T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNTERNET TABANLI BİLGİ ERİŞİMİ DESTEKLİ BİR OTOMATİK ÖĞRENME SİSTEMİ

Erdinç UZUN
Doktora Tezi
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Yrd. Doç. Dr. Erdem UÇAR
II. Danışman: Yrd. Doç. Dr. Yılmaz KILIÇASLAN
2007
EDİRNE

T.C.
TRAKYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

İNTERNET TABANLI BİLGİ ERİŞİMİ DESTEKLİ BİR OTOMATİK ÖĞRENME SİSTEMİ

Erdinç UZUN

DOKTORA TEZİ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

Bu tez 15/06/2007 tarihinde aşağıdaki jüri tarafından kabul edilmiştir.

Yrd. Doç. Dr. Erdem UÇAR
(Danışman)

Yrd.Doç. Dr. Yılmaz KILIÇASLAN
(II.Danışman)

Prof.Dr. H.Erol AKATA
(Üye)

Prof.Dr.Şaban EREN
(Üye)

Yrd.Doç.Dr. Tahir ALTINBALIK
(Üye)

Doktora Tezi
Trakya Üniversitesi Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Bölümü

ÖZET

Bu tez, Türkçe için alt öğeleme listelerinin otomatik olarak elde edilmesi görevini gerçekleştirmek için planlanan web-tabanlı bir sistemi sunar. Zamir düşmesi, seyrek gösterimli bir dil ve serbest sıralaması özellikleri olan Türkçe doğal dil işleme görevleri için ilginç ve zorlukları olan bir uygulama alanı sağlar. Tez; bilgi erişimi, doğal dil işleme ve makine öğrenmesi alanlarına katkıda bulunmayı amaçlar. Öncelikle, doğal dil işleme ve makine öğrenmesi çalışmalarını kullanan çoklu derlemin otomatik olarak oluşturulmasını sağlayan bir web-tabanlı yaklaşım önereceğiz. Bunun için, arama motorlarını kullanarak internet üzerinden dilbilimsel Türkçe cümleleri toplayan ve hal durum bilgileri açısından bunları işaretleyen bir araç geliştirildi. İkincil olarak; rastgele seçilmiş Türkçe fiillere ait alt öğeleme listelerini elde etmek için oluşturulan derleme çeşitli makine öğrenme metotları uygulanmıştır. Üçüncül olarak; veri boyutunun metotların performansına etkisini anlamak için bu veri boyutu farklı boyutlarda birkaç alt kümeye bölünmüştür. Son olarak; özellikle gözetimli ve gözetimsiz metotların arasındaki farka odaklanan deneylerimizde kullanılan metotların karşılaştırmalı değerlendirilmesi önerildi.

Tezin organizasyonu şu şekildedir. İlk bölüm, bilgi erişimi, alt öğeleme listesi ve makine öğrenmesi kavramları hakkında ön bilgiler verir. Ayrıca, bu bölüm ilgili çalışmalara ve bilgisayımsal bakış açısıyla incelenecek bir dil olarak Türkçe'nin ayırt edici özelliklerine temas edecektir. İkinci bölüm, deneylerde kullanılan bazı makine öğrenmesi algoritma ve tekniklerini tanıtır. Üçüncü bölümde, doğal dil çalışmaları için uygun büyük bir veri seti olan "web olarak derlem" görüşü anlatılacaktır. Dördüncü bölüm, önerilen sistemin tasarımını ve uygulamasını verir. Beşinci bölüm, deneylerimizdeki sonuçları raporlar ve performansın farklı veri boyutlarına etkisini gözlemler. Ayrıca, deneylerde kullanılan metotların bir karşılaştırmalı

değerlendirilmesini sağlar. Tez, altıncı bölümde ana bulgular ve sonuçların özeti ile bitirilmektedir.

Anahtar Kelimeler: Alt öğeleme listesinin otomatik elde etme, makine öğrenmesi metotları, bir derlem olarak web

Doctorate Thesis
Trakya University Graduate School of
Natural and Applied Sciences
Department of Computer Engineering

ABSTRACT

This thesis presents a web-based system that is intended to perform the task of automatic acquisition of subcategorization frames for Turkish. As a pro-drop, a referentially sparse and free word order language, Turkish provides an interesting and challenging domain of application for natural language processing tasks. The thesis aims to contribute to the fields of information retrieval, natural language processing and machine learning in the following respects. Firstly, we offer a web-based approach to the automatic construction of corpora to be used in natural language processing and machine learning work. To this effect, we implemented a tool that collects grammatical Turkish sentences from internet via search engines and annotates them with respect to case marking information. Secondly, various machine learning methods were applied to the generated corpus in order to acquire the subcategorization frames of a set of randomly chosen Turkish verbs. Thirdly, we divided our set of patterns into several subsets of different sizes to understand effect of data size on the performance of methods. Lastly, we offer a comparative evaluation of the methods used in our experiments, focusing particularly on the distinction between supervised and unsupervised methods.

The thesis is organized as follows. The first chapter gives a brief account of the concepts of information retrieval, subcategorization frame and machine learning. Moreover, this chapter touches upon the relevant literature and the peculiarities of a Turkish as a language to be investigated from a computational point of view. The second chapter introduces some machine learning algorithms and techniques used in our experiments. In the third chapter, we describe the view of web as a corpus that is the largest data set available for natural language studies. In the fourth chapter, the design

and implementation aspects of the proposed system are given. The fifth chapter reports on the results of our experiments and provides a comparative evaluation of the methods used in the experiments along with observations on the effect of data size on the performances. The thesis ends with a summary of major findings and conclusions in chapter six.

Keywords: Automatic acquisition of subcategorization frames, machine learning methods, web as a corpus

TEŞEKKÜR

Bu çalışmanın hazırlanması esnasında bana yol gösteren, bu alanda çalışmam için beni teşvik eden, yardımlarını ve desteklerini benden esirgemeyen değerli hocalarım Yrd. Doç. Dr. Erdem UÇAR, Yrd. Doç. Dr. Yılmaz KILIÇASLAN ve Prf. Dr. H. Erol AKATA'ya teşekkür ederim.

Doktora tezi savunma jurinde yer alarak bilgi ve tecrübelerinden yararlandığım Prf. Dr. Şaban EREN veYrd. Doç. Dr. Tahir ALTINBALIK'a teşekkür ederim.

Çalışmalarım sırasında değerli katkılarıyla bana yardım eden ve ortak çalışmalar yaptığımız arkadaşlarım Arş. Gör. Volkan AGUN ve Arş. Gör. Özlem AYDIN'a; ayrıca çalışabilmem için gerekli ortamı sağlayan tüm mesai arkadaşlarıma çok teşekkür ederim.

Tez çalışmam sırasında bana her zaman destek olan ve sabır gösteren babam Necdet UZUN ve annem Saime UZUN'a çok teşekkür ederim.

Ayrıca çalışma hayatımda bana destek olan ve yardımlarını esirgemeyen Bilgisayar Mühendisliği bölüm hocalarıma ve çalışma arkadaşlarıma çok teşekkür ederim.

Haziran 2007

İÇİNDEKİLER

BÖLÜM 1.

GİRİŞ

1.1. Alt öğeleme listesi

Alt öğeleme listesi (Subcategorization Frame) bir kelimenin çevresine aldığı sözdizimsel argümanların sayısı ve tipini gösterir. Alt öğeleme listesi sözdizimsel dilbilim teorilerinde kullanılan önemli bir kavramdır. Chomsky (1965) alt öğeleme fikrinin kullanıldığı ilk gramatikal teoridir. Bundan sonra ortaya çıkan birçok gramatikal teori[1] ve kural tabanlı[2] ayrıştırıcılar için temel parça olma özelliğini devam ettirmiştir. Başka bir deyişle, alt öğeleme listesi bilgisi dilbilim ve bilişimsel dilbilim alanlar için anahtar bileşen olma özelliğine sahiptir.

1.2. Alt öğeleme listelerinin hazırlanması ve önemi

Alt öğeleme listelerinin bulunması yüksek kapsamlı ayrıştırıcılar için önemli bir engel oluşturur. Birçok bilişimsel dilbilim çalışmasında alt öğeleme listesi bilgileri manüel olarak girilmektedir. Ancak, bu durumun bir takım sorunlara sebebiyet verdiği bazı araştırmacıların dikkatini çekmiştir [Manning, 1993] [Briscoe ve Caroll, 1997] [Basili vd., 1997] [Basili ve Vindigni, 1998] [Sarkar ve Zeman 2000] [Korhonen vd., 2006]. Bu araştırmacılara göre:

- Çok emek gerektiren bir iştir.
- Manüel yapıldığı için hatalar kaçınılmazdır.
- Tamamlanmış listelerin genişletilmesi çok masraflı olacaktır.

---

1 Standard and Extended Standard Theories of Transformational Generative Grammar [Chomsky, 1965, 1975, 1977] [Jackendoff, 1972], Government-Binding (GB) Theory [Chomsky 1981, 1982], Minimalist Program [Chomsky 1993, 1995] [Weinberg, 2001] [Lasnik 2002] [Uriagereka 2002], Generalized Phrase Structure Grammar (GPSG) [Gazdar vd., 1985], Head-Driven Phrase Structure Grammar (HPSG) [Pollard ve Sag 1987, 1994], Lexical-Functional Grammar (LFG) [Bresnan 1978, 1982] [Levin vd., 1983].

2 Manning (1993), Briscoe ve Carroll (1993), XTAG (1995), Korhonen (1998)

- Özel alanlara ve değişimlere adapte olunması kolay olmayacaktır.

Aslında, birçok sözlük alt öğeleme listelerini araştırmacılara sağlar. Ancak, Manning (1993) belirttiği gibi bu bilginin elle girilmesi hem can sıkıcı hem de zahmetli bir iştir. Ayrıca, bu gibi sözlüklerde birçok dil veya özel alanlar için hazırlanmamış olabilir.

1.3. Otomatik olarak alt öğeleme listelerinin bulunması

Makine öğrenmesi yöntemlerini kullanarak alt öğeleme liste bilgilerinin toplanması ve önceki bölümlerde belirtilen problemlerin üstesinden gelinmesi mümkündür. Makine öğrenmesi bilgisayarın "öğrenme" işlemini otomatik olarak yapmasını sağlayacak algoritma ve tekniklerin gelişimi anlamına gelir. Son yirmi yılda, alt öğeleme listesinin bulunması üzerine birçok çalışma yapılmıştır. Bu çalışmalar çoğu İngilizce üzerine yapılan çalışmalardır [Webster ve Marcus, 1989] [Brent, 1991, 1992, 1993] [Manning, 1993] [Ushioda vd.,1993] [Ersan ve Charniak, 1996] [Briscoe ve Caroll 1997] [Carroll ve Minnen, 1998] [Carroll ve Rooth, 1998] [Korhonen vd. 2000] [Sarkar ve Tripasai, 2002] [Korhonen vd., 2006]. Ayrıca, İtalyanca [Basili vd., 1997] [Basili ve Vindigni, 1998], Almanca [De Lima 1997], Çekçe [Sarkar ve Zeman 2000] [Zeman ve Sarkar 2000] [Zeman 2002], Japonca [Kawahara vd., 2000; Kawahara ve Kurohashi 2001], Yunanca [Maragoudakis vd. 2000, 2001], Çince [Zhang ve Zhou 2003], Bulgarca [Marinov 2004, Marinov ve Hemming 2004], Fransızca [Chesley ve Salman-Alt 2006] üzerine çeşitli çalışmalar yapılmıştır. Ancak bildiğimiz kadarıyla Türkçe üzerine makine öğrenmesi tekniklerini kullanarak alt öğeleme listesi bilgilerinin bulunması çalışması yoktur. Türkçe dilbilimsel açıdan bakıldığında zamir düşmeli (pro-drop), seyrek gösterimli (referentially sparse) ve serbest sözcük sıralamalı (free word order) bir dildir. İngilizce ile bu açıdan farklılıklar gösterir. Bu farklılıkları sebebiyle makine öğrenme alanındaki araştırmacılar için zor bir dil olarak görülebilir (bkz. Bölüm 1.6.).

1.4. Makine öğrenmesi ve Derlem Olarak İnternet

Bir problemin makine öğrenmesi teorisi ile çözülebilmesi için iki unsur önem taşır:

- Hangi algoritma veya metodun kullanılacağı
- Kullanılacak metinin seçimi ve hazırlanması

Makine öğrenmesi üzerine birçok algoritma ve teknik geliştirilmiştir. Bu teknikler ve algoritmalar çok çeşitli alanlarda kullanılmaktadır. Bölüm 2'de kullanılacak makine öğrenmesi algoritma ve teknikleri üzerinde ayrıntılı bir şekilde durulacaktır.

Diğer çalışmalar incelendiğinde genelde derlem denilen dilbilimsel çalışmalar için oluşturulmuş özel metin kaynakları kullanılır. Türkçe için Metu Derlemi göze çarpmaktadır. Fakat bu derlem her ne kadar çoğu çalışma için yeterli olsa da 2 milyon kelime içermesi bakımından bazı fiilleri hiç içermemekte veya çok az sayıda içermektedir. Bizim çalışmamız açısından çok örneğe ihtiyaç vardır. Son yedi yılda yapılan çalışmalara bakıldığında internetin derlem olarak kullanılma fikri yaygınlaşmaya başlamıştır. Bölüm 3'te kullanılabilecek metinlerin elde edilmesi ve hazırlanması üzerinde durulacaktır.

1.5. Bir fiilin alt öğeleme listesi

Alt öğeleme listesinin makine öğrenmesi tekniklerini kullanarak bulunması çalışmalarında genelde fiiller üzerinde durulmaktadır. Yaptığımız çalışmada, bir fiile ait durumların otomatik olarak saptanması üzerinedir. Türkçede bir fiil yanına belli sayıda kelime ve bu kelimelere bağlı hal ekleri almaktadır. Türkçe 6 adet hal eki bulunur:

- Yalın hal (nominative)
- -i hali (yükleme durumu, accusative)

- -e hali (yönelme durumu, dative)
- -den hali (ayrılma durumu, alative)
- -de hali (bulunma durumu, locative)
- -ile hali (instrumental)

Türkçede bir fiil belli bir sayıda kelime alacağını belirtmiştik. Örneğin aşağıdaki cümleyi incelediğimizde:

"Ali Ayşe'ye kitabı verdi."
"Ali Ayşe-e kitap-i verdi."

"vermek" fiili yanına 3 adet kelime ve kelimelerin sırasıyla yalın, -e ve -i hallerini aldığını görmekteyiz. Bu örneğe bakarak vermek fiilinin alt öğeleme listesini kolayca bulabiliriz. Fakat Türkçe açısından her zaman bir takım zorluklar karşımıza çıkar.

1.6. Türkçede alt öğeleme listesinin bulunmasında karşılaşılacak zorluklar

Türkçe, alt öğeleme listesinin makine öğrenmesi tekniklerini kullanarak bulunması açısından oldukça zor bir dildir. Türkçenin başlıca zorlukları şunlardır:

- Zamir düşmesi özelliği olan bir dil olması
- Seyrek gösterimli bir dil olması
- Serbest sözcük sıralaması olması

Türkçede cümle içinde zamirler gerçekleştirilmemiş bir durumda olabilir. Örneğin;

"Ali Ayşe-den kitab-ı ödünç aldı. Ertesi gün ver-di."
"Ali Ayşe-den kitap-i loan take-past following day give-past"

Ali borrowed the book from Ayşe. The following day, he gave (back) it to her.

İlk cümlede “al” fiili alt öğeleme listesine yalın hal, -den hali ve -i hali durumlarını almıştır. Fakat ikinci cümlede normal koşullarda “ver” fiili yalın, -e ve -i hallerini almak zorunda olsa da tüm zamirler düşmeye uğramıştır. İngilizce cümle incelendiğinde zamirlerin ikinci cümlede devam ettiği görülmektedir.

Türkçede bir cümle içinde alt öğeleme listesinde bulunması gereken öğelerin bir kısmı ve hatta hepsi cümle içinde olmayabilir. Örneğin;

Ertesi gün Ali Ayşe’ye kitabı verdi.	Ertesi gün Ayşe’ye kitabı verdi.
Ertesi gün Ali Ayşe’ye verdi.	Ertesi gün Ayşe’ye verdi.
Ertesi gün Ali kitabı verdi.	Ertesi gün kitabı verdi.
Ertesi gün Ali verdi.	Ertesi gün verdi.

Örnek 1.1. Türkçe cümle örnekleri

İngilizce cümlede ise böyle düşmeler olmaz. Diğer bir deyişle, Türkçe seyrek gösterimli bir dildir.

Türkçenin diğer bir zorlayıcı özelliği ise serbest sözcük sıralaması [Erguvanlı, 1984] [Kılıçaslan, 2004] olmasıdır. Örneğin İngilizce bir cümleyi ele alalım.

“Ali gave (back) the book to Ayşe”

İngilizce açısından kelimelerin durumu ve pozisyonu anlamı değiştirmektedir. Türkçe olarak ise:

Ali Ayşe’ye kitabı verdi.	Ali verdi Ayşe’ye kitabı.
Ali Ayşe’ye verdi kitabı.	Verdi Ali kitabı Ayşe’ye.
Ali kitabı Ayşe’ye verdi.	Verdi kitabı Ayşe’ye Ali.
Ali kitabı verdi Ayşe’ye.	Verdi kitabı Ali Ayşe’ye.
Ali verdi kitabı Ayşe’ye.	Verdi Ali kitabı Ayşe’ye.

Örnek 1.2. Türkçe cümle örnekleri

Kurulabilecek cümlelerin 8 âdeti yukarı yazdık. Türkçede 4 kelimelik bir cümle 24 farklı durum oluşturulabilir. Bu da Türkçe bir cümlenin ayrıştırma işlemini zorlaştırır.

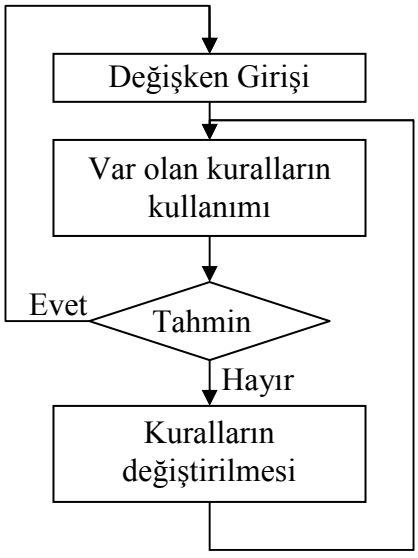
BÖLÜM 2.

MAKİNE ÖĞRENMESİ

2.1. Öğrenme nedir?

Öğrenme davranışların değişmesi olarak tanımlanabilir. En yalın tanımla bilebilme ve yapabilmedir. Öğrenmenin sınırı yoktur, yaşam boyu devam eder. İnsanoğlu her şeyi bilmek doğaya hâkim olmak ister. Bugünkü uygarlığı da insanın öğrenme merakı yaratmıştır.

İnsan soyut olaylar ve örnekler yardımı ile öğrenebilir. Yeni ve farklı olaylarla karşılaşarak her geçen gün "bilgi deposunu" zenginleştirir. İnsan geçmişten ders alıp bilgi deposunu yenileme özelliğine sahiptir. Bu yenileme işlemi ile öğrenmeye "geri beslemeli öğrenme" denir. Geri beslemeli öğrenmede tecrübe ön plana çıkar.[Nabiyev, 2005] (Şekil 2.1.)



Şekil 2.1. Geri Beslemeli Öğrenmenin Genel Yapısı

2.2. Makine öğrenmesi

Yapay Zekânın alt bir dalı olan Makine Öğrenmesi (Machine Learning) ise, bilgisayarların "öğrenme" işlemini sağlayacak algoritma ve tekniklerin gelişimini ile ilgili bir çalışma alanıdır. Makine öğrenmesi, insanlardaki geri beslemeli öğrenmenin yapısının bilgisayara tanışmış hali olarak görebilir. (Şekil 2.1.)

Makine öğrenmesi; Doğal Dil İşleme, Konuşma ve El Yazısı Tanıma, Nesne Tanıma, Bilgisayar Oyunları, Robot Hareketleri, Arama Motorları ve Tıbbi Teşhis gibi birçok alanda kullanılır.

Makine öğrenmesi üç önemli aşamadan oluşur:

1. Dokümanların Hazırlanması
2. Öğrenme Metotların Uygulanması
3. Öğrenmenin Performansının Değerlendirilmesi

Makine öğrenmesinde öncelikle öğrenme yapılacak veri setinin uygulanacak öğrenme metoduna uygun bir şekilde hazırlanması gerekir. Öğrenme metodunda istatistiksel yöntemler kullanılır. Geliştirilen yeni metotlar da istatistiksel temelli metotlardır. Yeni bir metot bulunduktan sonra bu metodun performansı ölçülür ve diğer metotlarla karşılaştırılması yapılır.

Makine öğrenmesi kavramın gelişimi ile birlikte çok farklı metotlar ortaya çıkmıştır. Bu metotları kullanın algoritma tekniklerine göre sınıflarsak:

- Gözetimsiz Öğrenme
- Gözetimli Öğrenme
- Yarı-Gözetimli Öğrenme
- Ödüllü Öğrenme
- Uyum Sağlama ile Öğrenme
- Öğrenme ile Öğrenme

Farklı uygulamaların analizinde faklı beklentiler olmaktadır. Makine öğrenmesi metotlarını bu beklentilere göre sınıflandırmakta mümkündür[Alpaydın, 2004].

- Sınıflandırma (classification): Geçmiş bilgilerin hangi sınıflara ait olduğu verildiğinde yeni gelen bilginin hangi sınıfa ait olduğunun bulunması işlemidir.
- Kümeleme: Geçmiş bilgilerin bilinmediği durumlarda verilerden birbirine benzerlerin yer aldığı kümelerin bulunması işlemidir.
- Eğri Uydurma (Regresyon): Geçmiş bilgilere karşılık gelen sınıflar yerine sürekli değerlerin yer aldığı problemlerdir.
- Özellik seçimi ve çıkarımı: Veriye ait birçok özellikten verinin kümesinin veya sınıfının değerlerini belirleyen özelliklerin belirlenmesidir.
- İlişki belirleme: Bulunan sınıf veya kümeler arasında ilişkilerin çözümlemesidir.

2.2.1. Gözetimsiz öğrenme

Gözetimsiz öğrenme (Unsupervised Learning) modeli gözlemlere bağlı bir makine öğrenmesi tekniğidir. Başka bir deyişle yöntem çıktı verilerinin kullanmadan sadece girdiler üzerinden öğrenme işlemini gerçekleştirmeye çalışır. Bu yöntem özellikle veri nesneleri kümesini toplamada kullanılır. [Hinton ve Sejnowski, 1999]

Gözetimsiz öğrenme tekniği, veri sıkıştırma işleminde kullanışlıdır. Veri sıkıştırma işleminde bir girdi seti üzerinden *olasılık dağılımı*na bağlı olan bir algoritmadır.

Gözetimsiz öğrenme teknikleri özellikle olasılık teorisine dayanan *kümeleme* işleminde kullanılır. Kümeleme farklı gruptaki benzer nesnelerin sınıflandırmadır. Kümeleme; makine öğrenmesi, veri madenciliği, örüntü tanıma ve resim analizi gibi birçok alanda kullanılan istatistiksel veri analizi için genel bir tekniktir.

2.2.2. Gözetimli öğrenme

Gözetimli öğrenme (Supervised Learning) eğitim verileri üzerinden bir fonksiyon üreten bir makine öğrenmesi tekniğidir. Başka bir deyişle, bu öğrenme tekniğinde algoritma girdilerle (etiketlenmemiş veri) çıktılar (etiketlenmiş veri) arasında eşleme yapan bir fonksiyon üretir. *Eğitim verisi*, girdi nesneleri ve istenilen çıktılarından oluşur. Fonksiyonun çıktıları, *eğri uydurma* (*regresyon*) olabilir ya da girdi nesnesinin *sınıf etiketlenmesi* (*sınıflama*) ile tahmin edilebilir.

Bir gözetimli öğrenme problemini çözmek için aşağıdaki adımlar düşünülmelidir:

1. Eğitim örneklerinin tiplerini belirleme: Örnek olarak kullanılan verilerin ne çeşit olduğuna karar verilmelidir. Mesela, el yazısı tek karakterden, tam bir kelimeden ya da tam bir cümleden oluşabilir.
2. Bir eğitim verisi toplama: Girdi nesneleri ve uygun çıktıların uzman bir kişi veya ölçümlerle toplanmasıdır.
3. Öğrenme fonksiyonunun girdi özellik gösterimi belirleme: Öğrenme fonksiyonunun doğruluğu girdi nesnesinin nasıl temsil edildiğine bağlıdır. Tipik olarak, bir girdi nesnesi nesneyi anlatan bir takım özellikleri içeren bir özellik vektörüne dönüştürülür. Özellik kümesi, boyut sorunu sebebiyle çok büyükte olmamalıdır.
4. Öğrenme fonksiyonunun yapısına ve uygun öğrenme algoritmasına karar verme: Örneğin, problemin çözümünde yapay sinir ağlarını mı yoksa karar verme ağaçlarını mı kullanılacağının seçilmesidir.
5. Tasarımın tamamlanması: Öğrenme algoritması toplanan eğitim seti üzerinden çalıştırılır. Öğrenme algoritmasını parametreleri eğitim kümesinin alt kümeleri(*onaylama kümesi*) üzerinden performansı optimize edecek şekilde ayarlanabilir. Yeni parametreler ayarlandıktan sonra, algoritmanın performansı farklı bir veri seti üzerinden tekrar ölçülebilir.

2.2.3. Yarı-Gözetimli öğrenme

Yarı-Gözetimli Öğrenme (Semi-supervised Learning) uygun sınıflama veya fonksiyonu bulmak için etiketlenmiş ve etiketlenmemiş örneklerin her ikisi de kullanılır. Genelde etiketlenmiş veri miktarı etiketlenmemiş veri miktarına göre daha azdır. Tüm etiketlenmemiş verileri de etiketlemek zordur. Bu sebeplerden dolayı yarı-gözetimli öğrenme pratik bir çözüm olarak düşünülebilir. [Blum ve Mitchell, 1998] [Chapelle vd., 2006] [Huang T-M., 2006]

2.2.4. Ödüllü öğrenme

Ödüllü öğrenmede (Reinforcement Learning) algoritma verilen bir gözlemin nasıl gerçekleşeceğinin politikasını öğrenir. Çevredeki her hareket bazı etkilere sahiptir. Çevre öğrenme algoritmasına yol gösteren bir geri besleme sağlar. [Littman ve Moore, 1996] [Sutton ve Barto, 1998] [Bertsekas ve Tsitsiklis, 1996] [Peters vd., 2003] [Fu ve Anderson, 2006]

Çevre tipik olarak *Markov Karar İşlemi* olarak formüle edilir. Markov Karar İşlemi, durumlara karar vermek için matematiksel bir çatı sağlar. Bu çatı kullanılarak ödüllü öğrenme algoritması dinamik programlama teknikleri ile öğrenme işlemini gerçekleştirir.

Ödüllü öğrenme, robot kontrolü, asansör sıralaması, telekomünikasyon, tavla ve satranç gibi çeşitli problem alanlarına uygulanabilir.

2.2.5. Uyum Sağlama ile öğrenme

Bu yöntem gözetimli öğrenme yöntemine benzer fakat kesin bir fonksiyona dayanmaz. Bunun yerine, yeni girdi ve çıktıları tahmin etmeye çalışır. *Uyum Sağlama (Tranduction)* 1990'lı yıllarda Vladimir Vapnik tarafından tanıtılmıştır. [Vapnik, 1998]

2.2.6. Öğrenmek için öğrenme

Öğrenmek için öğrenme (Learning to learn) önceki deneyimleri temel alarak kendi tümevarımsal eğilimi ile öğrenen bir algoritmadır. Özelikle Thrun (1996), Caruana (1997) ve Baxter (2000) bu öğrenme metodu özelliklerini açıklamışlardır.

Geliştirilen çoğu metot altı öğrenme tipinden ilk ikisini kapsamaktadır. Tezdeki çalışmalar açısından özellikle ilk iki öğrenme tipi daha ön plana çıkmaktadır. Bu sebeplerden dolayı, bu tezde gözetimli ve gözetimsiz öğrenme tipleri üzerine geliştirilen metotlar üzerinde durulacaktır.

2.3. Önemli Kavramlar

Eğitim Verisi (Eğitim Kümesi – Örnekleme Kümesi)(Training Set): girdi vektörlerinin ve cevap (çıktı) vektörlerini içerir ve özellikle gözetimli öğrenme yönteminde kullanılır.

Çapraz Doğrulama (Cross Validation): Eğitim verisi alt kümelere ayrılır. Tek alt kümeyi eğitim için kullanıp diğer kalan kümeleri doğrulama işlemi için kullanılır. Bu işlem çapraz bir şekilde tüm alt kümeler için tekrarlanır. Bu işleme *çapraz doğrulama* denir.

Gürültü (Noise): Konusu dışı ve anlamsız bilgi anlamına gelir. Makine öğrenmesi algoritmalarının amacı gürültüyü azaltıp doğru sınıflara ulaşmayı amaçlar. Gürültü, eğitim verisi açısından çok önemlidir. Eğer seçilen eğitim verisi az gürültülü ise bu durumda yeni gelen eğitim verilerinin tahmini zorlaşır.

Aşırı Beslenme (Over fitting): örnek verinin çok küçük veya büyük olması, gürültü miktarı ve çakışmalar gibi sebeplere aşırı yüklenme denir.

2.4. Dokümanın hazırlanması

Bir metin dokümanı makine öğrenmesi metotlarını kullanarak sınıflamak veya kümelemek için öncelikle dokümanın bir hazırlama işleminden geçmesi gerekir. İşleme hazırlama dokümanı makine öğrenmesi teknikleri için uygun bir duruma getirme işlemidir. Hazırlama işlemi için genelde terim sayma modeli (term count model) ve vektör uzayı-modeli (vector-space model) kullanılır. [Özgür, 2004]

Dokümanların hazırlanması 5 farklı işlemden oluşur:

- Dokümanların Ayrıştırılması
- Gereksiz Kelimelerin Temizlenmesi
- Kelime Köklerinin Tespiti
- Terim Ağırlıkları
- Boyutsal İndirgeme

2.4.1. Terim Sayma Modeli

Terim sayma modeli özellikle internetteki arama motorları gibi bilgi erişim sistemlerinde kullanılan bir basit modeldir. Bu model, indeksleme ve ilişki değerlendirme işlemlerinde kullanılır.

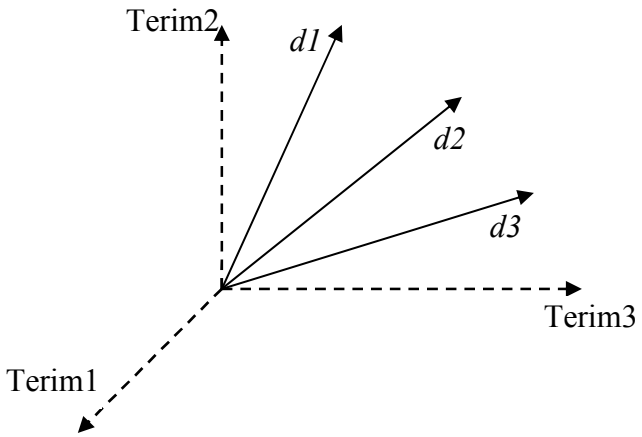
$$Genel\_Terim = \frac{D}{d(n)}$$

D: bir arama seti veya veritabanındaki doküman sayısı
d(n): arama terimi *(n)* içeren doküman sayısı

Bu modelin kötü tarafı fazla tekrarlayan terimleri ön plana çıkarmasıdır.

2.4.2. Vektör-Uzayı Modeli

Vektör-uzayı modeli bilgi filtreleme, bilgi erişimi, indeksleme ve ilişki değerlendirmede kullanılan matematiksel bir modeldir. İlk defa Cornell Üniversitesi tarafından 1960 yılında geliştirilen Smart Information Retrieval System [Salton, 1968] adındaki bir bilgi erişim sisteminde kullanılmıştır. Bu modelde, her doküman vektör *d* ile gösterilir. Vektör *d* deki her boyut dokümanların terim uzayında farklı bir terimi belirtir. (Şekil 2.2.)



Şekil 2.2. Vektör-Uzayı Modeli

Bir terim, bir tek kelime, kök veya tümcecik olarak tek bir kelime ile gösterilir. Tümcecik birden fazla kelimeden oluşabilir. Tümcecikler, istatistiksel yöntemlerle veya doğal dil işleme yöntemleriyle belirlenebilir. İstatistiksel teknikler kelimelerin bir arada kullanılma sıklıklarına bakılarak çıkarılabilir [Cohen ve Singer, 1996]. Doğal dil işleme teknikleri konusunda ise Fuernkranz ve ark. (1998) çalışmaları vardır. Tümcecikler elle de belirlenebilir fakat bu Web gibi büyük bir doküman kümesi içinde hamallık gerektiren bir iştir. Terimler, kelimeler, kök veya tümcecikler hazırlandıktan sonra bir vektör uzayı içine yerleştirilir.

Oluşan her *d* vektör arasında bir açı oluşur. Eğer bu açıyı hesaplanır ve karşılaştırırsa vektör uzayındaki ilişkiler hakkında bilgi edinilmiş olunur:

$$\cos\theta = \frac{d1 \cdot d2}{\|d1\| \cdot \|d2\|}$$

Kümeleme ve sınıflama algoritmaları kullanmak için iki doküman arasında benzerlik ölçümünü belirlenmesinde bu formül kullanılır. *cos* değerine göre benzerlik oranı hesaplanır. Eğer *cos* sıfır çıkarsa herhangi bir ilişkinin olmadığı anlamına gelir.

2.4.3. Bir dokümanın hazırlanması sırasında dikkat edilecek işlemler

2.4.3.1. Dokümanın Ayrıştırılması

Ayrıştırma işlemi genelde internet dokümanlarından HTML ve XML gibi etiketlerden temizlenmesi ile yapılır. Böylece doğal bir metine ulaşılır.

2.4.3.2. Gereksiz Kelimelerin Temizlenmesi

Zamirler, ilgeçler ve bağlaçlar bir doğal metinlerde çok sık tekrarlayan kelimeler frekans sorunlarına yol açabilir. Bu sebepten dolayı bu kelimelerin önceden belirlenip temizlenmesi gerekir. Örneğin Smart Sistem 571 adet gereksiz kelimeyi almamaktadır [Salton, 1975].

2.4.3.3. Kelime Köklerinin Tespiti

Kelimeler eklerle birleştiğinde aynı anlama sahip olmasına rağmen içerdikleri karakterler farklıdır. Doğru anlama ulaşabilmek için eklerden kurtulup köke ulaşılması gerekir. Örneğin İngilizce için:

Cat : Cats : s
Falling : fall: ing
tanned : tann : ed

Örnek 2.1. İngilizce için bazı kelime örnekleri

İngilizce için genelde Porter's Stemming Algorithm [Porter, 1980] kullanılır. Fakat Türkçe için bu tür bir algoritma çok daha karmaşık olacaktır. Bu konuda Oflazer (1994) çalışmalar vardır.

2.4.3.4. Terim Ağırlığı

Terim ağırlığı, bir doküman içindeki terime ait bilgi anlamına gelir. Bir doküman (vektör *d*) içinde farklı ağırlıklara sahip terimler vardır. Denklem olarak ifade edecek olursak:

$$d = (w_1, w_1, ..., w_{|T|})$$

w_i : *i*.nci terimin ağırlık değeri,

$|T|$: bir doküman içindeki farklı terimlerin sayısı anlamına gelir.

Yukarıdaki formüldeki ağırlık değeri duruma göre iki farklı şekilde ifade edilebilir.

- Bir doküman içindeki ilişkili kelimelerin doküman içindeki görülme sayısı hesaplanarak.
- Eğer dokümanların çoğunda görülüyorsa, doküman içinde değeri az ise ayırt edici bir ağırlık değeri belirlenerek.

Terim ağırlığı belirlemede çeşitli yaklaşımlar karşımıza çıkar. Hazırlanan makine öğrenmesi tekniğine uygun olarak aşağıdaki yaklaşımlardan biri veya bir kaçı kullanılabilir:

- Yanlış/Doğru Ağırlığı: Bu yaklaşım en basit terim ağırlığı hesaplama yöntemidir. Eğer terim doküman içinde varsa 1 yoksa 0 değerini terim ağırlığı olarak belirlenir.

$$w_i = \begin{cases} 1 & if \quad tf_i > 0 \\ 0 & otherwise \end{cases}$$

tf_i: terim frekansı

- Terim Frekansı Ağırlığı: Bu yaklaşımda son derece basit bir yaklaşımdır. Bu yaklaşımda doküman içinde terimin geçme frekansına terim ağırlığını belirler.

$$w_i = tf_i$$

- Terim Frekansı × Ters Doküman Frekansı Ağırlığı: İlk iki yaklaşım sadece bir dokümanı göz önüne alarak ağırlık değerini hesaplarlar. Aslında bir makine öğrenmesi yaklaşımında genelde birden fazla doküman vardır. Bu yaklaşım diğer dokümanları da göz önüne alarak bir terim ağırlığı hesaplama yoluna gider. Bu değer hesaplanırken doküman içindeki geçme frekansı terimin görüldüğü tüm dokümanları ters orantısı ile çarpılarak bulunur. Salton, Wong ve Yang (1975) tarafından önerilen klasik vektör modeli terim ağırlığı belirleme denklemi:

$$w_i = tf_i \cdot \log \frac{N}{N_i}$$

N: toplam doküman sayısı

N_i: terimi içeren doküman sayısı

$\log \frac{N}{N_i}$: doküman frekansını tersi

Bu yaklaşım diğer dokümanların göz önüne alınması terimin ağırlık değerini az da olsa etkilemektedir.

- Uzunluk Normalizasyon ile Terim Frekansı × Ters Doküman Frekansını: Bu yaklaşım doküman uzunluklarını göz önüne alan bir yaklaşımdır. Amaç her doküman vektörünün uzunluklarını normalizasyon işlemini yapıp terim ağırlık değerini hesaplamaktır. Salton ve Buckley (1988) göre denklemi:

$$w_i = \frac{tf_i \cdot \log \frac{N}{N_i}}{\sqrt{\sum_{j=1}^{|T|} \left[tf_i \cdot \log(\frac{N}{N_i}) \right]^2}}$$

2.4.3.5. Boyutsal İndirgeme

Makine öğrenmesinde boyutsal indirgeme çok boyutlu uzayı az daha miktarda uzaya indirgeme işlemidir. Boyutsal indirgeme sayesinde daha kesin sonuçlar alınabilecek bir uzaya ulaşılabilir. En genel boyutsal indirgeme teknikleri:

- Bilgi Kazancı
- Karşılıklı Bilgi
- Ki-Kare Testi
- Terim Kuvveti
- Doküman Frekansı

2.4.3.5.1. Bilgi Kazancı (Information Gain)

Bilgi Kazancı (BK), kategori tahmini için kazanılan bilgi bitlerinin sayısının hesaplanmasıdır. Bu yöntemde BK; muhtemel kategoriler kümesi $\{c_1, c_2, ..., c_m\}$ olmak üzere her terim *t* için aşağıdaki formülle hesaplanır.

$$BK(t) = -\sum_{i=1}^{m} P(c_i) \cdot \log P(c_i) + P(t) \cdot \sum_{i=1}^{m} P(c_i|t) \cdot \log P(ci|t) + P(\bar{t}) \cdot \sum_{i=1}^{m} P(c_i|\bar{t}) \cdot \log P(ci|\bar{t})$$

$P(c_i)$: c_i .nci kategorinin olasılık değeri

$P(t)$: *t* terimin olasılık değeri

$P(\bar{t})$: *t* terimini içermeme olasılığı

Terimlerin BK değerlerine bakarak belli bir eşik değerinin altında kalan terimler özellik uzayından çıkarılabilirler.

2.4.3.5.2. Karşılıklı Bilgi (Mutual Information)

Karşılıklı Bilgi (KB), kelime ilişkileri istatistiksel olarak modellenmesi çok sık olarak kullanılan bir yöntemdir [Manning ve Schutze, 1999]. Bu yöntemin denklemi:

$$KB(t, c) = \log P(t|c) - \log P(t)$$

Terimlerin KB değerlerine bakılarak belli eşiğin altında kalan terimler elenmiş olur. Fakat bu formülle özellikle yüksek frekanslı değerler özellik uzayı içinde kalırken bazı umulmayan terimlerde özellik uzayı içinden çıkmış veya eklenmiş olabilir. Bu

sebepten dolayı KB değeri hesaplanırken farklı matematiksel hesaplara gidilebilir. Örneğin:

$$KB_{avg}(t) = \sum_{i=1}^{m} P(c_i) \times KB(t, c_i) \text{ veya,}$$

$$KB_{\max}(t) = \max_{i=1}^{m} \{KB(t, c_i)\}$$

Bu farklı matematiksel hesaplar ile farklı terim uzayları elde edilebilir.

2.4.3.5.3. Ki-Kare Testi (Chi-Square)

Ki-Kare testi bir ilişki ölçümü için kullanılır. İstatistiksel formülü:

$$\chi^2 = \sum_i \sum_j \frac{(f_{ij} - \hat{f}_{ij})}{\hat{f}_{ij}}$$

f_{ij} : i satır ve j sütundaki gözlenen frekans

$\hat{f}_{ij}$: i satır ve j sütunun beklene frekansı

Makine öğrenmesinde ise χ^2 belirli bir terimin ve belirli bir kategorinin bağımlılık derecesinin ölçülmesidir. Doküman sınıflama görevinde kullanılan χ^2 Yang ve Pedersen (1997), Ng ve ark. (1997), Spitters (2000) tarafından önerilen formül:

$$\chi^2(t, c) = \frac{N \times (A_{ct} \cdot D_{ct} - C_{ct} \cdot B_{ct})^2}{(A_{ct} + C_{ct}) \times (B_{ct} + D_{ct}) \times (A_{ct} + B_{ct}) \times (C_{ct} + D_{ct)}}$$

Bu formül 2×2 bir olasılık tablosuna göre oluşturulmuştur. Bu tabloda ilk satır belirli bir terimi (t) içeren doküman sayısını, ikinci satır belirli bir terim içermeyen doküman sayısını, birinci sütun belli bir kategoriye (c) ait olan doküman sayısını ve ikinci sütun ise belli bir kategoriye ait olmayan doküman sayısını içerir. Formüle göre:

A_{ct} : belirli bir kategori (c) ve belirli bir terimi (t) içeren doküman sayısı

B_{ct} : belirli bir kategori dışında belirli bir terimi içeren doküman sayısı

C_{ct} : belirli bir terim dışında belirli bir kategoriyi içeren doküman sayısı

D_{ct} : belirli bir kategori ve terim dışındaki doküman sayısı

N : toplam doküman sayısını

χ^2 testine bağlı olarak iki farklı ölçüm formülü daha hesaplanabilir:

$$\chi^2_{avg}(t) = \sum_{i=1}^{m} P(c_i) \times \chi^2(t, c_i) \text{ veya}$$

$$\chi^2_{\max}(t) = \max_{i=1}^{m} \{\chi^2(t, c_i)\}$$

Bu yöntemde belli bir eşik değeri altındaki χ^2 değerleri elenir.

2.4.3.5.4. Terim Kuvveti (Term Strength)

Terim Kuvveti yöntemi, yakından ilişkili dokümanlarda bir terimin muhtemel görünmesini göz önüne alarak terimin önemi tahmin edilir.

$$TK(t) = P(t \in d_j \,|\, t \in d_j)$$

İlk aşamada bu yöntemde, bir eşik değerini geçen ikili dokümanları bulmak için dokümanların bir eğitim kümesi kullanılır. İkinci aşamada ilk dokümanda bulunan bir

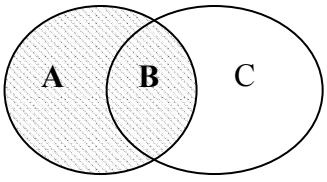
terimin ikinci dokümanda koşullu bulunma olasılığı temel alınarak *TK* hesaplanır. d_i ve d_j ilişkili farklı iki doküman kümesidir.

2.4.3.5.5. Doküman Frekansı (Document Frequency)

Doküman frekansı, bir terimin görüldüğü doküman sayısıdır. Bu yöntemde de, eşik değerinin altında kalan terimler elenir. Bu yöntem seyrek terimlerin bulunmasında kullanılabilir. Ancak, diğer yöntemlere göre az kullanılan bir yöntemdir. Yang ve Pedersen (1997) bu yöntemin metin sınıflama üzerine incelemişlerdir. Yang ve Pedersen göre farklı yöntemlerle karşılaştırıldığında hesaplama karmaşıklığı bakımından en sade ve en etkili yöntem olduğu saptanmıştır.

2.5. Performans Ölçümleri

Dokümanların değerlendirilmesinde performansını ölçmek için genelde bütünsel oranlı doğruluk (recall), kısmi oranlı doğruluk (precision) ve F-ölçümü (F-Measure) kavramları kullanılır [van Rijsbergen, 1979] [Manning ve Schutze, 1999].



A: İlgili dokümanlar (erişilmeyenler)
B: İlgili dokümanlar (erişilenler)
C: İlgisiz dokümanlar (erişilenler)

Bütünsel oranlı doğruluk (B.O.D.), erişilen ilgili doküman sayısının veritabanındaki toplam ilgili kayıtların sayısı oranıdır.

$$B.O.D. = \frac{A}{A+B} \times 100\%$$

Kısmi oranlı doğruluk (K.O.D.), erişilen ilgili doküman sayısının erişilen toplam ilgili ve ilgisiz kayıtlara oranıdır.

$$K.O.D. = \frac{A}{A+C} \times 100\%$$

Bütünsel oranlı doğruluk ve kısmi oranlı doğruluğun harmonik ortalamasına F-ölçümü denir.

$$F = \frac{2 \times B.O.D. \times K.O.D.}{B.O.D. + K.O.D.}$$

2.6. Gözetimsiz öğrenme

Gözetimsiz öğrenme, etiketlenmemiş dokümanlar üzerinden hiçbir ek bilgi ve aracı olmadan dokümanı kümelemeye çalışan makine öğrenmesi tekniğidir. Kümelemenin amacı benzer veri parçalarını sınıflayarak veya gruplayarak veri miktarını azaltmaya çalışmaktır. Kümeleme için geliştirilen algoritmalar otomatik olarak kategori ve sınıflar oluşturmamızı sağlar [Jardine ve Sibson, 1971] [Sneath ve Sokal, 1973]. Geliştirilen kümeleme metotları sayesinde insan faktörü minimize edilmiş olur. Kümeleme teknikleri kısımlara ayırma, hiyerarşik metotlar ve hipotez testleri olmak üzerine üçe ayırabiliriz [Anderberg, 1973] [Hartigan, 1975] [Jain ve Dubes, 1988] [Jardine ve Sibson, 1971] [Sneath ve Sokal, 1973] [Tryon ve Bailey, 1973].

- Kısımlara Ayırarak Öğrenme
  - Single Pass Algoritması
  - K-Means Algoritması
- Hiyerarşik Öğrenme
- Hipotez Testleri

2.6.1. Kısımlara ayırma metodu

Kısımlara Ayırma, veri seti üzerinden ayrıştırma işlemi yaparak ayrık kümeler elde etmeye çalışır. Algoritmayı minimize eden bir ölçüt fonksiyonuna sahiptir. Bu fonksiyon sayesinde kümeleme işlemini yerine getirmeye çalışır.

Kısımlara ayırma metodu üzerine çeşitli algoritmalar geliştirilmiştir. Alt bölümlerde çok bilinen Single Pass ve K-Means Algoritmaları hakkında bilgi verilecektir.

2.6.1.1. Single Pass Algoritması

En temel kısımlara ayırma işlemi Single Pass yöntemi olarak bilinir [Hill, 1968]. Bu yöntem aşağıdaki adımları içerir:

1. İlk nesneyi ilk kümenin kitle merkezi yap.
2. Diğer nesneye geç ve nesne için benzerlik değeri hesapla.
3. Benzerlik değeri belli bir eşik değerlerini geçerse yeni bir küme oluştur. Eğer eşik değeri var olan bir nesneye yakın çıkarsa bu nesneye ekleme yap. Eğer nesneler bitmediyse Adım 2 ye geri dön.

Single Pass Algoritması Örneği

Dokümanları (*d*) ve terimleri (*t*) olan bir küme olsun.

	t1	*t2*	*t3*	*t4*	*t5*
d1	1	2	0	0	1
d2	3	1	2	3	0
d3	3	0	0	0	1
d4	2	1	0	3	0
d5	2	2	1	5	1

Tablo 2.1. Single pass algortiması için veriler

İlk terim ait değerler kitle merkezi (*c1*) olarak belirlenir. Bu durumda,

$$c1 = <1, 3, 3, 2, 2>$$

değerlerinden oluşur. Bu aşamadan sonra *t2* geçilir. *t2* değerleri *c1* değerleri ile benzerlik karşılaştırması yapan bir fonksiyona gönderilir. Bu fonksiyon bir eşik değeri ile karşılaştırılır. Eğer eşik değerini aşarsa yeni bir kitle merkezi açılır, aşmıyorsa terim aynı kitle merkezine eklenir. Örneğin eşik değeri 10 olsun.

$$fonk(t2, c1) = 1*2 + 1*3 + 0*3 + 1*2 + 2*2 = 11$$

fonk(t2, c1) = 11 > 10 olduğuna göre c1 kümesine eklenir. Bu durumda yeni *c1*:

$$c1 = <3/2, 4/2, 3/2, 3/2, 4/2>$$

fonk(t3, c1) = t3 * c1 = 6 < 10 yeni bir kitle merkezi açılacağı anlamına gelir. (*c2*)

$$c1 = \{t1, t2\}$$
$$c2 = \{t3\}$$

Bu aşamadan sonra yeni gelen terim iki kitle merkez ile de karşılaştırılır.

$$fonk(t4, c1) = <0, 3, 0, 3, 5> . <3/2, 4/2, 3/2, 3/2, 4/2>$$
$$= 0 + 12/2 + 0 + 9/2 + 20/2 = 20.5$$

$$fonk(t4, c2) = <0, 3, 0, 3, 5> . <0, 2, 0, 0, 1>$$
$$= 0 + 6 + 0 + 0 + 5 = 11$$

Her ikiside 10'dan büyük bu durumda denklemi maksimum yapan değer seçilir. Yani t4, c1 kümesi içine eklenir:

$$c1 = \{t1, t2, t4\}$$
$$c2 = \{t3\}$$

Bu durumda *c2* nin kitle merkezi değerleri değişmez.

$$c2 = <0, 2, 0, 0, 1>$$

ama *c1* değeri ekleme yapıldığı için değişir.

$$c1 = <3/3, 7/3, 3/3, 6/3, 9/3>$$

t5 ekleme işleminde yine iki fonksiyona bakılır.

$$fonk(t5, c1) = <1, 0, 1, 0, 1> . <3/3, 7/3, 3/3, 6/3, 9/3>$$
$$= 3/3 + 0 + 3/3 + 0 + 9/3 = 5$$

$$fonk(t5, c2) = <1, 0, 1, 0, 1> . <0, 2, 0, 0, 1>$$
$$= 0 + 0 + 0 + 0 + 1 = 1$$

Her iki fonksiyonda eşik değeri (10) geçmeyi başaramamıştır. Bu durumda yeni bir kitle merkezi yani küme açılır:

$$c1 = \{t1, t2, t4\}$$
$$c2 = \{t3\}$$
$$c3 = \{t5\}$$

2.6.1.2. K-Means Algoritması

En popüler kısımlara ayırma algoritması ise *K-means* algoritmasıdır [MacQueen, 1967]. K-means verilen bir veri seti üzerinden belirli sayıda kümeyi (*k* adet) sınıflamak için geliştirilmiş en sade ve basit algoritmadır. Bu yöntemin adımları:

1. K adet noktayı K adet nesne ile eşleştir ve başlangıç kitle merkezlerini belirle.
2. Her nesneyi en uygun gruba ata ve K kitle merkezini hesapla.
3. Her obje için, tekrar K adet kitle merkezini hesapla.
4. Yeni oluşan grubu geçmişteki grup ile kıyasla. Eğer aynı ise işleme devam et. Fakat farklı ise işlemi sonlandır.

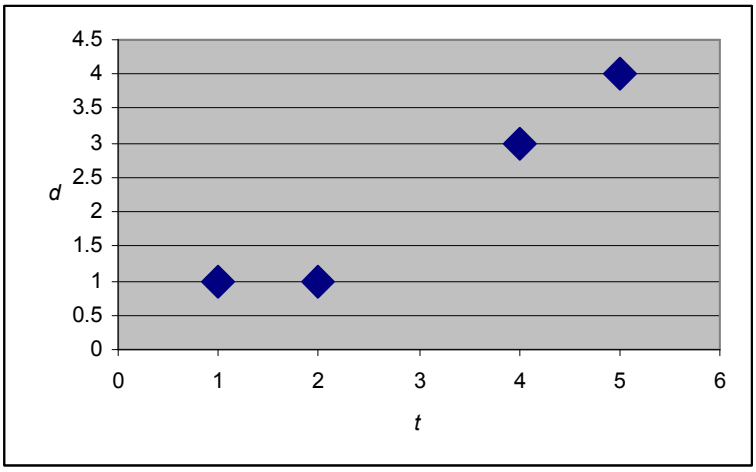
K-Means Algoritması Örneği

Dokümanları (*d*) ve terimleri (*t*) olan bir küme ve K değerimiz 2 olsun.

	d1	*d2*
T1	1	1
T2	2	1
T3	4	3
T4	5	4

Tablo 2.2. K-Means algoritması verileri

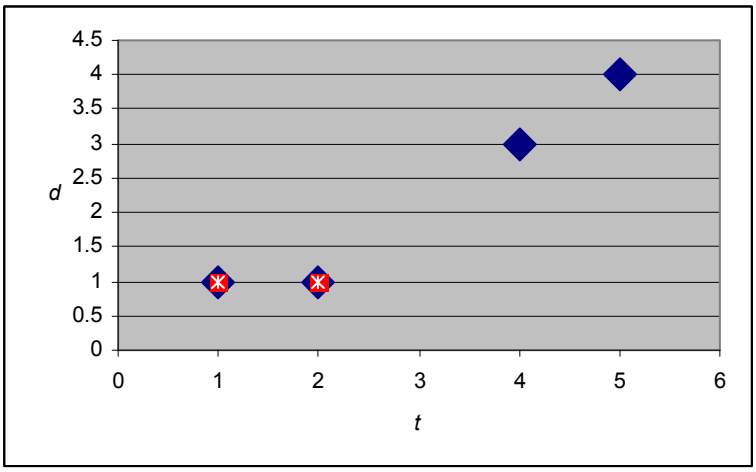
Bu tabloyu (X,Y) koordinat ekseni üzerine yerleştirelim.



Grafik 2.1. K-Means algoritması ilk durum

Adım 1.

Öncelikle başlangıç kitle merkezleri belirlenmeli. Tablodaki ilk iki değer kitle merkezi (*c1* = (1, 1) ve *c2* = (2,1)) olsun.



Grafik 2.2. K-Means algoritması adım 1'den sonra

Adım 2.

Bu adımda koordinatlar arasındaki uzaklık değerleri hesaplanır. Uzaklık değerlerini hesaplamak için *öklit uzaklık* formülünü kullanabiliriz.

$$Fark_e = \sqrt{\sum_{i=1}^{n} (t_i + d_i)^2}$$

Öncelikle ana tablodaki değerlerini kullanıp ilk matrisi oluşturalım.

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \leftrightarrow \begin{matrix} t \\ d \end{matrix}$$

Ardından öklit uzaklık formülünü kullanarak yeni matrisi oluşturalım. Bu matrisi oluşturmak için *c1* ve *c2* değerleri ile c_n= (4, 3) ve c_m=(5, 4) arasındaki öklit uzaklık değerlerini hesaplayalım. Örneğin uzaklık değerleri

c1 ve c_n arasında: $\sqrt{(4-1)^2+(3-1)^2} = 3.61$ veya

c2 ve c_n arasında: $\sqrt{(4-2)^2+(3-1)^2} = 2.83$ şeklinde hesaplanabilir.

Uzaklık değerleri hesaplandıktan sonra oluşan matris: (İterasyon 0 = I_0)

$$I_0 = \begin{bmatrix} 0 & 1 & 3.61 & 5 \\ 1 & 0 & 2.83 & 4.24 \end{bmatrix}$$

Adım 3.

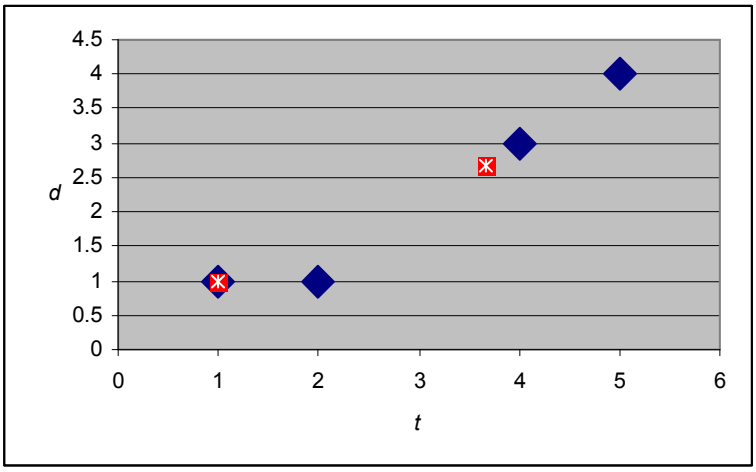
Matris oluştuktan sonra minimum değerlere 1 ve maksimum değerleri 0 diyerek bir grup matrisi oluşturalım. Grup (G_0) matrisine göre:

$$G_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \leftrightarrow \begin{matrix} grup1 \\ grup2 \end{matrix}$$

Grup1 için *c1* = (1, 1) alınabilir. Fakat grup2 üç elemandan oluşmaktadır. *c2* bulmak için bu üç elemanın ortalaması alınır. Bu durumda *c2* hesaplanırsa:

$c2 = (\frac{2+4+5}{3}, \frac{1+3+4}{3}) = (3.67, 2.67)$ olur.

Yeni oluşan değerleri (XY) koordinat sistemine yerleştirelim:



Grafik 2.3. K-Means algoritması adım 3'ten sonra

Adım 4.

Adım 2 teki işlemler aynen tekrarlanır. Yani yeni oluşan c1 ve c2 değerleri ana tablodaki değerlerle uzaklıkları tekrar hesaplanır.

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \leftrightarrow \begin{matrix} t \\ d \end{matrix} , c1 = (1, 1) , c2 = (3.67, 2,67)$$

Bu değerleri kullanarak hesaplanan uzaklık matrisi (İterasyon 1 (I_1)):

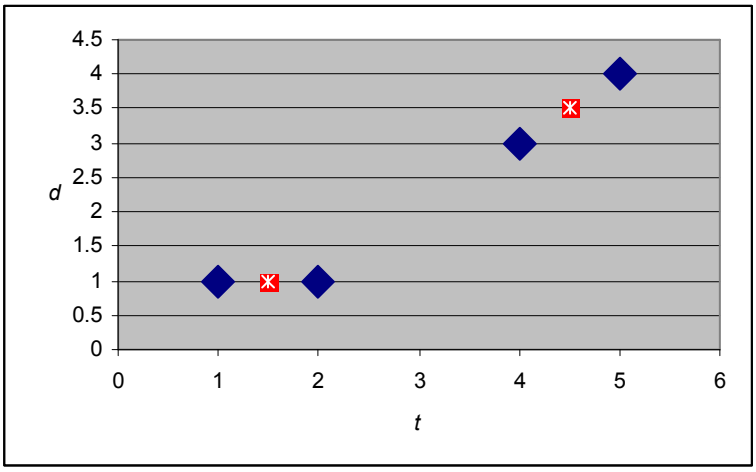
$I_1 = \begin{bmatrix} 0 & 1 & 3.16 & 5 \\ 3.14 & 2.36 & 0.47 & 1.89 \end{bmatrix}$ ve bu matrisi adım 3 teki gibi gruplanırsa (G_1):

$$G_1 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \leftrightarrow \begin{matrix} grup1 \\ grup2 \end{matrix}$$

Bu gruplama matrisine göre yeni oluşan *c1* ve *c2* değerleri:

$$c1 = (\frac{1+2}{2}, \frac{1+1}{2}) = (1.5,1) , \; c2 = (\frac{4+5}{2}, \frac{3+4}{2}) = (4.5,3.5)$$

Yeni oluşan *c1* ve *c2* değerlerine göre (X, Y) koordinat sistemi:



Grafik 2.4. K-Means algoritması son durum

Adım 5.

Adım 2 ve Adım 3 yeni oluşan değerlere göre tekrar hesaplandığında:

$$\begin{bmatrix} 1 & 2 & 4 & 5 \\ 1 & 1 & 3 & 4 \end{bmatrix} \leftrightarrow \begin{matrix} t \\ d \end{matrix} \text{ , } c1 = (1.5,\ 1) \text{ , } c2 = (4.5,\ 3{,}5)$$

Bu değerleri kullanarak hesaplanan uzaklık matrisi (İterasyon 2 (I_2)):

$I_2 = \begin{bmatrix} 0.5 & 0.5 & 3.20 & 4.61 \\ 4.30 & 3.54 & 0.71 & 0.71 \end{bmatrix}$ ve bu matrisi adım 3 teki gibi gruplanırsa (G_2):

$$G_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \leftrightarrow \begin{matrix} grup1 \\ grup2 \end{matrix}$$

Adım 6.

G_2 ve G_1 matrislerinde değişme olmaması durumu *c1* ve *c2*'nin en iyi değerlerine ulaşıldığını göstermektedir. Bu adımda işlem sonlandırılır.

2.6.2. Hiyerarşik kümeleme

Hiyerarşik Kümeleme, küçük kümeleri büyükleri ile birleştirme veya büyük kümeleri bölme yoluyla ilerler. Kümeleme metodu kurallara bakarak bu birleştirme veya bölme işlemini yapar. İşlem sonunda tüm nesnelerin ilişkilerini gösteren bir ağaç yapısı oluşur [Johnson, 1967]. Bu yöntemin adımları:

1. Her nesneyi bir küme olarak ata. (Bu durumda N tane nesne için N adet küme oluşur.) $N \times N$ bir matris oluştur.
2. Matrisi kullanarak nesneler arası uzaklık değerlerini hesapla. Yeni oluşan matris üzerinden en fazla benzeyen nesneleri grupla.
3. (N-1) gelininceye kadar gruplanan nesnelere bağlı yeni matris oluştur. Adım 2. ye geri dön.

Hiyerarşik Kümeleme Örneği

Dokümanları (*d*) ve terimleri (*t*) içeren bir küme olsun.

	d1	*D2*	*d3*
t1	9	3	1
t2	10	2	9
t3	1	9	4
t4	6	5	5
t5	1	10	3

Tablo 2.3. Hiyerarşik kümeleme verileri

Öncelikle terimlerin birbirine yakınlıklarını verecek Uzay Matrisi oluşturulur. Bu matris oluşturulurken terimler arasındaki *öklit uzaklık* formülünü kullanılabilir.

$$Fark_e = \sqrt{\sum_{i=1}^{n} (t_i + d_i)^2}$$

Örneğin *t1* ile *t2* ve *t3* ile *t4* arasındaki ilişki hesaplanırken;

$t1 \Rightarrow \begin{bmatrix} 9 \\ 3 \\ 1 \end{bmatrix}$, $t2 \Rightarrow \begin{bmatrix} 10 \\ 2 \\ 9 \end{bmatrix}$ iken öklit uzaklık formülü: $\Rightarrow \sqrt{(9-10)^2 + (3-2)^2 + (1-9)^2}$

$t3 \Rightarrow \begin{bmatrix} 1 \\ 9 \\ 4 \end{bmatrix}$, $t4 \Rightarrow \begin{bmatrix} 6 \\ 5 \\ 5 \end{bmatrix}$ iken öklit uzaklık formülü: $\Rightarrow \sqrt{(1-6)^2 + (9-5)^2 + (4-5)^2}$

Bu şekilde tüm terimler arasındaki ilişkiler hesaplanırsa aşağıdaki tablo elde edilir:

	t1	*t2*	*t3*	*t4*	*t5*
T1	0	2.5	10.44	4.12	11.75
T2	-	0	12.5	6.4	13.93
T3	-	-	0	6.48	1.41
T4	-	-	-	0	7.35
T5	-	-	-	-	0

Tablo 2.4. Hiyerarşik kümeleme başlangıç

Şimdi terimler arasındaki ilişkilerin saptanmasına sıra geldi. Adım adım terimlerin ilişkilerini inceleyelim.

Adım 1.

En az fark 1.41 ile *t3* - *t5* arasındadır.

c1 = {*t3, t5*}

Adım 2.

Küme belirlendikten sonra tablodaki değerler birleştirilmelidir. *t3 - t5* birleştirilirken ana tablodaki değerlerin ortalamaları alınır. Yani *t3 - t5* yeni değerleri = [1, 9.5, 3.5] olur. Yeni değerlere göre tablonun *t3 - t5* değerleri tekrar hesaplanır. Bu bilgiye göre birleştirilmiş tablo:

	t1	*t2*	*t4*	*t3 - t5*
t1	-	2.5	4.12	10.9
t2		-	6.4	9.1
t3			-	6.9
t3 - t5				-

Tablo 2.5. Hiyerarşik kümeleme adım 2

Adım 3.

Tekrar adım 1 deki işlem tekrarlanır. Yani bu sefer 2.5 değerine sahip *t1 – t2* yeni bir küme oluşturur.

$$c1 = \{t3, t5\}$$
$$c2 = \{t1, t2\}$$

Adım 4.

	t1 - t2	*t4*	*t3 - t5*
t1 - t2	-	3.7	9.2
t4		-	6.9
t3 - t5			-

Tablo 2.6. Hiyerarşik kümeleme son adım

Yine yakınlık değerlerine bakıldığında *t4*, t1 – t2 ile daha yakın olduğu görülmektedir. Bu durumda:

$$c1 = \{t3, t5\}$$
$$c2 = \{\{t1, t2\},t4\}$$

2.6.3. Hipotez testleri

Hipotez bilinmeyen kütle parametreleriyle ilgili olarak bu kütleden seçilen bir örnek grubu aracılığıyla yapılan varsayım olarak tanımlanır. Örnekleme sonucu elde edilen değerler kullanılarak, ana kütlenin kabulü ya da reddi yapılır. Ancak hipotezlerin test edilmeleri için önce kurulmaları gerekmektedir ve doğru hipotez kurmak çok önemli bir konudur. Hipotez, bir parametrenin değeri veya değerleri ile ilgili olarak kurulur. Bu açıdan ele alındığında, genel olarak iki tip Hipotez kurulması söz konusudur.

1. Ana kütlenin belirli bir frekans fonksiyonu olduğu varsayılır.
2. Ana kütleyi karakterize eden belirli bir değerin olduğu varsayılır ki, kalite kontrolünde kullanılan hipotezler bu türdendir.

Hipotez kurulduktan sonra rasgele bir örnekleme yapılır. Bu örneklemeden elde edilen değerlerden yararlanılarak red hipotezi kabul veya reddedilir. Bu işleme test işlemi adı verilir. Öyleyse hipotez testi sıfır hipotezinin kabulü veya reddi için ortaya konulan bir karar kuralıdır.

Test işleminde karşıt nitelikte iki hipotez vardır.

H0 =Sıfır Hipotezi
H1 =Alternatif Hipotez

Örnek grubundan alınan istatistiksel ölçümlerin her zaman ana kütlenin istatistiksel ölçülerine eşit olması beklenemez. Dolayısıyla, örnek grubuna dayanarak elde edilecek sonuçlara göre alınan karar bir hata içerebilir. Alınan karara ve kütle parametresinin gerçek değerine bağlı olarak iki tip hata yapılabilir.

- Birinci Tip Hata
- İkinci Tip Hata

Bilinmeyen ana kütle parametresinin gerçek değeri karşısında ya H0 ya da H1 doğrudur. Öyleyse H0 hipotezi ya doğrudur ya da yanlıştır, ya kabul edilecek ya da reddedilecektir. Doğruluk ve yanlışlık bir kriter, kabul ya da red ikinci kriter olduğunda dört durum ortaya çıkar.

1. H0 hipotezi ana kütle parametresinin gerçek değeri karşısında doğru ise ve örnekleme sonucunda H0 kabul edilmişse bir hata söz konusu değildir.
2. H0 hipotezi ana kütle parametresinin karşısında doğru ise fakat örnekleme sonucunda reddedilmişse bir yanlışlık yapılmıştır. Buna birinci tip hata denir ve α ile gösterilir.
3. H0 hipotezi ana kütle parametresinin gerçek değeri karşısında yanlış ise ve örnekleme sonucunda reddedilmişse, bir hata söz konusu değildir.
4. H0 hipotezi ana kütle parametresinin gerçek değeri karşısında yanlış ise ve örnekleme sonucunda kabul edilmişse, bir yanlışlık yapılmıştır. Buna ikinci tip hata adı verilir ve β ile gösterilir.

Birinci tip hata (α) üretici riski ya da satıcı riski olarak adlandırılır. Genellikle α = 0,05 kabul edilir ve α değerine testin anlamlılık düzeyi adı verilir.

İkinci tip hatalar (β) tüketici riski ya da alıcı riski olarak adlandırılır ve genellikle β = 0,10 kabul edilir.

H0 hipotezini reddederken ne ölçüde hatanın göze alındığı testin anlamlılık düzeyini belirler, sıfır hipotezinin red olasılığı da testin kuvvetini gösterir [Gümüşoğlu, 2000].

5. bölümünde alt öğeleme listesi bulmada en çok kullanılan hipotez testlerinden maksimum benzerlik ve T-Puanı testleri üzerinde durulacaktır. Ayrıca Briscoe and Carroll'ın önerdiği öğrenme tekniği de bu bölümde anlatılacaktır.

2.7. Gözetimli öğrenme

Gözetimli öğrenme yönteminde kategori yapısının ve verilerin hiyerarşik düzeninin önceden bilindiği varsayılır. Öncelikle etiketlenmiş bir eğitim kümesine ihtiyaç vardır. Bu küme üzerinden çeşitli öğrenme algoritmaları kullanılıp bilinen sınıf etiketleri ile eşleştirme yapılır. Daha sonra, değişik veri setleri kullanılarak bu eşleşmenin başarısı ölçülmeye çalışılır. Bu bölümde 6 farklı gözetimli öğrenme yaklaşımı incelenecektir.

- K - En Yakın Komşu Algoritması
- Karar Verme Ağaçları
- Naive Bayes Sınıflandırıcı
- Bayesian.Net Sınıflandırıcı
- Yapay Sinir Ağları
    - Çok katmanlı Perceptrons
- Destek Karar Makinesi

2.7.1. K - En yakın komşu algoritması

K-En yakın komşu algoritması bir örnek tabanlı öğrenme veya tembel öğrenme tipidir. Veri madenciliği, istatistiksel örüntü tanıma, resim işleme ve birçok alanda kullanılan gözetimli öğrenme yöntemlerinden biridir. Özellikle el yazısı tanıma, uydu resimleri ve EKG (electrocardiogram) gibi uygulama alanlarında başarılı bir şekilde kullanılmaktadır.

K-En yakın komşu sınıflandırılması eğitim örnekleri taban alan nesneleri sınıflamak için geliştirilmiş bir algoritmadır. Bu algoritmada, eğitim örnekleri çok boyutlu bir özellik uzayında eşleştirilir. Uzay, eğitim örneklerin sınıf etiketleri tarafından bölgelere ayrılır. Uzaydaki bir nokta en uygun sınıfa atanır. Bu atama işleminde genellikle daha önce kullandığımız öklit uzaklık formülü kullanılır [Dasarathy, 1991] [Mäkelä ve Pekkarinen, 2004] [Franco-Lopez, 2001].

Algoritma eğitim evresi ve gerçek sınıflama evresi olmak üzere ikiye ayrılır. Algoritmanın eğitim evresi özellik vektörlerinin ve eğitim örneklerinin sınıf etiketlerin depolanmasını içerir. Gerçek sınıflama evresinde, örnek vektörler eğitim örnekleri üzerinden hesaplanır. Yeni vektörün tüm depolanan vektörlere uzaklığı hesaplanır ve k en yakın örnekler seçilir. Yani, yeni noktanın uygun sınıfı bulunmaya çalışır.

Algoritma k değeri örnek verilere bağlı olarak belirlenir. Özellik k'nın yüksek seçilmesi sınıflamadaki gürültünün temizlenmesi açısından önemlidir. Bu seçim işlemi için çapraz doğrulama gibi parametre optimizasyon teknikleri kullanılabilir. $k = 1$ en yakın komşu algoritması olarak adlandırılır.

Algoritmanın geliştirilmesi son derece basittir, ama özelikle eğitim boyutunun büyümesi yoğun matematiksel işlem gerektirir. Özellikle bu algoritma uzaklık hesaplama sayısını düşürmeye yönelik birçok optimizasyon çalışması yapılmıştır. Bu çalışmaların bazıları:

- Doğrusal Tarama
- Kd-trees
- BallTrees
- Metrik Trees
- Locality-sensitive hashing (LSH)
- Toplayıcı En Yakın Komşu

Bu algoritma beş adımdan oluşur:

1. Öncelikle k değeri belirlenir.
2. Hedef noktaya olan öklit uzaklıkları hesaplanır.

3. Uzaklıklar sıralanır ve en minimum uzaklığa bağlı olarak en yakın komşular bulunur.
4. En yakın komşu kategorileri toplanır.
5. En uygun komşu kategorisi seçilir.

K-En Yakın Komşu Sınıflandırılması Örneği

İki adet terim ve bu terimlerin sonuçlarına bağlı yanlış/doğru değerlerini alan bir sınıf değerimiz olsun ve eklenecek nokta olarak ta (3, 7) değerlerini seçelim.

t1	*t2*	*Sınıflama*
7	7	Sınıf_1
7	4	Sınıf_1
3	4	Sınıf_2
1	4	Sınıf_2

Tablo 2.7. K-En yakın veriler

Adım 1.

k değeri 3 olarak seçilir.

Adım 2.

Eklenecek noktaya bağlı öklit uzaklıklarını hesaplayalım.

$$Fark_e = (t_{1i} - t_1)^2 + (t_{2i} - t_2)^2$$

t1	*t2*	*(3, 7) koordinatları için Öklit Uzaklıkları*
7	7	$(7-3)^2 + (7-7)^2 = 16$
7	4	$(7-3)^2 + (4-7)^2 = 25$
3	4	$(3-3)^2 + (4-7)^2 = 9$
1	4	$(1-3)^2 + (4-7)^2 = 13$

Tablo 2.8. K-En yakın adım 2

Adım 3.

Uzaklıklar sıralanır ve en minimum uzaklığa bağlı olarak en yakın komşular bulunur.

t1	*t2*	*(3, 7) koordinatları için Öklit Uzaklıkları*	*Sıralama*	*3-En Yakın Komşu*
7	*7*	$(7-3)^2+(7-7)^2=16$	***3***	Evet
7	*4*	$(7-3)^2+(4-7)^2=25$	***4***	Hayır
3	*4*	$(3-3)^2+(4-7)^2=9$	***1***	Evet
1	*4*	$(1-3)^2+(4-7)^2=13$	***2***	Evet

Tablo 2.9. K-En yakın adım 3

Adım 4.

En yakın komşu kategorileri toplanır.

t1	*t2*	*(3, 7) koordinatları için Öklit Uzaklıkları*	*Sıralama*	*3-En Yakın Komşu*	*Sınıflama*
7	7	$(7-3)^2+(7-7)^2=16$	3	Evet	Sınıf_1
7	4	$(7-3)^2+(4-7)^2=25$	4	Hayır	-
3	4	$(3-3)^2+(4-7)^2=9$	1	Evet	Sınıf_2
1	4	$(1-3)^2+(4-7)^2=13$	2	Evet	Sınıf_2

Tablo 2.10. K-En yakın adım 4

Adım 5.

En uygun komşu kategorisi seçilir. Yukarıdaki duruma göre 2 adet sınıf 2 var iken 1 adet sınıf 1 vardır. Bu durumda (3, 7) noktaları için sınıf_2 tercih edilir.

Avantajları

- Özellikle gürültü içeren verilerde çok olumlu sonuçlar verir

- Eğitim kümesi büyük olduğunda çok etkilidir

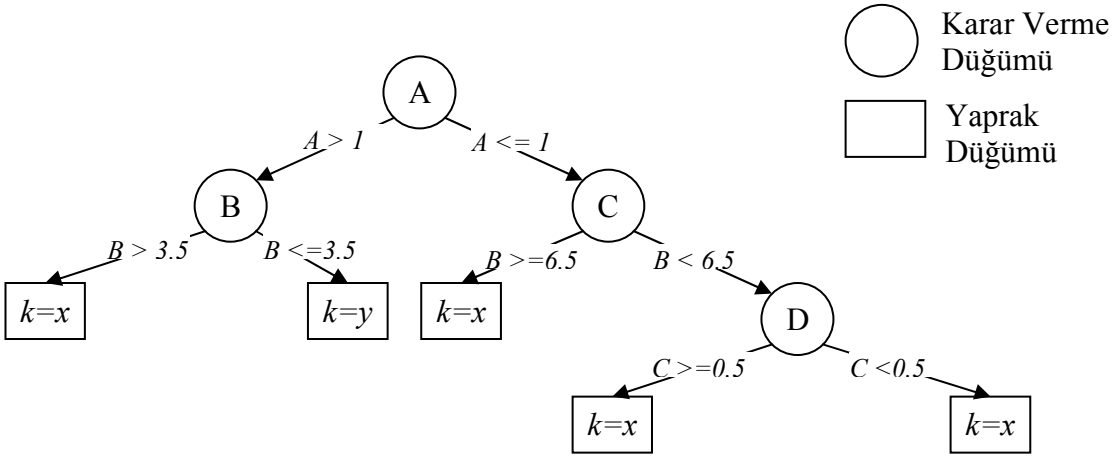
Dezavantajları

- *k* değerini belirlemek gerekir
- İyi sonuçlar üretmek için kullanılan uzaklık tipine ve kullanılan özelliklere bağlı olarak *uzaklık tabanlı öğrenme* çok açık değildir. Sadece belirli özellikler ön plana çıkarır.
- Tüm eğitim örnekleri için uzaklık hesaplandığı için hesaplama maliyeti çok yüksektir. Kd-Tree gibi indekslemeye sahip algoritmalarla bu hesaplama maliyeti düşürülebilir.

2.7.2. Karar verme ağaçları

Karar verme ağaçları, verileri ağaç yapısı formunda gösteren bir sınıflayıcıdır. Her ağaç düğümü en azından bir yaprak düğümü veya bir karar verme düğümünden oluşur (Şekil 2.3.): [Mitchell, 1997] [Winston, 1992]

- *Yaprak düğümü*: Örneğin hedef özelliğinin değerini gösterir.
- *Karar verme düğümü*: Bir tek özellik değerini gerçekleştirmeyi bazı testlerle belirler. Bu testler dallanma ve alt ağaçları ortaya çıkarır.



Şekil 2.3. Basit bir karar ağacı modeli

Karar verme ağacı, ağacın kökünden başlayarak ve yaprak düğümlere doğru hareket ederek örnek üzerinden sınıflamayı sağlamak için kullanılır. Karar verme ağacı sınıflandırma üzerinden bilgi öğrenmek için tipik bir tümevarımsal yaklaşımdır. Karar verme ağacı için 4 adet anahtar gereksime ihtiyaç vardır:

1. *Özellik değeri tanımı*: nesne veya durum terimlerle ifade edilmiş olmalıdır. Bu terimlerin oluşturulması algoritma tarafından yapılmalıdır.
2. *Önceden tanımlanan sınıflar* (Hedef özellik değerleri - Gözetimli Veri): Örneklerden atanan sınıflar önceden saptanmalıdır.
3. *Ayrık sınıflar*: Bir durum bir özel sınıfa bağlı olur veya olmaz. Ama her zaman için durum sayısı sınıf sayısına göre çok fazla olmalıdır.
4. *Yeterli veri*: Genellikle eğitim durumlarının sayısı yüzlerce veya hatta binlerce olmalıdır.

Karar verme ağaçlarının kurulması

Karar verme ağaçlarını öğrenmek için geliştirilen birçok algoritma, muhtemel karar ağacını doğrultusunda tepe-aşağı bir şekilde ve arama yöntemlerini yenileyerek çekirdek algoritma üzerindeki yapılan değişimlerdir. Karar verme programları eğitim durumları kümesinden bir karar ağacı oluşturur.

İlk karar verme algoritması Quanlin (1975) tarafından oluşturulan ID3 (Iterative Dichomotomizer (version) 3) tür [Quinlan, 1986]. Daha sonraki sürümler C4.5 [Quanlin, 1993] ve C5 [Rakotomalala ve Lallich, 1998] algoritmalarını içerir.

Quinlan ilk olarak ID3 algoritmasını Sidney Üniversitesinde geliştirmiştir. Bu algoritmayı 1975'teki kitabında anlatmıştır.

function ID3
Input: (R: hedef olmayan özellik kümesi,
C: hedef özellikler,
S: eğitim kümesi) Karar verme ağaçını döndür;
begin
If S is boş,
Tek düğüm döndür ve başarısızlık değeri ver;

If S hedef özellikler için tümü aynı olan kayıtları içerirse,
Değeri ile birlikte tek bir yaprak düğümü döndür;

If R boş ise,
Bir yaprak düğümü ile birlikte S'nin kayıtlarında bulunan hedef özellik değerlerinin frekans değerini döndür [Bu durumda hatalı sınıflanan örnekler ve çeşitli hatalar olabilir];

Let A, R'deki özeliklerin arasından Gain(A, S) ile özellik oluştur;

Let {aj| j=1,2, .., m} A özellik değerlerini oluştur.

Let {Sj| j=1,2, .., m} A için aj değerleri ile sıralı kayıtları içeren S'nin alt kümelerini olşutur;

Etiketlenen A ve etiketlenen a1, a2, …, am değerlerini kök ile birlikte bir ağaç döndür (ID3(R-{A}, C, S1), ID3(R-{A}, C, S2),…,ID3(R-{A}, C, Sm);

Alt küme değeri boş oluncaya kadar, ID3 alt kümeler {Sj| j=1,2, .., m} için tekrarlı bir şekilde uygula;
end;

Algoritma 2.1. Karar verme algoritması

ID3 eğitim örneklerinin özellikleri doğrultusunda arama yapar ve verilen örnekleri en iyi şekilde ayırarak özellik çıkarır. Eğer özellik eğitim kümesini en iyi şekilde sınıfladığında ID3 durur, aksi takdirde "en iyi" özelliği bulmak için m (m = bir özelliğin muhtemel değerlerinin sayısı) adet kısımlara ayrılmış alt küme üzerinde tekrarlı bir şekilde işlem yapar. Bu algoritma, en iyi değerleri seçer fakat daha önceki seçimleri yeniden ele almak için geçmişe bakmaz. Geriye bakmadığı içinde hatalı sınıflamalara sebebiyet verebilir.

Karar verme ağacının odak noktası, ağaçtaki her düğümü test ederek özellikleri seçmektir. Bu özellik seçimi ile pek çok benzer olmayan sınıf dağılımı için düzensizlik kavramı kullanılır.

Hangi özellik en iyi sınıflandırıcıdır?

Karar verme algoritmasındaki tahmin kıstası ağaçtaki her karar düğümünü test ederek özellik seçimini yapmaktır. Yani, amaç örnekleri sınıflamak için en faydalı özelliği seçmeye çalışmaktır. Bir özelliğin en iyi nicel ölçümü, *bilgi kazancı* olarak bilinen istatistiksel bir yöntemdir. (bak Bölüm 2.4.3.5.1.) Bilgi kazancı, verilen özelliğin hedeflenen sınıflamaya uygun olarak eğitim örneklerini nasıl en iyi şekilde ayrılacağını ölçer. Bu ölçüm bir ağaç büyürken her adımda aday özellikler arasından seçimde kullanılır.

Düzensizlik – örnek kümenin homojenlik ölçümü

Bilgi kazancını tam olarak belirlemek için *düzensizlik (entropi - düzensizliğin bir ölçüsü)* adı verilen genellikle bilgi teorisinde kullanılan bir ölçüm tanımlanması gerekir. Düzensizlik, gelişigüzel toplanan örnekler üzerinden tanımlanır. Hedef kavramın pozitif ve negatif örneklerini içeren S kümesinin düzensizliği ikili sınıflama ile şu şekilde tanımlanır:

$$Entropy(S) = -p_p \log_2 p_p - p_n \log_2 p_n$$

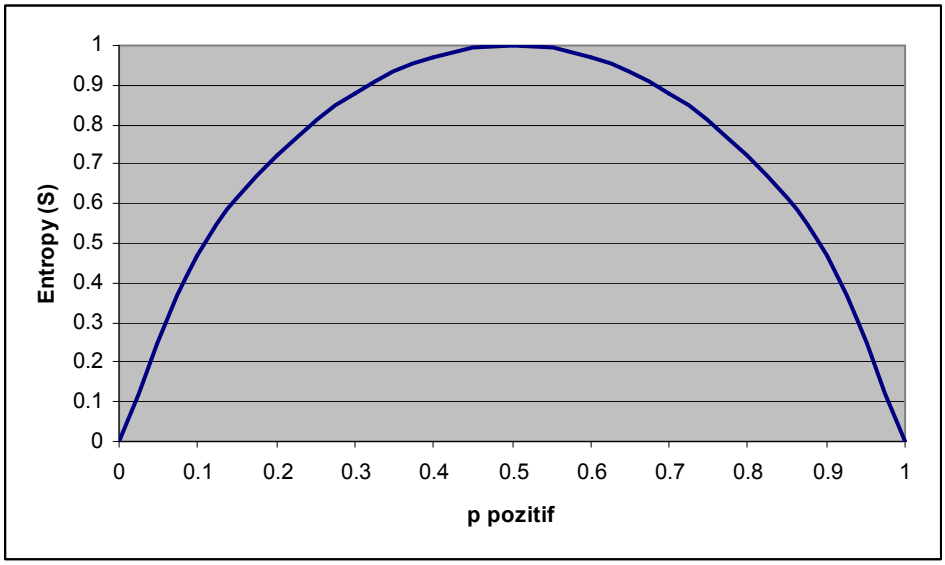
p_p : S'deki pozitif örneklerin oranı

p_n : S'deki negatif örneklerin oranı

Örnek olarak gösterecek olursak; 25 adet örnekten oluşan, 15 pozitif ve 10 adet negatif örnek içeren bir S kümemiz olsun. Bu durumda S'in düzensizliği:

$$Entropy(S) = -(\frac{15}{25}) \log_2 \frac{15}{25} - (\frac{10}{25}) \log_2 \frac{10}{25} = 0.970$$

Eğer p_p ve p_n oranlarının ikisinden biri 0 ise $Entopy(S) = 0$ olur. p_p ve p_n oranları en fazla 1 değerini alabilir. Ayrıca bu değerler eşit olduğunda $Entopy(S) = 1$ değerini alır. Bu formülasyona göre karşımıza şöyle bir grafik çıkar. (Şekil 2.3.)



Şekil 2.4. P pozitif durumları için Entropi değerleri

Bu örnekteki hedef sınıflama ikili idi. Yani sadece pozitif ve negatif durumları içeren ikili bir formülasyona sahipti. Eğer c adet durumuz var ise bu durumda formülasyon:

$$Entropy(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

p_i : S'in sınıf i ye ait oranı

Bilgi kazanmada düzensizlikte beklenen indirgemenin ölçümü

Verilen düzensizlik, eğitim verisini sınıflamada bir özelliğin etkinliğinin ölçümünde tanımlayabilir. *Bilgi kazanma* olarak adlandırılan kullanılan ölçüm, düzensizlikteki beklenen indirgemedir. Bilgi kazanmanın A özellikleri ve S eğitim kümesine bağlı formülasyonu:

$$Gain(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Values(A): A özellikleri için tüm muhtemel değerin kümesi

S_v: A özelliklerinin *v* değeri için S'nin alt kümesi

Yeni özellikleri seçme ve eğitim verilerinin kısımlara bölme işlemi her sonlanmayan durum için tekrarlar. Bu işlem her yeni yaprak notu için iki durum söz konusu olur:

1. Gelen yeni özellik ağaçta zaten bulunabilir veya,
2. Yaprak düğümlerle birleşmiş eğitim örnekleri tümü aynı hedef özellik değerlerine sahip olabilir(yani düzensizlik sıfırdır).

ID3 algoritmasının sorunları

Karar verme ağaçları ile öğrenmede ağacın aşırı derecede nasıl büyüceği, sürekli özelliklerin elle girilmesi, en uygun özellik seçme ölçümünün seçilmesi, eksik özellik değerleri ile birlikte eğitim verisinin elle girilmesi ve sayısal etkinliğin geliştirilmesi gibi sorunlarla karşılaşılmıştır. Bu bölümde ID3 algoritması için belirtilen sorunlar üzerine çeşitli tavsiyelerde bulunulacaktır.

Aşırı beslenmiş veriden kaçınma

ID3 algoritması uygun veri koşullarında iyi çalışması rağmen verideki gürültü veya eğitim verisinin çok küçük olması gibi zorluklarla karşılaşılabilir. Bu durumlardan biriyle karşılaşıldığında en iyi yöntemlerden biri eğitim örnekleri uygun hale getirmeye çalışmaktır. Bunun içinde örnekleme sayısı arttırılır. Bu durumda ise karşımıza aşırı beslenme sorunu çıkar.

Aşırı beslenme, karar verme ağaçları ve birçok öğrenme metodu için önemli bir sorundur. Aşırı beslenmeden kaçınmak için çeşitli yöntemler vardır. Bu yöntemleri iki grupta toplamak mümkündür:

1. Eğitim verisinin mükemmel olarak sınıflandığı noktaya ulaşıldığında ağacın büyümesini durdurma yaklaşımları.
2. Ağacı budayarak, verideki aşırı beslenmeyi azaltan yaklaşımlar.

İlk yaklaşımlar çok direk olmasına rağmen, budama yöntemini kullanan ikinci yaklaşımlar pratikte çok başarılı olabilir. Birinci yaklaşımlardaki sorun büyümenin nerede duracağına karar vermedeki zorluktur.

Önceden durdurma veya budama yöntemleriyle doğru ağaç boyutunu bulunup bulunamayacağına bakılmaksızın, en önemli anahtar soru en uygun ağaç boyutunu belirlemek için hangi kıstasın kullanılacağını tespit etmektir. Bu yaklaşımlarda:

- Ağaçtan budanan düğümlerin yararını ölçmek için eğitim örneklerinden ayrı bir örnekleme kümesi kullanmak.
- Eğitim için tüm uygun veriyi kullanmak, ama dikkate değer düğümlerinin genişleme olup olmayacağını tahmin etmek için kullanılan istatistiksel testler uygulamak.
- Kodlanan eğitim örnekleri ve karar ağacı, kodlanan boyut minimize edildiğinde duraksayan ağaç büyümesi için kesin karmaşıklık ölçümünü kullanmak. (Bu yaklaşım Minimum tanımlama uzunluğu (Minimum Description Length) prensibi olarak adlandırılan sezgiselliğe dayanır.)

Bu yaklaşımlara göre mevcut veri iki ayrı kümeye ayrılır:

- *Eğitim kümesi*: öğrenme hipotezi tarafından kullanılır.
- *Doğrulama kümesi*: Sonradan gelen veriler üzerinde bu hipotezin başarısının ölçülmesi ve hipotezin genişleme-budama etkisinin değerlendirilmesi için kullanılır.

Sürekli – Değerli özelliklerin birleştirilmesi

ID3'ün başlangıç tanımı özellikleri kısıtlamaktadır. İlk önce, öğrenme ağacı tarafından tahmin edilen hedef özellik değerleri ayrık değerler olmalıdır. İkinci olarak, ağacın karar verme düğümlerinde test edilen özellikler ayrık değerler olmalıdır. Bu ikinci kısıtlama, öğrenme ağacındaki sürekli-değerli karar verme özellikleri birleştirilebilmesi için kolaylıkla çıkarılabilir. Bu tanımlanan yeni ayrık-değerli özellikler dinamik olarak başarılabilir. Özellikle, sürekli-değerli A değerleri için,

algoritma eğer $A < c$ için doğru olan yeni A_c özelliği dinamik olarak yaratılabilir. Buradaki sorun eşik değeri olan *c*'nin en iyi değerini nasıl seçileceğidir. Açıkça, en iyi bilgi kazanmayı sağlamak için *c* değeri bulunmaya çalışılır. Sürekli A özelliklerine göre örnekleri sıralayarak, sonra kendi hedef sınıfında farklı olan bitişik örnekleri tanımlayarak, A'nın ilgili değerleri arasından aday eşik değerleri üretilir. Bilgi kazanma açısından uygun sınır değerlerine yaklaşıncaya kadar aday eşik değerleri arasından maksimize eden değer seçilir.

Eğitim örnekleri ile eksik özellik değerlerinin elle girilmesi

Belirli durumlarda, mevcut veri bazı özellikler için eksik değerler içerebilir. Bu gibi durumlarda, değerleri bilinen özellikler için diğer örneklemelere dayalı eksik özellik değerleri tahmin edilir.

Birinci strateji, eksik özellik değerlerinin üstesinden gelmek için n adet düğümde eğitim kümesi içine en genel değerleri atamaktır. Alternatif olarak, *c(x)* sınıflandırılmasına sahip n adet düğümde örnekler içine en genel değerlerde atanabilir. *A(x)* için bu tahmin edilen değerleri kullanan ayrıntılı eğitim örnekleri var olan karar verme ağacı algoritması tarafından direk kullanılabilir.

İkinci strateji, daha karmaşık bir yöntem *A(x)* en genel değeri atamak yerine A'nın muhtemel olası değerlerinin olasılıklarını atamaktır. Bu olasılık değeri A için gözlenen çeşitli değerlerin frekanslarına bağlı olarak tahmin edilir. Mesela, verilen bir örnekleme içinden 10 adet değer seçelim. Bu değerlerin altısı $A = 1$ ve dörtü $A=0$ olsun. Bu durumda $A(x)=1$ için olasılık 0.6 iken $A(x) = 0$ için olaslık ise 0.4 alınır.

Avantajları

- Karar verme ağaçları anlaşılır kurallar üretebilir.
- Karar verme ağaçları daha az hesaplaya ihtiyaç duyarak sınıflama işlemini gerçekleştirir.

- Karar verme ağaçları sürekli ve kategorik değişkenlerin her ikisi de elle işlenebilir.
- Karar verme ağaçları tahmin ve sınıflama için önemli olan alanları açık bir şekilde gösterir.

Dezavantajları

- Karar verme ağaçları sürekli değerleri tahmin görevleri için daha az elverişlidir.
- Karar verme ağaçları özellikle ufak eğitim örneklerinde ve birçok sınıfta sınıflama problemlerine sebebiyet verir.
- Karar verme ağaçlarında hesaplama olarak eğitim aşaması masraflı olabilir. Karar verme ağacının büyüme işlemi hesaplama olarak pahalıdır. Her düğümde, alanlara ayrılmış her aday kendisine ait en iyi bölüm bulunmadan önce sıralanmalıdır. Bazı algoritmalarda alanların birleşimi kullanılır ve böylece bir arama en uygun duruma getirilebilir. Budama algoritmaları ayrıca aday alt ağaçları oluşturma ve karşılaştırma açısından pahalı bir yöntemdir.
- Karar verme ağaçları dörtgen biçiminde olmayan alanlarda iyi sonuç vermeyebilir. Birçok karar verme ağaçları algoritması sadece bir zamanda bir tek alanı inceler. Bu karar verme uzayında kayıtların gerçek dağılımı uygun olmayan dikdörtgen alanlara yol açar.

2.7.3. Naive Bayes sınıflandırıcı

Naive Bayes sınıflandırıcı kuvvetli bağımsız varsayımlarla Bayes teoremini temel alan olasılıklı bir sınıflayıcıdır. Bu olasılık modelinde altı çizilecek nokta *bağımsız özellik modeli* olmasıdır. Yalın tasarımına ve görünüşte basitleştirilmiş varsayımlara rağmen naive Bayes sınıflandırıcı gerçek dünya durumlarında beklenenden çok daha iyi sonuçlar vermektedir [Domingos ve Pazzani, 1997] [Rish, 2001] [Hand ve Yu, 2001] [Mozina vd., 2004] [Maron, 1961] [McCallum ve Nigam, 1998].

Olasılık modeline bağlı olarak, naive Bayes sınıflandırıcı gözetimli öğrenmede çok etkin bir şekilde kullanılabilir. Birçok uygulama, naive Bayes modellerinin parametre tahmini için maksimum benzerlik metodunu kullanır.

Naive Bayes olasılık modeli

Teorik olarak, sınıflandırıcı için olasılık modeli koşullu bir modeldir:

$$p(C|F_1,...F_n)$$

C: bağımlı sınıf değişkeni

$F_1,...F_n$: koşullu birkaç özellik değişkeni

Buradaki problem eğer n özelliklerinin sayısı çok büyük veya özelliklerin değerleri çok büyük olursa, model için kurulacak olasılık tablosunu kurmak zorlaşır. Bu yüzden bu modeli daha kontrol edilebilir duruma getirmeliyiz.

Bayes teoremini kullanırsak,

$$p(C|F_1,...F_n) = \frac{P(C)\,p(F_1,...F_n|C)}{p(F_1,...F_n)}$$

Bu formüle göre pay C'ye bağlı olmasına rağmen, payda C'ye bağlı değildir. Bu durumda payda kısmına C'yi katacak olursak:

$$\begin{aligned}
&p(C,F_1,...,F_n)\\
&= p(C)\,p(F_1,...,F_n|C)\\
&= p(C)\,p(F_1|C)\,p(F_2,...,F_n|C)\\
&= p(C)\,p(F_1|C)\,p(F_2|C,F_1)\,p(F_2,...,F_n|C)
\end{aligned}$$

Bu durumda koşullu bağımsızlık varsayımları ortaya çıkar. $i \neq j$ için her özellik F_i diğer özellik F_j 'nin koşullu bağımsızıdır. Formül olarak:

$$= p(F_i|C, F_j) = p(F_i|C)$$

Bu durumda bağlantı modeli şu şekilde yazılabilir:

$$p(C, F_1, ..., F_n) = p(C)p(F_1|C)p(F_2|C)...$$
$$= p(C)\prod_{i=1}^{n} p(F_i|C)$$

Bağımsızlık varsayımları üzerine, C sınıf değişkeninin koşullu dağılımı şöyle yazılabilir:

$$p(C, F_1, ..., F_n) = \frac{1}{Z} p(C)\prod_{i=1}^{n} p(F_i|C)$$

Z: eğer özellik değişkenlerinin değerleri biliniyorsa Z $F_1, ..., F_n$ bağlı bir ölçekleme katsayısıdır.

Avantajları

- Eğitim ve değerlendirme işlemi çok hızlıdır
- Gerçek dünya sorunlarında şaşırtıcı derecede iyi sonuçlar vermektedir

Dezavantajları

- Çok karmaşık sınıflama problemleri çözmede yetersiz kalabilir

2.7.4. Bayesian ağları

Bayesian ağları, değişken kümesi üzerine bağlantı olasılık dağılımda verilen bağımsızlıkları gösteren bir direkt çevrimsiz grafiktir. Düğümler; bir ölçülen parametre, bir örtülü değişken veya bir hipotez olabilen herhangi bir değişken çeşidi ile gösterilebilir. Bunlar, gösterilen tesadüfî değişkenlerle sınırlandırılmamıştır. Bu bilginin ne alacağına bakılmaksızın görülen düğümler ilgilenilen değişkenlere ve iki düğüm arasındaki kenarlar değişkenler arasındaki muhtemel bağımlılığa tekabül eder. X ve Y iki düğümü arasındaki kenarların bulunmaması X ve Y bağımsız hale getiren bazı Z değişkenler kümesi vardır anlamına gelir. Eğer X ve Y bir kenar tarafından bağlı değilse, X ve Y bağımsız kılan Z kümesi d-ayırma denilen grafiksel durum ile tanımlanabilir. [Heckerman, 1997] [Jensen, 2001] [Pearl, 2000] [Neil vd., 2005] [Castillo vd., 1997]

Normal bir olasılık dağılımı $P_r(X_1,...,X_n)$ zincir kuralı olasılığını kullanarak $\prod_{i=1}^{n} P_r(X_i|X_{i-1},...,X_1)$ çarpımında bileşenlerine ayrılabilir. Ancak, bir değişken X_i sadece $\{X_1,...,X_{i-1}\}$'nin Pa_i alt kümesine bağımlı ve geri kalanlar bağımsız durumda olabilir. Daha sonra, $P_r(X_i|X_{i-1},...,X_1)$ $P_r(X_i|P_{ai})$ olarak yazılabilir ve genel olasılık dağılımı $\prod_{i=1}^{n} P_r(X_i|P_{ai})$ olarak yazılabilir. Bundan sonra, kendi düğümüne sahip her X_i değişkeni izin verilerek ve Pa_i den X_i'de her düğümden direkt çizgiler çizilerek direkt çevrimsiz grafik inşa edilebilir. Sonuç grafiği olarak $P_r(X_1,...,X_n)$'nın bir Bayesian ağı oluşur.

Bir Bayesian ağı bir direkt çevrimsiz grafiktir ki:

- Düğümler rasgele değişkenlerle temsil edilir,
- Kavisler, değişkenler arasındaki istatistiksel bağımlılık ilişkileri ve değerleri verilen her değişken için yerel olasılık dağılımları ile temsil edilir.

Eğer *A* düğümden *B* düğümüne bir kavis var ise, değişken B değişken A bağımlı olur ve A, B'nin ebeveyni olarak adlandırılır. Eğer $i \in \{1,...,N\}$ şeklindeki X_i'nin her değişken için ebeveyn değişkenler kümesi $ebeveyn(X_i)$ olarak gösterilebilirse değişkenleri birleşik dağılımı yerel dağılımların bir neticesidir:

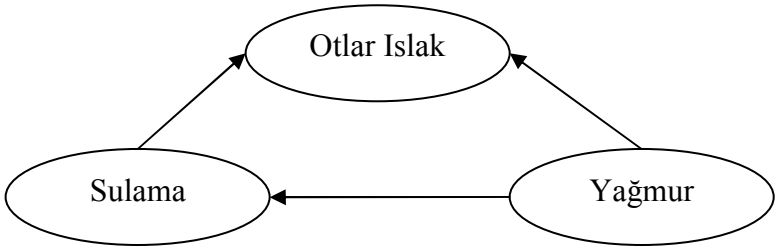
$$P_r(X_1,...,X_n) = \prod_{i=1}^{n} P_r(X_i | ebeveyn(X_i))$$

Eğer X_i hiç bir ebeveyn sahip değilse kendisinin yerel olasılık dağılımı *koşulsuz* olduğu söylenebilir, aksi takdirde *koşullu* olur. Eğer düğüm ile sunulan değişken gözlenen ise düğüm bir ifade düğümü olduğu söylenir.

Bayesian ağlarının bir avantajı eksiksiz bileşik dağılımlar yerine doğrudan bağımlılık ve yerel dağılımların sezgiyle anlaşılması sebebiyle kolaydır.

Bayesian ağları örneği

Sulama ve yağmur gibi otların ıslak olması sebebiyet verecek iki durum olsun. Bu durum Bayesian ağları ile modellenebilir. Burada her değişken *Y* (Yanlış) ve *D* (Doğru) iki muhtemel durum alabilir. İlişki değerleri aşağıdaki tabloda gösterilmiştir.



Yağmur	
D	Y
0.4	0.6

	Sulama	
Yağmur	D	Y
Y	0.4	0.6
D	0.01	0.99

		Otlar Islak	
Sulama	Yağmur	D	Y
Y	Y	0.0	1.0
Y	D	0.8	0.2
D	Y	0.9	0.1
D	D	0.99	0.01

Tablo 2.11. Bayesian Ağları için veriler

Bu örneğe göre, Pr*(I=doğru | S=doğru, Y=yanlış) = 0.9 olduğu görülüyor*, bu nedenle Pr*(I=yanlış | S=doğru, Y=yanlış) = 0.1,* çünkü her satırın toplamı 1 olmak zorundadır.

Bu örnek için bağlantı olasılık fonksiyonu:

$$\Pr(I, S, Y) = \Pr(I \mid S,Y) \times \Pr(S \mid Y) \times \Pr(Y)$$

Bu model "Otlar ıslak olduğunda yağmur yağma olasılığı nedir?" gibisinden bir soruya yukarıdaki formülü kullanarak cevap verebilir.

$$\Pr(Y = 1 \mid I = 1) = \frac{\Pr(I = 1, Y = 1)}{\Pr(I = 1)} = \frac{\sum_{S \in \{1,0\}} \Pr(I = 1, S, Y = 1)}{\sum_{S,Y \in \{1,0\}} \Pr(I = 1, S, Y)}$$

$$= \frac{P(I = 1, S = 1, Y = 1) + P(I = 1, S = 0, Y = 1)}{P(I = 1, S = 1, Y = 1) + P(I = 1, S = 0, Y = 1) + P(I = 1, S = 1, Y = 0) + P(I = 1, S = 0, Y = 0)}$$

$$= \frac{0.8 * 0.99 * 0.2 + 0.99 * 0.01 * 0.2}{0.8 * 0.99 * 0.2 + 0.99 * 0.01 * 0.2 + 0 * 0.6 * 0.8 + 0.9 * 0.4 * 0.8} = \%35.7688$$

Avantajları

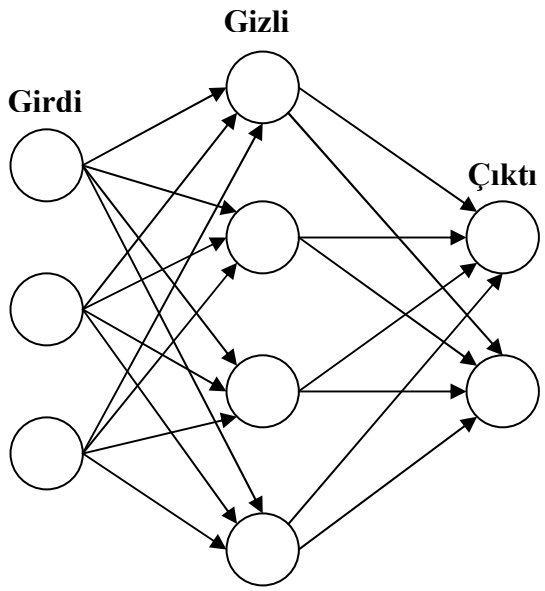
- Bir uzman bilgisi sağlar.
- Yeni bir veri geldiğinde güncel tutulması daha kolaydır.

Dezavantajları

- Algoritma gelişigüzel ve öznel olduğu için oluşan Bayesian ağı her zaman çok sağlıklı olmaz.

2.7.5. Yapay Sinir Ağları

Yapay sinir ağları basit biyolojik sinir sisteminin çalışma şekli simüle edilerek tasarlanan programlama yaklaşımıdır. Simüle edilen sinir hücreleri (nöronlar) içerirler ve bu nöronlar çeşitli şekillerde birbirlerine bağlanarak ağı oluştururlar. Bu ağlar öğrenme, hafızaya alma ve veriler arasındaki ilişkiyi ortaya çıkarma kapasitesine sahiptirler. Diğer bir ifadeyle, sinir ağları, normalde bir insanın düşünme ve gözlemlemeye yönelik doğal yeteneklerini gerektiren problemlere çözüm üretmektedir. Bir insanın, düşünme ve gözlemleme yeteneklerini gerektiren problemlere yönelik çözümler üretebilmesinin temel sebebi ise insan beyninin ve dolayısıyla insanın sahip olduğu yaşayarak veya deneyerek öğrenme yeteneğidir. [Abdi, 2003] [Abdi vd., 1999] [Anderson, 1995] [Michael, 1995] [Wilkes ve Wade, 1997] [Mandic ve Chambers, 2001]



Şekil 2.5. Yapay sinir ağları örneği

Yapay sinir ağları, ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem biriminden (nöronlar) oluşan matematiksel sistemlerdir (Şekil 2.5.). Bir işlem birimi, aslında sık sık transfer fonksiyonu olarak anılan bir denklemdir. Bu işlem birimi, diğer nöronlardan sinyalleri alır; bunları birleştirir, dönüştürür ve sayısal bir sonuç ortaya

çıkartır. Genelde, işlem birimleri kabaca gerçek nöronlara karşılık gelirler ve bir ağ içinde birbirlerine bağlanırlar; bu yapı da sinir ağlarını oluşturmaktadır.

Sinirsel (*neural*) hesaplamanın merkezinde dağıtılmış, adaptif ve doğrusal olmayan işlem kavramları vardır. Sınır ağları, geleneksel işlemcilerden farklı şekilde işlem yapmaktadırlar. Geleneksel işlemcilerde, tek bir merkezi işlem birimi her hareketi sırasıyla gerçekleştirir. Sinir ağları ise her biri büyük bir problemin bir parçası ile ilgilenen, çok sayıda basit işlem birimlerinden oluşmaktadır. En basit şekilde, bir işlem birimi, bir girdiyi bir ağırlık kümesi ile ağırlıklandırır, doğrusal olmayan bir şekilde dönüşümünü sağlar ve bir çıktı değeri oluşturur. İlk bakışta, işlem birimlerinin çalışma şekli yanıltıcı şekilde basittir. Sinirsel hesaplamanın gücü, toplam işlem yükünü paylaşan işlem birimlerinin birbirleri arasındaki yoğun bağlantı yapısından gelmektedir. Çoğu yapay sinir ağı, benzer karakteristiğe sahip nöronlar tabakalar halinde yapılandırılırlar ve transfer fonksiyonları eş zamanlı olarak çalıştırılırlar. Hemen hemen tüm ağlar, veri alan nöronlara ve çıktı üreten nöronlara sahiptirler.

Yapay sinir ağlarının ana öğesi olan matematiksel fonksiyon, ağın mimarisi tarafından şekillendirilir. Daha açık bir şekilde ifade etmek gerekirse, fonksiyonun temel yapısını ağırlıkların büyüklüğü ve işlem elemanlarının işlem şekli belirler. Sinir ağlarının davranışları, yani girdi veriyi çıktı veriye nasıl ilişkilendirdikleri, ilk olarak nöronların transfer fonksiyonlarından, nasıl birbirlerine bağlandıklarından ve bu bağlantıların ağırlıklarından etkilenir.

Yapay sinir ağları geniş bir alana uygulanmaktadırlar. Bu alanları gruplayacak olursak:

- Fonksiyon tahmini ve regresyon analizi: zaman serileri tahmini ve modellemesi
- Sınıflandırma: örüntü ve düzen tanıma, algılama ve sırasal karar verme
- Veri işleme: filtreleme, kümeleme, kör sinyal ayırma ve sıkıştırma

Sinir ağları sistem tanıma ve kontrolü(taşıt kontrolü, işlem kontrolü), oyunlar ve karar verme(satranç, tavla, yarış), örüntü tanıma(radar sistemleri, yüz tanıma, nesne

tanıma), düzen tanıma(işaret, konuşma, el yazısı tanıma), medikal teşhis, finansal uygulamalar, veri madenciliği, e-posta spam filtreleme gibi uygulama alanlarında kullanılmaktadır.

Yapay sinir ağları üzerine Hebb Kuralı veya Öğrenmesi(Hebb Rule / Hebbian Learning), Perceptron Öğrenme, Çok Katmanlı Perceptron Öğrenme(Multilayer Perceptron Learning), Uyarlanabilir Rezonans Teorisi(Adaptive Resonance Theory), Geriye-Çoğalma (Back-Propagation), İki Yönlü Çağrışımlı Öğrenme(Bi-Directional Associative Memory), Olaslıklı Ağlar(Probalistic Networks) ve Destek Karar Makinesi(Support Vector Machine) birçok yöntem geliştirilmiştir. Uygulama aşamasında özellikle Çok Katmanlı Perceptron Öğrenme ve Destek Karar Makinesi yöntemleri üzerinde durduk. Bundan sonraki iki bölümde bu yöntemleri anlatacağız.

Avantajları

- Etkili doğrusal olmayan sınıflandırıcıdır.
- Sürekli temel fonksiyonların yapımında iyi çözümler sunar.
- Gürültülü data iyi bir şekilde elle yönetilebilir.

Dezavantajları

- Çok karmaşık problemlerin çözümünde zorlanır.

2.7.5.1. Çok katmanlı Perceptrons

Çok katmanlı perceptrons, perceptrons olarak adlandırılan basit sinir hücreleri ağıdır. Temel fikir olan tek perceptron (single perceptron) ilk olarak 1958 yılında Rosenblatt tarafından tanıtılmıştır. Perceptron birçok değeri girdi olarak alıp tek bir çıktı üretir. Çıktıyı oluştururken girdi ağırlıklarına uygun olarak bir lineer kombinasyon oluşturulur ve bazı lineer olmayan etkinleştirme fonksiyonu yoluyla çıktılara yerleştirilir [Minsky, 1995]. Matematiksel formül olarak:

$$y = \ell(\sum_{i=1}^{n} w_i x_i + b) = \ell(w^T x + b)$$

w : ağırlık vektörleri,

x : girdi vektörleri,

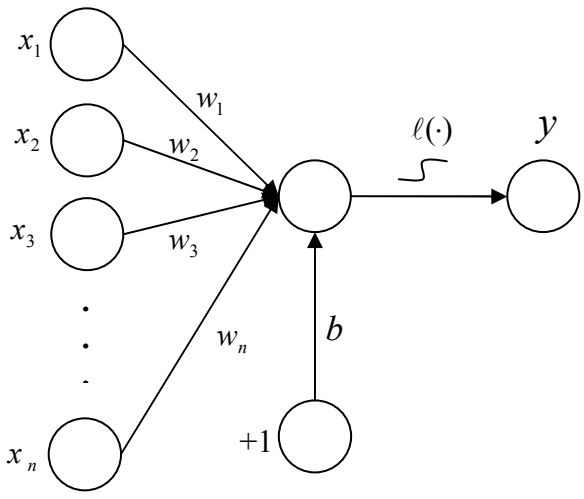
b : eğilim,

ℓ : etkinleştirme fonksiyonu gösterilir.

Rosenblatt, Heaviside [Kainen vd., 2000] adım fonksiyonunu etkinleştirme fonksiyonu olarak kullanmıştır. Günümüzde etkinleştirme fonksiyonu olarak logistic sigmoid $\frac{1}{(1+e^{-x})}$ veya hyperbolic tangent $\tanh(x)$ kullanılır. Aslında bu iki fonksiyonda birbiriyle bağlantılır:

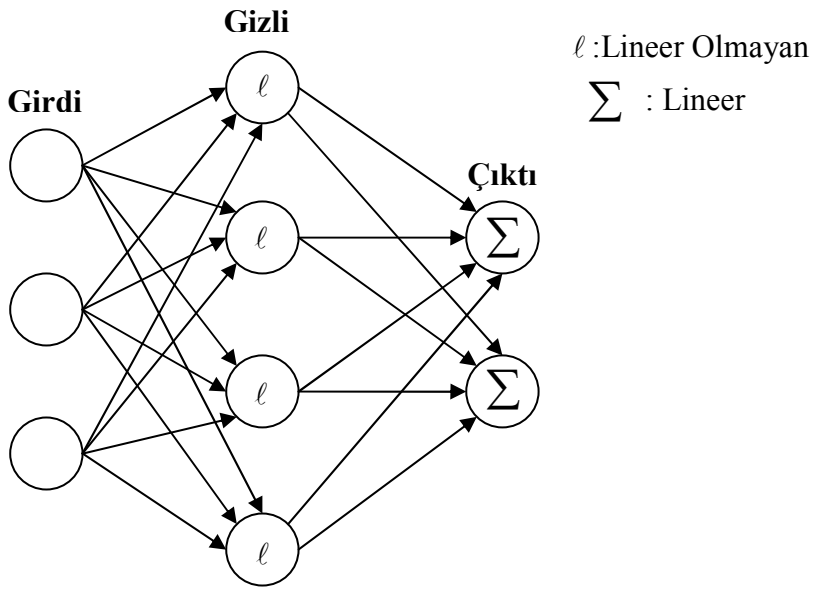
$$\frac{(\tanh(x)+1)}{2} = \frac{1}{(1+e^{-2x})}$$

Matematiksel olarak elverişli oldukları için genelde bu fonksiyonlar kullanılır. Bu yapı çok katmanlı perceptrons ağları güçlü ve doğrusal olmayan eşleştirmeleri modellemeyi sağlar.



Şekil 2.6. Tek percptron'un akış grafiği

Sınırlı sayıda eşleştirme olanak sağladığı için tek perceptron kullanışlı bir yöntem değildir. Çok katmanlı perceptron girdi, gizli ve çıktı katmanlarından oluşur. İşlemler ağda adım adım yürütülür.



Şekil 2.7. Çok katmanlı perceptron'un akış grafiği

Çok katmanlı perceptron formülasyonu:

$$x = f(s) = B\ell(As + a) + b$$

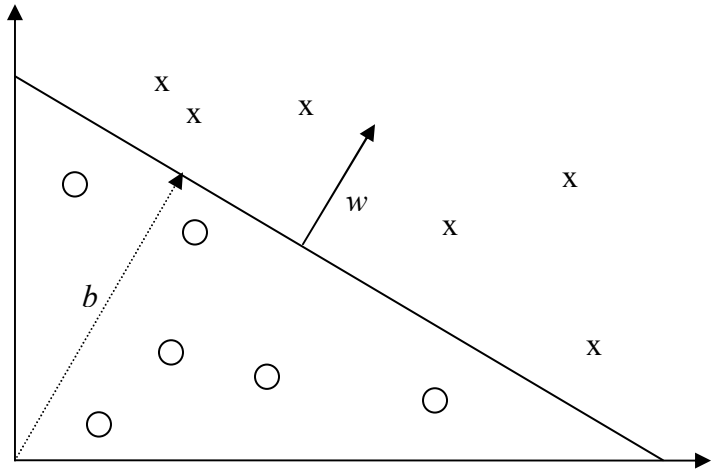
s : girdi vektörleri,
x : çıktı vektörleri,
A : ilk katmanın ağırlık matrisi,
a : ilk katmanın eğilim değeri,
B : ikinci katmanın ağırlık matrisi,
b : ikinci katmanın eğilim değeri,
ℓ : lineer olmayan elementleri ifade eder.

Tek katmanlı ağlara göre çok katmanlıların avantajı gizli katmanda ortaya çıkar. Hornik ve ark. (1989) ve Funahashi (1989) verilen herhangi bir doğruluk değerinde

birçok gizli ünitenin elde edilebilir olduğunu herhangi bir sürekli fonksiyonla $f : R^n \rightarrow R^m$ yapılabileceğini göstermişlerdir.

2.7.6. Destek karar makineleri

Destek karar makineleri (Support vector machines) yüksek boyutlu verilerde sıkça kullanılan bir ayrım metodudur. Veri üzerinde eğitilerek uygulanır. Eğitim için konveks optimizasyonunda kullanılan herhangi bir metot seçilebilir. Temelde düşük boyutta lineer olarak ayrılamayacak bir veri kümesini, daha yüksek boyuta taşıyarak bir düzlem yardımıyla ayırmayı sağlar. Destek karar makinelerinin mantığını anlamak için, lineer ayrımın nasıl işlediğine bakmamız gerekir [Cristianini N. ve Shawe-Taylor J., 2000] [Kecman, 2001] [Herbrich, 2002].



Şekil 2.8. Fisher Doğrusal Diskriminantı

Yukarıdaki şekilde Fisher Lineer Diskriminantı (Fisher Lineer Discriminant) görülmektedir. Burada iki farklı tipteki veri birbirinden lineer olarak ayrılmaktadır. Bu lineer doğru veya ayrım fonksiyonu aşağıdaki şekilde ifade edilir:

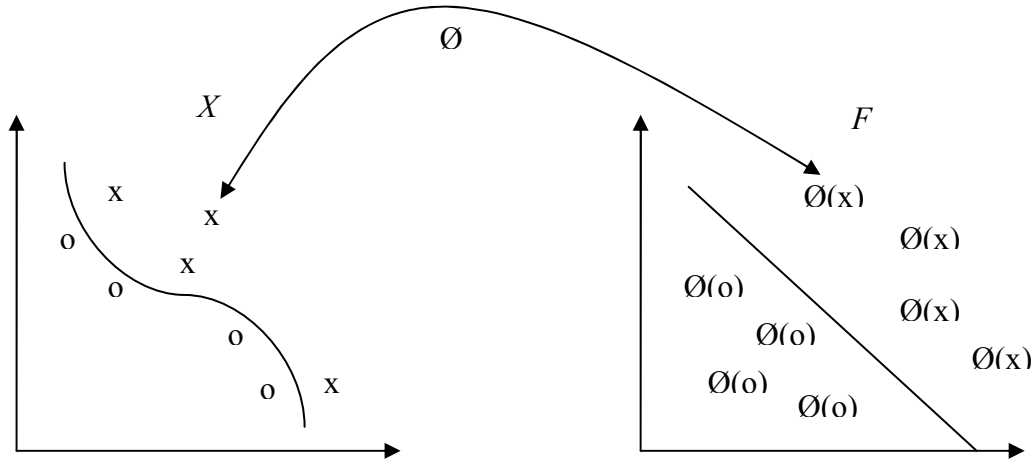
$$f(x) = \sum_{i=1}^{n} w_i x_i + b$$

Burada x ayrım yapacağımız girdi verisi, b ise eşik değer veya orijine uzaklık olarak tanımlanmaktadır. w ise ağırlık matrisi yada veriye bağlı olarak bir vektördür. Bu vektör veri eğitim aşamasında şekillenir.

Eğer f(x) $\geq$ 0 ise o zaman x vektörüne ait sınıf pozitif, tersi ise negatif olarak değerlendirilir. Kısaca karar fonksiyonu sgn($f(x)$) dir. Bu fonksiyon daha sonra destek karar makinelerinde görünen ayrım fonksiyonunun basit halidir.

Lineer Ayrım metotlarından Rosenblatt's Perceptron [Rosenblatt, 1958], bu yapıya en uygun olan eğitim algoritmasını içerisinde barındırmaktadır.

Destek karar makineleri, yapı itibariyle Perceptron algoritmasına benzemesine rağmen aralarında çok önemli bir fark vardır. Bu fark *çekirdek* (kernel) fonksiyonu ile sağlanmaktadır. Çekirdek fonksiyonun Destek karar makinelerine sağladığı avantaj aşağıda gösterilmektedir.



Şekil 2.9. $\phi(x)$ çekirdek fonksiyonunun etkisi

Bu şekilde $\phi(x)$ ile ifade edilen fonksiyon çekirdek fonksiyonudur. Bu fonksiyonun görevi aslında lineer olarak ayrılamayan veriyi, daha yüksek bir boyuta

taşıyarak bir düzlem ile ayrım yapılmasını sağlar. Çeşitli çekirdek fonksiyonları bulunmaktadır, örneğin

- $\exp(\frac{-\|x-z\|^e}{\sigma^2})$: Üstsel çekirdek (exponential kernel)
- $\frac{1}{\sqrt{\|x-x'\|^2+c^2}}$: Ters çoklu ikinci derece çekirdek (inverse multiquadratic)
- $-\sqrt{\|x-x'\|^2+c^2}$: Çoklu ikinci derece çekirdek (multiquadratic)
- $\|x-x'\|^{2n}\ln\|x-x'\|$: İnce eğri tabaka (thin plate spline)

Destek karar makineleri çekirdek kullandıkları için yukarıda bahsettiğimiz Fisher Lineer Diskriminantından ayrılıyorlar. Aşağıda, Destek karar makinelerinin ayrım fonksiyonu gösterilmektedir.

$$y = sign((w \cdot \phi(x)) + b)$$

Bu ayrım fonksiyonunu lineer ayrım fonksiyonlarından ayıran tek şey, *w* ağırlık matrisinin eğitim şeklidir. Bu ayrımın lineer bir düzlem şeklinde yapılabilmesi için, w'nin karmaşıklığının en düşük olması gerekir. Bu da ancak ikinci dereceden (Quadratic) optimizasyon teknikleri ile mümkün olabilir. Aşağıdaki optimizasyonun neye göre yapılması gerektiği çıkarımlar göz önüne alınmadan basitçe verilmiştir.

$$\min_{\|w\|\hbar} \frac{1}{2}\|w\|^2, subject\_to(??)$$

Yukarıdaki ikili (dual) optimizasyonu çözmek için Lagranj çarpanlarının kullanılması gerekir, Lagranj açılımı yapıldıktan sonra ikili optimizasyon problemi aşağıdaki şekli alacaktır.

$$\max_\alpha \sum_{i=1}^n \alpha_i - \frac{1}{2}\sum_{i,j=1}^n \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

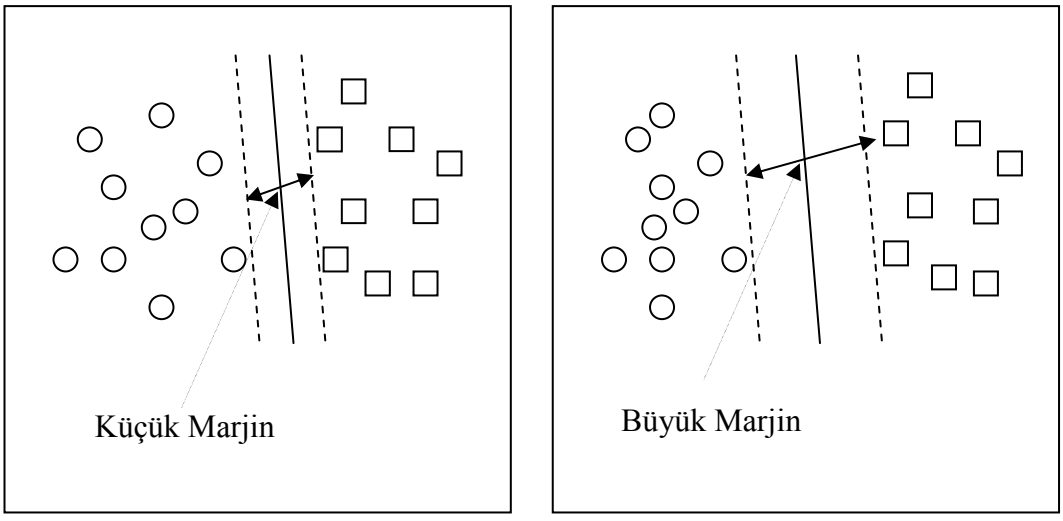
$subject\_to$

$\alpha_i >= 0, i = 1,...n,$

$\sum_{i=1}^{n} \alpha_i y_i = 0$

Burada $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$ olduğundan bütün işlemler aslında çekirdek fonksiyonunun üzerinde gerçekleşmektedir. Dolayısıyla burada yapılacak bir optimizasyonun tüm veri üzerinde üstsel bir etki yapacağı ve veriyi non-lineer olarak ayıracağı düşünülebilir. Yukarıdaki optimizasyon işlemi için konveks optimizasyonunda kullanılabilen tüm yöntemler geçerli olacaktır. Bu da demek oluyor ki optimizasyon sonucunda bulunan tüm lokal minimum yada lokal maksimum noktaları tektir ve globaldir. Destek karar makinelerinde en çok kullanılan optimizasyon tekniği Sequential Minimum Optimizasyon (SMO) dur [Platt, 1998]. SMO hızlı ve kararlı bir tekniktir.

Çeşitli destek karar makineleri bulunmaktadır. Verinin gürültü içerip içermemesine bağlı olarak en çok kullanılan iki yöntem bulunmaktadır. Bunlar *maksimal marjin (maximal margin)* ayrımcısı ve *esnek marjin (soft margin)* ayrımcısıdır.



Şekil 2.10. Esnek marjin ve maksimal marjin

Maksimal marjin ayrımcısında amaç marjini oldukça geniş tutmaktır. Bunu yapabilmek içinde marjine yakın olan tüm vektörler seçilerek bu vektörleri ayırabilecek en geniş alanı bulmak gerekir. Esnek marjin ayrımcısında ise kullanılan yardımcı vektörler marjini optimum tutacak şekilde seçilir. Dolayısıyla esnek marjin ayrımcısı ayrımı yapmak için daha genel bir düzlem belirler. Bu da esnek marjin ayrımcısının veriyi aşırı besleme (over fitting) olasılığını düşürür. Maksimal marjin ise veriyi aşırı besleyebileceğinden gürültüye daha duyarlı olacak, gürültülü veri için kötü sonuçlar üretecektir.

Görüldüğü gibi destek karar makinelerindeki temel mantık, veri lineer olarak ayrılabilecek uzaya (Hilbert Uzayı) taşımak ve üstsel optimizasyon ile yardımcı vektörleri belirleyerek, bunlar üzerinden ayrım marjinini belirlemektir.

BÖLÜM 3.

SINIRSIZ BİR DERLEM OLARAK İNTERNET

3.1. Dilbilimsel çalışmalarda kullanılan metin türleri

3.1.1. Derlem

Derlem (Corpus) için farklı tanımlar yapılabilir:

- Büyük ve yapısal metinler kümesidir.
- Dilbilimsel bilginin koleksiyonudur, yazılı veya kaydedilen konuşmalar şeklinde olabilir.
- Doğal olarak meydana gelen metinlerden dilin çeşitliliğini ve durumunu belirlemek amacıyla seçilen ve bir araya getirilen metinlerdir.
- Doğal Dil İşleme alanında kullanılmak için, yazılı veya sözlü metinlerden oluşturulmuş özel bir veritabanıdır; kelimeleri hızlı şekilde bulma ve işleme gibi özel işlemleri yapmaya izin verir

Derlemler, özellikle dilbilimsel çalışmalarda istatistiksel analiz, denetim olayları ya da onaylanmış dilbilimsel kurallar gibi özel alanlarda kullanılır.

Derleme kavramı bilgisayar tabanlı dil çalışmaları için 1960'lı yıllarda bir milyon kelimelik Brown Corpus ile karşımıza çıkar. Günümüzde en çok kullanılan derlem konuşulan ve yazılı metinlerden oluşan yüz milyon kelimelik British National Corpus (BNC)'tur. Bu derlemlerin yanı sıra Amerikan English Corpus (100 milyon kelime), Bank of English (525 milyon kelime), Helsinki Corpus (10 milyon kelime), Longman-Lancaster Corpus (14.5 milyon kelime), Oxford English Corpus (1 milyar kelime üzeri)

ve Scottih Corpus (4 milyon kelime) dilbilim dünyası tarafından tercih edilen derlemlerdir. Türkçe için geliştirilen yegâne derlem ise 2006 yılında tamamlanmış 2 milyon kelimelik METU Turkish Corpus'tur.

3.1.2. Çoklu Derlem

Derlemler genellikle konu sınırlı metin kümeleri iken çoklu derlem (corpora) birçok farklı konudaki metinlerden oluşur. Çoklu derlem kavramı sayısal dilbilim üzerine yapılan Vancouver'deki 1989 ACL konferansında dilbilim dünyasına duyurulmuştur. Birçok araştırmacı ilk olarak çoklu derlem fikrine karşı çıkmıştır. Fakat daha sonraları çoklu derlemlerle yapılan uygulamalar sayesinde dilbilim dünyası çoklu derlem fikrini benimsemeye başlamıştır. Curch ve Mencer (1993) özel sayı olarak çıkan makalelerinde sayısal dilbilim ile çoklu derlem arasındaki ilişkiyi göstermişlerdir. Çoklu derlemler sayısal dilbilim çalışmalarının yanı sıra ses tanıma, çeviri uygulamaları ve dil öğrenme gibi alanlarda da kullanılmıştır.

3.1.3. Sınırsız bir derlem: İnternet

Birçok araştırmada, derlem veya çoklu derlemler yetersiz kalabiliyordu. Bu durumu fark eden Manning ve Schütze (1999, sayfa 20) istatistiksel çalışmalarda ilgilenilen alan için yeterli miktarda verinin olmadığı durumlarda elde bulunan tüm metinlerin kullanılabileceğini önermişlerdir. Bu gibi durumlar için, ilk akla gelen en geniş metin alanı internet olarak düşünülebilir.

İnternet, birçok dil için farklı tiplerde çok fazla metin içeren bir ortamdır. Bu metinler çeşitli doğal dil çalışmaları için kaynak olarak kullanılabilir. İnternette 10 trilyonun üzerinde web sayfası bulunmaktadır. Bu web sayfalarının kaynak olarak kullanılabileceği fikri ilk olarak 1999'daki ACL konferansında tartışılmaya başlamıştır [Mihalcea ve Moldovan, 1999] [Resnik, 1999]. Özellikle arama motorları bu çalışmalarda kullanılabileceği belirtilmiştir. Fakat bu konu üzerinde ilk başta

internettekini verinin güvenirliliği ve uygunluğu konusunda tartışmalar yaşanmıştır. Kalgarriff 2001 yılında web'in kullanılabilirliğini kanıtlamışlardır [Kilgarriff, 2001]. 2001 yılındaki yaptığı çalışma 2003 yılında özel bir sayı olarak makale olarak yayınlanmıştır [Kilgarriff ve Grefenstette, 2003]. Bu çalışmada özellikle arama motorları üzerinde durulmuş ve arama motorlarının eksiklikleri ortaya konmuştur. Bu eksiklikleri ortadan kaldırmak için son yıllarda birçok çalışma yapılmıştır [Fletcher, 2004] [Baroni ve Bernardi, 2004], [Baroni ve Sharoff, 2005] [Rayson vd., 2006] [Sharoff, 2006]. Bunun yanı sıra web'i derlem olarak kullanılmasını sağlayan çeşitli uygulamalarda geliştirilmiştir. (Gsearch system [Corley ve Haywood, 2000], WebCorp [Kehoe ve Renouf, 2002], KWiCFinder [Fletcher, 2004], the Linguist's Search Engine [Kilgarriff, 2003] [Resnik and Elkiss, 2003]. Bu çalışmaların çoğu İngilizce üzerine yapılan çalışmalardır.

3.1.4. Etiketlenmiş veri

Her ne kadar derlem ve çoklu derlemler özellikle kelime tabanlı istatistiksel çalışmalarda yeterli olsa da, daha geniş çalışmalar için cümlelerin işaretlenmesi gerekir. En basit işaretleme şekli metinde geçen tüm kelimelerin fiil, isim, sıfat ve benzeri şekilde işaretlenmesidir. Bu işaretleme şekline *kategori bilgisinin etiketlenmesi* (part-of-speech tagging ya da POS-tagging) adı verilir. Diğer bir işaretleme ise kelimelerin arasındaki ilişkilerin çözümlenip işaretlenmesidir. Bu işaretleme işlemi ise dilbilimsel ayrıştırıcılar (linguistic parsers) tarafından yapılır. Bu çalışma için bölüm 4.2.3'te kullanılan işaretleme tekniği üzerinde durulacaktır.

3.2. İnternet ortamındaki veriye erişim

3.2.1. İnternette verinin depolanma biçimleri

3.2.1.1. HTML

HTML web sayfalarının hazırlanmasında kullanılan bir biçimleme dilidir. İnternet ortamındaki veri en temel olarak web sayfaları içinde HTML *(Hyper Text Markup Language*) etiketleri arasında tutulur. HTML etiketleri tarayıcı tarafından görüntülenecek sayfasının görselliği ayarlayan etiketlerdir. Bu etiketler tarayıcı tarafından görüntülenmezler. Tarayıcının amacı sadece ham veriyi kullanıcıya vermektir.

Teknik olarak HTML, Standard Generalized Markup Language (SGML) Document Type Definition (DTD) olarak tanımlanır. SGML ilk olarak IBM tarafından 1960'ların sonlarında, değişik bilgisayar ortamlarında belge taşıma sorununa çözüm olarak GML (General Markup Language) olarak geliştirilmiştir. Zaman içinde GML, SGML olarak International Standards Organization (ISO) tarafından standart haline getirildi.

Bir SGML belgesi üç ana parçadan oluşur. İlk parça, etiket ile normal metni birbirlerinden ayırmak için hangi karakter setinin kullanılacağını tanımlar. İkinci parça, etiketlerin uygun olarak kullanılacağı belge tipini tanımlar. Üçüncü parça ise, belgenin asıl metnini ve işaret etiketlerini içerir. Bu üç parçanın hepsi aynı fiziksel dosya içinde olmak zorundadırlar. Bütün HTML tarayıcıları aynı SGML karakter setini ve belge tipini kabul ederler, böylece kullanıcı yalnız metin içeriğini düşünür.

HTML üç kısımdan oluşur. Bunlar;

1. <html> ; Ana bloktur. Tüm html dosyaları "<html> </html>" arasında yer alır.

2. <head> ; Tanımlamalar yani kullanıcının browserde görmediği bölümdür. Bu kısımda site ile ilgili açıklamalar, arama motorları için anahtar kelimeler, site başlığı, CSS ve javasciptler gibi bölümler bulunur.
3. <body> ; Sayfa üzerinde görülecek her şeyin yazıldığı bölümdür. Yani sayfa içeriğinin başlangıç ve bitiş bloğu denebilir.

HTML komutlarının yazılışı:

Html komutları onu yorumlayan tarayıcının anlaması için etiketler kullanılarak yazılır. Bu etiket " < komut1 > metin < / komut1 > " şeklindedir. Mesela yazıları kalın yazmak için;

 cümle veya kelime

kullanılır. Burada < > işaretleri arasındaki "b" bold (kalın) yazı için kullanılan bir html komutudur. Aynı şekilde bu yazıyı kalın ve kırmızı renkte yapmak istersek;

 cümle veya kelime

yazmamız gerekir. Burada dikkat etmemiz gereken nokta font etiketinin içinde bulunan color="red" bölümüdür. Eğer bir yazıya herhangi bir özellik vermek istersek font etiketini kullanıp içerisinde özellikleri belirtmeliyiz. Merhaba yazısının fontunu verdana yapmak istersek

<font color="red" face="verdana"> yazmamız gerekirdi.

3.2.1.2. XML

XML (Extensible Markup Language) HTML ile pek çok açıdan benzerlik gösteren bir biçimleme dilidir. Verinin tanımlanması ve tarif edilmesi için kullanılır. HTML'deki yapının aksine XML'de kullanılacak olan etiketlerin önceden tanımlı değildir. Yani bir XML dokümanının yapısı tamamıyla kullanıcı tarafından oluşturulur. Verinin tarif edilmesi için DTD adı verilen yapılar kullanılmaktadır. XML ve DTD'nin

birlikte kullanılması ile dokümanlar kendini tarif eden bir yapı halini alırlar. XML ve HTML arasındaki en belirgin fark XML'in verinin kendisiyle ilgilenmesi HTML'in ise verinin sunumuyla ilgilenmesidir. Buna bağlı olarak HTML dokümanları veriye ilişkin şekillendirme bilgilerini içerirken XML dokümanları ise verinin tanım bilgilerini içermektedir. XML'in tasarım amaçlarından biri de verinin taşınmasıdır. Bahsedilen bu özellikleri incelendiğinde XML'in pek çok önemli işlevi yerine getirdiği görülmektedir. Burada önemli bir nokta olarak XML'i HTML'in yerine geçecek bir dil olarak düşünmek yerine HTML'in tamamlayıcısı olacak olan bir dil şeklinde düşünmek uygundur. Günümüz bilişim dünyasına bakacak olduğumuzda XML'in her alanda karşımıza çıktığını görmekteyiz. Bu nedenle XML bir anlamda geleceğin web dili olarak tanımlamak mümkündür.

3.2.1.3. Diğer depolama biçimleri

Bir web sayfası sadece HTML veya XML etiketlerinden oluşmaz. Sayfanın arkasında ilk bakışta anlamakta zorlanacağımız komutlar bulunmaktadır. Bu komutlar sayfa içinde bulunan her şeyin nasıl gösterileceğini kullanılan tarayıcıya anlatmak için kullanılır. Yani sayfada bulunan resim, flash, yazı gibi materyallerin yan yana anlaşılır görünmesi için HTML kullanmak zorundayız.

Bu dosyaların yanı sıra metin, kelime işlemci, pdf, sunum ve ghost birçok dosya biçimine internet üzerinden erişmek mümkündür. Arama motorlarının çoğu bu dosyaların içinden de veriler çekip indekslemektedir. Bu sayede bir arama yaptığımızda bu dosyalara ulaşabiliriz.

3.2.2. HTML ve XML ayrıştırıcılar

HTML bir belge etiketlerinin iyi şekilde analiz edilmesi sonucu kolayca gerekli HTML etiketlerinden ayrıştırması yapılabilir. Ayrıca bu ayrıştırma işlemi internette çeşitli kütüphaneler mevcuttur. Bu kütüphanelerden en başlıca olanları:

- HTML Parser: Java için geliştirilmiş bir kütüphane topluluğudur. HTML etiketleri içinden metinlerin ve bağlantıların kolayca ayrıştırılmasını sağlar.
- HTML Parser for PHP: PHP için geliştirilmiş benzer özelliklere sahip bir kütüphane topluluğudur.
- DIHtmlParser: Delphi için geliştirilmiş bir kütüphanedir.

Aslında bir web sayfası içinden sadece istenen metin, cümle veya kelimelerin çekilmesi için etiketlerin çok iyi şekilde yorumlanması yeterli olabilir. Bölüm 4.2.2'de Delphi'deki Web Browser bileşenini kullanarak etiket içinden gerekli verinin elde edilmesi ayrıca anlatılacaktır.

XML ayrıştırıcılara bakacak olursak DOM (Document Object Model) ve SAX (Simple API for XML) ayrıştırıcılar göze çarpmaktadır.

DOM ile XML içeriğini, stilini ve yapısını görüp değiştirebilirsiniz. DOM, XML belgeyi alıp kök elemanlardan başlayarak bir ağaca yerleştirir. Bu yerleşim tamamlandıktan sonra DOM metotlarını kullanarak ağaç üzerinde dolaşabilirsiniz.

SAX ise David Magginson tarafından yürütülen açık kaynak kodlu bir üründür. DOM W3C tarafından onaylanmış olmasına rağmen SAX böyle bir özelliği yoktur. DOM'dan bahsederken tüm XML belgesi için bir ağaç yapısı oluşturulduğunu ve bunun belleğe yerleştirildiğini söylemiştik. İşte SAX ayrıştırıcıları bu soruna çözüm olarak ortaya çıkmıştır. Özellikle sistem kaynaklarının yetersiz kalabileceği durumlar için kullanılmaktadır. DOM ve SAX kısaca karşılaştırırsak:

DOM	SAX
DOM tüm XML belgesini okur ve tamamını hafızaya bir ağaç yapısı olarak yerleştirir.	SAX belgeyi okur ve okuduğu kısım ya da elemente ait tanımladığımız metodu çağırır
DOM tüm belgeyi hafızaya yüklediğinden rastgele erişime (Random Access) izin verir.	SAX ise ancak sıralı erişime (Sequential Access) izin verir.
DOM tüm belgeyi hafızaya yerleştirdiğinden yavaştır ve büyük XML belgeleri için tercih edilmez.	SAX daha hızlıdır ve daha az hafızaya ihtiyaç duymaktadır. Bu nedenlerden ötürü büyük XML belgelerinde tercih edilmektedir ve WEB uygulamalarında daha popülerdir
Bazı DOM versiyonlarında hafızaya yerleştirilen XML belgesini değiştirmek için metotlar vardır. (UPDATE)	Bu tarz metotlar SAX için bulunmamaktadır.

Tablo 3.1. DOM ve SAX'ın karşılaştırılması

3.3. Arama motorları

3.3.1. Arama motoru nedir?

Arama motoru (search engine), Internet üzerindeki web sitelerini başlıklarına, açıklamalarına, anahtar kelimelerine ve içeriklerine göre indeksleyen sistemlerdir.

İnternet üzerinde yüzlerce hatta binlerce arama motoru bulunmaktadır. Bunların bir kısmı kendi alanlarındaki web sitelerini listelemekte, bir kısmı yerel alanlarda hizmet vermektedir (www.trakya.edu.tr). Bir kısmı da dünya üzerindeki her türlü web sitesini listelemektedir (www.google.com, www.yahoo.com).

Arama motorları 3 temel parçadan oluşur:

- *Spider:* İlki "spider,bot,ant" gibi isimlere sahip olan programlar. Bu programlar interneti dolaşıp sayfaları tespit edip veritabanlarına kaydederler.
- *Veritabanı:* Spider tarafından ziyaret edilen, her sayfanın kopyası burada saklanır. Sayfanızda yaptığınız değişiklikler, spider tekrar uğrayıncaya kadar arama motorlarında yer almaz. Arama yapıldığında, sayfanız veritabanındaki son haline göre değerlendirilip, sıralanır. Bu yüzden kullanıcılar, ölü bağlantılar veya çok farklı içeriklerle karşılaşabilmektedir. Google.com'da "Önbellekten oku" komutuyla kayıtlı sayfanın Google'un veritabanındaki halini görebilirsiniz.
- *Sıralama mekanizması:* Üçüncü parçada sıralama mekanizmasıdır. Kullanıcın yaptığı aramaya göre, en uygun şekilde sayfaları sıralamaya çalışır. Her arama motorunda bu parçalar farklı çalışır. Bazılarında, sayfanızda çerçeve, resim ilişkileri kullanmanız sorun yaratabilir. Bazılarında ise veritabanına alt etiketleri kaydetmez.

Her biri kendi yöntemini geliştirerek, ziyaretçilerine en iyi sonuçları getirmeye çalışır. Arama motorları, sayfanızı inceleyip kelimelerin kullanılma oranlarını, yerlerini ve şekillerini ölçer. Bir kelime diğerlerine göre, çok sık olarak, hem başlıkta, hem yazılarda, hem resimlerin açıklamalarında kullanılmışsa, o sayfanın o kelimeye yönelik içeriğe sahip olduğu kanısına varılır. O kelimeyle arama yapıldığında, o sayfa, kelimeyi daha az kullanan ya da başlığında o kelimeye yer vermeyen sitelere göre daha üst sırada yer alır.

3.3.2. Arama motorlarına HTML tabanlı erişim

Herhangi bir tarayıcından bir arama işlemi yaptığımızda arama motoru sunucusu tarafında bir HTML belgesi oluşturulup arama yapılan bilgisayara sonuçlar gönderilir. Bu sırada tarayıcının adres çubuğu bölümünde çeşitli değişimler olur. Örneğin Google üzerinden bir arama yaptığımızda aşağıdaki satırlarla karşılaşırız.

http://www.google.com.tr/search?hl=tr&q=gidecek

"search?" arama yapılacağı anlamına gelmektedir. "q" parametresi arama yapılan kelime içeriği içerir. "&" karakteri ile farklı parametreler birbirine bağlanır. "hl" parametresi ise dil ara yüzü hakkındaki bilgiyi taşır. Bu parametreler hariç görüntülenecek sayfası için "num" ve sonuç dili için "lr" parametreleri kullanılır. Örneğin "gidecek" kelimesi için ilk 100 kaydı ve dil ara yüzü - sonuç dili olarak Türkçe'yi seçersek adres çubuğunu şu satırları yazmamız yeterlidir.

http://www.google.com.tr/search?hl=tr&q=gidecek&num=100&lr=tr

3.3.3. Arama motorlarına API'ler üzerinden erişim

Google ve Yahoo gibi arama motorlarına özel API'ler üzerinden de ulaşmak mümkündür. Bu API'lere bir önceki bölümdeki parametrelere benzer parametreler gönderilerek XML belgesi elde edilebilir. XML belgesinin HTML belgeye göre çözümlenmesi çok daha kolaydır. Fakat her iki API'de kısıtlar içerir. Örneğin Google API sistemine üye olunması gerekir. Ayrıca her üye her gün sadece 1000 sorgu yapabilmesine olanak tanınmaktadır. Yahoo API ise belli bir IP üzerinden günlük belli sayıda sorgu alınabilmesi olanak tanır. Bu kısıtlardan dolayı API kullanmak şu anlık pek uygun bir metot değildir.

3.3.4. Arama motorlarında karşılaşılan zorluklar

Arama motorları her ne kadar çok hayranlık uyandırıcı yönleri olsa da, dilbilimsel açıdan bazı eksiklikler göze çarpmaktadır. Kilgariff ve Grefentette (2003) göre bu eksiklikler:

- Bir arama motoru yeterince örnek içermeyebilir (maksimum 1000 veya 5000)
- Her örnek için yeterince içerik mevcut olmayabilir. (Google 10 adet kelime çevresinde çalışabilir)
- Arama motorları belli bir ölçüte göre seçim yapar, genelde dilbilimsel perspektif göz ardı edilir.
- Arama motorundan elde edilen istatistiksel sonuçlar güvenilmez olabilir.

Türkçe açısından arama motorlarına baktığımızda çalışmamız açısından üç eksik göze çarpar:

- Arama motorlarına kök kelime verildiğinde sadece kök kelimeye ait bilgiler görüntülenir. Oysa Türkçe bir kelimenin yanına binlerce ek alabilen bir dildir.
- Arama sonuçlarında Türkçe için Unicode sorunu ile karşılaşılabilir. Bunun sebebi bazı yazılım ve web uygulamalarının Türkçe karakterleri desteklememesinden kaynaklanmaktadır.
- Arama sonuçlarından uygun cümlelerin seçimini yapmak zordur. Özellikle durak karakterlerinin veya kelimelerin tayini yapılmalıdır.

3.3. Morfolojik Analiz

3.3.1. Kelime Bulucu

Kelime bulucu (tokenizer), metin içindeki cümlelerin içindeki kelimelerin ayrıştırılmasında kullanılır. Mesela iki kelime arasında bir boşluk ayırt edici bir özellik olarak kullanılabilir. Örneğin "Ali kitabı Ayşe'ye verdi" cümlesini kelime bulucuya gönderdiğimizde aşağıdaki çıktı ile karşılaşırız:

Ali
Kitabı
Ayşe'ye
Verdi

Örnek 3.1. Türkçe cümle örneği

3.3.2. Kategori Bilgisi Etiketleyici

Kelime Bulucu tarafından bulunan kelimeler *kategori bilgisi etiketleyici* (Part-of-speech tagging) sayesinde kelimenin fiil, isim, sıfat ve benzeri özellikleri işaretlenir. Örneğin "Ali kitabı Ayşe'ye verdi" cümlesini kategori bilgisi etiketleyiciye gönderdiğimizde aşağıdaki gibi bir çıktı üretilir:

<kişi_ismi> Ali </>
<isim> Kitabı </>
<kişi_ismi> Ayşe'ye </>
<fiil> Verdi </>

Örnek 3.2. Türkçe cümle örneğinin kelimelerinin etiketlenmesi

3.3.3. Kök Bulucu

Kök bulucu (stemmer), verilen kelimenin eklerini ayırarak kök kelimeye ulaşmaya çalışır. Kök bulucu özellikle kategori bilgisi etiketleyici ile bir arada çalışır. Örneğin "Ali kitabı Ayşe'ye verdi" cümlesini kök bulucuya gönderdiğimizde:

<kişi_ismi kök = Ali> Ali </>
<isim kök=kitap hal=i > Kitabı </>
<kişi_ismi kök=Ayşe hal=e > Ayşe'ye </>
<fiil kök=ver zaman=di kişi=o> Verdi </>

Örnek 3.3. Türkçe cümle örneğinin kelimelerinin ayrıntılı bir şekilde etiketlenmesi

İngilizce, Almanya ve Fransızca gibi dillerde kök bulmak Türkçeye göre çok daha kolaydır. Örneğin İngilizcede bir fiil için 4 durum vardır. Türkçe ise sondan eklemeli bir dil olduğu için bir fiil için 1000'lerce değişik durumdan oluşabilir. Kemal Oflazer'e göre bir fiil yaklaşık 40,000 farklı ek alabilir [Jurafsky ve Martin, 2000 - syf: 59]. Örneğin gitmek fiilini ele alalım. İngilizce için:

go (goes) Going Went Gone

Örnek 3.4. İngilizce için bir fiilin durumları

Türkçe için ise binlerce farklı durum olabilir. Örneğin gitmek fiili aşağıdaki durumda olabilir:

Gidemiyormuşum Git: kök, yönelme: e, olumsuzluk: mi, şimdiki zaman: yor, şart: muş, kişi eki: um

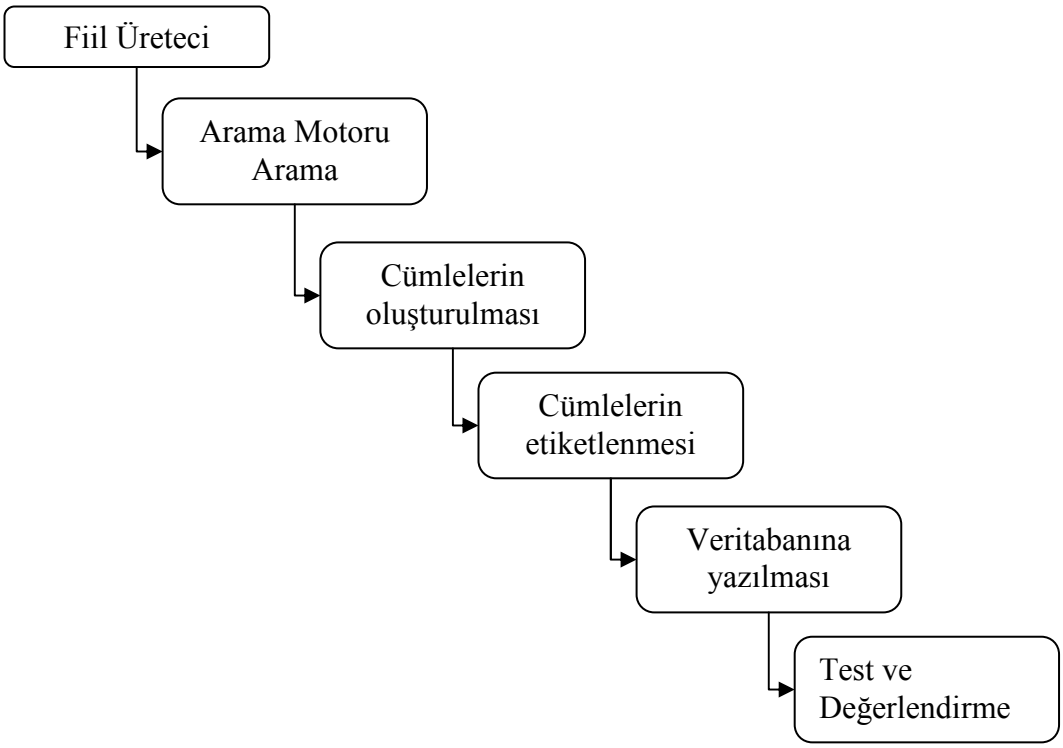
Örnek 3.5. Türkçe için bir fiilin alabileceği durumlara örnek

Türkçe için bu takı analizi açık kaynak kod olarak geliştirilen "zemberek" programı ile çözülebilir. (https://zemberek.dev.java.net/)

BÖLÜM 4.

TASARIM ve UYGULAMA

4.1. Sistemin tasarımı



Şekil 4.1. Sistemin çağlayan modeli

Sistem 6 ana bileşen üzerine kurulmuştur. Bu bileşenler:

- Fiil Üreteci: Bir kök fiilin alabileceği tüm durumları üretilmesi
- Arama motorunda arama: Bir kök fiilin tüm durumlarının arama motorları üzerinde sorgulanmasını yapılması

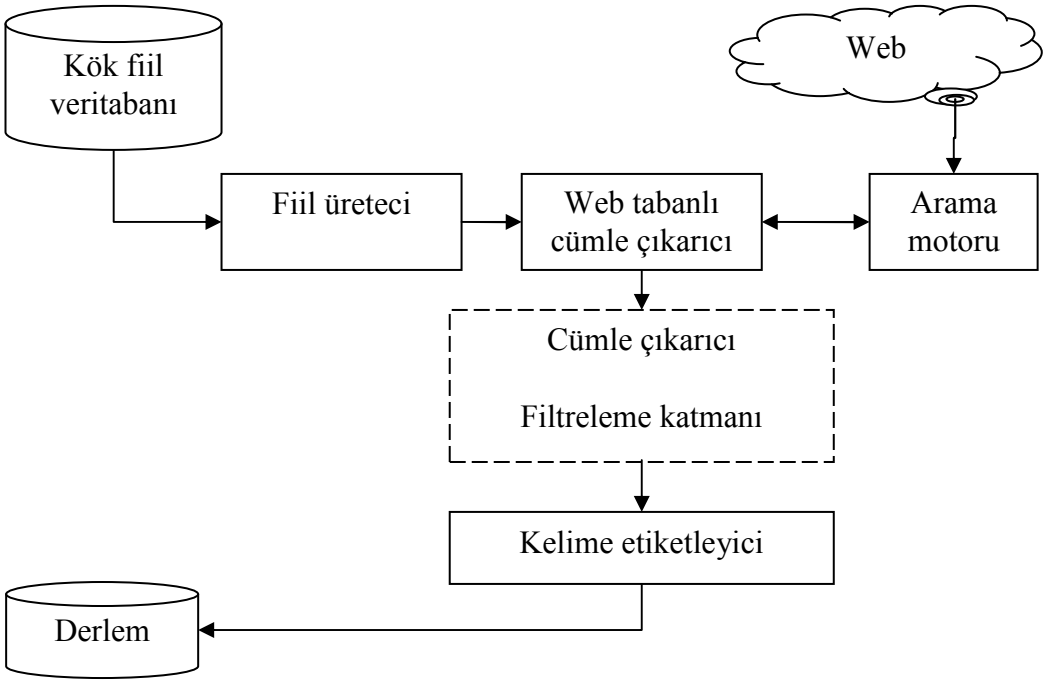
- Cümlelerin oluşturulması: Sorgulama sonuçlarından uygun metinlerin seçimi ve cümlelerin üretilmesi
- Cümlelerin etiketlenmesi: Cümlelerin öğrenme işlemi için uygun şekilde etiketlenmesi
- Veritabanına yazılması: Cümle ve cümlenin etiket kodları veritabanına yazılması
- Test ve Değerlendirme: Bir makine öğrenme metodu ile cümlelerin test edilmesi ve test sonuçlarının değerlendirilmesi

4.2. Geliştirilen uygulama

Geliştirilen uygulama için üç ana bileşenden oluşmaktadır. Bu bileşenler:

- Fiil üreteç
- Web tabanlı cümle çıkarıcı
- Kelime etiketleyici

Fiil üreteci veritabanından aldığı kök fiillerin çeşitli durumlarını üreterek web tabanlı cümle çıkarıcı uygulamasına gönderir. Web tabanlı cümle çıkarıcısı arama motorundan aldığı sayfaları ayrıştırarak uygun cümleleri sayfalar içinden çıkarmaya çalışır. Çıkan cümleler iki adet filtrelemeden geçer. Birinci filtreleme illegal karakterlerin siler veya düzeltmeye çalışır iken ikinci filtreleme ise uygun Türkçe cümleleri tespit eder. Çıkan cümleler kelime etiketleyicisine gönderilir. Kelime etiketleticisi teker teker kelimeleri etiketleyip veritabanına yazar. (Şekil 4.2.)



Şekil 4.2. Geliştirilen uygulama

4.2.1. Fiil Üreteci

Türkçede bir fiil olumsuzluk, edilgen, yapabilirlik, belirli kip, belirli bir zaman, belirli bir kişi ve soru eklerinden birini veya tümünü alabilir. Bu ekleri 4 ana bölümde toplayabiliriz.

Birinci bölüm tüm fiillerin alabileceği ekler:

- Olumsuz eki: -me / -ma
- Edilgen eki: -il / -in / -ıl / -ın
- Yapabilirlik: -ebil / -abil
- Soru Eki: -mi / -mı

İkinci bölüm fiilin alabileceği zamanlar bölümü:

- Geçmiş zaman: -di / -dı / -du / -dü / -ti / -tı / -tu / -tü
- Dolaylı geçmiş: -miş / -mış / -muş / -müş
- Geniş: -r
- Şimdiki zaman: -iyor / -ıyor / -uyor / -üyor
- Gelecek zaman: -ecek / -acak
- Gereklilik zaman: -meli / -malı
- Dilek şart: -se / -sa
- İstek: -e / -a

Üçünü bölüm fiilin alabileceği kipler:

- Bildiri
- Anlatma: -di / -dı / -du / -dü / -ti / -tı / -tu / -tü
- Söylenti: -miş / -mış / -muş / -müş
- Şart: -se / -sa

Dördüncü bölüm ise fiilin alabileceği kişi ekleridir:

- Ben: m
- Sen: -n
- O: -
- Biz: -k ve -z
- Siz: (s)-ınız / -iniz / -unuz / -ünüz
- Onlar: - / -ler / -lar

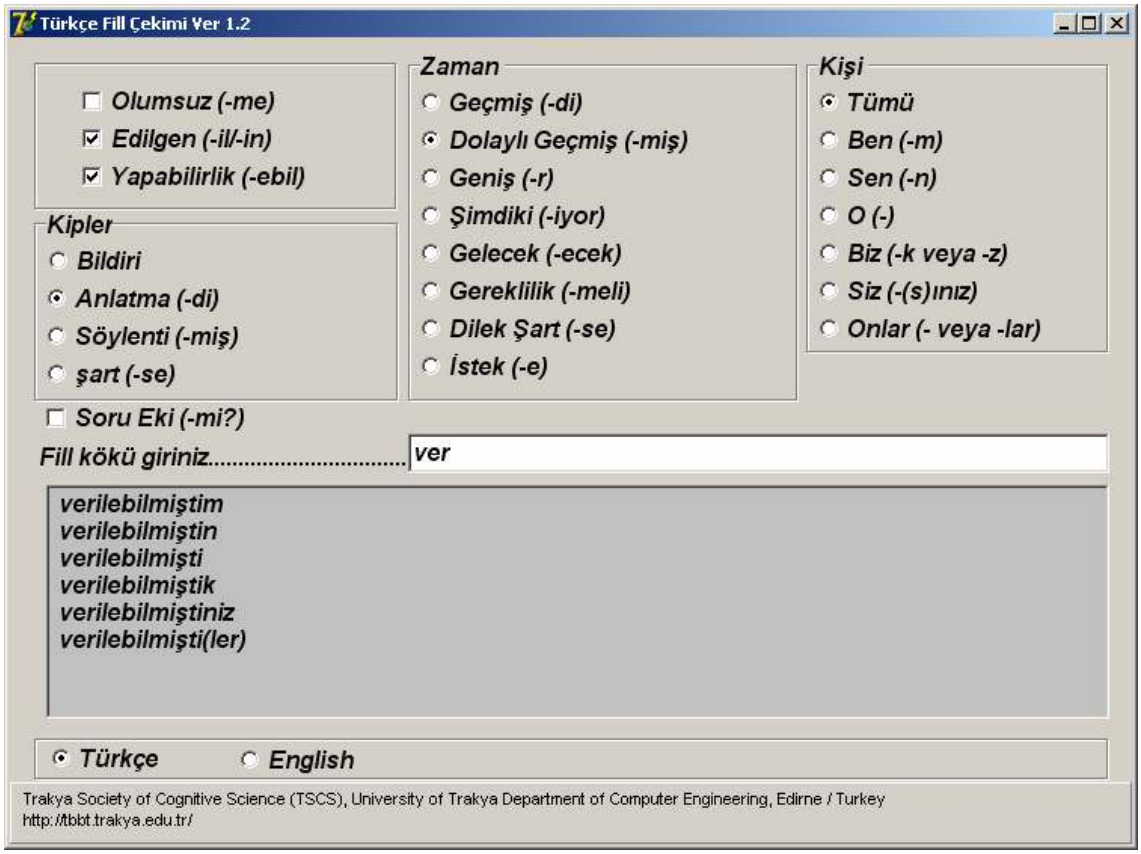
Bir fiil birinci bölümdeki durumlardan hiçbirini veya her birini içerebilir. Diğer kalan üç bölümde kendi arasında çeşitli kombinasyonlar oluşturabilir ama her bölüm içinden sadece bir ek kullanılabilir. Örneğin “verilebilmiştim” fiilini inceleyelim.

Ver - il - ebil - miş - ti - m

Ver: fiil kökü
-il: edilgen yapı eki
-ebil: yapabilirlik eki
-miş: geçmiş zaman eki
-ti: anlatma kipi
-m: ben kişi eki

Örnek 4.1. Türkçe için bir fiilin alabileceği durumlara örnek

Türkçede üç adet kural dışı durum söz konusudur. Birincisi, Geçmiş zaman eki var iken kesinlikle söylenti kip eki getirilemez. Diğerleri dilek şart ve istek zamanlarından sonra şart kip eki getirilemez. Bu durumları da göz ardı ettiğimizde Türkçe için bir fiil 1198 tane durumdan oluşur. (Şekil 4.3.)



Şekil 4.3. Fiil Üreteç Uygulaması

Uygulama için zamanları içeren 8 adet fonksiyon geliştirilmiştir. Bu fonksiyonlarda yanı sıra sesli harf kontrol fonksiyonu, edilgen, yapabilirlik, olumsuzluk ve soru eki fonksiyonları bulunmaktadır. Örneğin bir geçmiş zaman fonksiyonu incelersek:

DiliGecmisZaman(kök fiil, olumsuzluk durumu, edilgen durumu, yapabilirlik durumu, anlatma kip durumu, söylenti kipi durumu, şart kipi durumu, soru durumu)

Fonksiyona kök fiil ve 7 adet durum gönderilir. Bu kök ve durumlara bağlı olarak fonksiyon gerekli çıktıyı üretir. Fonksiyonların diğer bir özelliği ise birbirlerini çağırabilmesidir. Örneğin geçmiş zaman fonksiyonu, anlatma kipi için tekrar kendini çağırır.

4.2.2. Web tabanlı cümle çıkarıcı

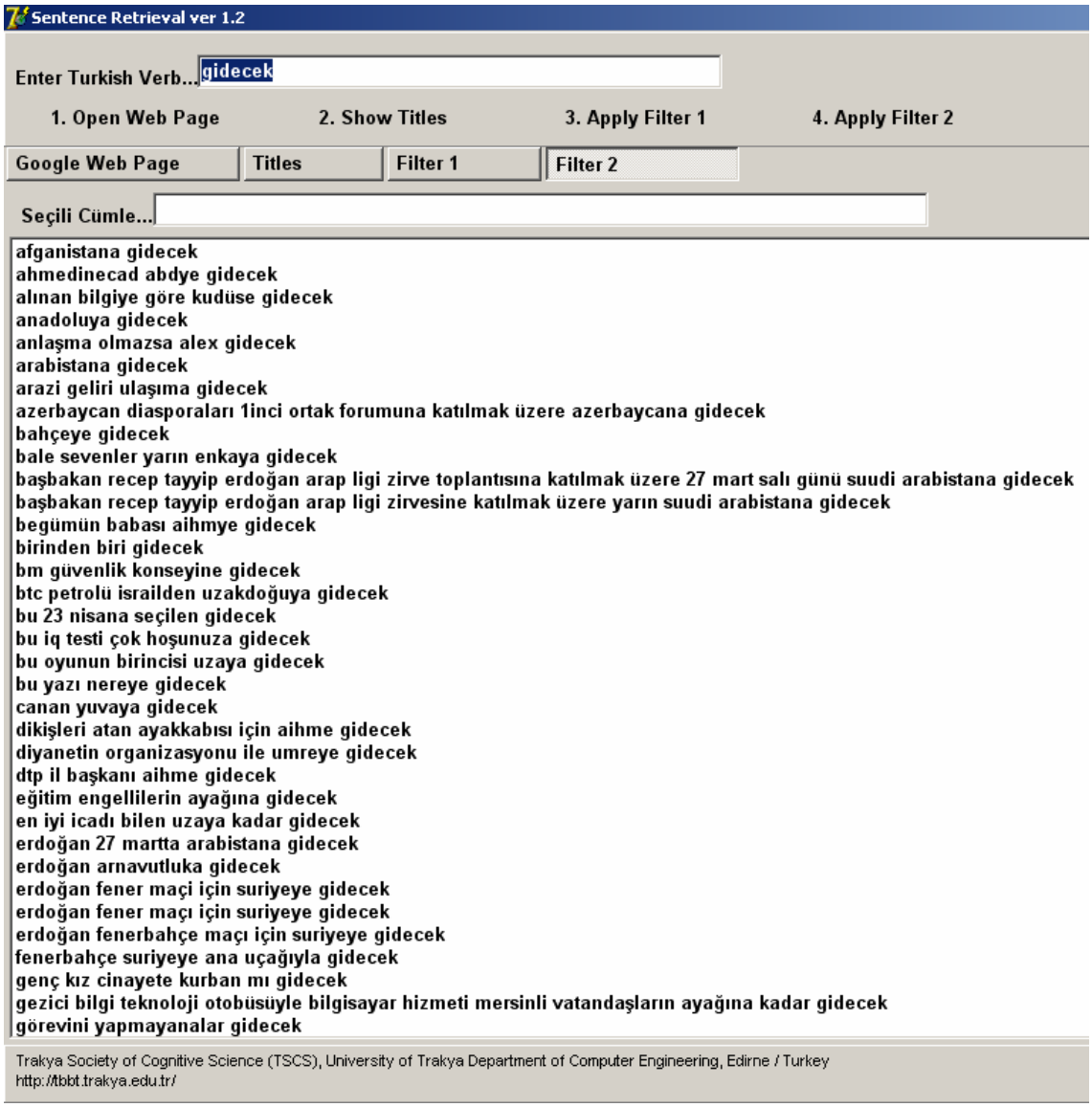
Web tabanlı cümle çıkarıcısın iki ana görevi vardır. (Şekil 4.4.)

- Arama motorlarına sorgu göndermek
- Alınan sorgu sonuçlarını ayrıştırıp, uygun cümleleri bulmak

Arama motorları sorgu gönderebilmek için otomatik olarak sorgu tümcesinin oluşturulması gerekir. Bu sorgu tümcesi arama motorlarının arama kriterlerine uygun olarak belirlenmelidir. Örneğin google arama yapmak için

http://www.google.com.tr/search?hl=tr&q=gidecek&num=100&lr=tr

türünde bir tümce oluşturulmalıdır. Bu tümce içinde q parametresi arama parametresidir. Bu parametre her fiilin duruma göre değişir. Sorgu hazırlandıktan sonra uygun programlama dili bileşeni ile sorgulanır ve sonuçların döndüğü bir web sayfası elde edilir. Örneğin Delphi için WebBrowser bileşeni kullanılır.



Şekil 4.4. Web tabanlı cümle çıkarıcı

Sorgu sonuçları alındıktan sonra oluşan html dosyanı ayrıştırılması işine geçilir. Çekilecek olan metinler belli etiketler arasında bulunmaktadır. Bu etiketler dinamik olarak oluşturulmaktadır. Bu yüzden etiketler uygun şekilde belirlenirse etiketler arasından uygun metinler çekilebilir. Sorgu sonucu üretilen Google sayfasında bir başlık, başlığın hemen altında bir açıklama ve bu başlığın içinde bir bağlantıdan oluşur. Başlık “ ” etiketi ile başlar ve “” etiketi ile biter. “%link%” arası bağlantılı olan web sayfasının adresi vardır. Bu adres değişken bir

değerdir. Açıklama bölümünü çekmek için ise “<td class=j><font size=-1>” etiketi ile başlar ve “<br>” etiketi ile sonlanır. (Yalancı Kod 4.1.)

metin_kopyala: metinden belli bir karakterden başlayıp istenilen karaktere kadar alan fonksiyon. Birincisi metin, ikincisi başlangıç noktası ve üçüncüsü bakılacak toplam karakter sayısı olmak üzere üç parametreden oluşuyor.
metin_al: uygun metinleri toplayan fonksiyon.

```
x = 0  :gösterge
temp_str: web sayfasından alınan metin
başla_link, basla_başlık, basla_açıklama = Yanlış: Doğru/Yanlış
metin_başlık, metin_açıklama = ' ' : geçici değişenler

WHILE metindeki tüm karakterler için

        IF metin_copyala(temp_str, x, 16) = '<a class=l href=' THEN
        x = x + 17;
        başla_link := Doğru;
        END IF

        IF (basla_link = Doğru) AND (temp_str[x]= '>') THEN
        x = x + 1;
        başla_başlık = Doğru;
        basla_link = Yanlış;
        END IF

        IF (basla_baslik = Doğru) AND (metin_kopyala(temp_str, x, 4)= '</a>') THEN
        x = x + 5;
        başla_başlık = Yanlış
        metin_al(str_başlık)
        str_başlık = ' ';
        END IF

        IF metin_kopyala(temp_str, x, 26)= '<td class=j><font size=-1>' THEN
        x = x + 26;
        başla_açıklama = Yanlış
        END IF

        IF (basla_açıklama = Doğru) AND
        (metin_kopyala(temp_str, x, 18)= '<br><span class=a>') THEN
        x = x + 18;
        başla_açıklama = Yanlış
        metin_al(str_açıklama)
        str_açıklama = ' ';
        END IF
```

```
IF başla_başlık THEN
  str_başlık := str_başlık + temp_str[x];

IF başla_açıklama THEN
  str_açıklama := str_açıklama + temp_str[x];

x = x + 1;

END WHILE
```

Yalancı Kod 4.1. Google başlık ve açıklama satırlarını ayrıştıran kod

Uygun olan metinler toplandıktan sonra filtreleme aşamasına geçilir. Filtrele iki ana aşamadan oluşur:

- İllegal kelimelerin silinmesi veya düzeltilmesi
- Türkçe açısından uygun cümlelerin belirlenmesi

Metin içindeki illegal karakterler iki sebebi olabilir. Metin içinde HTML etiketleri kalmış olabilir (Tablo 4.1.) veya bazı Türkçe karakterler Türkçe karakter desteklemeyen programlar sebebiyle başka karakterlere dönüşebilir (Tablo 4.2.).

Etiket	Açıklama
	Kelimelerin Kalın yazılmasını sağlar
	Kalın yazılmanın bitişi
	Kelimelerin İtalik yazılmasını sağlar
	İtalik yazılmasının bitişi

Tablo 4.1. İllegal HTML karakterleri

Ã‡ , Ã§	Ç
Ä□ , ÄŸ	Ğ
Ä° , Ã□	İ
Ä± , Ã½	I
Ã– , Ã¶	Ö
Å□ , ÅŸ	Ş
Ãœ , Ã¼	Ü

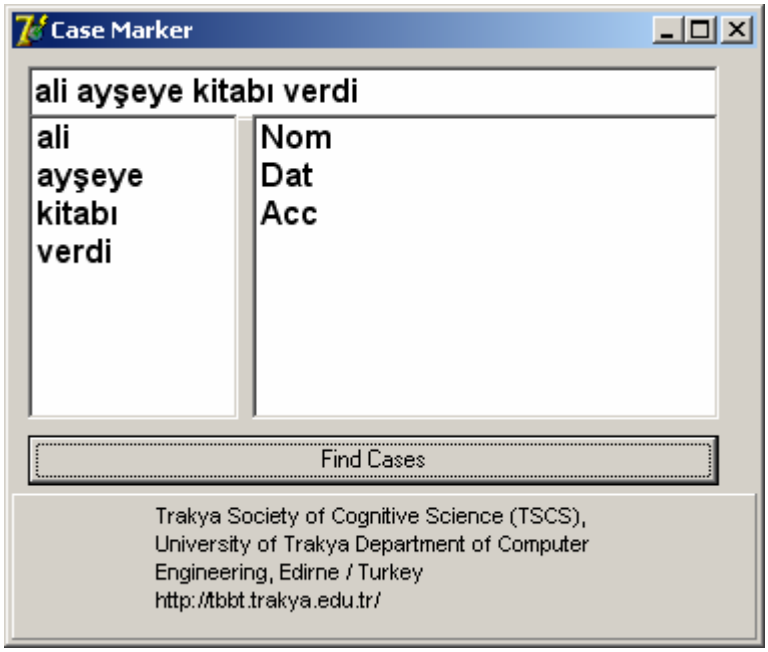
Tablo 4.2. Değiştirilmesi gereken değişeme uğrayan bazı Türkçe karakterler

â€™ , ' , €œ , => , " , > , < , € , Ã¾ , □ , ¤ , ¢ , ¦ , «

Tablo 4.3. Silinmesi gereken karakterler

4.2.3. Kelime Etiketleyici

Kelime etiketleyici cümlelerin içindeki kelimelerin hal durumlarına karar vermeye çalışır. (Şekil 4.5.)



Şekil 4.5. Kelime etiketleyici

Kelime etiketleyici 6 hal durum ekine karar verir. Bu hal ekleri:

- Yalın hal (nominative - nom)
- -i hali (accusative - acc)
- -e hali (dative - dat)
- -den hali (ablative - abl)
- -de hali (locative - loc)
- -ile hali (instrumental - inst)

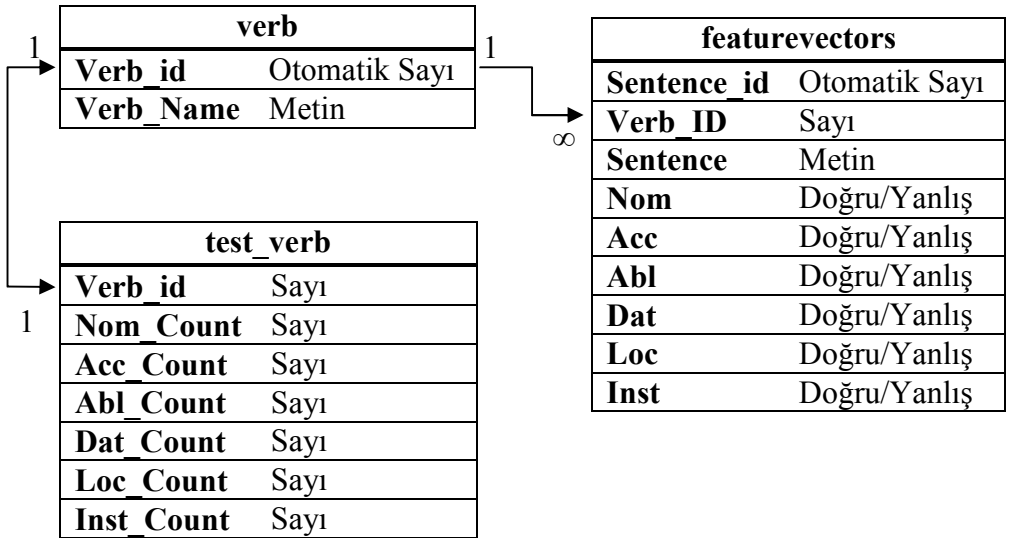
Kelime etiketleyici uygulaması öncelikle sözlük veritabanından kelimenin kök olup olmadığının kontrolünü yapar. Eğer kök ise kelime üzerinde hiçbir işlem yapmaz, başka bir deyişle kelime yalın hal içerir. Eğer kelime kök değilse hal fonksiyonlarına gönderilir ve kelimenin hal eki tespit edilmeye çalışılır.

```
kelime: cümlenin içerdiği kelimeyi tutar
kelime_hal: kelimenin hal ek durumu
veritabanı_var: kelimenin veritabanında olup olmadığını döndüren fonksiyon (Doğru / Yanlış değeri döndürür)
acc, abl, dat, loc, inst: kelimenin hal durum ekini içerip içermediğini döndüren fonkiyon (Doğru / Yanlış değer döndürür)

x=0: kelime göstergesi

WHILE tüm kelimeler – 1

  IF veritabanı_var ( kelime[x] ) THEN
    kelime_hal = nom;
  ELSE
    IF acc ( kelime[x] ) THEN kelime_hal = acc;
    IF abl ( kelime[x] ) THEN kelime_hal = abl;
    IF dat( kelime[x] ) THEN kelime_hal = dat;
    IF loc( kelime[x] ) THEN kelime_hal = loc;
    IF inst( kelime[x] ) THEN kelime_hal = inst;
  END IF

END WHILE
```

Yalancı Kod 4.2. Kelime etiketleyici

4.6. Veritabanı Tasarımı

Veritabanı fiillerin tutulduğu "verb" tablosu, internetten elde edilen cümlelerin tutulduğu "featurevectors" tablosu ve test işlemleri için kullanılacak olan "test_verb" tablosu olmak üzere üç tablodan oluşmaktadır.



Şekil 4.6. Veritabanı tasarımı

"verb" tablosunda kök kelime çekilir. Fiil üreteci bir fiile ait tüm durumları üretir(Şekil 4.7.). Web tabanlı cümle çıkarıcı internetten üretilen fiile ait tüm durumları sorgular ve cümleler elde eder. "Sentence_ID", "Verb_ID", "Sentence" değerleri veritabanına yazılır. Oluşan her cümle kelime etiketleyicisi gider. Kelime etiketleyesi "Nom", "Acc", "Abl", "Dat", "Loc", "Inst" değeri doğru veya yanlış şeklinde işaretler (Şekil 4.8.).

Yeterince cümle çekildikten sonra acc, nom, abl, dat, loc, inst değerleri sayılıp "test_verb" tablosuna konur. "test_verb" tablosu öğrenme aşamasında kullanılan bir tablodur.

verbs : Tablo

Verb_ID	Verb_Name
2	bul
3	ver
4	söyle
5	gir
6	bak
10	yaşa
11	anlat
13	gör
18	sor

Şekil 4.7. “verb” tablosu

featureVectors : Tablo

SentenceID	Verb_ID	Sentence	Nom	Acc	Abl	Dat	Loc	Inst
2894	2	eksik parçamı bulmuşum	☑	☑	☐	☐	☐	☐
2895	2	2006 türk patent ödülleri sahiplerini buldu	☑	☑	☐	☐	☐	☐
2896	2	ahmet hakan mona rozayi buldu	☑	☑	☐	☐	☐	☐
2897	2	altın örümcek sahiplerini buldu	☑	☑	☐	☐	☐	☐
2898	2	altın portakallar sahiplerini buldu	☑	☑	☐	☐	☐	☐
2899	2	amerikan müzik ödülleri sahiplerini buldu	☑	☑	☐	☐	☐	☐
2900	2	bgh mutasaddiki suçlu buldu	☑	☑	☐	☐	☐	☐
2901	2	bilim adamları yeni element buldu	☑	☑	☐	☐	☐	☐
2902	2	bitirme projesi ödülleri sahiplerini buldu	☑	☑	☐	☐	☐	☐
2903	2	bond kızını buldu	☑	☑	☐	☐	☐	☐
2904	2	cehennemi yaşadı canan ile cenneti buldu	☑	☑	☐	☐	☐	☑

Şekil 4.8. “featurevectors” tablosu

BÖLÜM 5.

ÖĞRENME YÖNTEMLERİNİN TEST EDİLMESİ ve DEĞERLENDİRİLMESİ

5.1. Toplanan Veri

Öncelikle MikroBeta Türkçe Sözlüğünden (www.turkcesozluk.org) rasgele 200 fiil seçilmiş ve bu fiillere ait alt öğeleme listeleri manüel olarak sisteme girilmiştir. Web tabanlı cümle çıkarıcı uygulaması sayesinde seçilen fiillere ait 16,841 adet örnek alt öğeleme listeleri oluşturulmuştur. Kelime etiketleyici ile toplanan örnekler etiketlenmiştir.

5.2. Gözetimsiz öğrenme

Alt öğeleme listelerin bulunması için gözetimsiz öğrenme tekniği olarak yaygın bir şekilde kullanılan testler *maksimum benzerlik* ve *T-Puanı* hipotez testleridir. Bu yöntemler saf istatistiksel sonuçlara dayanır. Ayrıca, Briscoe and Carroll'ın önerdiği teknikte gözetimsiz öğrenme tekniklerinden biridir. Bu bölümde bu üç teknik hakkında genel bilgiler ve sonuçlar verilecektir.

5.2.1. Maksimum benzerlik

Dunning (1993) göreceli olarak ufak örneklemelerde maksimum benzerlik hipotez testinin iyi sonuçlar verdiğini göstermiştir. Bu hipotez testinde gözlenen hal durumunun (f) dağılımı bağımsız fiil (v) dağılımıdır. Yani bu hipotez $p(f \mid v) = p(f \mid \neg v)$ şeklinde ifade edilebilir. Bu hipotez gözlenen hal durumu ve fiil ilişkisine bağlıdır. Bu hipotez testi *f* ve *v* içeren 4 formül kullanır. Bu formüller:

1. $k_1 = c(f, v)$	Verilen fiil *v* için bulunan belli bir hal durumunun (*f*) sayısı
2. $n_1 = c(v) = c(f, v) + c(\neg f, v)$	Verilen fiil sayısı (*v*)
3. $k_2 = c(\neg f, v)$	Belli bir hal durumu (*f*) hariç o fiil (*v*) içindeki tüm hal durumlarının geçme sayısı
4. $n_1 = c(\neg v) = c(f, \neg v) + c(\neg f, \neg v)$	Belli bir hal durumunun (*f*) belirli fiil (*v*) hariç diğer tüm fiillerde bulunma sayısı

Maksimum benzerlik formülasyonu:

$$-2\log\lambda = 2[\log L(p_1, k_1, n_1) + \log L(p_2, k_2, n_2) - \log L(p\ , k_1, n_2) - \log L(p, k_2, n_2)]$$

burada,

$$\log L(p\ , k\ , n\ ) = k\log p + (n-k)\log(1-p)$$

ve

$$p_1 = \frac{k_1}{n_1},\ p_2 = \frac{k_2}{n_2},\ p\ = \frac{k_1 + k_2}{n_1 + n_2}$$

5.2.2. T-Puanı

Verilerin arasındaki ilişkileri tespit etmeye çalışan diğer bir hipotez testi T-Puanıdır. Bir önceki bölümdeki tanımlara kullanarak T-Puanı formülü:

$$T = \frac{p_1 - p_2}{\sqrt{\sigma^2(n_1, p_1) + \sigma^2(n_2, p_2)}}$$

burada,

$$\sigma(n, p) = np(1 - p)$$

Bu hipotez testi *p1* ve *p2* arasındaki bağımlılık olup olmadığını test eder. Eğer T değeri belli bir eşik değerinin üzerinde ise hal durumu *f* fiilin alt öğeleme listesine dâhil edilir.

5.2.1. Briscoe ve Carroll'ın tekniği

Briscoe ve Carroll (1997) metinsel çoklu derlemlerden alt öğeleme sözlüğü yapmak için bir teknik ve uygulama sistemi önermişlerdir. Sistemleri ayırt edici 160 alt öğeleme sınıfını tanıyabilecek, fiil tahminleri ile her sınıfa atama yapabilecek ve bağıl frekanslarına bağlı olarak bunları değerlendirebilecek özelliklere sahiptir. Sistemleri etiketleyici, kök bulucu, olasılıklı LR ayrıştırıcı, hal durumu çıkarıcı, hal durumu sınıflayıcı ve hal durumu tahmin edici olmak üzere 6 bileşenden oluşmaktadır. Bu teknik Brent (1993) tarafından önerilen binomial frekansın hipotez testine dayanır. Verilen bir fiil için toplam örnekleme kümeleri *n* ve örnekleme kümelerini içeren her sınıf için girdi olan *m* sistem tarafından tanımlanır. Ayrıca her alt öğeleme sınıfı *i* için durumu için bir olasılık tahmin edilmelidir. Briscoe ve Carroll bu değeri tahmin için Alvey NL tools(ANLT, Boguraev et al. 1987) sözlüğünü ve etiketlenmiş bir derlem olarak Susanne derlemini (Sampson, 1995) kullanmışlardır. Öncelikle ANLT sözlüğü üzerinden her alt öğeleme sınıfı için olasılık değerleri hesaplanmıştır. Bu olasılık değeri için:

$$p(f-i) = (1 - \frac{|anlt\_fiiler\_sunı\_i|}{|anlt\_fiiler|}) \frac{|örnekleme\_i|}{|tüm\_örnekler|}$$

Devamında binomial dağılım formülü kullanılır. Burada n deneme için m kez görülen alt öğeleme durumu ve bir üst formülde bulunan p değeri kullanılır. Binomial dağılıma göre:

$$P(m,n,p) = \frac{n!}{m!(n-m)!} p^m (1-p)^{n-m}$$

m defa veya daha fazla kez gerçekleşme olasılığı hesaplarken:

$$P(m+,n,p) = \sum_{i=m}^{n} P(i,n,p)$$

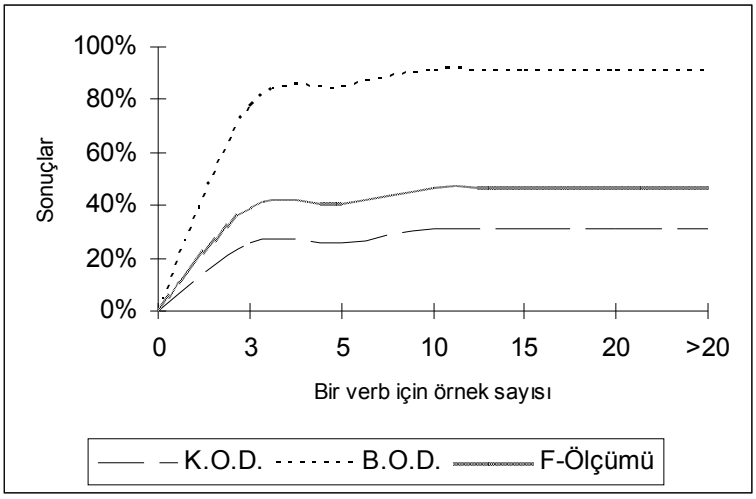
formulü kullanılır.

Son olarak, ortaya çıkan $P(m,n,p(v-i))$ değerinin alt öğeleme listesine dâhil edilip edilmeyeceğine karar verilir. Bunun için hipotez testlerinde olduğu gibi bir eşik değeri belirlenir. Bu eşik değerine göre bulunan alt öğelemenin fiilin alt öğeleme listesine dâhil edilip edilmeyeceğine karar verilir.

5.2.4. Gözetimsiz öğrenme test sonuçları

5.2.4.1. Maksimum benzerlik sonuçları

Saf istatiksel metotlardan biri olan Maksimum benzerlik, örneklemedeki bir hal durumunun frekansına bağlı bir hipotez testidir. Bu hipotez testine göre sonuçlar:



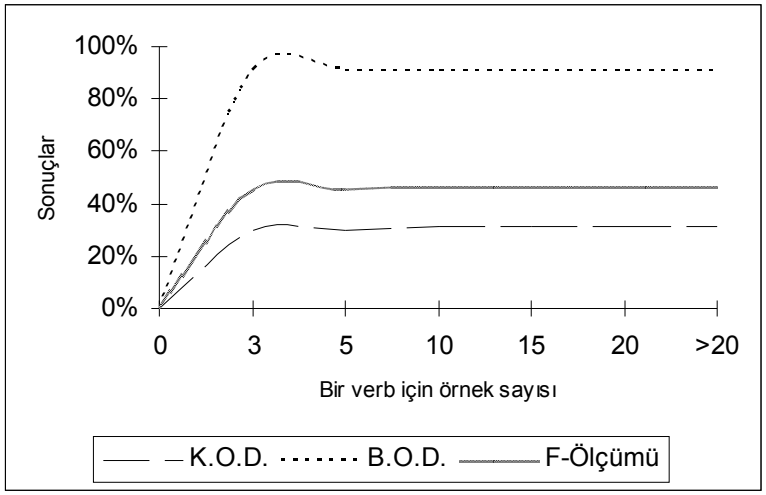
Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	26	77	39
5	26	85	40
10	31	91	46
15	31	91	46
20	31	91	46
>20	31	91	46

Şekil 5.1. Maksimum benzerlik sonuçları

Maksimum benzerlik sonuçlarını incelediğimizde, özellikle her fiil için 10 adet örnekleme ve sonrasında öğrenme sabitleşmektedir. Sistemin kısmi oranlı doğruluğu yüzde 31'de kalırken, bütünsel doğruluk oranı ise yüzde 91'e çıktığını ve böylelikle F-ölçümü değerinin yüzde 46 kaldığı görülmektedir.

5.2.4.2. T-Puanı sonuçları

Diğer bir saf istatiksel hipotez testi T-Puanı'dır. Bu test verilerin arasındaki ilişkileri tespit etmeye çalışır. Bu testin sonuçları:



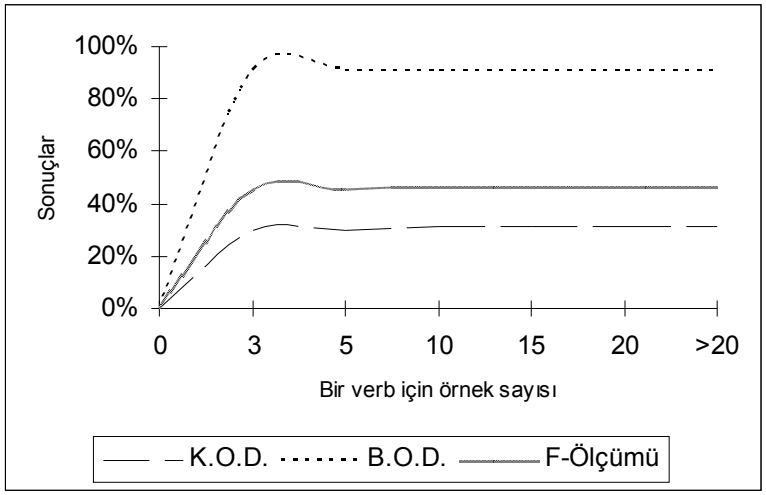
Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	30	91	45
5	30	91	45
10	31	91	46
15	31	91	46
20	31	91	46
>20	31	91	46

Şekil 5.2. T-Test sonuçları

T-Test sonuçlarında da Maksimum benzerlik sonuçlarına benzer sonuçlar bulundu. Yine 10 adet örnekleme ve sonrasında kısmi oranlı doğruluk, bütünsel oranlı doğruluk ve F-ölçümü yüzdeleri sırasıyla yüzde 31, yüzde 91 ve yüzde 46 olarak görülmektedir.

5.2.4.3. Briscoe ve Carroll'ın tekniği sonuçları

Briscoe ve Carroll tarafından önerilen örnekleme sonuçlarının sözlükten alınan istatistiksel sonuçlarla birleştirildiği tekniktir. Bu tekniğin sonuçları:



Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	61	88	72
5	61	88	72
10	61	88	72
15	60	88	71
20	60	88	71
>20	60	88	71

Şekil 5.3. Briscoe ve Carroll tekniği sonuçları

Ön bilgi kullanıldığı için sonuçlarda bir iyileşme sağlansa da örnekleme büyüklüğünün sistemi hiç etkilemediği saptanmıştır. Kısmi oranlı doğruluk ve bütünsel oranlı doğruluk sırasıyla yüzde 60 ve yüzde 88 civarında kalırken bu sonuçlara bağlı olarak F-ölçümü değerinin yüzde 71 geldiği görülmüştür.

5.2.4. Gözetimsiz öğrenme sonuçlarının değerlendirilmesi

Saf istatistiksel sonuçlara bağlı olan maksimum benzerlik ve T-test hipotez testleri ile beklenen sonuçlar elde edilememiştir. Briscoe ve Carroll'ın tekniği sonuçları bütünsel oranlı doğrulukta bir miktar düşme olsa da kısmi oranlı doğruluk değerinin yaklaşık iki kat artmasını sağlamıştır. Briscoe ve Carroll'ın tekniğindeki iyileşmenin sebebi sözlükten alınan istatistiksel sonuçların neticesinde olduğu görülmektedir. Bu sonuca göre Türkçe için bir ön-bilgi kullanılmasının gerektiğini göstermektedir.

Test	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
Maksimum Benzerlik	31	91	46
T-test	31	91	46
Briscoe ve Carroll'ın tekniği	60	88	71

Tablo 5.1. Gözetimsiz öğrenme sonuçları

Bu testleri kullanan diğer dilleri incelediğimizde çok daha başarılı sonuçlara ulaşıldığı görülmektedir. Yunanca [Maragoudakis vd., 2000], Bulgarca [Marinov, 2004] ve Çekçe [Sarkar ve Zeman, 2000] [Zeman ve Sarkar, 2000] üzerine yapılan testlerde maksimum benzerlik ve T-test'in F ölçümü sonuçları sırasıyla yüzde 77, yüzde 77 ve yüzde 79 olarak saptandığı görülmüştür. Briscoe ve Caroll (1997) İngilizce üzerine yaptıkları çalışmada ise yaklaşık yüzde 81 bütünsel oranlı doğruluk ve kısmi oranlı doğruluk değerlerine ulaşmışlardır. Bu sonuçlara göre Türkçede makine öğrenmesi ile daha iyi sonuçlara ulaşabilmek için ön-bilginin farklı bir şekilde aranması gerektiği görülmektedir.

5.3. Gözetimli öğrenme

Gözetimli öğrenme metotları için Weka yazılımını kullanıldı. Bunun için veritabanına kayıtlı verileri Weka yazılımının anlayabileceği ARFF dosya formatına çeviren bir ara yazılım geliştirildi. Bu bölümde ara yazılım sayesinde ARFF dosyalarını ürettikten sonra bu dosyalar çeşitli algoritmalara gönderilip çıkan sonuçların değerlendirilmesi yapılacaktır. Bu bölümde değerlendirilmesi yapılacak algoritmalar:

- K - En Yakın Komşu Sınıflandırılması
- Karar Verme Ağaçları
- Bayesian Sınıflandırma
- Bayesian.Net Sınıflandırma
- Çok katmanlı Perceptrons
- Destek Karar Makinesi

5.3.1. Weka hakkında ön bilgi

Weka makine öğrenmesi algoritmalarını içeren açık kaynak koda sahip bir yazılımdır. Weka içindeki algoritmalara Java kodu üzerinden kolaylıkla ulaşılabilir. Weka veri hazırlama, sınıflama, regresyon, kümeleme ve ilişki kuralları için çeşitli araçlar içerir.

Weka girdi olarak ARFF (Attribute-Relation File Format) dosya yapısını kullanır. Bir ARFF dosyası özellikleri ve örnekleri içeren bir ASCII metin dosyasıdır. ARFF dosyaları Waikato Üniversitesi Bilgisayar Mühendisliği Bölümünde makine öğrenme projeleri için geliştirilmiştir. [Witten ve Frank, 2005]

ARFF dosyaları iki ana bölümden oluşur. Birinci bölüm Başlık (Header) bilgilerini, ikinci bölüm ise Veri (Data) bilgilerini içerir.

Başlık bölümünde ilişki (relation) ismi, özelliklerin (attribute) listesi ve tiplerini içerir. Örneğin alt öğeleme listesi için hazırladığımız başlık;

```
@relation subcat
@attribute acc numeric
@attribute dat numeric
@attribute abl numeric
@attribute loc numeric
@attribute inst numeric
@attribute empty numeric
@attribute class{_Accusative,_Dative,_Ablative,_Locative,_Instrumental,
_Empty,_Accusative_Dative,_Accusative_Ablative,_Accusative_Locative,
_Accusative_Instrumental,_Dative_Ablative,_Dative_Locative,
_Dative_Instrumental,_Ablative_Locative,_Ablative_Instrumental,
_Locative_Instrumental,_Accusative_Dative_Ablative,
_Accusative_Dative_Locative, _Accusative_Dative_Instrumental,
_Dative_Ablative_Locative,_Dative_Ablative_Instrumental,
_Accusative_Dative_Ablative_Locative,_Accusative_Dative_Ablative_Instrumental,
_Dative_Ablative_Locative_Instrumental,
_Accusative_Dative_Ablative_Locative_Instrumental}
```

Örnek 5.1. ARFF dosyası başlığı

Başlık bölümü takip eden bölüm ise Veri (Data) bölümüdür. Veri bölümündeki bilgiler özellik isimlerinin her birini aynı sırada içermelidir. Örneğin;

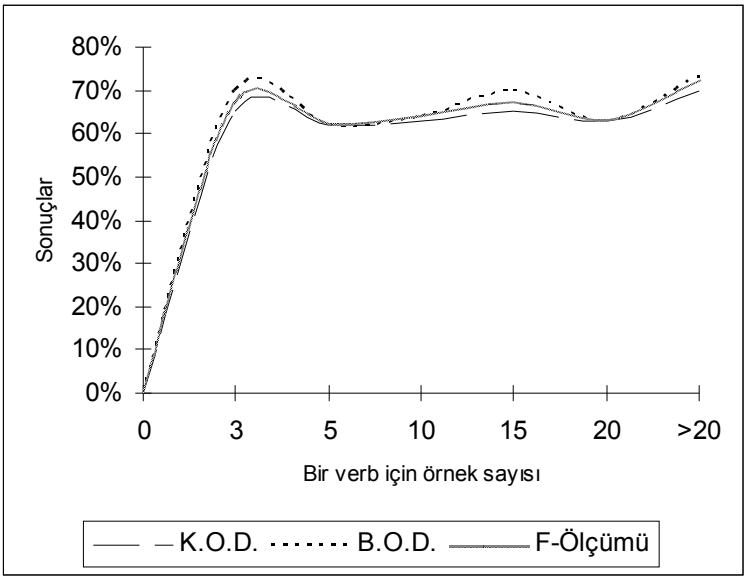
```
0.7137681159420289,0.11956521739130435,0.09057971014492754,
0.32971014492753625,0.036231884057971016,0.08695652173913043,_Accusative

0.46808510638297873,0.5815602836879432,0.07092198581560284,
0.1773049645390071,0.07092198581560284,0.12056737588652482,_Accusative_Dative

0.07692307692307693,0.4965034965034965,0.3076923076923077,
0.17482517482517482,0.11188811188811189,0.07692307692307693,_Ablative

0.1015625,0.6796875,0.0859375,0.125,0.0234375,0.1640625,
_Dative_Ablative_Instrumental

0.7666666666666667,0.03333333333333333,0.06666666666666667,
0.05555555555555555,0.022222222222222223,0.2,_Accusative
```

Örnek 5.2. ARFF dosyası verisi

5.3.2. Gözetimli öğrenme test sonuçları

5.3.2.1. K - En Yakın Komşu Sınıflandırılması sonuçları

Bir örnek tabanlı öğrenme tekniği olan K - En yakın komşu sınıflandırılması sonuçlarının Briscoe ve Carroll tekniğine yakın sonuçlar olduğu görülmektedir. Bu öğrenme tekniği için "lazy.IBK" kütüphanesi kullanılmıştır. Bu kütüphane sayesinde elde edilen sonuçlar:



Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	65	70	67
5	62	62	62
10	63	64	64
15	65	70	67
20	63	63	63
>20	70	73	72

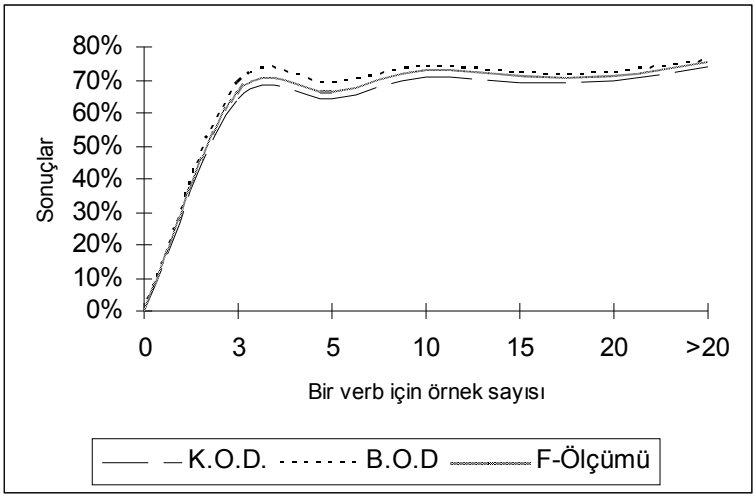
Şekil 5.4. K-En yakın komşu sonuçları

K-En Yakın komşu metodunda veri artışlarının öğrenmeye olumlu bir etkisi olacağını beklenmesine rağmen tutarsız sonuçlarla karşılaşılmıştır. Bu verilere göre bu metot yüksek miktarda artışlarda olumlu etkilenmekteyken, az miktarda (5 ve 2 gibi) sistemi fazla etkilememektedir. Ancak, örnekleme sayısı 20'nin çok üzerinde olduğunda

en yüksek sonuçlara ulaşılmıştır. 20'nin üzerinde kısmi oranlı doğruluk, bütünsel oranlı doğruluk ve F-ölçümü sırasıyla yüzde 70, yüzde 73 ve yüzde 74 değerleri ile karşılaşılmıştır.

5.3.2.2. Karar Verme Ağaçları sonuçları

Verileri ağaç yapısı formunda sınıflama kullanılabilecek gözetimli tekniklerden diğeridir. Bu teknik için "trees.J48" kütüphanesinden yararlanılmıştır. Bu kütüphane kullanılarak elde edilen sonuçlar:



Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	66	69	67
5	66	69	67
10	71	75	73
15	71	75	73
20	73	77	75
>20	75	75	75

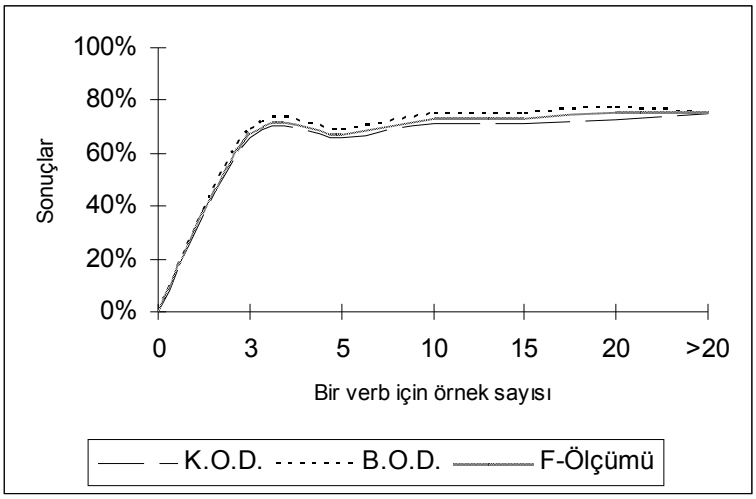
Şekil 5.5. Karar verme ağacı sonuçları

Sonuçlara göre karar verme ağaçları örnekleme sayısının öğrenme üzerindeki etkisini açıkça gösteren metotlardan biridir. 3 örnekleme sayısı için kısmi oranlı doğruluk, bütünsel oranlı doğruluk ve F-ölçümü sırasıyla yüzde 66, yüzde 69 ve yüzde

67 iken 20 örnekleme sayısı 20'nin üzerinde olduğu durumda yaklaşık yüzde 8'lik bir artış sağlanmıştır.

5.3.2.3. Naive Bayesian Sınıflandırma sonuçları

Naive Bayes sınıflandırıcı kuvvetli bağımsız varsayımlarla Bayes teoremini temel alan olasılıklı bir sınıflayıcıdır. Bu sınıflandırıcı için Weka içindeki "Bayes.NaiveBayes" kütüphanesi kullanılmıştır. Bu kütüphane sayesinde elde edilen sonuçlar:



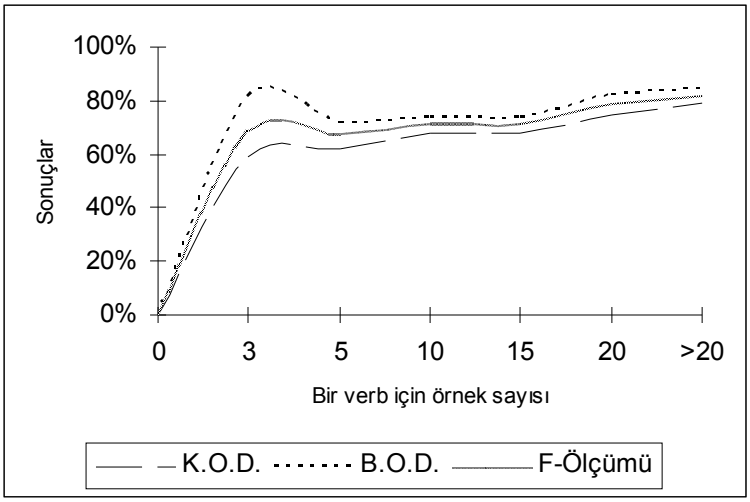
Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	64	69	66
5	64	69	66
10	71	74	73
15	69	72	71
20	70	72	71
>20	74	76	75

Şekil 5.6. Bayesian sınıflandırıcı sonuçları

Bayesian sınıflandırıcı sonuçları, karar verme ağaçlarına çok yakın olduğu görülmektedir. Bu metotta da örnekleme sayısı arttıkça sistem olumlu bir şekilde etkilenmekte ve 20'nin üzerindeki örnekleme sayıları için kısmi oranlı doğruluk ve bütünsel oranlı doğrulukta yüzde 74 ve yüzde 76 sonuçlara ulaşılırken, bu doğruluk oranlarına bağlı F-ölçümü değeri ise yüzde 75'tir.

5.3.2.4. Bayesian.Net Sınıflandırma sonuçları

Bayes teoremini temel alan ve verilen bağımsızlıkları gösteren bir direkt çevrimsiz grafik olarak gösteren bir diğer sınıflandırıcı Bayesian.Net sınıflandırıcısıdır. Bu sınıflandırıcı için Weka içindeki "Bayes.BayesNet" kütüphanesi kullanılmıştır. Bu kütüphane üzerinden elde edilen sonuçlar:



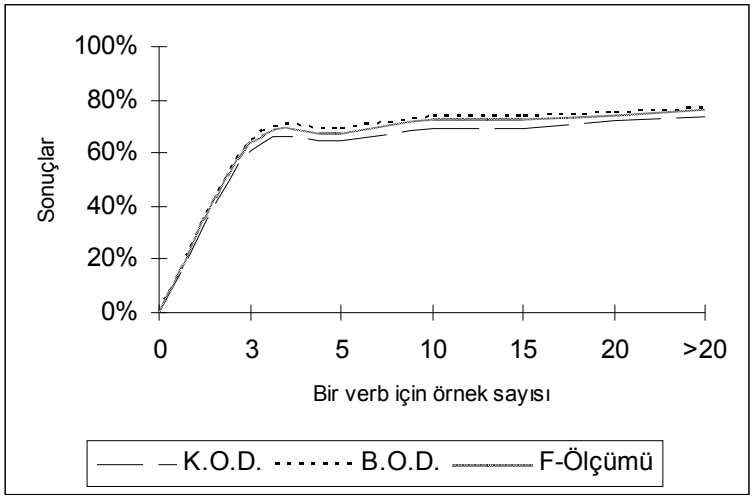
Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	59	81	68
5	62	72	67
10	68	74	71
15	68	74	71
20	75	82	78
>20	79	84	81

Şekil 5.7. Bayesian.Net sınıflandırıcı sonuçları

Bayesian sınıflandırıcının geliştirilmiş bir hali olan Bayesian.Net sınıflandırıcı ile daha iyi sonuçlara ulaşılmıştır. 3 örnekleme için her ne kadar kısmi oranlı doğruluk yüzde 59 kalsa da bütünsel oranlı doğruluk yüzde 81 gibi yüksek bir değer çıkmıştır. 20 örnekleme fazla olan durumlar için ise kısmi oranlı doğruluk %20'lik bir artış göstermiş ve bütünsel oranlı doğruluk yüzde 84 olduğu görülmüştür. Bu oranlara bağlı olarak F-ölçümü değeri ise yüzde 81 olmuştur.

5.3.2.5. Çok katmanlı Perceptrons sonuçları

Perceptron birçok değeri girdi olarak alıp tek bir çıktı üreten bir sınıflayıcıdır. Bu sınıflayıcı için "fuction.MultilayerPerceptron" kütüphanesi kullanıldı. Bu kütüphane sayesinde elde edilen sonuçlar:



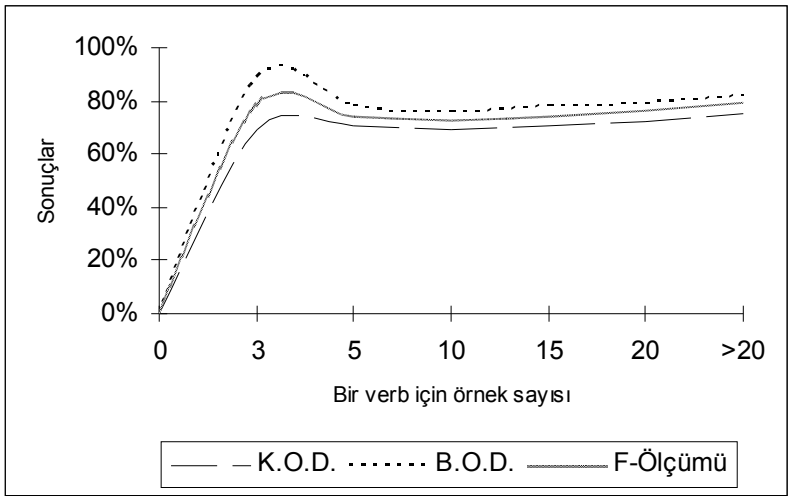
Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	61	65	63
5	65	69	67
10	69	74	72
15	69	74	72
20	72	75	74
>20	74	77	76

Şekil 5.8. Çok katmanlı Perceptrons sonuçları

Yapay zekâ algoritmalarından biri olan çok katmanlı perceptron metodu da örnekleme sayısının artışından olumlu bir şekilde etkilendiği görülmektedir. 3 örnekleme için kısmi oranlı doğruluk, bütünsel oranlı doğruluk ve F-ölçümü sırasıyla yüzde 61, yüzde 65 ve yüzde 63 iken 20 örneklemeden fazla olan durumlar için yaklaşık yüzde 13 bir artış sağlanmıştır.

5.3.2.6. Destek Karar Makinesi sonuçları

Destek karar makineleri, yüksek boyutlu verilerde sıkça kullanılan bir ayrım metodudur. Bu metod içinde "functions.SMO" kütüphanesinden yararlanılmıştır. Bu kütüphane kullanılarak elde edilen sonuçlar:



Örnekleme Sayı	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
3	69	89	78
5	71	78	74
10	69	76	72
15	71	78	74
20	72	79	76
>20	75	82	79

Şekil 5.9. Destek karar makinesi sonuçları

Destek karar makinesi metodunda 3 örnekleme için kısmi oranlı doğruluk yüzde 69 kalmasına rağmen bütünsel oranlı doğruluk yüzde 89 gibi bir yüksek orandadır. 20 örnekleme sayısından fazla olan durumlar için ise kısmi oranlı doğruluk, bütünsel oranlı doğruluk ve F-ölçümü sırasıyla yüzde 75, yüzde 82 ve yüzde 79 değerlerine ulaşılmaktadır.

5.3.3. Gözetimli öğrenme sonuçlarının değerlendirilmesi

Test	K.O.D. (%)	B.O.D. (%)	F-Ölçümü (%)
K - En Yakın Komşu Sınıflandırma	70	73	72
Karar Verme Ağaçları	75	75	75
Bayesian Sınıflandırma	74	76	75
Bayesian.Net Sınıflandırma	79	84	81
Çok katmanlı Perceptron	74	77	76
Destek Karar Makinesi	75	82	79

Tablo 5.2. Gözetimli öğrenme sonuçları

Türkçe için alt öğeleme listesinin makine öğrenmesi ile elde edilmesi için en uygun metotların Bayesian.Net sınıflandırıcı ve destek karar sistemleri olduğu görülmektedir. Özellikle Bayesian.Net sınıflandırıcı ile kısmi oranlı doğruluk, bütünsel oranlı doğruluk ve F-ölçümünde sırasıyla elde edilen yüzde 79, yüzde 84 ve yüzde 81 değerleri ile tüm sonuçlar arasında en iyi öğrenme metodu olduğu göze çarpmaktadır. Karar verme ağaçları, Bayesian sınıflandırma ve çok katmanlı perceptron sonuçları ise birbirine son derece yakındır. K-En yakın komşu sınıflandırma ise en düşük sonuçların elde edildiği metottur.

Bu arada, yapay zekâ tekniklerini kullanan çok katmanlı perceptron ve destek karar makinesi algoritmaları diğer algoritmalarla karşılaştırıldığında son derece yavaş kalan algoritmalardır.

Saf istatistiksel sonuçlara bağlı olan maksimum benzerlik ve T-test hipotez testleri ile elde edilen sonuçlar son derece kötüdür. Briscoe ve Carroll'ın tekniği kullanılan sözlük bilgisi sayesinde sonuçlarda iyileşme sağlanmıştır. Örnek veriler üzerinden öğrenme işlemini gerçekleştiren gözetimli öğrenme teknikleri sayesinde çarpıcı bir iyileşme sağlanmıştır. Özellikle Bayesian.Net metodu sayesinde son derece iyi sonuçlara ulaşılmıştır.

BÖLÜM 6.

SONUÇLAR

Bu tezin birincil amacı, makine öğrenmesi tekniklerini kullanarak internet üzerinden Türkçe bir fiile ait alt öğeleme listesinin öğrenilmesini sağlamaktır. Alt öğeleme listelerinin tahmini için arama motorları üzerinden cümle toplayan bir araç geliştirilmiştir. Burada arama motorlarının Türkçe açısından zorlukları karşımıza çıkmaktadır.

- Arama motorlarından sadece kök fiili aramak çok yetersiz sonuçlara sebebiyet vermektedir.
- Arama sonuçları içinde Türkçeye uymayan karakter ve cümlelerle karşılaşılmaktadır.

Bu zorlukları aşmak için Fiil Üreteci, Web Tabanlı Cümle Çıkarıcı ve Kelime Etiketleyici olmak üzere üç ana modülden oluşan bir yazılım geliştirilmiştir. Bu yazılımlar sayesinde ortaya çıkan zorluklar aşılmaya çalışılmıştır.

Tezin sonuçları, deneysel gözlemler ve teorik tartışma olmak üzere iki ana bölümde incelenebilir.

DENEYSEL GÖZLEMLER

1. Maksimum Benzerlik ve T-Test gibi saf istatistiksel metotların Türkçe için alt öğeleme listesinin otomatik olarak elde edilmesi görevi için yetersiz kaldığı gözlemlenmiştir (F-ölçümü yüzde 46).
2. Dilbilimsel ön-bilgi kullanan Briscoe ve Carroll'ın tekniğinin bu görev için daha iyi sonuçlara ulaştığı görülmüştür (F-ölçümü yüzde 71). Bu, ön-bilgi kullanmanın bu görev için önemli olduğunu göstermektedir.

3. Örnek veriler üzerinden öğrenme işlemini gerçekleştiren gözetimli öğrenme teknikleri sayesinde daha iyi sonuçlara ulaşılabildiği görülmüştür. Özellikle Bayesian.Net sınıflandırıcısı sayesinde en iyi sonuçlara ulaşılmıştır (F-ölçümü yüzde 81).

Gözlemlere göre, biri negatif biri pozitif olmak üzere iki ayrı teorik sonuç karşımıza çıkmıştır.

TEORİK TARTIŞMA

1. Saf istatistiksel metotların, Türkçede alt öğeleme listesinin otomatik olarak elde edilmesi için uygun olmadığı görülmektedir.
2. Ön-bilgi kullanan metotlarla bir iyileşme sağlanmaktadır. Özellikle örnek veriler üzerinden öğrenme yapan gözetimli öğrenme teknikleri sayesinde daha iyi sonuçlara ulaşılabilmektedir.

Bu tez, Türkçe alt öğeleme listelerinin makine öğrenmesi ile internet üzerinden elde edilmesi için dilbilimsel bir ön-bilgiye ihtiyaç olduğunu göstermektedir. Türkçenin ön-bilgiye ihtiyaç duymasının üç ana sebebi vardır.

- Zamir düşmesi özelliği olan bir dil olması
- Seyrek gösterimli bir dil olması
- Serbest sözcük sıralaması olması

Bu açılardan Türkçenin diğer dillerle karşılaştırıldığında makine öğrenmesi açısından çok daha zor bir dil olduğu görülmektedir. Özellikle (Çekçe, Yunanca, Bulgarca gibi) diğer dillerin saf istatistiksel öğrenme metotları ile çok iyi sonuçlara ulaşabilmelerine rağmen Türkçe için bu metotlar yetersiz kalmaktadır.

Bu tez ayrıca derlem olarak interneti kullanması açısından diğer ilgili çalışmalardan ayrılmaktadır. Çalışmada arama motorlarının yarattığı zorluklar aşılmaya çalışılmış ve Türkçeye uygun gramatikal cümleler toplayan bir araç geliştirilmiştir.

Bu tez için geliştirilmiş olan uygulamalar ve kullanılan makine öğrenmesi teknikleri gelecek çalışmalar için temel olabilecek niteliktedir. Geliştirilen web tabanlı cümle çıkarıcı uygulaması kelime anlamı belirginleştirme çalışmalarında kullanılabilir. Ayrıca bu uygulama geliştirilerek otomatik derlem ve sözlük oluşturma gibi çalışmalar da gerçekleştirilebilir. Kullanılan makine öğrenmesi teknikleri ise anafora çözümleme, metin sınıflama ve bilgi çıkarımı gibi doğal dil işlemenin alt alanlarında da kullanılabilir.

KAYNAKLAR

1. Abdi H., 2003, "Neural Networks", In M. Lewis-Beck, A. Bryman, T. Futing (Eds): Encyclopedia for research methods for the social sciences. Thousand Oaks (CA): Sage, syf. 792-795
2. Abdi H., Valentin D. ve Edelman B.E., 1999, "Neural Networks", Thousand Oaks: Sage
3. Alpaydın E., 2004, "Introduction to Machine Learning", The MIT Press, Syf: 3-6
4. Anderberg M. R., 1973, "*Cluster Analysis for Applications*" Academic Press, New York, NY
5. Anderson J. A., 1995, "An Introduction to Neural Networks", ISBN 0-262-01144-1
6. Bertsekas D. P. ve Tsitsiklis J., 1996, "Neuro-Dynamic Programming", Athena Scientific, ISBN 1-886529-10-8
7. Baroni M. ve Bernardini S.,2004, BootCaT: "Bootstrapping corpora and terms from the Web", In Proceedings of LREC 2004, syf. 1313-1316
8. Baroni M. ve Sharoff S., 2005, "Creating specialized and general corpora using automated search engine queries", Presented at the Corpus Linguistics 2005 Web as Corpus Workshop, Birmingham, UK
9. Basili R., Pazienza M.T. ve Vindigni M., 1997, "Corpus-Driven Unsupervised Learning of Verb Subcategorization Frames", *Number 1321 in LNAI*, Heidelberg, Germany, Springer-Verlag
10. Basili R. ve Vindigni M., 1998, "Adapting a Subcategorization Lexicon to a Domain", *Proceedings of the ECML'98 Workshop TANLPS: Towards adaptive NLPdriven systems: linguistic information, learning methods and applications*, Chemnitz, Germany
11. Baxter J., 2000, "A model of inductive bias learning", Journal of Artificial Intelligence Research, 12:149-198
12. Bickel B., 2003, "Referential Density in Discourse and Syntactic Typology", *Language* 79, syf. 708-36

13. Blum A. ve Mitchell T., 1998, "Combining labeled and unlabeled data with co-training", *COLT: Proceedings of the Workshop on Computational Learning Theory*, Morgan Kaufmann, syf. 92-100
14. Boguraev B. ve Briscoe E., 1987, "Large Lexicons for Natural Language Processing: Utilising the Grammar Coding System of the Longman Dictionary of Contemporary English", *Computational Linguistics* 13.4, syf. 219-240
15. Brent M., 1991, "Automatic Acquisition of Subcategorization Frames from Untagged Text", *Proceedings of the 29th Meeting of the ACL*, Berkeley, CA, syf. 209–214
16. Brent M., 1993, "From Grammar to Lexicon: Unsupervised Learning of Lexical Syntax", *Computational Linguistics,* 19.3: syf. 243–262
17. Brent M., 1994, "Acquisition of Subcategorization Frames Using Aggregated Evidence from Local Syntactic Cues", *Lingua*, 92, Reprinted in *Acquisition of the Lexicon*, L. Gleitman and B. Landau, eds., MIT Press, Cambridge, MA, pp. 433–470
18. Bresnan J., 1978, "*A Realistic Transformational Grammar*," *Linguistic Theory and Psychological Reality*, M. Halle, J. Bresnan and G. A. Miller, eds. Cambridge, Mass. And London: MIT Press
19. Bresnan J., 1982, "*The Mental Representation of Grammatical Relations*", syf. 1-59, Cambridge, Mass. And London: MIT Press, 1982
20. Briscoe T. ve Carroll J., 1993, "Generalized Probabilistic LR Parsing for Unification-based Grammars", *Computational Linguistics* 19.1: 25-60
21. Briscoe T. ve Carroll J., 1997, "Automatic Extraction of Subcategorization from Corpora," *Proceedings of the 5th ACL Conference on Applied Natural Language Processin*g, Washington, DC.
22. Carroll J. ve Minen G., 1998, "Can Subcategorisation Probabilities Help a Statistical Parser", *Proceedings of the 6th ACL/SIGDAT Workshop on Very Large Corpora (WVLC-6)*, Montreal, Canada
23. Carroll G. ve Rooth M., 1998, "Valence Induction with a Head-lexicalized PCFG", *Proceedings of the 3rd Conference on Empirical Methods in Natural Language Processing (EMNLP 3)*, Granada, Spain
24. Caruana R.,1997, "Multitask learning: A knowledge-based source of inductive

bias. Machine Learning", 28:41-75

25. Castillo E., Gutiérrez J.M. ve Hadi S.A., 1997, Expert Systems and Probabilistic Network Models. New York: Springer-Verlag, ISBN 0-387-94858-9
26. Chapelle O., Schölkopf B. ve Zien A., 2006, "*Semi-Supervised Learning*", MIT Press, Cambridge, MA
27. Chesley P. Ve Salmon-Alt S., 2006, "Automatic Extraction of Subcategorization Frames for French", *Language Resources and Evaluation Conference (LREC 2006)*, Genua, Italy
28. Chomsky N., "Aspects of the Theory Syntax", 1965, Cambridge, MIT Press
29. Chomsky N.,1975, "Reflections on Language", New York: Pantheon
30. Chomsky N., 1997, "Essays on Form and Interpretation", New York: North Holland
31. Chomsky N., 1981, "Lectures on Government and Binding", Dordrecht: Foris
32. Chomsky N., 1982, "Some Concepts and Consequences of the Theory of Government and Binding", Cambridge, Mass. And London: MIT Pres
33. Chomsky N., 1993, "A Minimalist Program for Linguistic Theory", The View from Building 20: Essays in Linguistics in Honor of Sylvain Bomberger, K. Hale and S.J. Keyser eds., syf. 1-52. Cambridge, MA: MIT Press
34. Chomsky N., 1995, "The Minimalist Program", Cambridge, MA: MIT Pres
35. Church K. W. ve Mercer R. L., 1993, "Introduction to the special issue on computational linguistics using large corpora", Computational Linguistics, 19(1):1-24
36. Cohen W. W. ve Singer Y., 1996, "Context-Sensitive Learning Methods for Text Categorization", Proceedings of the 19th Annual ACM SIGIR Conference
37. Corley M. ve Haywood S., 2000, "Parsing modifiers: The case of bare-NP adverbs", In Proceedings of the 21st annual meeting of the Cognitive Science Society, syf 126-131
38. Dasarathy B. V., editor, 1991, "Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques", ISBN 0-8186-8930-7
39. De Lima F., 1997, "Acquiring German Prepositional Subcategorization Frames from Corpora", *Proceedings of the 5th Workshop on Very Large Corpora (WVLC-5)*

40. Domingos P. ve Pazzani M., 1997, "On the optimality of the simple Bayesian classifier under zero-one loss". *Machine Learning*, 29:103–137
41. Erguvanlı E. E., 1984, "*The Function of Word Order in Turkish Grammar*", University of California Publications in Linguistics, Vol. 106. Berkeley and Los Angeles: University of California Press
42. Ersan M. ve Charniak E., 1996, "A Statistical Syntactic Disambiguation Program and What It Learns", *Connectionist, Statistical and Symbolic Approaches in Learning for Natural Language Processing*, Vol.1040 of Lecture Notes in Artifical Intelligence, S. Wermter, E. Riloff, and G. Scheler, eds., Springer - Verlag, Berlin, syf. 146-159
43. Fletcher W., 2001, "Concordancing the Web with KWiCFinder", in Proc. 3rd North American Symposium on Corpus Linguistics and Language Teaching
44. Fletcher B., 2004, "Making the Web more useful as a source for linguistic corpora", In Connor, U. and Upton, T. (eds.) Corpus linguistics in North America 2002, Amsterdam: Rodopi.
45. Franco-Lopez H., Ek A. R. ve Bauer M. E., 2001 "Estimation and mapping of forest stand density, volume, and cover type using the k-nearest neighbors method", Remote Sensing of Environment, Volume 77, Issue 3, Syf. 251-274
46. Fu W.-T. ve Anderson J. R., 2006, "Recurrent Choice to Skilled Learning Model. Learning: A Reinforcement Learning Model. Learning: A Reinforcement Learning Model. Journal of Experimental Psychology: General", 135 (2), syf. 184-206
47. Fuernkranz J., Mitchell T. ve Riloff E., 1998, "A Case Study in Using Linguistic Phrases for Text Categorization on the WWW", Sahami, M., editor, In Learning for Text Categorization: Papers from the 1998 AAAI Workshop (Technical Report WS-98-05)
48. Funahashi K., 1989, "The Approximate Realization of Continuous Mappings", *Neural Networks*, Vol.2, syf.183-192
49. Gazdar G., Klein E., Pullum G. K. ve Sag I. A., 1985, "*Generalized Phrase Structure Grammar*", Oxford: Basil Blackwell; Cambridge, Mass.: Harvard University Pres
50. Gümüşoğlu Ş., 2000, "İstatiksel Kalite Kontrolü ve Toplam Kalite Yönetimi

Araçları", ISBN : 975 486 977 1, Yayınevi : Beta Basım Yayın

51. Hand D.J. ve Yu K., 2001, "Idiot's Bayes - not so stupid after all?" International Statistical Review. Vol 69 part 3, pages 385-399. ISSN 0306-7734
52. Hartigan J., 1975, "*Clustering Algorithms*", Wiley, New York, NY.
53. Heckerman D., 1999, "A Tutorial on Learning with Bayesian Networks. In Learning in Graphical Models", M. Jordan, ed.. MIT Press, Cambridge, MA
54. Herbrich R., 2002, "Learning Kernel Classifiers", ISBN 0-262-08306-X, The MIT Press
55. Hill D. R., 1968, "A vector clustering technique", In Samuelson (ed.), *Mechanised Information Storage, Retrieval and Dissemination*, North-Holland, Amsterdam
56. Hinton G. ve Sejnowski T.J. (editorler), 1999, "Unsupervised Learning and Map Formation: Foundations of Neural Computation", MIT Press, ISBN 0-262-58168-X
57. Hornik K., Stinchcombe M. ve White H., 1989, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks,* Vol.2, No.5, pp.359-366
58. Huang T-M., Kecman V. ve Kopriva I., 2006, "Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semisupervised and Unsupervised Learning", Springer-Verlag, Berlin, Heidelberg, syf. 260, 96 illus., Hardcover, ISBN 3-540-31681-7
59. Jackendoff R., 1972, "*Semantic Interpretation in Generative Grammar*", Cambridge, Mass.: MIT Pres
60. Jain A. K. ve Dubes R. C., 1988, "*Algorithms for Clustering Data*", Prentice Hall, Englewood Cliffs, NJ.
61. Jardine N. ve Sibson R., 1971, "*Mathematical Taxonomy*", Wiley, London
62. Jensen F. V., 2001, "Bayesian Networks and Decision Graphs", Springer
63. Johnson S. C., 1967, "Hierarchical Clustering Schemes", *Psychometrika*, 2:241-254
64. Jurafsky D. ve Martin J. H., 2000, "Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition", Prentice Hall, ISBN: 0130950696
65. Kainen P.C., Vogt A., ve uKrková V., 2000, "An integral formula for Heaviside

neural networks", International Journal on Neural and Mass-Parallel Computing and Information Systems, syf.313-319

66. Kecman V., 2001, "Learning and Soft Computing: Support Vector Machine, Neural Networks, and Fuzzy Logic Models", ISBN 0-262-11255-8, The MIT Press
67. Kehoe A. ve Renouf A., 2002, "WebCorp: Applying the Web to Linguistics and Linguistics to the Web", WWW2002 Conference, Honolulu, Hawaii
68. Kilgarriff A., 2003, "Linguistic search engine", In Kiril Simov, editor, Shallow Processing og Large Corpora: Workshop held in association with Corpus Linguistics 2003, Lancaster, March
69. Korhonen A., 1998, "Automatic Extraction of Subcategorization Frames from Corpora - Improving Filtering with Diathesis Alternations," *Proceedings of the ESSLLI 98 Workshop on Automated Acquisition of Syntax and Parsing*, Saarbrucken, Germany, syf. 49-56
70. Korhonen A., Gorrell G. ve McCarthy D., 2000, "Is Hypothesis Testing Useful for Subcategorization Acquisition?," *Technical Report UCAM-CL-TR-491*. Computer Laboratory, University of Cambridge
71. Korhonen A., Gorrell G. ve McCarthy D., 2000, "Statistical Filtering and Subcategorization Frame Acquisition", *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, Hong Kong
72. Korhonen A., Krymolowski Y. ve Briscoe T., 2006, "A Large Subcategorization Lexicon for Natural Language Processing Applications", *Proceedings of the 5th international conference on Language Resources and Evaluation*. Genova, Italy
73. Kawahara D., Kaji N. ve Kurohashi S., 2000, "Japanese Case Structure Analysis by Unsupervised Construction of a Case Frame Dictionary", *Proceedings of the 18th conference on Computational linguistics*, Saarbrücken, Germany, syf. 432-438
74. Kawahara D. ve Kurohashi S., 2001, "Japanese Case Frame Construction by Coupling the Verb and Its Closest Case Component", *Proceedings of the first international conference on Human language technology research*, San Diego, syf.1-7

75. Kılıçaslan Y., 2004, "Syntax of Information Structure in Turkish," *Linguistics*, 42-4, syf. 717-765
76. Kilgarriff A., 2001, "The Web as corpus", Proceedings of Corpus Linguistics 2001
77. Kilgarriff A. ve Grefenstette G., 2003, "Introduction to the special issue on the Web as corpus", Computational Linguistics 29(3), syf. 333-347
78. Lasnik H., 2002, "The minimalist program in syntax", TRENDS in Cognitive Sciences 6:10, syf. 432-437
79. Levin L., Rappaport M. ve Zaenen A., 1983, "*Papers in Lexical Functional Grammar*", Bloomington: Indiana University Linguistics Club
80. Littman M. L. ve Moore A. W., 1996, "Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research 4", syf. 237–285
81. MacQueen J. B., 1967, "Some Methods for classification and Analysis of Multivariate Observations", *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1:281-297
82. Mandic D. ve Chambers J., 2001, "Recurrent Neural Networks for Prediction: Architectures, Learning algorithms and Stability", Wiley
83. Maragoudakis M., Kermanidis K. and Kokkinakis G., 2000, "Learning Subcategorization Frames from Corpora: A Case Study for Modern Greek", *Proceedings of COMLEX 2000*, Workshop on Computational Lexicography and Multimedia Dictionaries, Kato Achaia, Greece, syf. 19-22
84. Maragoudakis M., Kermanidis K., Fakotakis N. ve Kokkinakis G., 2001, "Learning Automatic Acquisition of Subcategorization Frames using Bayesian Inference and Support Vector Machines", ICDM '01, *The 2001 IEEE International Conference on Data Mining*, San Jose, syf. 623-625
85. Marinov S., 2004, "Automatic Extraction of Subcategorization Frames for Bulgarian" *Proceedings of 16th ESSLLI Student Session*, Nancy, France
86. Marinov S. ve Hemming C., 2004, "Automatic Extraction of Subcategorization Frames from the Bulgarian Tree Bank", Unpublished manuscript, Graduate School of Language Technology, Göteborg, Sweden
87. Manning D. C., 1993, "Automatic Acquisition of a Large Subcategorization

Dictionary from Corpora", *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, Ohio, syf. 235-242

88. Manning, D. C. ve Schutze H., 1999, "Foundations of Statistical Natural Language Processing", The MIT Press, Cambridge, Massachusetts, USA
89. Maron M. E., 1961, "Automatic Indexing: An Experimental Inquiry." Journal of the ACM (JACM) 8(3):404–417
90. Mozina M., Demsar J., Kattan M. ve Zupan B., 2004, "Nomograms for Visualization of Naive Bayesian Classifier". In Proc. of PKDD-2004, syf. 337-348
91. Mäkelä H. ve Pekkarinen A., 2004, "Estimation of forest stand volumes by Landsat TM imagery and stand-level field-inventory data", Forest Ecology and Management, Volume 196, Issues 2-3, Syf 245-255
92. McCallum A. ve Nigam K., 1998, "A Comparison of Event Models for Naive Bayes Text Classification". In AAAI/ICML-98 Workshop on Learning for Text Categorization, pp. 41-48. Technical Report WS-98-05. AAAI Pres
93. Michael A.A. (Ed.), 1995, "The Handbook of Brain Theory and Neural Networks"
94. Mihalcea R. ve Moldovan D., 1999, "A method for word sense disambiguation of unrestricted text", In Proc. 37th Meeting of ACL, syf 152-158
95. Minsky M. L. ve Papert S.A., 1969, "Perceptrons", (Cambridge, MA: MIT Press)
96. Mitchell T., 1997, "Decision Tree Learning", *Machine Learning*, The McGraw-Hill Companies, Inc.,syf. 52-78
97. Nabiyev V.V., 2005, "Yapay Zeka", ISBN 975 347 985 9,Seçkin Yayıncılık, 2. Baskı, Syf: 591
98. Neil M., Fenton N. ve Tailor M., 2005, "Using Bayesian Networks to model Expected and Unexpected Operational Losses", Risk Analysis: An International Journal, Vol 25(4), syf. 963-972
99. Ng, H. T., W. B. Goh, and K. L. Low, 1997, "Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization", 20th Ann Int ACM SIGIR Conference on Research and Development in Information Retrieval , syf. 67-73
100. Oflazer K., 1994, "Two-level description of Turkish morphology. Literary and Linguistic Computing", 9, 175-198
101. Oflazer K. ve Kuruöz I., 1994, "Tagging and Morphological Disambiguation of

Turkish Text", Proceedings of the 4th ACL Conference on Applied Natural Language Processing, Stuttgart

102. Özgür A., 2004, "Supervised and unsupervised machine learning techniques for text document categorization", Yüksek Lisans Tezi
103. Quinlan R.J., 1986, "Induction of decision trees", Machine Learning, 1:81-106
104. Quinlan R.J., 1993, "C4.5: Programs for Machine Learning", Morgan Kaufmann.
105. Pearl J., 2000, "Stuart Russell. Bayesian Networks. UCLA Cognitive Systems Laboratory", Technical Report (R-277)
106. Peters J., Vijayakumar S., Schaal S., 2003, "Reinforcement Learning for Humanoid Robotics", IEEE-RAS International Conference on Humanoid Robots
107. Platt J., 1998, Sequential Minimal Optimization: "A Fast Algorithm for Training Support Vector Machines", Microsoft Research Technical Report MSR-TR-98-14
108. Pollard C. ve Ivan A. S., 1987, "*Information-Based Syntax and Semantics*", Vol. 1: Fundamentals. CSLI Lecture Notes no. 13. Stanford: Center for the Study of Language and Information (University of Chicago Pres tarafın dağıtımı yapılmıştır.)
109. Pollard C. ve Ivan A. S., 1994, "*Head-Driven Phrase Structure Grammar*", Chicago, London: University of Chicago Press and CSLI Publications
110. Porter M. F., 1980, "An Algorithm for Suffix Stripping", Program, Vol. 14,syf. 130-137.
111. Rakotomalala R. ve Lallich S., 1998, "Handling noise with generalized entropy of type beta in induction graphs algorithm", in Proceedings of International Conference on Computer Science and Informatics, syf. 25-27
112. Rayson P., Archer D., Baron A. ve Smith N., 2006, "Tagging historical corpora - the problem of spelling variation", In proceedings of Digital Historical Corpora, Dagstuhl-Seminar 06491, International Conference and Research Center for Computer Science, Schloss Dagstuhl, Wadern, Germany, December 3rd-8th 2006
113. Resnik P. ve Elkiss A, "The Linguist's Search Engine: An Overview", Proceedings of ACL 2005 (Demonstration Section), 2005
114. Rish I., 2001, "An empirical study of the naive Bayes classifier", IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence

115. Rosenblatt F., 1958, "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", Cornell Aeronautical Laboratory, Psychological Review, v65, No. 6, syf. 386-408

116. Resnik P., 1999, "Mining the web for bilingual text", In Proc. 37th Meeting of ACL, syf 527-534

117. Salton G., 1968, "Automatic Information Organization and Retrieval"

118. Salton G. ve Buckley C., 1988, "Term Weighting Approaches in Automatic Text Retrieval", Information Processing and Management, Vol. 24, No. 5, syf. 513-523

119. Salton G., Yang C. ve Wong A., 1975, "A Vector-Space Model for Automatic Indexing", Communications of the ACM, Vol. 18, No. 11, syf. 613-620

120. Sarkar A. ve Zeman D., "Automatic Extraction of Subcategorization Frames for Czech", 2000, *Proceedings of the International Conference on Computational Linguistics*, COLING-00, Saarbrucken, Germany, syf. 691-697

121. Sarkar A. ve Tripasai W., 2002, "Learning Verb Argument Structure from Minimally Annotated Corpora", *Proceedings of the 19th International Conference on Computational Linguistics: COLING 2002*. Taipei, Taiwan

122. Sharoff S., 2006, "Translation as problem solving: uses of comparable corpora", In Proc. of Third International Workshop on Language Resources for Translation Work, Research & Training at LREC2006, Genoa, May, 2006

123. Sneath P. H. A. ve Sokal R. R., 1973 "*Numerical Taxonomy*" Freeman, San Francisco, CA.

124. Spitters M., 2000, "Comparing Feature Sets for Learning Text Categorization", Proceedings of RIAO 2000

125. Sutton R. S. ve Barto A. G., 1998, "Reinforcement Learning", MIT Press, ISBN 0-262-19398-1

126. Thrun S., 1996, "Is learning the n-th thing any easier than learning the first?", In Advances in Neural Information Processing Systems 8, syf. 640--646. MIT Press

127. Tryon R. C. ve Bailey D. E., 1973, "*Cluster Analysis*", McGraw-Hill, New York, NY.

128. Uriegereka J., 2002, "*Derivations: Exploring the Dynamics of Syntax*", London and New York: Routledge

129. Ushioda A., Evans A. D., Gibson T. ve Waibel A., 1993, "The Automatic Acquisition of Frequencies of Verb Subcategorization Frames from Tagged Corpora", *Proceedings of the Workshop on Acquisition of Lexical Knowledge from Text,* B. Boguraev and J. Pustejovsky, eds., Columbus, OH, syf. 95–106

130. van Rijsbergen C. J., 1979, "*Information Retrieval*", Butterworths, London

131. Vapnik V. N., 1998, "*Statistical learning theory*", New York: Wiley, syf. *339-371*

132. Webster M. ve Marcus M., 1989, "Automatic Acquisition of the Lexical Frames of Verbs from Sentence Frames", In Proc. *27th Meeting of the ACL*, pp. 177–184

133. Weinberg A., 2001, "A Minimalist Theory of Human Sentence Processing", *Working Minimalist*, S.D. Epstein and N. Hornstein eds., syf. 283-315, Cambridge, MA: MIT Pres

134. Wilkes A.L. ve Wade N.J., 1997, "Bain on Neural Networks", Brain and Cognition 33: 295–305

135. Winston P., 1992, "Learning by Building Identification Trees", *Artificial Intelligence*, Addison-Wesley Publishing Company, syf. 423-442

136. Witten I. H. ve Frank E., 2005, "Data Mining: Practical Machine Learning Tools and Techniques (Second Edition) ", ISBN: 0-12-088407-0, Morgan Kaufmann

137. Yang Y. ve Pedersen J. P., 1997, "A Comperative Study on Feature Selection in Text Categorization", The Fourteenth International Conference on Machine Learning, syf. 412-420

138. XTAG Research Group., 1995, "A Lexicalized Tree Adjoining Grammar for English", *Technical Report IRCS 95-03*, University of Pennsylvania

139. Zeman D. ve Sarkar A., 2000, "Learning Verb Subcategorization from Corpora: Counting Frame Subsets", *2nd International Conference on Language Resources and Evaluation (LREC2000)*. Athens, Greece

140. Zeman D., 2002, "Can Subcategorization Help a Statistical Dependency Parser?", *Proceedings of the 19th international conference on Computational linguistics*, Vol. 1, Taipei, Taiwan, syf. 1-7

141. Zhang Y. ve Zhou Q., 2003, "Chinese Partial Parser for Automatic Extraction of Verb Grammatical Collocations", *Proceedings of 2003 International Conference on Natural Language Processing and Knowledge Engineering*, syf. 677-682

TEZ SIRASINDA YAPILAN ÇALIŞMALAR

Uluslararası Kongre ve Sempozyum Bildirileri

1. Uzun E., Uçar E. ve Kılıçaslan Y., 2004, "A New Approach to Developing A Data Exchange Language as An Alternative to XML", International Scientific Conference'2004 (UNITECH'04), Gabrovo, Bulgaria, November 18-19, Volume I, pp. 318-321.

2. Kılıçaslan Y., Uzun E., Volkan A. ve Uçar E., 2007, "Automatic Acquisition of Subcategorization Frames for Turkish with Purely Statistical Methods", INISTA 2007, International Symposium on INnovations in Intelligent SysTems and Applications June 20-23, 2007 - Istanbul, Turkey (Accepted)

3. Uzun E., Kılıçaslan Y. ve Uçar E. 2007, "Web Based Sentence Collector", Ninth International Scientific Conference'2007 (SMOLYAN-2007), Smolyan, Bulgaria (Accepted)

Ulusal Kongre ve Sempozyum Bildirileri

1. Uzun E., Uçar E. ve Kılıçaslan Y., 2006, "XML'in Zaman ve Yer Etkinliği Açısından İncelenmesi", Bilgi Teknolojileri Kongresi IV, Akademik Bilişim 2006, 0-11 Şubat 2006, Bidiri Kitabı Sayfa: 509

ÖZGEÇMİŞ

Erdinç UZUN, 22 Aralık 1978 yılında Tekirdağ ili Çorlu ilçesinde doğdu. İlköğretimini Velimeşe Atatürk İlkokulu, Orta ve Lise öğrenimini Çorlu Mehmet Akif Ersoy Anadolu Lisesinde tamamladıktan sonra 1997 yılında Trakya Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği bölümünü kazandı ve bu bölümden 2001 yılında mezun oldu. 2001 yılı Eylül ayında Trakya Üniversitesi Mühendislik Mimarlık Fakültesi Bilgisayar Mühendisliği Bölümünde Araştırma Görevlisi olarak çalışmaya başlayan Erdinç UZUN aynı sene Yüksek Lisans çalışmalarına başladı. Yüksek lisansını 2003 yılında başarıyla bitirdi. 2003 yılında Trakya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Ana Bilim Dalı'nda doktora çalışmalarına başladı. Erdinç UZUN halen Trakya Üniversitesi Bilgisayar Mühendisliği Bölümü'nde Araştırma Görevlisi olarak görev yapmaktadır.