildiğinde, ya sistem erken sonuçlara ulaşır (örneğin, işlem basit bir yazım denetimiyse) ya da sistem eşleştirilmemiş ek kalmayıncaya kadar diğer olası çözümleri izlemeye çalışır. Basitleştirilmiş bir blok ayrıştırıcı blok şeması şekil-4'te gösterilmiştir.

Genel olarak ayrıştırıcı mekanizma dilden bağımsızdır. Ancak doğru olası sonuçlar bir kez bulunduğunda, ardından semboller veya büyük harflerin ön işleme aşamasında temizlenmiş olmaları nedeniyle, bunların girdi sözcükte doğru kullanılıp kullanılmadığını kontrol etmek için bir son işlem (post-processing) daha gerçekleşir. Bu, kısaltmaları ve "Ahmet'in" veya "prof.e" ifadelerinde olduğu gibi özel harfler içeren sözcükleri ayrıştırma açısından özellikle önemlidir. Bu işlem dile bağlıdır ve bütün dil uygulamalarında tanımlanan bir yardımcı sınıf tarafından gerçekleştirilir.

Türk dillerinde önek yoktur. Türkçede vurgulama önekleri gibi, "mas+mavi", "bem+beyaz", bir kaç yaygın olmayan istisna vardır, fakat bu önekler belirli bir düzen izlemez. Ne var ki, bazı başka dillerden geçmiş sözcükler önek içerir. Mevcut durumda Zemberek önekleri ayrıştırmaz.

Ayrıca, Zemberek bazı yapım eklerini de ayrıştırmaz; örneğin, -it eki; yak→yakıt veya kes→kesit gibi. Bu tür kelimelerin ekli halleri sözlükte bulunur. Bununla birlikte, bazı yapım ekleri kök sözcük diye işaretlenerek Zemberek'e uygulanmıştır. Bunun bir örneği Türkçeye özel yansıma sözcükleridir.

Ayrıştırıcı için, sözcük sonuna gelen ekleri çıkarma [9] veya harf temelli ek sonlu otomatı kullanma gibi farklı yaklaşımlar denenebilirdi, fakat mevcut mekanizma iyi çalıştığı için söz konusu mekanizmalar uygulanmamıştır.



Şekil 4

Yazım Denetleyicisi

Yazım denetleme işlemi için standart morfolojik çözümleyici kullanır. Ancak, yazım denetiminde bütün ayrıştırıcı sonuçlarına gerek olmadığından performans için standart ayrıştırıcı bütün sonuçları bulmaya çalışmaz, bir karşılık bulunursa erken vakitte sonuçları bildirir.

Hata Toleranslı Ayrıştırıcı ve Sözcük Önerme

Hata toleranslı ayrıştırıcı, kök adaylarını bulmada ve kapsamlı bir ek oluşturma ve ekleme denemesinde olduğu gibi standart ayrıştırıcıya benzer biçimde çalışır. Mevcut uygulamada, hata toleranslı kök seçicisi Damerau-Levenshtein hata mesafesi algoritmasını kullanır, böylece sistemde bir karakter aralıklı hatalar bulunabilir. Mevcut sistemde şu anda kullanılmasa da Jaro-Winkler benzerlik algoritması da mevcuttur. Hata toleranslı ayrıştırıcıdan olası adayları bulduktan sonra Zemberek ayrıca girilen sözcüğün aslında yanlışlıkla birleşik yazılmış iki farklı kelime olup olmadığını da kontrol eder. Son olarak, Zemberek, sözcüğün Türkçeye özel karakterleri kullanmaksızın yazılıp yazılmadığını görmek için (örneğin Türkçe karakterler olmayan bir tuş takımı ile yazılmış Türkçe bir metin) ASCII toleranslı ayrıştırıcı kullanır. Sonrasında sonuçlar, en iyi olası önermelere ulaşmak için kök halindeki sözcük kullanımı sıklığını kullanarak sınıflandırılır. Bu sistem iyi bir belirsizlik giderimi algoritması ile daha da güçlendirilebilir.

Sözcük Oluşturma ve Ek İçeriği Çıkarma

Zemberek sözcük oluşturma fonksiyonuna da sahiptir. Bu morfolojik ayrıştırmadan daha basit bir işlemdir, kök sözcük nesnesi ve bir ek listesi gerektirir. Sistem temel olarak ekleri oluşturur ve bunlar oluşturulmuş sözcüğün sonuna getirilir. Ayrıştırma işlemine benzer biçimde özel durumlar da uygulanır. Örneğin, Türkmen dili için,

Kök: {it: Ad}Ekler {AD_ KUCULTME + AD_E HALİ} "itjegize"i (köpekceğize) üretir.

Bu mekanizma ayrıca ek içeriklerini ihtiva eden özel bir liste üretmek için de kullanılır. İşlem hem ayrıştırıcıyı hem de yapıcıyı kullanır. Ayrıştırıcı kökü ve ekleri (fakat ek halindeki sözcükleri değil sadece ek türünü) bulur ve sonrasında kelime oluşturucu sözcüğü yeniden yaratır, fakat bu kez üretilen ek içeriklerini depolar. Örneğin, Türkçe "koyunlara" sözcüğü için ayrıştırıcı şunu üretir:

Kök:[koyun, AD]

Ekler: [AD_ÇOĞUL+AD_E-HALİ]

Oluşturucudaki sonucu kullanırız, üretilen:

[koyun]+[lar,a]

İstatistikler

Zemberek, harfler, sözcükler, sözcük türleri, kökler, ek düzenleri, sözcük ikilileri (bigram), hece hesapları, hece ikilileri vb. için istatistiksel bilgi üretmek amaçlı temel istatistiksel analiz araçlarını sağlar. Kütüphane genişleyebilir bir istatistik raporlama ara birimine sahiptir.

Kullanım ve Uygulamalar

Kütüphanenin çoğu yüksek seviye işlemine Zemberek adı verilen özel bir erişim sınıfını kullanarak erişilebilir. Zemberek kullanarak Türkmen dilinde yazım denetimi için, bu iki kod sırası yeterlidir.

```
Zemberek z = new Zemberek(new Turkmence())

boolean result = z.denetle("pişige");
```

Zemberek, platform bağımsızlığı ve hız için Java dil ve platformunu kullanılmıştır.

Zemberek, i hali hazırda sadece akademik araştırmalar [10] [11] için değil, aynı zamanda OpenOffice.org [6] ve Türkçe Linux dağıtımı Pardus [7] gibi gerçek hayat uygulamalarında da kullanılmaktadır. Pardus bünyesinde Zemberek bir sunucu olarak açılışta başlatılır ve KDE uygulamalarının Kspell'e ilettikleri istekler zpspell isimli küçük bir uygulama ile Zemberek sunucusuna gönderilir. Firefox İnternet tarayıcısı ve Thunderbird e-posta uygulaması da benzer şekillerde yazım denetimi isteklerini bu sunucuya iletirler. Bu sayede Pardus işletim sistemi bünyesinde yazım denetimi yapan tüm uygulamalara Türkçe hizmeti verilebilmiştir.

Performans

Zemberek'in tasarımında sistemin performansına da öncelik verildi, bunun en önemli nedeni sistemin uygulamalarda da kullanılmasıdır. Özellikle yazım denetleme fonksiyonunun hızlı olması gereklidir. Performans için kodda yapılan bazı değişiklikler:

- *İkili (binary) kök dosyasının kullanımı kök ağacının oluşturulma süresini ve uygulama başlama zamanını azaltmıştır.*
- *Alfabe karakterin doğrudan eşlenmesi için büyük karakter dizileri kullanılmıştır.*
- *Basit sözlük veya liste yapılarıyla karşılaştırıldığında oldukça karmaşık olsa da özel bir kök ağacının kullanımı genel olarak daha iyi performans sağlamıştır.*
- *Yazım denetiminde erken sonuç bildirimi ve seçimlik sözcük önbelleği kullanılır.*
- *Eklerin olası ilk harflerini hesaplama ve standart ayrıştırma sürecinde bir ek içeriğini gerçek anlamda oluşturmadan ve karşılaştırmadan önce bu veriyi kontrol etme.*
- *Bazı ekler, istatistiksel olarak diğerlerinden daha fazla izleme olanağına sahiptir. Dolayısıyla ek*

dosyası seçimlik olarak "öncelikli ek" bilgisi içerir. Bu nedenle, bazı ekler ayrıştırma işleminde önceliğe sahip olur. Eklere ilişkin istatistik veriler, Zemberek'in istatistiksel bilgi toplama aracı kullanılarak toplanmıştır.

Bunların yanında başka performans iyileştirmeleri de mümkündür, fakat bunlar getirecekleri karmaşıklık nedeniyle şimdilik uygulanmamıştır.

Yazım denetimi performansı, 71.925 sözcükten oluşan bir Türkçe romanla test edilmiştir. Ortalama test sonuçları saniyede 75.000 sözcüktür. Yazım denetimiyle karşılaştırıldığında morfolojik analiz daha yavaş bir işlemdir. Aynı test nesnesi için Zemberek saniyede 12.000 sözcük analiz edebilir. Test ortamı: AMD Athlon 64 3000+ CPU, 512MB bellek, Java SE 6 Beta-2.

Diğer Türk Dilleri

Halihazırda Zemberek Türkçe ve Türkmen dillerinin tam uygulamasını içermektedir. Gerçek ek ve kök sözcük verisine girmeye hazır iskelet halinde bir Azerice ve Tatarca uygulamaları vardır. Bunun ardından, Özbek ve Tatar gibi Latin alfabesini kullanan Türk dilleri gelebilir. Kiril Alfabesi kullanan Kırgız ve Kazak dilleri de her ses için ayrı sembolü olması anlamında Latin Alfabesi'ne benzediği için uygulaması oldukça kolay olacaktır.

İran'da konuşulan Uygur, Azeri dili ve Osmanlı Türkçesi Arap Alfabesi kullandığından daha büyük bir engel oluşturuyor. Arap Alfabesi sesliler için yeterince sembol içermediği için, işleme sokulmadan önce sözcükleri Latin Alfabesi'ne çevirmek gibi özel işlemler gerektirir.

Gelecek Çalışmalar

Zemberek kütüphanesinde geliştirilecek çeşitli noktalar şöyle sıralanabilir:

• Diğer Türk dilleri için gerekli çalışmalar tamamlanmalıdır. Mevcut yapıdaki Azerice ve Tatarca çalışma iskeletlerinin tamamlanması, Kiril ve Arap Alfabesi kullanan Türk dilleri için gerekli çalışmaların başlatılması gerekmektedir.

• Yapılan çözümlemelerin net olarak belirlenmesi için gerekli olan ve NLP için çok önemli olan bir belirsizlik giderme sistemine ihtiyaç vardır.

• Çoklu sözcüklü kök sözcükleri ve deyimlerin eklenmesi gereklidir.

• Türk dilleri için açık bir küllliyatın (Corpus) ve kelime ağının (wordnet) bulunmaması daha fazla ve daha gelişmiş çalışmalar yapılmasını engellemektedir. Web tabanlı ve kullanımı tamamen özgür bir Crpus ve Wordnet ülkemizdeki dil araştırmaları için çok önemli bir kaynak olacaktır.

Sonuç

Zemberek, Türk dilleri için DDİ alanında çalışmak isteyen herkes için kullanımı kolay ve açık bir kütüphanedir. MPL lisansı sayesinde isteyen herkes rahatlıkla kodu inceleyebilir ve değiştirebilir. Şimdilik sadece en temel DDİ işlemlerinde kullanılsa da Türk dilleri araştırmalarında büyük bir boşluğu doldurduğuna inanıyoruz. Ne var ki, Zemberek sadece DDİ alanında küçük bir alanda kullanılabilmektedir. Zaman içinde daha zorlu konulara gireceğini umut ediyoruz. Zemberek projesi ile ilgili bilgilere ve kodun son haline http://code.google.com/p/zemberek adresinden erişilebilir.

Kaynaklar

[1] Türk dilleri hakkında genel bilgi http://en.wikipedia.org/wiki/Turkic_languages

[2] İlyas Çiçekli, Türkçe ve Kırım Tatarcası Arasında Bir Çeviri Sistemi, in: Bilgisayar Destekli Dil Bilimi Çalıştayı Bildirileri, Ankara, Turkey, 2005, s. 123-132.

[3] Oflazer Kemal Two-level Description of Turkish Morphology, Literary and Linguistic Computing, sayı. 9, No:2, 1994.

[4] Atalay Nart B., Oflazer Kemal ve Say Bilge, The Annotation in The Turkish Treebank, in Proceedings of the EACL Workshop on Linguistically Interpreted Corpora - LINC, Nisan 13-14, Budapeşte, Macaristan, 2003.

[5] Oflazer Kemal, Say Bilge, Hakkani-Tür Dilek Zeynep, ve Tür Gokhan. Building a Turkish Treebank, chapter in Building and Using Parsed Corpora, Anne Abeille Editor, Kluwer Academic Publishers, Eylül 2003.

[6] Türkçe Resmi Open Office.org sayfası: http://www.openoffice.org.tr

[7] Pardus proje sayfası: http://pardus.org.tr

[8] Allen James, Natural Language Processing (second edition), The Benjamin/Cummings Publishin Company, Inc., 1995.

[9] Eryiğit, G. ve Adali, E., 2004. An Affix Stripping Morphological Analyzer For Turkish, Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, İnnsbruck, Avusturya.

[10] Guychmyrat Amanmyradov, Türkçe Türkmence Bilgisayarlı çeviri sistemi, Karadeniz Teknik Üniversitesi FBE Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi, Haziran 2006.

[11] Görkem Özbek, Siddharth Jonathan, TURKALATOR, A Suite of Tools for Augmenting English-to-Turkish Statistical Machine Translation, Natural Language Processing Final Project. Haziran 2006.

[12] Özlem Çetinoğlu, A Prolog Based Natural Lan-guage Processing Infrastructure for Turkish. Yüksek Lisans Tezi, Boğaziçi Üniversitesi, 2001.