ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI
2013-YL-058

TÜRKÇE DOKÜMANLAR İÇİN YAZAR TANIMA

Özcan KOLYİĞİT

Danışmanı:
Yrd. Doç. Dr. Rıfat AŞLIYAN

AYDIN

ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Matematik Anabilim Dalı Yüksek Lisans Programı öğrencisi Özcan KOLYİĞİT tarafından hazırlanan "Türkçe Dokümanlar İçin Yazar Tanıma" başlıklı tez, 03.09.2013 tarihinde yapılan savunma sonucunda aşağıda isimleri bulunan jüri üyelerince kabul edilmiştir.

	Ünvanı, Adı Soyadı	Kurumu	İmzası
Başkan :	Yrd.Doç.Dr. Rıfat AŞLIYAN	ADÜ Fen Edebiyat Fak.	.....
Üye :	Yrd.Doç.Dr. Korhan GÜNEL	ADÜ Fen Edebiyat Fak.	.....
Üye :	Yrd.Doç.Dr. Refet POLAT	Yaşar Üni. Fen Edebiyat Fak.	.....

Jüri üyeleri tarafından kabul edilen bu yüksek lisans tezi, Enstitü Yönetim Kurulunun ……sayılı kararıyla **……...** tarihinde onaylanmıştır.

Prof. Dr. Cengiz ÖZARSLAN
Enstitü Müdürü

ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Bu tezde sunulan tüm bilgi ve sonuçların, bilimsel yöntemlerle yürütülen gerçek deney ve gözlemler çerçevesinde tarafımdan elde edildiğini, çalışmada bana ait olmayan tüm veri, düşünce, sonuç ve bilgilere bilimsel etik kuralların gereği olarak eksiksiz şekilde uygun atıf yaptığımı ve kaynak göstererek belirttiğimi beyan ederim.

24/07/2013

Özcan KOLYİĞİT

ÖZET

TÜRKÇE DOKÜMANLAR İÇİN YAZAR TANIMA

Özcan KOLYİĞİT

Yüksek Lisans Tezi, Matematik Anabilim Dalı
Tez Danışmanı: Yrd. Doç. Dr. Rıfat AŞLIYAN
2013, 57 sayfa

Günümüzde, yazar tanıma çalışmaları, teknolojinin gelişmesi ve bilginin yaygınlaşması ile ortaya çıkan bir takım sorunlara çözüm üretmek için yapılmaktadır. Bu sorunlardan bazıları yazarı belli olmayan dokümanların yazarlarının belirlenmesi ve yazarının kim olduğundan tam olarak emin olunamayan metinlerin yazarlarının belirlenmesidir. Bu çalışmada, Türkçe dokümanlar için yazar tanıma sistemleri geliştirilmiştir. Sistemlerin eğitilmesinde ve test edilmesinde kullanılmak üzere, gazetelerden seçilen 6 yazara ait köşe yazıları kullanılmıştır. Yazarların 70'er makalesinden oluşan 420 dokümandan oluşan bir derlem hazırlanmıştır. Bu dokümanlardan 20'şer tanesi eğitim için, 50'şer tanesi test için kullanılmıştır. İlk olarak, 6 yazara ait dokümanlar toplanmış, daha sonra her yazara ait 20 doküman birleştirilerek tek bir doküman haline getirilmiştir. Bu şekilde elde edilen 6 doküman için sözcük, gövde, hece ve karakter *n*-gramlarının öznitelik vektörleri belirlenmiştir. K-En Yakın Komşu algoritması için öznitelik vektörleri belirlenirken her yazar için vektör uzunlukları 120, 180 ve 240 olarak seçilmiş, oluşan öznitelik vektörleri için K-En Yakın Komşu algoritmasıyla test edilmiştir. En başarılı sonuçlar, vektör boyu 120 olduğunda elde edildiğinden diğer metotlar için de vektör boyu 120 olarak kullanılmıştır. Geliştirilen sistemler eğitildikten sonra test edilerek doğruluk ve F-ölçüsü değerlerine göre birbirleriyle karşılaştırılmıştır.

Anahtar sözcükler: Yazar Tanıma, K-En Yakın Komşu, Çok Katmanlı Algılayıcı, Destek Vektör Makinesi, LVQ, *n*-gram.

ABSTRACT

AUTHOR RECOGNITION FOR TURKISH DOCUMENTS

Özcan KOLYİĞİT

M.Sc. Thesis, Department of Mathematics
Supervisor: Assist. Prof. Dr. Rıfat AŞLIYAN
2013, 57 pages

Today, the studies of author recognition have been made for providing the solutions of the problems which occur by developing and growing of information technology. Some of these problems are to specify the authors who the papers are exactly written by. In this study, some systems about author recognition for Turkish documents have been developed. For generating the systems, we have used the columns which belong to six authors in some newspapers. A corpus which includes totally 420 documents is constructed for training and testing of the systems. Each author has seventy documents. Twenty documents of every author are used for training operation. But, the other documents are utilized for testing stage. The features of word, stem, syllable, character and their *n*-grams are decided for each documents of these six author. Author recognition systems have been developed with the methods as K-Nearest Neighbor, Support Vector Machine, Multi-Layer Perceptron and Learning Vector Quantization. The feature vectors' lengths of the systems developed by K-Nearest Neighbor have been chosen as 120, 180 and 240. Because the most successful results are obtained as the length of the feature vectors is 120, we have used this length for the other methods. After the developed systems are trained the methods, the systems have been tested and evaluated according to accuracy and F-measure values.

Key words: Author Recognition, K-Nearest Neighbor, Multi-Layer Perceptron, Support Vector Machine, Learning Vector Quantization, *n*-gram.

ÖNSÖZ

Hayatım boyunca bana destek olan aileme ve dostlarıma, üniversite eğitimime ilk başladığım günden itibaren bütün eğitimim boyunca özveriyle bilgisini, sabrını ve insani ilgisini esirgemeyen tez danışmanım Yrd. Doç. Dr. Rıfat Aşlıyan'a, bilgisini ve deneyimini her zaman bizlerle paylaşan saygıdeğer hocam Yrd. Doç. Dr. Korhan Günel'e,

Sonsuz teşekkürler...

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

EKLER DİZİNİ

1. GİRİŞ

Yazar tanıma çalışmalarında amaç, yazarı bilinmeyen metinlerin yazarını tespit etmek veya yazarının kim olduğundan tam olarak emin olunamayan metinlerin yazarlarının belirlenmesidir (Stamatatos, 1999). Türkçe için yazar tanıma alanında ilk çalışmalar 1999 yılında yapılmaya başlanmış ve günümüzde, yapılan çalışmaların sayısı hızla artmaktadır.

Aynı doküman üzerinde, yazarlık iddia eden iki kişiden hangisinin dokümanın gerçek yazarı olduğunun tespiti için yazar tanıma uygulamalarından faydalanılır (Fung, 2003).

İlk yazar tanıma çalışmaları Stamatatos ve arkadaşları(Stamatatos, 1999) tarafından yapılmıştır. Sözdizimsel stil özelliklerinin çeşitli kombinasyonlarını kullanarak dokümanların yazarlarını belirlemeye yönelik bir çalışma yapmışlardır. Yunanca dokümanlar üzerinde çalışmışlardır.

Peng vd.(2003), her yazarın en çok kullandığı belli sayıdaki *n*-gramlardan oluşan bir vektör oluşturmuş, daha sonra en yakın komşuluk algoritmasını kullanarak dokümanların yazarlarını belirleyen bir çalışma yapmışlardır.

Fung (2003), Federalist yayınlarının yazarlık özelliklendirilmesi için Destek Vektör Makinesi sınıflandırıcısını kullanılmıştır. Çalışmada Federalist yayınlar "as", "of" ve "on" kelimelerinin üç boyutlu uzayında bir düzlemle ayrılmışlardır. Bir takım fonksiyonel kelimeler kullanılarak Destek Vektör Makinesi uygulanmış ve yayınlar birbirinden ayrılmıştır.

Diri ve Amasyalı (2003), Türkçe metinler üzerinde ilk çalışmaları yapmışlardır. Dokümanın içeriğini ve belirlenen 22 farklı stil özelliğini kullanarak dokümanların yazarlarını belirleyen sınıflandırma yöntemleri ile çalışmışlardır. 18 yazara ait 20'şer dokümandan oluşan bir derlem oluşturmuşlardır.

Doküman içeriğine bağlı sınıflandırmada Naive Bayes metodunu kullanmışlardır. Stil özelliklerine göre sınıflandırmada kendi geliştirdikleri Automatic Author Detection for Turkish Text (AADTT) metodunu kullanmışlardır.

Diri ve Amasyalı (2006), Türkçe metinlerde yazar, tür ve cinsiyete bağlı sınıflandırma yapan bir sistem geliştirmişlerdir. Bu çalışmalarında da Naive Bayes, Destek Vektör Makineleri, C 4.5 ve Rastgele Orman yöntemlerini kullanmışlardır.

Amasyalı vd. (2006), farklı öznitelik vektörleri kullanarak Türkçe dokümanların yazarlarının belirlenmesini amaçlayan bir çalışma yapmışlardır. Türkçenin 2 ve 3-gram'larını, Türkçede sık geçen sözcükleri, dilbilgisel ve istatistiksel özellikleri kullanarak 10 farklı öznitelik vektörü çıkarmışlardır. Daha sonra yine Naive Bayes, Destek Vektör Makineleri, C 4.5 ve Rastgele Orman yöntemlerini kullanmışlardır.

Bu çalışmada, 6 yazara ait 70'er doküman, toplamda 420 doküman kullanılmıştır. Bu dokümanlardan 20'şer tanesi sistemin eğitilmesi için 50'şer tanesi sistemin test edilmesinde kullanılmıştır. Dokümanlar ilk olarak ön işlemden geçirilmiştir. Dokümanlardaki tüm harfler küçük harfe dönüştürülmüş, noktalama işaretleri ve rakamlar silinmiştir.

Temizlenmiş dokümanlar üzerinde 4 farklı yaklaşım ve 4 yöntem uygulanmıştır. Temizlenmiş dokümanların sözcük, en uzun gövde ve hece tabanlının 1-gram, 2-gram ve 3-gramları; karakter tabanlının 2-gram, 3-gram, 4-gram, 5-gram ve 6-gramları için frekanslar belirlenmiştir.

Dokümanlardaki en uzun gövdelerin ve hecelerin belirlenmesinde L-M (Longest-Match Algoritması) (Kut vd., 1995) ve RASAT Heceleme Algoritması (Aşlıyan ve Günel, 2011) kullanılmıştır.

Sınıflandırma metodu olarak Destek Vektör Makinesi (DVM) (Joachims, 1998; Yang ve Liu, 1999), ÇKA (Çok Katmanlı Algılayıcı), LVQ (Öztemel, 2003) ve K-NN (K-En Yakın Komşu) (Soucy ve Mineau, 2001) metotları kullanılmıştır.

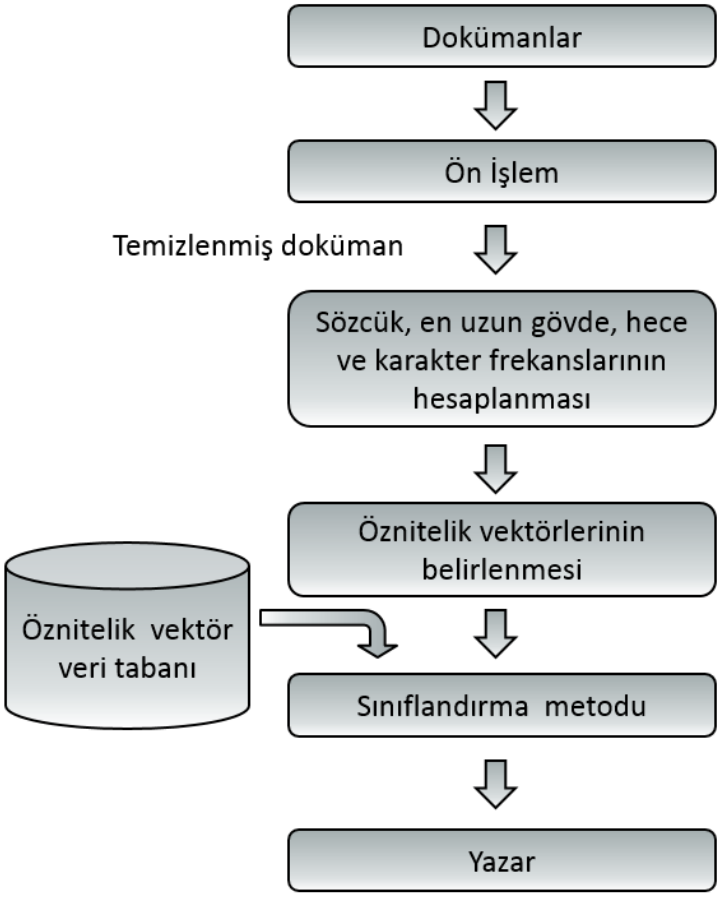
2. SİSTEM MİMARİSİ

Bu çalışmada, sözcük, gövde, hece ve karakter tabanlı K-NN, ÇKA, DVM ve LVQ metotlarının kullanıldığı Türkçe metin dokümanları üzerinde çalışan yazar tanıma sistemi geliştirilmiştir. Bu sistemler için ilk olarak Türkçe dokümanlardan oluşan bir derlem hazırlanmıştır.

Şekil 2.1'de bu sistemler genel olarak bir kaç safhadan oluşmaktadır. Sisteme alınan dokümanlar, ön işlem safhasında, öncelikle bir temizleme işleminden geçirilmiş, dokümanlarda bulunan noktalama işaretleri kaldırılmış büyük harfler küçük harflere dönüştürülmüştür.

Sonraki aşamada, sözcük, en uzun gövde, hece, karakter ve *n*-gramlarının her bir doküman içindeki frekansları hesaplanıp normalize edilmiştir. Böylece, her doküman bir vektörle temsil edilmiştir. Geliştirilen sistemlerde K-NN, ÇKA, DVM ve LVQ olmak üzere dört farklı sınıflandırma metodu kullanılmıştır. K-NN metodu için vektör uzunluğu 120, 180 ve 240 olarak belirlenmiştir. En iyi sonuçlar vektör boyu 120 olduğunda elde edildiğinden ÇKA, DVM ve LVQ metotları için vektör boyu120 olarak kabul edilmiştir.

Sistem, bir dokümanı alıp hangi yazara ait olduğuna K-NN metoduyla karar verirken öznitelik vektör veritabanındaki örneklerle benzerliklerine göre karar vermektedir. Diğer metotlarda ise eğitim safhasında her bir sınıf için sistem modelleri oluşturulur ve bu modellerle dokümanların hangi yazara ait olduğu belirlenir.



Şekil 2.1. Yazar tanıma sisteminin genel yapısı

3. MATERYAL VE METOT

3.1 Özniteliklerin Elde Edilmesi

Öznitelikler oluşturulurken dokümanlarda sıkça geçen ve yazarın belirlenmesinde ayırt edici özelliğe sahip olmayan ifadeler dikkate alınmamıştır. Bunun gerçekleştirebilmek için her yazara ait diğer yazarlar tarafından daha az tercih edilen sözcük, gövde, hece ve karakter *n*-gramları belirlenmiştir. Özniteliklerin belirlenmesinde 0,5 eşik değeri uygulanmıştır. Örneğin, eğitim için kullanılan 120 doküman içerisinde A yazarına ait dokümanlarda "bilgisayar" sözcüğünün bulunma olasılığı 0,042 ve diğer yazarların dokümanlarında ise 0,021'den az ise bu sözcük, A yazarı için öznitelik olabilir. Bu şekilde öznitelikler elde edilmiştir.

3.1.1. Gövdeleme Algoritması

Gövde tabanlı öznitelik vektörleri elde etmek için öncelikle tüm dokümanlar gövdeleme sisteminden geçirilmiştir. Daha sonra gövde *n*-gramları oluşturulmuştur. Bu *n*-gramlardan oluşan tüm dokümanlar için frekansları belirlenmiş, sınıflandırma metotları kullanılarak eğitim ve test aşamaları gerçekleştirilmiştir.

Bu çalışmada kullanılan gövdeleme algoritması L-M(Longest-Match) algoritmasıdır (Kut vd., 1995). Longest-Match algoritması aşağıda verilmiştir.

1- Sözcüğe noktala işareti ile eklenmiş olan ekleri kaldır.
2- Sözcüğü sözlükte ara.
3- Eğer sözcük sözlükte bulunursa 5. adıma geç.
4- Eğer sözcük tek bir harften oluşuyorsa 6.adıma geç. Aksi halde sözcükteki son harfi kaldırıp 2. adıma geç.
5- Sözlükte bulunan sözcüğü gövde olarak seçip 7. adıma geç.
6- Aranan sözcüğü bulunamayan kayıtlara ekle.
7- Çıkış.

3.1.2. Heceleme Algoritması

Heceleme algoritmaları doğal dil işlemenin birçok alanında yaygın olarak kullanılmaktadır. Özellikle dil tanıma, sözcük düzeltme ve metin tanıma gibi alanlarda hecelerin dil birimi olarak kabul edilerek başarılı sonuçlar vermektedir. Hece algoritmaları genel olarak kök merkezli ve veritabanlı olmak üzere iki yaklaşıma göre yapılmaktadır (Adsett vd.; Marchand, 2009).

Türkçe, eklemeli bir dildir ve heceler en fazla beş harften meydana gelmektedir. Heceler, sadece bir tane sesli harf içermektedir. Tek başına bir tane sesli harf de hece olabilmektedir. Çizelge 3.1'de hece yapıları ve buna bağlı olarak hece örnekleri verilmektedir. Burada V, sesli harfi; C ise sessiz harfi temsil etmektedir.

Günümüzde, literatürde üç tane hece algoritması (Ucoluk ve Toroslu, 1997; Akın, 2005; Aşlıyan ve Günel, 2011) dikkati çekmektedir. Bu çalışmamızda, diğerlerinden daha hızlı ve daha başarılı olan RASAT (Aşlıyan ve Günel, 2011) heceleme algoritmasını kullandık. RASAT algoritmasındaki heceleme mantığı şu şekildedir. İlk olarak hash fonksiyonu kullanarak bir sözcükteki sesli harfler tespit edilir. Sonra, sözcükteki ardışık sesli harfler arasındaki sessiz harflerin sayısına göre uygun olan sessiz harfler bölünerek hecelere ayrılır.

Çizelge 3.1. Türkçedeki hece yapıları

HECE YAPILARI	**ÖRNEK HECELER**
V	A, e, ı, i, o, ö, u, ü
VC	Ab, aç, ak, az, ek, ...
CV	Ba, be, bı, ra, ...
CVC	Bel, gel, cam, gör, kal, ...
VCC	Alt, üst, ilk, ...
CCV	Bre, gri, ...
CVCC	Kurt, kırk, kalk, renk, ...
CCVC	Tren, krem, ...
CCVCC	Tvist, ...

3.1.3 Morfolojik Analiz

Morfoloji, kelimelerin yapısının, biçimlerinin incelenmesini ve türünün belirlenmesini sağlayan dilbiliminin bir dalıdır. Türkçede morfolojik analiz yapmak güçtür. Çünkü, kök ve gövdelere sondan eklenen ekler sözcüğün anlamını değiştirebilmektedir. Bu nedenle Türkçe metinler için kök analizi ve ek analizi mutlaka yapılmalıdır.

Kök Analizi: Türkçe dokümanlarda geçen sözcüklerin köklerinin tespiti için öncelikle kök veri tabanı oluşturulmalıdır. Dokümanlarda geçen sözcükler içerisinde veri tabanındaki kökler aranır. Burada dikkat edilmesi gereken sözcüğün doğru kökünün tespit edilmesidir. Örneğin; "çalışmak" sözcüğünde kök analizi yapılarak "çal" tespit edilebilir. Bu durumda sözcüğün kökü yanlış belirlenmiş olur. Bu nedenle kök analizi yanında ek analizi de yapılmalıdır.

Ek Analizi:Türkçede ekler yapım eki ve çekim eki olmak üzere ikiye ayrılır. Çekim ekleri sözcüklerin sonuna eklenerek cümle içerisinde anlam bağlantısı kurmalarını sağlar. Fakat, yapım ekleri isim ya da fiil kök veya gövdelerine eklenerek onların anlamını bazen de türünü değiştirirler. Bu nedenle morfolojik analiz yapılırken sözcükte bulunan çekim ekleri atılırken yapım ekleri atılmaz. Yapım eki almış kökler gövde olarak ele alınırlar.

Türkçe sözcüklerde kök ve gövdelerin belirlenmesinde bazı metotlar kullanılmıştır. Bunlar:

1. AF Algoritması (Solak ve Can, 1994)
2. Longest-Match (L-M) Algoritması (Kut vd., 1995)
3. Identified Maximum Match Algoritması (Köksal, 1975)
4. FindStem Algoritması (Sever ve Bitirim, 2003)
5. Solak ve Oflazer Algoritması (Solak ve Oflazer, 1993)

6. Root Reaching Method without Dictionary (Cebiroğlu ve Adalı, 2002)

7. Extended Finite State Approach (Oflazer, 1999)

3.2 Kullanılan Yöntemler

3.2.1 K-En Yakın Komşu Algoritması

Sınıfları belli olan bir örnek kümesindeki gözlem değerlerinden yararlanarak, örneğe katılacak yeni bir gözlemin hangi sınıfa ait olduğunu belirlemek amacı ile K-En Yakın Komşu algoritması (K-Nearest Neighbors Algorithm) kullanılmaktadır.

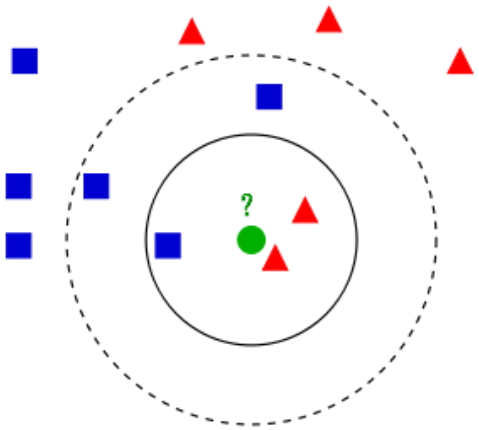
Bu yöntem, örnek kümedeki gözlemlerin her birinin, sonradan belirlenen bir gözlem değerine olan uzaklıklarının hesaplanması ve en küçük uzaklığa sahip k sayıda gözlemin bulunduğu sınıfın seçilmesi esasına dayanmaktadır.

Örneğin, $k = 3$ için yeni bir eleman sınıflandırılmak istensin. Bu durumda eski sınıflandırılmış elemanlardan en yakın 3 tanesi alınır. Bu elamanlar hangi sınıfa dâhil ise, yeni eleman da o sınıfa dâhil edilir.

Uzaklıkların hesaplanmasında Öklid uzaklık formülü kullanılabilir. Aralarındaki uzaklık hesaplanacak $\boldsymbol{x}$ ve $\boldsymbol{y}$ vektörleri için aşağıdaki Öklid uzaklık formülü kullanılabilir.

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\sum_{k=1}^{p} \left(x_{ik} - x_{jk}\right)^2} \tag{3.1}$$

Şekil 3.1'de K-NN algoritması ile ilgili basit bir örnek verilmiştir. Mavi karelerden ve kırmızı üçgenlerden oluşan iki sınıfımız olsun. Yeşil daire ise sınıfını belirlemek istediğimiz test verimiz olsun. Eğer $k = 3$ seçilirse dairemize yakın iki üçgen bir kare olduğundan üçgen sınıfını seçmeliyiz. Fakat $k = 5$ seçilirse dairemize yakın 3 kare 2 üçgen olduğundan kare sınıfını seçmeliyiz. Bu nedenle k'nın seçimi kritiktir.



Şekil 3.1.K-En Yakın Komşu algoritması

Bu çalışmada, ilk olarak 6 yazara ait dokümanlar toplanmış, daha sonra her yazara ait 20 doküman birleştirilerek tek bir doküman haline getirilmiştir. Bu şekilde elde edilen 6 dokümanlara göre sözcük, en uzun gövde, hece ve karakter öznitelik vektörleri belirlenmiştir.

Ön işlem uygulanmış dokümanlardaki sözcüklerin, en uzun gövdelerin, hecelerin ve karakterlerin frekansları normalize edilerek hesaplanmıştır. 6 yazara ait frekanslar oluşturulduktan sonra her yazar için diğer yazarlar tarafından daha az tercih edilen (0,5'den daha az) sözcükler, gövdeler, heceler ve karakterler seçilerek öznitelik vektörleri oluşturulmuştur.

Her yazar için oluşturulan öznitelik vektörleri birleştirilerek öznitelik veri tabanı oluşturulmuştur. Altı yazarın her biri için öznitelik sözcük, en uzun gövde, hece ve karakter sayısı 20, 30 ve 40 seçilerek 3 farklı öznitelik vektörü oluşturulmuş, ayrı ayrı test edilmiştir.

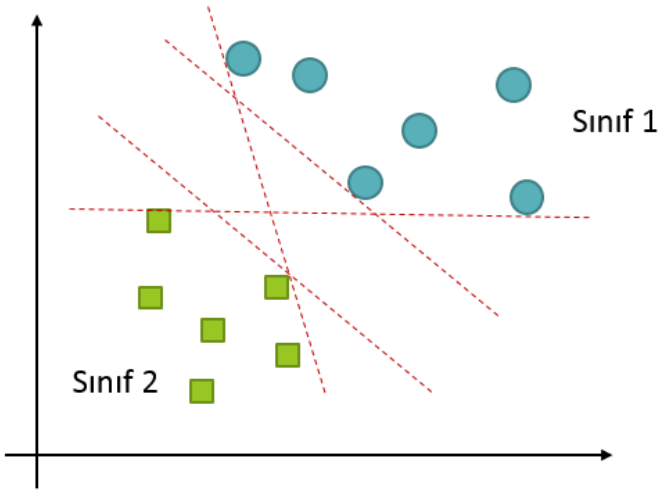
Eğitim seti ve test seti oluşturulurken her dokümanın öznitelik vektörlerindeki 120, 180 ve 240 sözcük, en uzun gövde, hece ve karakter için frekansları hesaplanmıştır. Oluşan 420 vektörün (her yazar için 70 adet) 20'şer tanesi eğitim için ayrılmış geri kalanları test için kullanılmıştır.

Test için K-NN metodu kullanılmıştır. Test setindeki her bir dokümanın eğitim setindeki her bir doküman ile arasındaki uzaklık Öklid uzaklık formülü kullanılarak hesaplanmıştır.k=1, k=3, k=5 ve k=7için K-NN metodu uygulanmıştır.

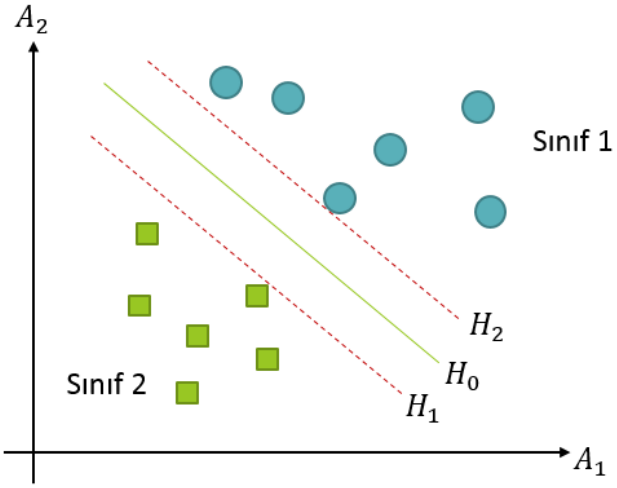
3.2.2 Destek Vektör Makinesi

Destek Vektör Makinesi (DVM) yöntemi, hem doğrusal hem de doğrusal olmayan fonksiyonlar yardımı ile bir veri grubunu sınıflandırır. Sınıflandırma yapmak için doğrular ya da çok boyutlu uzaylarda hiper düzlemler kullanılabilir. Bu yöntemin amacı, bu doğru ya da hiper düzlemler arasından en uygun olanı bulmaktır (Öztemel, 2003).

Şekil 3.2'de iki boyutlu uzayda doğrusal olarak ayrılabilen veri grubu için tercih edilebilecek doğrular gösterilmiştir. Şekil 3.3'te iseH_1 ve H_2 hiper düzlemleri arasındaki muhtemel en büyük boşluk kullanılarak elde edilen sınıflandırıcı hiper düzlem gösterilmektedir. BuradakiH_0 düzlemine optimal ayırma hiper düzlemi adı verilir.

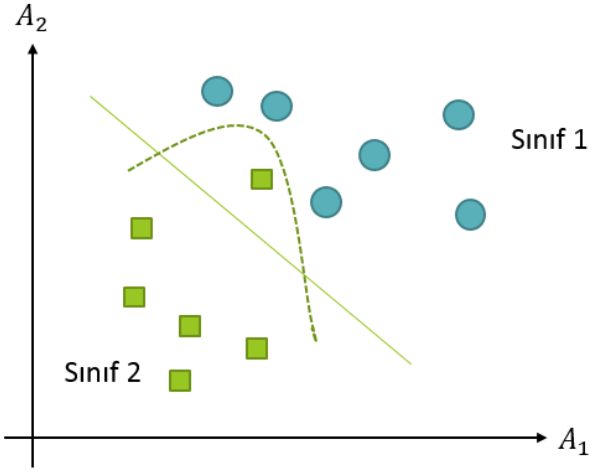


Şekil 3.2. Doğrusal sınıflandırıcılar

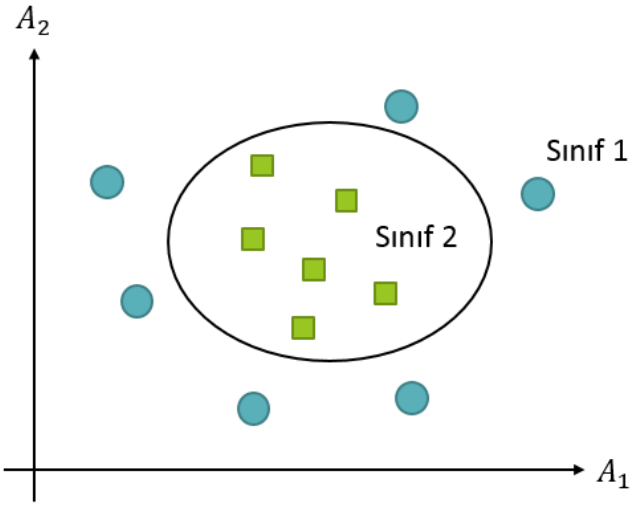


Şekil3.3. Hiper Düzlem

Doğrusal olarak ayrılamayan veri grupları için Şekil 3.4 ve 3.5'teki gibi doğrusal olmayan sınıflandırıcılar kullanılabilir.



Şekil 3.4. Doğrusal olmayan sınıflandırıcılar



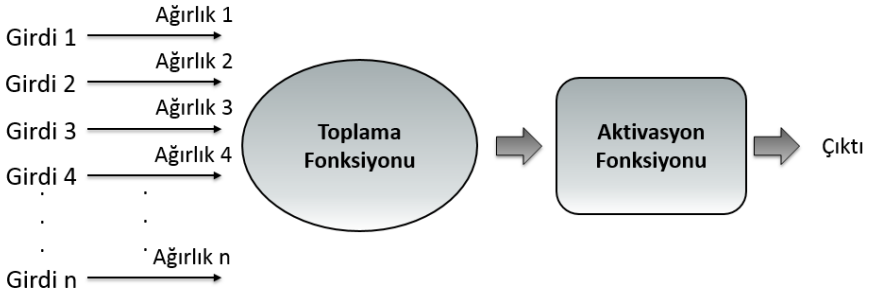
Şekil 3.5. Doğrusal olmayan sınıflandırıcılar

DVM metodu, başarılı bir sınıflandırma metodu olduğu için sınıflandırma ve yazar tanıma çalışmalarında sıklıkla kullanmaktadır (Diederich vd. 2003).

3.2.3 Yapay Sinir Ağları

Yapay sinir ağları, insan beyninden esinlenerek geliştirilmiş, insan beyninin gerçekleştirebildiği öğrenme, yeni bilgiler oluşturabilme ve veriler arasında ilişki kurabilme yeteneklerinin gerçekleştirilebildiği bilgisayar sistemleridir.

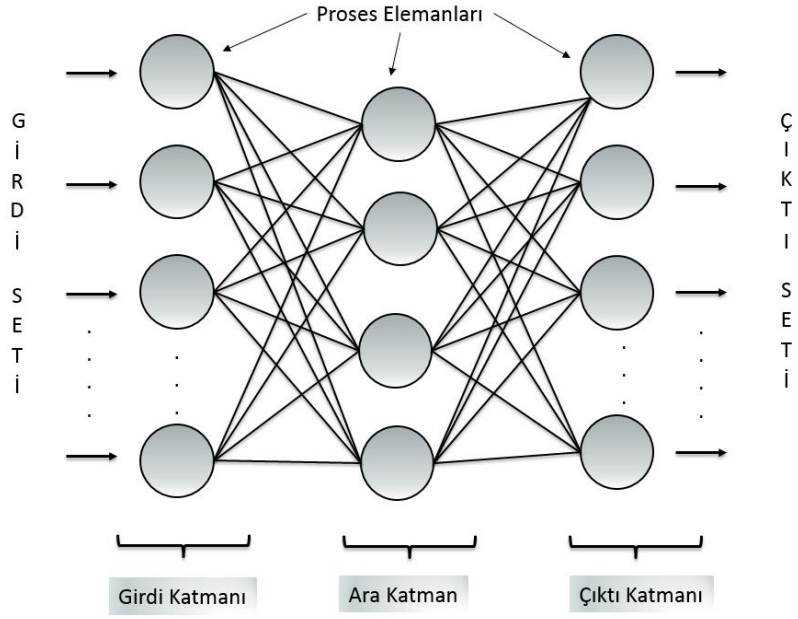
Yapay sinir ağları öğrenme işlemini örnekler yardımı ile yaparlar. Birbirine bağlı proses elemanı adı verilen yapay sinir hücrelerinden oluşurlar. Proses elemanları arasındaki bağlantıların her birinin ağırlık değeri vardır. Yapay Sinir ağının sahip olduğu bilgi bu ağırlıklarda saklanmaktadır. Şekil 3.8'de bir yapay sinir hücresinin yapısı verilmiştir.



Şekil 3.6. Yapay Sinir hücresinin yapısı

Yapay Sinir Ağlarını oluşturan proses elemanları ağ içerisinde rastgele bulunmazlar. Genel olarak hücreler katmanlar halinde bulunurlar. Yapay Sinir ağlarında dış dünyadan bilgilerin alındığı ve herhangi bir işleme uğramadan ara katmanlara aktarıldığı bir girdi katmanı; girdi katmanından alınan bilgilerin işlendiği bir veya daha fazla sayıda bulunabilen ara katmanlar; ara katman veya katmanlardan alınan bilgilerin işlendiği ve çıktıların dış dünyaya gönderildiği bir çıktı katmanı bulunur. Şekil 3.7'de bir yapay sinir ağının genel yapısı gösterilmiştir.

Yapay sinir ağlarında proses elemanlarının ağırlıklarının belirlenmesine ağın eğitilmesi denir. Ağ oluşturulurken rastgele atanan ağırlık değerleri ağa örnekler gösterildikçe değiştirilirler. Her örnekte ağın çıktıları istenilen değere yaklaştırılmaya çalışılır. Bu olaya ağın öğrenmesi denir. Yani ağ her örnekte ağırlıklarını değiştirerek örneklerin genel yapısı hakkında bilgi edinir. Ağırlıkların değiştirilmesi belli kurallara göre gerçekleştirilmektedir. Bu kurallara öğrenme kuralları denir. Yapay sinir ağları modelleri için birçok öğrenme kuralı geliştirilmiştir.



Şekil 3.7. Yapay sinir ağının genel yapısı

Ağın eğitilmesi için ilk olarak ağa gösterilen örneğe göre çıktısı belirlenir. Daha sonra, bu çıktının beklenen değere yakınlığına göre ağırlıklar değiştirilir. Her örnekte bu tekrarlanarak ağın eğitilmesi tamamlanır. Ağ test edilirken eğitim sırasında gösterilmeyen örnekler ağa gösterilir ve ağın performansı belirlenir. Test aşamasında ağda herhangi bir değişiklik yapılmaz.

Yapay sinir ağlarından en çok kullanılan modellerden bazıları aşağıda listelenmiştir.

- Algılayıcılar
- Çok katmanlı algılayıcılar (ÇKA)
- Vektör Kuantizasyon modelleri (LVQ)
- Kendi kendini organize eden model (SOM)
- Adaptif Rezonans Teorisi modelleri (ART)
- Hopfield ağları
- Elman ağı

3.2.3.1 Çok Katmanlı Algılayıcı (ÇKA)

Çok Katmanlı Algılayıcı modeli Rumelhart ve arkadaşları (1986) tarafından XOR probleminin çözümüne yönelik olarak geliştirilmiştir. Bu model ilk yapay sinir ağları modellerinden farklı olarak aralarında doğrusal olmayan ilişkiler bulunan veri gruplarını iki veya daha fazla sınıfa ayırabilmesidir(Öztemel, 2003). Çizelge 3.2'de XOR problemi verilmiştir. Bu problem üzerinde çalışma yapılmasının nedeni basit algılayıcı modelinin bu probleme çözüm üretemediğinin görülmesidir.

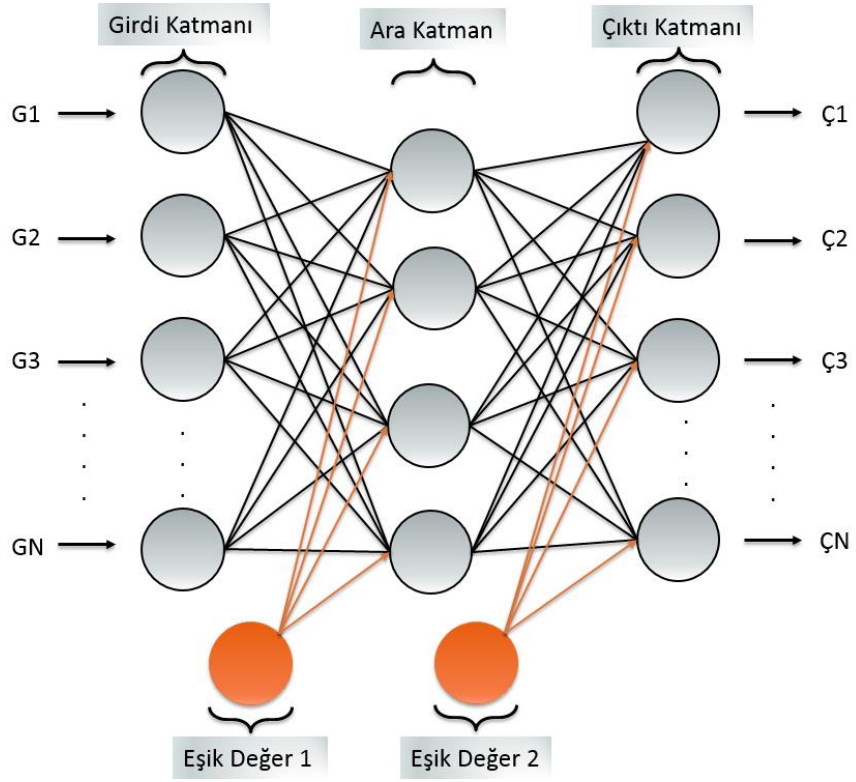
Çizelge 3.2. XOR problemi

	Girdi 1	Girdi 2	Çıktı
Örnek 1	0	0	0
Örnek 2	0	1	1
Örnek 3	1	0	1
Örnek 4	1	1	0

ÇKA ağlarının genel yapısı Şekil 3.8'de gösterilmiştir. ÇKA modelinde girdi katmanı, ara katman ve çıktı katmanı yanı sıra eşik değerler bulunur.

Çok Katmanlı Algılayıcı modeli, öğretmenli öğrenme stratejisini kullanır. Yani ağa hem girdiler hem de girdilerden elde edilmesi gereken çıktılar verilir. Başlangıçta rastgele atanan ağırlıklar, ağa örnekler sunuldukça değiştirilir. Böylelikle ağ kendisine gösterilen örneklerden genelleme yaparak problemin çözümüne yönelik bir yapay sinir ağı oluşturur.

Öğrenme gerçekleşirken önce ileri doğru hesaplama daha sonra geri doğru hesaplama yapılır. İleri doğru hesaplama safhasında ağ verilen örnekler için bir çıktı belirler. Bu çıktı elde edilmesi gereken çıktıyla karşılaştırılır. Ağın ürettiği hata değerleri hesaplanır. Geri hesaplama safhasında ise hatanın azaltılması için ağırlıkların değiştirilir. Gerçekleşen çıktılar ile beklenen çıktılar arasındaki fark kabul edilebilir düzeye inene kadar ileri hesaplama ve geri hesaplama işlemleri gerçekleştirilir.



Şekil 3.8. ÇKA modeli

3.2.3.2 LVQ Modeli

LVQ (Learning Vector Quantization) modeli destekleyici öğrenme stratejisine sahip yapay sinir ağı modelidir (Öztemel, 2003). Öğretmenli öğrenme stratejisinden farklı olarak destekleyici öğrenme stratejisinde eğitim için ağa hem girdi değerleri hem de beklenen çıktı değerleri verilmez. Yalnızca, LVQ ağına, girdi değerleri için üretilen çıktı değerlerinin doğru ya da yanlış olduğu belirtilmektedir.

LVQ modelinde amaç çok boyutlu bir vektörü bir vektörler seti ile ifade etmektir. Yani, bir vektörü belirli sayıda vektörler ile temsil etmektir. Eğitim sırasında ağ,

girdi vektörlerinin hangi vektör seti ile ifade edileceğini belirler. Bu vektör setine referans vektörleri denir.

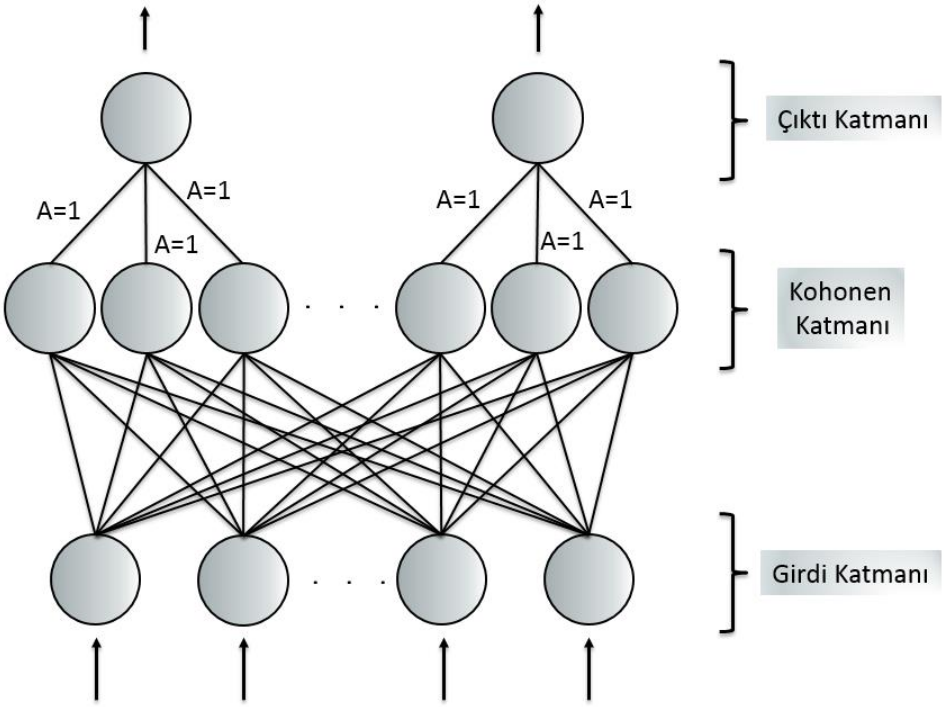
Bu modelde, çıktılardan sadece biri 1 değerleri 0 değerlerini alır. Çıktı değerinin 1 olması girdinin ilgili çıktının temsil ettiği sınıfa ait olduğunu göstermektedir.

Eğitim sırasında girdi vektörünün, referans vektörleri arasından hangisine daha yakın olduğu en yakın komşu kuralı ile belirlenmektedir. Böylece girdi vektörünün ona en yakın vektör setinin temsil ettiği gruba dâhil olduğu varsayılmaktadır. Eğer girdi vektörü doğru sınıflandırılmışsa girdi vektörüne en yakın vektörün ağırlıkları değiştirilir. Bu öğrenme stratejisine destekleyici öğrenme stratejisi, çıktıların belirlenmesinde kullanılan stratejiye ise "kazanan her şeyi alır" stratejisi denilmektedir.

LVQ ağı, ÇKA ağlarında olduğu gibi 3 katmandan oluşmaktadır. ÇKA modelinden farklı olarak ara katmana Kohonen katmanı denmektedir. Kohonen katmanının farklı yanı ise girdi katmanındaki her proses elemanı ile tam bağlantılı fakat çıktı katmanındaki proses elemanları ile kısmi bağlantılı olmasıdır. Kohonen katmanındaki her proses elemanı çıktı katmanında yalnız bir proses elemanı ile bağlantılıdır. Bu ağırlıklar 1'de sabitlenmiştir. Yani öğrenme sırasında girdi katmanı ile Kohonen katmanı arasındaki ağırlıklar değiştirilir.

Kohonen katmanındaki proses elemanları birbiri ile yarışırlar. Kazanan proses elemanının çıktısı 1 diğerleri 0 değerini alır.

LVQ modelinin topolojik yapısı Şekil 3.9' da gösterilmiştir.



Şekil 3.9. LVQ modeli

4. SİSTEMİN UYGULANMASI VE BULGULAR

4.1. Veri Tabanının Oluşturulması

Yazar tanıma sistemi oluşturulurken ilk olarak veri tabanı oluşturulmuştur. Rastgele seçilmiş 6 yazara ait köşe yazıları gazetelerin internet sitelerinden alınmıştır. Daha sonra, bu dokümanlar için öncelikle sözcük frekansları, en uzun gövdeleme (Longest-Match) algoritması ile dokümanlarda geçen sözcüklerin gövdeleri tespit edilerek gövde frekansları, RASAT heceleme algoritması kullanılarak hecelere ayrılan dokümanlarda hece frekansları ve son olarak karakter frekansları belirlenmiştir. Sözcük, en uzun gövde ve hece tabanlı veri tabanı hazırlanırken 1-gram, 2-gram ve 3-gram, karakter tabanlı veri tabanı hazırlanırken 2-gram, 3-gram, 4-gram, 5-gram ve 6-gram için frekanslar hesaplanmıştır.

4.2. Sistemin Değerlendirilmesi

Bu çalışmada, sonuçların değerlendirilmesinde Kesinlik skoru (Precision), Hassasiyet skoru (Recall), ortalama doğruluk (Accuracy) değerleri ve F-ölçüsü (F-measure) değerleri kullanılmıştır. Yazar tanıma sistemi temelde bir doküman sınıflandırma sistemidir. Bu nedenle, bu çalışmada doküman sınıflandırmada en çok kullanılan yaklaşım ve metotlar kullanılmıştır.

Doküman sınıflandırmada çok sık karşılaşılan kıyaslama yöntemlerinden olan kesinlik skoru ve hassasiyet skoru değerlerinin hesaplanmasında kullanılan Doğru Pozitif (DP), Doğru Negatif (DN), Yanlış Pozitif (YP) ve Yanlış Negatif (FN) değerleri sonucu "Pozitif" ve "Negatif" olan iki sınıf için dört farklı olası sonuçları verir (Yılmaz,2013). DP, pozitif sınıfında olan dokümanların, pozitif olarak sınıflandırılmış dokümanların sayısıdır, yani doğru sınıflandırmış olur. YP, negatif sınıfında olan dokümanların, pozitif olarak sınıflandırılmış dokümanların sayısıdır, yani yanlış sınıflandırmış olur. YN, pozitif sınıfında olan dokümanların, negatif olarak sınıflandırılmış dokümanların sayısıdır; yanlış sınıflandırılır. DN, negatif sınıfında olan dokümanların, negatif olarak sınıflandırılmış dokümanların sayısıdır; doğru sınıflandırılmış olur.

DP, pozitif sınıfındaki dokümanların doğru sınıflandırma sayısını, DN ise negatif sınıfındaki dokümanların doğru sınıflandırma sayısını gösterir. Her iki durumda da

doğru sınıflandırma yapmış olunur. Bu olasılık sonuçları Çizelge 4.1.'de hata matrisi olarak adlandırılan bir matrisle gösterilir. Hata matrisinde köşegen elemanları doğru tanıma sayısını, köşegen dışı elemanlar ise yanlış karar sayısını gösterir.

Çizelge 4.1. Hata matrisi

	"Pozitif" Sınıfı	**"Negatif" Sınıfı**
Test Sonucu "Pozitif"	DP	YP
Test Sonucu "Negatif"	YN	DN

Kesinlik skoru bir sınıftaki doğru olarak sınıflandırılan dokümanların sayısının, o sınıftaki toplam doküman sayısına oranını; hassasiyet skoru ise bir sınıftaki doğru olarak sınıflandırılan dokümanların sayısının, sistemin o sınıf olarak tespit ettiği toplam doküman sayısına oranını verir.

Kesinlik skoru ve hassasiyet skoru değerlerini Denklem 4.1 ve Denklem 4.2'de gösterecek olursak:

$$Kesinlik\ skoru = \frac{DP}{DP+YP} \tag{4.1}$$

$$Hassasiyet\ skoru = \frac{DP}{DP+YN} \tag{4.2}$$

Kesinlik skoru ve hassasiyet skoru değerlerinin her ikisinin de aynı anda iyi olması amacıyla F-ölçüsü değeri hesaplanır. F-ölçüsü değeri Denklem 4.3'deki gibi hesaplanır.

$$F - \text{ölçüsü} = \frac{2 \times Kesinlik\ skoru \times Hassasiyet\ skoru}{Kesinlik\ skoru + Hassasiyet\ skoru} \tag{4.3}$$

4.3 Sistemin Eğitilmesi ve Test Edilmesi

Yazar tanıma sistemi oluşturmak için 6 farklı yazara ait 70'şer dokümandan oluşan bir derlem hazırlanmıştır. Bu dokümanlardan 20 tanesi eğitim için 50 tanesi test için kullanılmıştır. K-En Yakın Komşu modelinde maksimum öznitelik sayısı 120, 180 ve 240 olacak şekilde eğitimler gerçekleştirilmiştir. En yakın *k* değeri olarak 1, 3, 5 ve 7 alınarak test sonuçları elde edilmiştir.

K-NN metodu kullanılarak sözcük, gövde, hece ve karakter tabanlı ortalama doğruluk değerleri ve F-ölçüsü değerleri sırasıyla Çizelge 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8 ve 4.9'da verilmiştir.

Çizelge 4.2. K-NN'ye göre sözcük tabanlı ortalama doğruluk değerleri

		k-değerleri	**Öznitelik Sayısı**		
			120	**180**	**240**
Sözcük tabanlı	**1-gram**	**1**	88,3	88,6	85,7
		3	**89,0**	88,0	85,4
		5	87,9	86,8	84,1
		7	87,2	85,7	83,4
	2-gram	**1**	78,6	76,3	75,6
		3	**79,4**	77,2	76,6
		5	78,6	75,8	75,3
		7	77,8	75,4	75,0
	3-gram	**1**	77,2	77,3	75,7
		3	**77,6**	76,6	75,1
		5	76,1	74,6	73,1
		7	74,8	73,6	72,9

K-NN metoduna göre sözcük tabanlı sistemlerde en başarılı sonuçlar öznitelik sayısı 120, *k* değeri 3 ve *n*-gram için 1 alındığında elde edilmiştir. Ortalama doğruluk oranı %89 ve F-ölçüsü değeri %66,7 olarak bulunmuştur.

Çizelge 4.3. K-NN'ye göre sözcük tabanlı ortalama F-ölçüsü değerleri

		k-değerleri	Öznitelik Sayısı		
			120	180	240
Sözcük tabanlı	1-gram	1	65,5	66,0	56,5
		3	**66,7**	64,3	55,5
		5	63,7	60,4	50,6
		7	62,0	56,8	48,7
	2-gram	1	35,9	27,2	23,3
		3	**36,7**	28,2	25,9
		5	32,7	22,6	20,7
		7	29,7	21,2	19,4
	3-gram	1	28,5	28,9	24,0
		3	**29,7**	28,8	20,7
		5	26,1	19,1	10,7
		7	20,4	13,3	9,1

Çizelge 4.4. K-NN'ye göre gövde tabanlı ortalama doğruluk değerleri

		k-değerleri	Öznitelik Sayısı		
			120	180	240
Gövde tabanlı	1-gram	1	87,4	87,8	88,2
		3	90,1	90,3	90,0
		5	**90,9**	89,4	89,6
		7	89,7	88,1	87,7
	2-gram	1	**83,7**	81,2	79,4
		3	82,6	80,7	79,0
		5	83,4	81,9	79,0
		7	82,3	79,7	76,9
	3-gram	1	77,3	**78,3**	75,6
		3	77,2	**78,3**	75,0
		5	76,4	75,4	74,4
		7	75,7	75,0	75,9

Gövde tabanlı sistemlerde ise en yüksek sonuçlar öznitelik sayısı 120, *k* değeri 5 ve *n*-gram değeri 1 alındığında bulunmuştur. Buna göre, ortalama doğruluk değeri %90,9, F-ölçüsü değeri ise %70,7 olarak tespit edilmiştir.

Çizelge 4.5. K-NN'ye göre gövde tabanlı ortalama F-ölçüsü değerleri

		k-değerleri	**Öznitelik Sayısı**		
			120	**180**	**240**
Gövde tabanlı	**1-gram**	**1**	62,3	63,3	64,5
		3	70,2	69,7	68,8
		5	**70,7**	67,2	67,3
		7	66,8	62,5	61,4
	2-gram	**1**	**52,1**	45,4	38,5
		3	46,6	42,3	35,7
		5	46,3	41,7	33,9
		7	42,8	37,0	26,9
	3-gram	**1**	30,2	**34,9**	23,1
		3	29,1	33,7	20,7
		5	27,2	22,1	17,2
		7	23,5	19,6	20,5

Hece tabanlı en yüksek değerler öznitelik sayısı 120, *k* değeri 3 ve 2-gram ele alındığında ortalama doğruluk değeri %90,1 ve F-ölçüsü değeri ise %69,5 olarak elde edilmiştir.

Çizelge 4.6. K-NN'ye göre hece tabanlı ortalama doğruluk değerleri

		k-değerleri	Öznitelik Sayısı		
			120	180	240
Hece tabanlı	1-gram	1	**87,7**	87,2	87,4
		3	87,0	87,2	87,0
		5	86,3	85,9	86,0
		7	85,6	85,3	85,2
	2-gram	1	89,0	89,1	88,8
		3	**90,1**	88,9	87,7
		5	89,7	86,7	86,1
		7	87,7	85,2	85,0
	3-gram	1	86,1	86,1	84,9
		3	86,1	**87,4**	86,7
		5	86,4	86,2	84,7
		7	85,7	84,8	83,3

Çizelge 4.7. K-NN'ye göre hece tabanlı ortalama F-ölçüsü değerleri

		k-değerleri	Öznitelik Sayısı		
			120	180	240
Hece tabanlı	1-gram	1	**61,5**	60,5	61,0
		3	58,3	59,1	58,6
		5	55,4	54,6	55,3
		7	52,5	52,4	52,1
	2-gram	1	67,0	66,5	65,4
		3	**69,5**	65,3	61,8
		5	67,8	57,3	56,0
		7	61,0	51,8	52,3
	3-gram	1	58,5	58,5	55,3
		3	57,2	**61,6**	57,5
		5	56,6	57,2	51,3
		7	54,5	51,6	47,1

Çizelge 4.8. K-NN'ye göre karakter tabanlı ortalama doğruluk değerleri

		k-değerleri	Öznitelik Sayısı		
			120	180	240
Karakter tabanlı	2-gram	1	**83,9**	83,6	83,4
		3	82,9	82,8	82,7
		5	81,9	81,8	81,8
		7	79,7	79,7	79,8
	3-gram	1	**87,0**	86,7	86,7
		3	86,0	86,6	86,6
		5	86,1	86,3	86,3
		7	85,8	86,7	86,7
	4-gram	1	89,7	89,0	89,8
		3	**90,7**	89,0	89,9
		5	90,0	89,1	89,4
		7	89,0	88,4	89,2
	5-gram	1	88,9	88,7	89,0
		3	**89,7**	89,2	89,1
		5	89,7	88,6	87,8
		7	88,0	86,7	86,2
	6-gram	1	87,1	88,0	89,0
		3	**90,0**	88,7	89,7
		5	88,9	87,1	88,1
		7	87,6	86,4	87,2

Karakter tabanlı en yüksek değerler öznitelik sayısı 120, *k* değeri 3 ve 4-gram ele alındığında ortalama doğruluk değeri %90,7 ve F-ölçüsü %70,7 olarak elde edilmiştir.

Çizelge 4.9. K-NN'ye göre karakter tabanlı ortalama F-ölçüsü değerleri

		k-değerleri	Öznitelik Sayısı		
			120	180	240
Karakter tabanlı	2-gram	1	**50,6**	49,9	49,3
		3	47,8	47,6	47,4
		5	44,6	44,3	44,4
		7	38,7	38,7	39,0
	3-gram	1	**59,9**	58,8	58,8
		3	55,4	57,3	57,3
		5	55,7	56,2	56,2
		7	54,2	56,5	56,5
	4-gram	1	67,4	65,6	67,9
		3	**70,7**	65,6	68,3
		5	68,7	65,6	66,1
		7	64,7	62,8	64,3
	5-gram	1	66,3	65,4	66,5
		3	**67,8**	66,7	65,9
		5	67,7	64,3	61,8
		7	61,7	58,3	57,5
	6-gram	1	61,2	63,7	66,6
		3	**68,5**	63,6	66,7
		5	63,6	57,2	59,8
		7	58,5	55,6	56,6

K-NN metodu ile elde edilen ortalama doğruluk değerleri ve F-ölçüsü değerleri incelendiğinde en yüksek değerlerin öznitelik boyu 120 olduğunda elde edildiği görülmektedir. Bu nedenle, diğer yöntemlerde de öznitelik boyu 120 olarak alınmıştır. ÇKA (Çok Katmanlı Algılayıcı), DVM (Destek Vektör Makinesi) ve LVQ (Learning Vector Quantization) metotlarına ait sonuçlar aşağıda değerlendirilmiştir.

Çizelge 4.10. Diğer metotlara göre ortalama doğruluk değerleri

		ÇKA	DVM	LVQ
Sözcük tabanlı	**1-gram**	**85,7**	**91,4**	**81,4**
	2-gram	84,4	90,8	81,4
	3-gram	77,4	83,4	73,4
Gövde tabanlı	**1-gram**	**87,9**	**91,4**	**83,1**
	2-gram	84,8	91,1	77,4
	3-gram	76,7	91,1	74,0
Hece tabanlı	**1-gram**	**87,2**	89,9	82,7
	2-gram	85,0	89,8	**83,3**
	3-gram	84,2	**91,3**	80,7
Karakter tabanlı	**2-gram**	85,3	85,7	79,6
	3-gram	85,9	87,6	83,2
	4-gram	85,2	88,9	**83,3**
	5-gram	**88,0**	**89,4**	**83,3**
	6-gram	87,6	88,0	82,8

Çizelge 4.11. Diğer metotlara göre ortalama F-ölçüsü değerleri

		ÇKA	DVM	LVQ
Sözcük tabanlı	**1-gram**	**58,0**	**72,8**	**47,2**
	2-gram	52,0	70,8	23,2
	3-gram	30,8	48,9	11,6
Gövde tabanlı	**1-gram**	**65,4**	**72,4**	**53,5**
	2-gram	53,0	71,6	30,1
	3-gram	27,5	71,6	14,8
Hece tabanlı	**1-gram**	**60,2**	68,4	51,7
	2-gram	54,3	66,6	**54,3**
	3-gram	54,3	**73,0**	44,5
Karakter tabanlı	**2-gram**	56,3	56,9	37,0
	3-gram	54,4	59,1	53,8
	4-gram	56,0	65,0	**54,3**
	5-gram	62,1	**65,5**	**54,3**
	6-gram	**63,6**	61,3	52,6

ÇKA metodu kullanılarak elde edilen ortalama doğruluk değerleri Çizelge 4.10'da ve F-ölçüsü sonuçları Çizelge 4.11'de verilmiştir. ÇKA metodu kullanılan sistemler için en iyi ortalama doğruluk değeri karakter tabanlı *n*-gramı 5 için %88 ve F-ölçüsü değeri gövde tabanlı *n*-gramı 1 için %65,4 olarak bulunmuştur. Sözcük tabanlı en yüksek ortalama doğruluk değeri %85,7 ve en yüksek F-ölçüsü değeri %58 olarak 1-gram için tespit edilmiştir. Gövde tabanlı en yüksek ortalama doğruluk değeri %87,9 ile 1-gram'da elde edilmiştir. Hece tabanlı en yüksek sonuçlar her iki ölçüde de 1-gram için bulunmuştur. Bu sonuçların ortalama doğruluk değeri %87,2 ve F-ölçüsü değeri %60,2'dir. Karakter tabanlı en yüksek F-ölçüsü değeri %63,6 ile 6-gram'da elde edilmiştir.

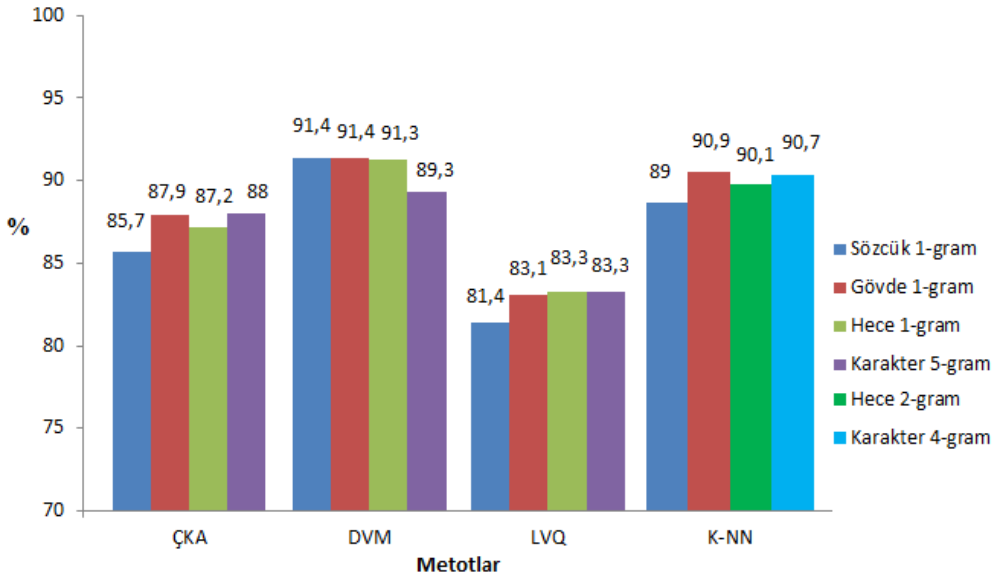
DVM (Destek Vektör Makinesi) metodu ile elde edilen ortalama doğruluk ve F-ölçüsü sonuçları Çizelge 4.10 ve 4.11'de verilmiştir. Sözcük tabanlı ortalama doğruluk değerleri ve F-ölçüsü değerleri içerisinde en yüksek bulgular sırasıyla %91,4 ve %72,8 ile 1-gram için kaydedilmiştir. Gövde tabanlı en yüksek sonuçlar her iki ölçüde de 1-gram'da elde edilmiştir. Ortalama doğruluk değeri %91,4 ve F-ölçüsü değeri %72,4 olarak ölçülmüştür. Hece tabanlı en yüksek sonuçlar ortalama doğruluk değeri %91,3 ile 3-gram'da, F-ölçüsü değeri %73 ile yine 3-gram'da bulunmuştur. Karakter tabanlı en yüksek ortalama doğruluk değeri %89,4 ile en yüksek F-ölçüsü değeri %65,5 ile 5-gram için elde edilmiştir.

LVQ metodu kullandığımızda bulduğumuz ortalama doğruluk değerleri Çizelge 4.10 ve F-ölçüsü değerleri Çizelge 4.11'de verilmiştir. Sözcük tabanlı ortalama doğruluk değerleri ile F-ölçüsü değerlerinde en yüksek sonuçlar her iki ölçü için 1-gramda tespit edilmiştir. Bu değerler sırasıyla %81,4 ve %47,2'dir. Gövde tabanlı en yüksek ortalama doğruluk değeri %83,1 olarak 1-gram'da, en yüksek F-ölçüsü değeri %53,5 olarak yine 1-gram'da elde edilmiştir. Hece tabanlı en yüksek ortalama doğruluk değeri 2-gram ile %83,3 ve en yüksek F-ölçüsü değeri 2-gram ile %54,3 olarak bulunmuştur.Karakter tabanlı en yüksek ortalama doğruluk değeri 4-gram ile 5-gram için eşit ve %83,3, en yüksek F-ölçüsü değeri 4-gram ile 5-gram için eşit ve %54,3 olarak tespit edilmiştir.

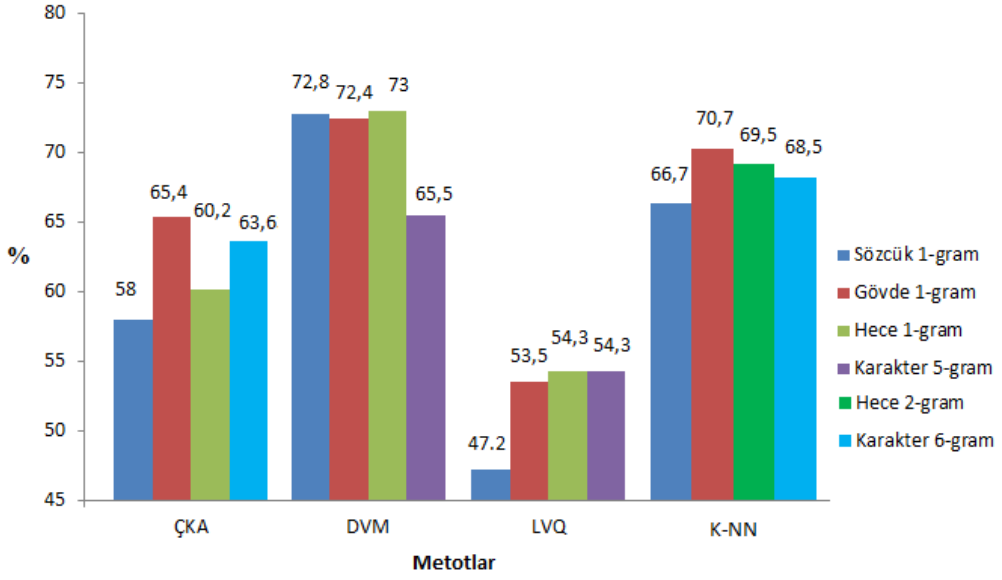
Şekil 4.1 ve 4.2'de metotlara göre ortalama doğruluk ve F-ölçüsü değerlendirme kriterleri için en başarılı değerler verilmektedir. Bu sonuçlara göre DVM metodunun en etkili metot olduğu görülmektedir. Daha sonraki en başarılı metot K-NN metodudur. LVQ metodu ise en başarısız metot olmuştur. Ortalama doğruluk değerlerine göre en başarılı sistem DVM metodu ile oluşturulan sözcük

ve gövde 1-gram tabanlı sistemler olmuştur. Başarını yüzdesi de 91,4 olarak ölçülmüştür.

F-ölçüsü değerlerine göre en başarılı metot, ortalama doğruluk değerlerinde olduğu gibi DVM metodudur. Ardından K-NN metodu gelmektedir. Fakat buradaki en başarılı sistem DVM metodu ile oluşturulan ve başarısı da %73 olan hece 1-gram tabanlı sistem olmuştur.



Şekil 4.1. En iyi ortalama doğruluk değerleri



Şekil 4.2. En iyi F-ölçüsü değerleri

5. TARTIŞMA VE SONUÇLAR

Bu çalışmanın asıl amacı yazar tanıma sistemi için kullanılan yöntemler arasında en iyi metodu ve yaklaşımı belirlemektir. Bunun yanında her metot için en iyi sonuçların hangi *n*-gramda hangi tabanlı yaklaşımda elde edileceği belirlenmeye çalışılmıştır.

Bu çalışmada elde edilen sonuçlara göre yazar tanıma sistemi için en başarılı metot Destek Vektör Makinesi metodudur.K-NN metodu başarı sıralamasında ikinci sıradadır. Çok katmanlı Algılayıcı metodu ise başarı sıralamasında üçüncü sırada yer almaktadır. LVQ metodu bu metotlar arasında en başarısız metot olarak belirlenmiştir.

K-NN metodu için en iyi sonuçlar genel olarak öznitelik sayısı 120 olarak alındığında elde edilmiştir. K-NN metodunda gövde ve hece tabanlı olarak 3-gram ele alındığında en iyi sonuçlar öznitelik sayısı 180 olarak alındığında elde edilmiştir. K-NN metodu için en iyi sonuçlar *k* değeri 3 olduğunda sözcük, hece ve karakter tabanlı, *k* değeri 5 olduğunda gövde tabanlı olarak elde edilmiştir.

Sözcük tabanlı olarak sonuçlar değerlendirildiğinde en iyi sonuçlar sözcük 1-gram ile elde edilmiştir. Gövde tabanlı sistemlerde en iyi sonuçlar, sözcük 1-gram ele alındığında elde edilmiştir. Hece tabanlı olarak değerlendirildiğinde ise en iyi sonuçlar K-NN ve LVQ metodunda 2-gram, ÇKA metodunda 1-gram ve DVM metodunda 3-gram ile elde edilmiştir. Karakter tabanlı sistemlerde en başarılı sonuçlara, K-NN metodunda 4-gram diğer metotlarda ise 5-gram ile ulaşılmıştır.

Bu metotlar kullanılarak daha farklı öznitelik sayıları ve *n*-gramlar kullanılabilir. Yazar tanıma çalışmalarında, sözcüklerin kökleri ve kök *n*-gramları tercih edilip uygulamalar geliştirilebilir.

Bu çalışmadaki özniteliklerin belirlenmesinde kullanılan metot dışında chi-squared distribution (ki-kare dağılımı), mutual information (karşılıklı bilgi), document frequency (doküman frekansı) yaklaşımları kullanılabilir.

KAYNAKLAR

Adsett, C. R., Y. Marchand, V. Keselj. 2009. Syllabification Rules Versus Data-Driven Methods in A Language With Low Syllabic Comlexity: The Case of Italian.**Computer Speech and Language**, 23(4): 444-463.

Akın, A.A., Akın, M.D. 2010. Zemberek an open source NLP framework for Turkish Languages. Technical Report, http://zemberek.gogle.com/files/zemberek_makale.pdf.

Amasyalı M.F., Diri B. 2006.Automatic Written Turkish Text Categorization in Terms of Author, Genre and Gender.**11th International Conference on Applications of Natural Language to Information Systems**.

Amasyalı M.F.,Diri B., Türkoğlu F. 2006.Farklı Özellik Vektörleri ile Türkçe Dokümanların Yazarlarının Belirlenmesi. **Turkish Symposium on Artificial Intelligence and Neural Networks(TAINN)**.

Aşlıyan, R., Günel, K. 2011. A Comparison of Syllabifying Algorithms for Turkish.**Advanced Research in Computer Science**, 3(1): 58-78.

Cebiroğlu, G., Adalı, E. 2002. Root Reaching Method Without Dictionary. Computer Engineering Department, Istanbul Technical University, Istanbul, Turkey.

Diri, B., Amasyalı M.F. 2003.Automatic Author Detection for Turkish Texts.**Artificial Neural Networks and Neural Information Processing**, pp. 138-141.

Diederich, J., Kindermann, J., Leopold, E., Paass, G. 2003. Authorship Attribution with Support Vector Machines.**Applied Intelligence**, 19(1): 109-123.

Fung, G., Mangasarian, O. 2003.The Disputed Federalist Papers: SVM Feature Selection via Concave Minimization.In **Proceedings of the 2003 Conference of Diversity in Computing**, pp. 42-46, Atlanta, Georgia, USA.

Gerritsen, C.M. 2003.Authorship Attribution Using Lexical Attraction. Master Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), USA.

Joachims, T. 1998. Text Categorization with Support VectorMachines:Learning with Many RelevantFeatures. In **Proceedings of the European Conference on Machine Learning ECML)**, pp. 137-142, Berlin.

Köksal, A. 1975. Automatic Morbhological Analysis of Turkish. Hacettepe University, Ankara, Turkey.

Kut, A., Alpkoçak, A., Özkarahan, E. 1995. Bilgi Bulma Sistemleri İçin Otomatik Türkçe Dizinleme Yöntemi. Bilişim Bildirileri, Dokuz Eylül University, İzmir, Turkey.

Marchand, Y., Adsett, C. R., Damper, R. I. 2009. Automatic Syllabification in English: A Comparison of Different Algorithms. **Language and Speech**, 52(5): 1-27.

Oflazer, K. 1999. Dependency Parsing With an Extended Finite State Approach. Department of Computer Engineering, Bilkent University, Ankara, Turkey.

Öztemel, E. 2003. Yapay Sinir Ağı Modeli (Öğretmenli Öğrenme) Çok Katamanlı Algılayıcı, Yapay Sinir Ağları Papatya Yayıncılık, İstanbul.

Peng, F., Schuurmans, D., Keselj, V., Wang, S. 2003.Language Independent Authorship Attribution using Character Level Language Models. **Conf. of the European Chapter of the Association for Computational Linguistics**(**EACL)**, pp. 267-274.

Rumelhart D.E, Hinton G.E, Williams R.J.1986.Learning representations by backpropogation errors.**Nature**, 323:533-536.

Sever, H., Bitirim, Y. 2003. FindStem: analysis and evaluation of a Turkish stemming algorithm. **String Processing and Information Retrieval (SPIRE)**, 10th International Symposium, pp.238-251.

Stamatatos, E. Fakotakis, N., Kokkinakis, G.,1999. Automatic Authorship Attribution. In **Proceedings of the 9th Conf. of the European Chapter of the Association for Computational Linguistics (EACL'99)**, pp. 158-164.

Solak, A., Can, F. 1994. Effects of Stemming On Turkish Text Retrieval. Departman of Computer Engineering and Information Sciences, Bilkent University, Ankara, Turkey.

Solak, A., Oflazer, K. 1993. Design and Implementation of a Spelling Checker for Turkish. Department of Computer Engineering and Information Science, Bilkent University, Ankara, Turkey.

Soucy, P., Mineau, G.W. 2001. A Simple K-NN Algorithm For Text Categorization**.** In **Proceedings of The First IEEE International Conference On Data Mining (ICDM_01)**, pp. 647–648, San Jose, CA.

Ucoluk, G., Toroslu, I. H. 1997. A Genetic Algorithm Approach for Verification of the Syllable Based Text Compression Technique. **Journal of Information Science**, 23(5): 365-372.

Yang, Y., Liu, X. 1999. A re-examination of text categorization methods. School of Computer Science, Carnegie Mellon University, Pittsburgh, USA.

Yılmaz, R. 2013. Türkçe Dokümanların Sınıflandırılması. Yüksek Lisans Tezi, Adnan Menderes Üniversitesi, Aydın.

EKLER

EK 1: Dokümanlardaki Metinleri Hece Haline Getiren Program

```
clear all
clc
dosyaisimleri=dir('Dosyalar\*.txt');
dosyasay=length(dosyaisimleri);
for k1=1:dosyasay
clear dosyaadi1 dosyaadi2
dosyaadi1=['Dosyalar\' dosyaisimleri(k1).name];
dosyaadi2=['Heceler\' dosyaisimleri(k1).name];
fid1=fopen(dosyaadi1,'r');
fid2=fopen(dosyaadi2,'w');
while 1
   clear keli1 kelime
   keli1= fgetl(fid1);
   if ~ischar(keli1), break, end
   keli2= fgetl(fid1);
[kelime,digerkelimeler] = strtok(keli1);
   while (0==isempty(kelime))
     clear hece;
    hece=hecele(kelime);
    for t=1:length(hece)
       if (strcmp(hece{t},'ÿ')==0)
          fprintf(fid2,'%s ',hece{t});
       end
    end
    [kelime,digerkelimeler] = strtok(digerkelimeler);
   end
end
fclose(fid1);
fclose(fid2);
end
```

EK 2: Heceleme Fonksiyonu

```
function sonucheceler=hecele(kelime)
      sesliler = 'AaEeIıİiUuÜüOoÖö';
      metin='';
      hecelermetin='';
      heceler={};
      hecelersay=0;
      hecesay=1;
      v1=0;
      v2=0;
      keliuzun=length(kelime);
      hecelerindis(hecesay)=0;
cik=0;
      for i=1:keliuzun
         for k=1:length(sesliler)
           if (kelime(i)==sesliler(k))
              v1=i;
              cik=1;
              break;
           end
         end
         if cik==1
           break;
         end
      end
         bas = v1+1 ;
          son = keliuzun;
        indis = bas;
        while  (indis <= son)
             sesli=0;
           for k=1:length(sesliler)
             if (kelime(indis)==sesliler(k))
              sesli=1;
              break;
             end
           end
       if (sesli==0)
              indis=indis+1;
               continue;
            else
               v2 = indis;
               sessizsay = v2 - v1 - 1;
               if (sessizsay == 0)
                  hecesay=hecesay+1;
                   hecelerindis(hecesay) = v2-1;
```

```
                    hecelersay=hecelersay+1;
                    heceler{hecelersay} = kelime(1+hecelerindis(hecesay - 1): ...
hecelerindis(hecesay)) ;
                    v1 = v2;
                    else
                    if (sessizsay <= 3)
                       hecesay = hecesay + 1;
                       hecelerindis(hecesay) = v2 - 2;
                       hecelersay=hecelersay+1;
                        heceler{hecelersay} = kelime(1+hecelerindis …
(hecesay -1):hecelerindis(hecesay)) ;
                        v1 = v2;
                       else
                       hecesay = hecesay + 1;
                       hecelerindis(hecesay) = v1 + 2;
                       hecelersay=hecelersay+1;
                       heceler{hecelersay} = kelime(1+hecelerindis …
(hecesay - 1):hecelerindis(hecesay)) ;
                       v1 = v2;
                     end
                end
  end
indis=indis+1;
 end

          hecesay=hecesay+1;
          hecelerindis(hecesay)=keliuzun;
          hecelersay=hecelersay+1;
          heceler{hecelersay} = kelime(1+hecelerindis …
(hecesay - 1):hecelerindis(hecesay)) ;
sonucheceler= heceler;
```

EK 3: Bütün Verileri Tek Dosya Haline Getiren Program

```
clear all
clc
 no=1;
for i=1:71
    fprintf('deniz - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
    if (i==71)
        dosyaadi='ilk20dosya';
        dosya=['deniz\' dosyaadi '.txt'];
    else
        dosyaadi=num2str(i);
        dosya=['deniz\' dosyaadi '.txt'];
    end
    fid=fopen(dosya);
    butunveriler.deniz.dosya.adi{i}=dosyaadi;
    butunveriler.deniz.dosya.toplam{i}=0 ;
 keli= fgetl(fid) ;
     keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
       frekans{j} =str2num(fgetl(fid)) ;
       butunveriler.deniz.dosya.toplam{i}= …
butunveriler.deniz.dosya.toplam{i} + frekans{j} ;
       j=j+1;
    end
    butunveriler.deniz.dosya.sozcukler{i}=sozcuk;
    butunveriler.deniz.dosya.frekanslar{i}=frekans;
    fclose(fid);
end
 no=1;
for i=1:71
    fprintf('ergun - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
    if (i==71)
        dosyaadi='ilk20dosya';
        dosya=['ergun\' dosyaadi '.txt'];
    else
        dosyaadi=num2str(i);
        dosya=['ergun\' dosyaadi '.txt'];
```

```
    end
    fid=fopen(dosya);
    butunveriler.ergun.dosya.adi{i}=dosyaadi;
    butunveriler.ergun.dosya.toplam{i}=0 ;

    keli= fgetl(fid) ;
    keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
       frekans{j} =str2num(fgetl(fid)) ;
       butunveriler.ergun.dosya.toplam{i}= …
butunveriler.ergun.dosya.toplam{i} + frekans{j} ;
       j=j+1;
    end
    butunveriler.ergun.dosya.sozcukler{i}=sozcuk;
    butunveriler.ergun.dosya.frekanslar{i}=frekans;
    fclose(fid);
end
no=1;
for i=1:71
    fprintf('gungor - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
    if (i==71)
        dosyaadi='ilk20dosya';
        dosya=['gungor\' dosyaadi '.txt'];
    else
        dosyaadi=num2str(i);
        dosya=['gungor\' dosyaadi '.txt'];
    end
    fid=fopen(dosya);
    butunveriler.gungor.dosya.adi{i}=dosyaadi;
    butunveriler.gungor.dosya.toplam{i}=0 ;
   keli= fgetl(fid) ;
    keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
```

```
      frekans{j} =str2num(fgetl(fid)) ;
      butunveriler.gungor.dosya.toplam{i}= …
butunveriler.gungor.dosya.toplam{i} + frekans{j} ;
      j=j+1;
    end
    butunveriler.gungor.dosya.sozcukler{i}=sozcuk;
    butunveriler.gungor.dosya.frekanslar{i}=frekans;
   fclose(fid);
 end
no=1;
for i=1:71
    fprintf('mehmet - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
    if (i==71)
       dosyaadi='ilk20dosya';
       dosya=['mehmet\' dosyaadi '.txt'];
    else
       dosyaadi=num2str(i);
       dosya=['mehmet\' dosyaadi '.txt'];
    end
    fid=fopen(dosya);
    butunveriler.mehmet.dosya.adi{i}=dosyaadi;
    butunveriler.mehmet.dosya.toplam{i}=0 ;
   keli= fgetl(fid) ;
    keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
       frekans{j} =str2num(fgetl(fid)) ;
       butunveriler.mehmet.dosya.toplam{i}= …
butunveriler.mehmet.dosya.toplam{i} + frekans{j} ;
      j=j+1;
    end
    butunveriler.mehmet.dosya.sozcukler{i}=sozcuk;
    butunveriler.mehmet.dosya.frekanslar{i}=frekans;
    fclose(fid);
end
no=1;
for i=1:71
    fprintf('mesut - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
    if (i==71)
```

```
        dosyaadi='ilk20dosya';
        dosya=['mesut\' dosyaadi '.txt'];
    else
        dosyaadi=num2str(i);
        dosya=['mesut\' dosyaadi '.txt'];
    end
    fid=fopen(dosya);
    butunveriler.mesut.dosya.adi{i}=dosyaadi;
    butunveriler.mesut.dosya.toplam{i}=0;
    keli= fgetl(fid) ;
    keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
       frekans{j} =str2num(fgetl(fid)) ;
       butunveriler.mesut.dosya.toplam{i}= …
butunveriler.mesut.dosya.toplam{i} + frekans{j} ;
       j=j+1;
    end
    butunveriler.mesut.dosya.sozcukler{i}=sozcuk;
    butunveriler.mesut.dosya.frekanslar{i}=frekans;
    fclose(fid);
end
no=1;
for i=1:71
    fprintf('ridvan - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
    if (i==71)
        dosyaadi='ilk20dosya';
        dosya=['ridvan\' dosyaadi '.txt'];
    else
        dosyaadi=num2str(i);
        dosya=['ridvan\' dosyaadi '.txt'];
    end
    fid=fopen(dosya);
    butunveriler.ridvan.dosya.adi{i}=dosyaadi;
    butunveriler.ridvan.dosya.toplam{i}=0 ;

    keli= fgetl(fid) ;
    keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
```

```
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
       frekans{j} =str2num(fgetl(fid)) ;
       butunveriler.ridvan.dosya.toplam{i}=butunveriler.ridvan.dosya.toplam{i} +
frekans{j} ;
       j=j+1;
    end
    butunveriler.ridvan.dosya.sozcukler{i}=sozcuk;
    butunveriler.ridvan.dosya.frekanslar{i}=frekans;
    fclose(fid);
end
no=1;
for i=1:1
    fprintf('Hepsi - Dosya No: %d \n',i);
    clear dosya dosyaadi ;
          dosyaadi='hepsi_ilk20toplam';
       dosya=['hepsi_ilk20toplam.txt'];
     fid=fopen(dosya);
    butunveriler.hepsi.dosya.adi{i}=dosyaadi;
    butunveriler.hepsi.dosya.toplam{i}=0 ;
    keli= fgetl(fid) ;
    keli= fgetl(fid) ;
    clear sozcuk frekans;
    j=1 ;
    while 1
      clear keli
       keli= fgetl(fid) ;
       if ~ischar(keli), break, end
       sozcuk{j}=keli ;
       frekans{j} =str2num(fgetl(fid)) ;
       butunveriler.hepsi.dosya.toplam{i}=butunveriler.hepsi.dosya.toplam{i} +
frekans{j} ;
       j=j+1;
    end
    butunveriler.hepsi.dosya.sozcukler{i}=sozcuk;
    butunveriler.hepsi.dosya.frekanslar{i}=frekans;
     fclose(fid);
 end
 save butunveriler.mat butunveriler
```

EK 4: Öznitelikleri Belirleyen Program

```
clear all
clc
 load butunveriler.mat;
 oznitelik sayisi=999;
 for i=1:oznitelik sayisi
 oznitelik_kelime{i,1}=butunveriler.hepsi.dosya.sozcukler{1}{i};
 oznitelik_inputfrekans(i,1)=0;
 end
for i=1:oznitelik sayisi
 deniz_oznitelik_inputfrekans(i,1)=0;
 ergun_oznitelik_inputfrekans(i,1)=0;
 gungor_oznitelik_inputfrekans(i,1)=0;
 mehmet_oznitelik_inputfrekans(i,1)=0;
 mesut_oznitelik_inputfrekans(i,1)=0;
 ridvan_oznitelik_inputfrekans(i,1)=0;
 end
 k=71;
 for i=1:oznitelik sayisi
     for j=1:length(butunveriler.deniz.dosya.sozcukler{k})
       if strcmp(oznitelik_kelime{i,1}, butunveriler.deniz.dosya.sozcukler{k}{j})
          deniz_oznitelik_inputfrekans(i,1)=
butunveriler.deniz.dosya.frekanslar{k}{j}/butunveriler.deniz.dosya.toplam{k};
          break;
       end
         end
 end
 k=71;
  for i=1:oznitelik sayisi
    for j=1:length(butunveriler.ergun.dosya.sozcukler{k})
      if strcmp(oznitelik_kelime{i,1}, butunveriler.ergun.dosya.sozcukler{k}{j})
         ergun_oznitelik_inputfrekans(i,1)=
butunveriler.ergun.dosya.frekanslar{k}{j}/butunveriler.ergun.dosya.toplam{k};
          break;
       end
    end
 end
 k=71;
 for i=1:oznitelik sayisi
         for j=1:length(butunveriler.gungor.dosya.sozcukler{k})

       if strcmp(oznitelik_kelime{i,1}, butunveriler.gungor.dosya.sozcukler{k}{j})
         gungor_oznitelik_inputfrekans(i,1)=
butunveriler.gungor.dosya.frekanslar{k}{j}/butunveriler.gungor.dosya.toplam{k};
```

```
          break;
       end
         end
 end
 k=71; for i=1:oznitelikSayisi
         for j=1:length(butunveriler.mehmet.dosya.sozcukler{k})
           if strcmp(oznitelik_kelime{i,1},
butunveriler.mehmet.dosya.sozcukler{k}{j})
          mehmet_oznitelik_inputfrekans(i,1)=
butunveriler.mehmet.dosya.frekanslar{k}{j}/butunveriler.mehmet.dosya.toplam{k
};
          break;
       end

     end
 end
k=71; for i=1:oznitelikSayisi
         for j=1:length(butunveriler.mesut.dosya.sozcukler{k})
           if strcmp(oznitelik_kelime{i,1},
butunveriler.mesut.dosya.sozcukler{k}{j})
         mesut_oznitelik_inputfrekans(i,1)=
butunveriler.mesut.dosya.frekanslar{k}{j}/butunveriler.mesut.dosya.toplam{k};
          break;
       end

     end
 end
  k=71;
 for i=1:oznitelikSayisi
      for j=1:length(butunveriler.ridvan.dosya.sozcukler{k})
       if strcmp(oznitelik_kelime{i,1}, butunveriler.ridvan.dosya.sozcukler{k}{j})
          ridvan_oznitelik_inputfrekans(i,1)=
butunveriler.ridvan.dosya.frekanslar{k}{j}/butunveriler.ridvan.dosya.toplam{k};
          break;
       end
      end
 end
 P=[deniz_oznitelik_inputfrekans(:,1) ergun_oznitelik_inputfrekans(:,1)
gungor_oznitelik_inputfrekans(:,1) mehmet_oznitelik_inputfrekans(:,1)
mesut_oznitelik_inputfrekans(:,1)   ridvan_oznitelik_inputfrekans(:,1)];
max_oznitelik_boyu=20;
theshold=0.50;
kelimesay=0;
kactane=0;
for i=1:length(P(:,1))
```

```
   sabit=P(i,1)*theshold;
   if ((sabit > P(i,2)) && (sabit > P(i,3)) && (sabit > P(i,4)) && (sabit > P(i,5))
&& (sabit > P(i,6)))
       kelimesay=kelimesay+1;
       Yeni_oznitelik_kelime{kelimesay,1}=oznitelik_kelime{i} ;
       YeniP(kelimesay,:)=P(i,:);
       kactane=kactane+1;
       if (kactane==max_oznitelik_boyu) , break , end
   end
end
oznitelik sozcukler.deniz.bas=1;
oznitelik sozcukler.deniz.son=kactane;
kactane=0;
for i=1:length(P(:,2))
    sabit=P(i,2)*theshold;
   if ((sabit > P(i,1)) && (sabit > P(i,3)) && (sabit > P(i,4)) && (sabit > P(i,5))
&& (sabit > P(i,6)) )
       kelimesay=kelimesay+1;
       Yeni_oznitelik_kelime{kelimesay,1}=oznitelik_kelime{i} ;
       YeniP(kelimesay,:)=P(i,:);
      kactane=kactane+1;
      if (kactane==max_oznitelik_boyu) , break , end
   end
end
öznitelik sozcukler.ergun.bas=oznitelik sozcukler.deniz.son+1;
oznitelik sozcukler.ergun.son=oznitelik sozcukler.ergun.bas+kactane-1;
kactane=0;
for i=1:length(P(:,3))
     sabit=P(i,3)*theshold;
   if ((sabit > P(i,1)) && (sabit > P(i,2)) && (sabit > P(i,4)) && (sabit > P(i,5))
&& (sabit > P(i,6)) )
        kelimesay=kelimesay+1;
       Yeni_oznitelik_kelime{kelimesay,1}=oznitelik_kelime{i} ;
       YeniP(kelimesay,:)=P(i,:);
        kactane=kactane+1;
        if (kactane==max_oznitelik_boyu) , break , end
     end
end
oznitelik sozcukler.gungor.bas=oznitelik sozcukler.ergun.son+1;
oznitelik sozcukler.gungor.son=oznitelik sozcukler.gungor.bas+kactane-1;
kactane=0;
for i=1:length(P(:,1))
sabit=P(i,4)*theshold;
   if ((sabit > P(i,1)) && (sabit > P(i,2)) && (sabit > P(i,3)) && (sabit > P(i,5))
&& (sabit > P(i,6)))
          kelimesay=kelimesay+1;
```

```
        Yeni_oznitelik_kelime{kelimesay,1}=oznitelik_kelime{i} ;
        YeniP(kelimesay,:)=P(i,:);
      kactane=kactane+1;
      if (kactane==max_oznitelik_boyu) , break , end
    end
end
öznitelikSozcukler.mehmet.bas=oznitelikSozcukler.gungor.son+1;
oznitelikSozcukler.mehmet.son=oznitelikSozcukler.mehmet.bas+kactane-1;
 kactane=0;
for i=1:length(P(:,1))
    sabit=P(i,5)*theshold;
 if ((sabit > P(i,1)) && (sabit > P(i,2)) && (sabit > P(i,3)) && (sabit > P(i,4)) &&
(sabit > P(i,6)) )
        kelimesay=kelimesay+1;
        Yeni_oznitelik_kelime{kelimesay,1}=oznitelik_kelime{i} ;
        YeniP(kelimesay,:)=P(i,:);
       kactane=kactane+1;
       if (kactane==max_oznitelik_boyu) , break , end
    end
end
öznitelikSozcukler.mesut.bas=oznitelikSozcukler.mehmet.son+1;
oznitelikSozcukler.mesut.son=oznitelikSozcukler.mesut.bas+kactane-1;
 kactane=0;
for i=1:length(P(:,1))
    sabit=P(i,6)*theshold;
 if ((sabit > P(i,1)) && (sabit > P(i,2)) && (sabit > P(i,3)) && (sabit > P(i,4)) &&
(sabit > P(i,5)) )
        kelimesay=kelimesay+1;
        Yeni_oznitelik_kelime{kelimesay,1}=oznitelik_kelime{i} ;
        YeniP(kelimesay,:)=P(i,:);
       kactane=kactane+1;
       if (kactane==max_oznitelik_boyu) , break , end
    end
end
öznitelikSozcukler.ridvan.bas=oznitelikSozcukler.mesut.son+1;
oznitelikSozcukler.ridvan.son=oznitelikSozcukler.ridvan.bas+kactane-1;
 oznitelikSozcukler.sozcukler=Yeni_oznitelik_kelime;
save oznitelikSozcukler.mat oznitelikSozcukler;
```

EK 5: LVQ Metodu İçin Eğitim Programı

```
clear all
clc
load oznitelikleriksozcukler.mat;
load butunveriler.mat;
 vektor_sayisi=50;
 topoloji =[vektor_sayisi vektor_sayisi vektor_sayisi vektor_sayisi vektor_sayisi
vektor_sayisi];
 Ogrenme_Orani = 0.5;
 Monoton_Azalma_Yuzdesi = 0.001;
 Hedef = 1;
 Maksimum_Iterasyon = 10;
  oznitelikleriksayisi=length(oznitelikleriksozcukler.sozcukler);
  for i=1:oznitelikleriksayisi
 oznitelik_kelime{i,1}=oznitelikleriksozcukler.sozcukler{i};
 oznitelik_inputfrekans(i,1)=0;
 end
  for k=1:20
 for i=1:oznitelikleriksayisi
 deniz_oznitelik_inputfrekans(i,k)=0;
 ergun_oznitelik_inputfrekans(i,k)=0;
 gungor_oznitelik_inputfrekans(i,k)=0;
 mehmet_oznitelik_inputfrekans(i,k)=0;
 mesut_oznitelik_inputfrekans(i,k)=0;
 ridvan_oznitelik_inputfrekans(i,k)=0;
 end
 end
   for k=1:20
 for i=1:oznitelikleriksayisi
      for j=1:length(butunveriler.deniz.dosya.sozcukler{k})
        if strcmp(oznitelik_kelime{i,1}, butunveriler.deniz.dosya.sozcukler{k}{j})
           deniz_oznitelik_inputfrekans(i,k)=
butunveriler.deniz.dosya.frekanslar{k}{j};
           break;
       end
      end
 end
end
  for k=1:20
 for i=1:oznitelikleriksayisi
     for j=1:length(butunveriler.ergun.dosya.sozcukler{k})
       if strcmp(oznitelik_kelime{i,1}, butunveriler.ergun.dosya.sozcukler{k}{j})
           ergun_oznitelik_inputfrekans(i,k)=
butunveriler.ergun.dosya.frekanslar{k}{j};
           break;
```

```
      end
    end
 end
end
for k=1:20
 for i=1:oznitelikSayisi
    for j=1:length(butunveriler.gungor.dosya.sozcukler{k})
      if strcmp(oznitelik_kelime{i,1}, butunveriler.gungor.dosya.sozcukler{k}{j})
         gungor_oznitelik_inputfrekans(i,k)=
butunveriler.gungor.dosya.frekanslar{k}{j};
         break;
     end
    end
 end
end
for k=1:20
 for i=1:oznitelikSayisi
     for j=1:length(butunveriler.mehmet.dosya.sozcukler{k})
       if strcmp(oznitelik_kelime{i,1},
butunveriler.mehmet.dosya.sozcukler{k}{j})
         mehmet_oznitelik_inputfrekans(i,k)=
butunveriler.mehmet.dosya.frekanslar{k}{j};
         break;
      end
     end
 end
end
  for k=1:20
 for i=1:oznitelikSayisi
    for j=1:length(butunveriler.mesut.dosya.sozcukler{k})
      if strcmp(oznitelik_kelime{i,1}, butunveriler. mesut.dosya.sozcukler{k}{j})
         mesut_oznitelik_inputfrekans(i,k)=
butunveriler.mesut.dosya.frekanslar{k}{j};
         break;
      end
    end
 end
 end
 for k=1:20
 for i=1:oznitelikSayisi
    for j=1:length(butunveriler.ridvan.dosya.sozcukler{k})
      if strcmp(oznitelik_kelime{i,1}, butunveriler. ridvan.dosya.sozcukler{k}{j})
         ridvan_oznitelik_inputfrekans(i,k)=
butunveriler.ridvan.dosya.frekanslar{k}{j};
         break;
      end
```

```
    end
 end
 end
YeniP=[deniz_oznitelik_inputfrekans(:,1:20) ergun_oznitelik_inputfrekans(:,1:20)
gungor_oznitelik_inputfrekans(:,1:20) mehmet_oznitelik_inputfrekans(:,1:20)
mesut_oznitelik_inputfrekans(:,1:20)  ridvan_oznitelik_inputfrekans(:,1:20)];
T(1,1:20)=1;
T(1,21:40)=2;
T(1,41:60)=3;
T(1,61:80)=4;
T(1,81:100)=5;
T(1,101:120)=6;
 X=YeniP;
Y=T;
for i=1:sum(topoloji)
   rng('shuffle');
   Agirliklar{i}= 0 + (1-(0)).*rand(size(X,1),1);
end
 disp('LVQ_X');
NET_LVQ_X=LVQ_X(X,Y,topoloji,Ogrenme_Orani,Monoton_Azalma_Yuzdesi
,Hedef,Maksimum_Iterasyon,Agirliklar);
```

EK 6: LVQ Fonksiyonu

```
function
NET=LVQ_X(P,T,topoloji,Ogrenme_Orani,Monoton_Azalma_Yuzdesi,Hedef,Ma
ksimum_Iterasyon,Agirliklar)
format long
NET.P=P;
 NET.T=T;
 NET.LVQ_Topoloji= [50 50 50 50 50 50];
 NET.Ogrenme_Orani=Ogrenme_Orani;
 NET.monoton_azalma_yuzdesi=Monoton_Azalma_Yuzdesi;
 NET.hedef=Hedef;
 max_iter=Maksimum_Iterasyon;
 NET.performans=0;
 NET.toplam_proseseleman=sum(NET.LVQ_Topoloji);
NET.Kohonen_Vektorleri = Agirliklar;
 iterasyon=1;
while ((NET.performans<NET.hedef) && (iterasyon<=max_iter))
 for i=1:length(NET.T)
input=NET.P(:,i);
 for j=1:NET.toplam_proseseleman
    uzakliklar(j)=sum((input-NET.Kohonen_Vektorleri{j}).^2).^0.5;
end
[K1 GlobalKazananProsesElemani]=min(uzakliklar);
    ToplamProsesElemani=0;
   for sinif=1:length(NET.LVQ_Topoloji)
   ToplamProsesElemani=ToplamProsesElemani+NET.LVQ_Topoloji(sinif);
    if (GlobalKazananProsesElemani<=ToplamProsesElemani)
      KazananSinif1=sinif;
      break;
    end
   end
    Toplam=0;
   for sinif=1:length(NET.LVQ_Topoloji)
       SinifVektorBas(sinif)=Toplam+1;
       Toplam=Toplam+NET.LVQ_Topoloji(sinif);
       SinifVektorSon(sinif)=Toplam;
   end
    yereluzakliklar(1:NET.toplam_proseseleman)=inf;
    for uz=SinifVektorBas(NET.T(i)):SinifVektorSon(NET.T(i))
       yereluzakliklar(uz)=sum((input-NET.Kohonen_Vektorleri{uz}).^2).^0.5;
   end
[K2 YerelKazananProsesElemani]=min(yereluzakliklar);
if (GlobalKazananProsesElemani==YerelKazananProsesElemani)
```

```
NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=NET.Kohonen_Vekto
rleri{GlobalKazananProsesElemani}+NET.Ogrenme_Orani*(input-
NET.Kohonen_Vektorleri{GlobalKazananProsesElemani});
        else
NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=NET.Kohonen_Vekto
rleri{GlobalKazananProsesElemani}-NET.Ogrenme_Orani*(input-
NET.Kohonen_Vektorleri{GlobalKazananProsesElemani});
NET.Kohonen_Vektorleri{YerelKazananProsesElemani}=NET.Kohonen_Vektorl
eri{YerelKazananProsesElemani}+NET.Ogrenme_Orani*(input-
NET.Kohonen_Vektorleri{YerelKazananProsesElemani});
   end
    ciktilar=simulation(NET,NET.P);
  t=0;
  for ind=1:size(NET.T,2)
     t=t+isequal(NET.T(ind),ciktilar(ind));
  end
   if((t/size(NET.T,2))>=NET.performans)
     clear NET_LVQ_X;
    NET_LVQ_X=NET;
  end
   NET.performans=t/size(NET.T,2);
 end
     fprintf('İterasyon: %d   Performans: %2.2f\n',iterasyon,NET.performans);
iterasyon=iterasyon+1;
NET.Ogrenme_Orani=NET.Ogrenme_Orani-
NET.monoton_azalma_yuzdesi*NET.Ogrenme_Orani;
 end
 save NET_LVQ_X.mat NET_LVQ_X;
 simulation(NET,NET.P)
```

EK 7: LVQ Metodu İçin Test Programı

```
clear all
clc
 load oznitelikSozcukler.mat;
load butunveriler.mat;
 sinif_isimleri={'deniz','ergun','gungor','mehmet','mesut' ,'ridvan'};
 oznitelikSayisi=length(oznitelikSozcukler.sozcukler);
oznitelik_kelime=oznitelikSozcukler.sozcukler;
load NET_LVQ_X.mat;
  for i=1:oznitelikSayisi
 oznitelik_kelime{i,1}=oznitelikSozcukler.sozcukler{i};
 oznitelik_inputfrekans(i,1)=0;
 end
  for k=21:70
 for i=1:oznitelikSayisi
 deniz_oznitelik_inputfrekans(i,k)=0;
 ergun_oznitelik_inputfrekans(i,k)=0;
 gungor_oznitelik_inputfrekans(i,k)=0;
 mehmet_oznitelik_inputfrekans(i,k)=0;
 mesut_oznitelik_inputfrekans(i,k)=0;
 ridvan_oznitelik_inputfrekans(i,k)=0;
 end
 end
  for k=1:20
 for i=1:oznitelikSayisi
      for j=1:length(butunveriler.deniz.dosya.sozcukler{k})
       if strcmp(oznitelik_kelime{i,1}, butunveriler.deniz.dosya.sozcukler{k}{j})
          deniz_oznitelik_inputfrekans(i,k)=
butunveriler.deniz.dosya.frekanslar{k}{j};
          break;
       end
     end
 end
end
  for k=1:20
 for i=1:oznitelikSayisi
      for j=1:length(butunveriler.ergun.dosya.sozcukler{k})
       if strcmp(oznitelik_kelime{i,1}, butunveriler.ergun.dosya.sozcukler{k}{j})
          ergun_oznitelik_inputfrekans(i,k)=
butunveriler.ergun.dosya.frekanslar{k}{j};
          break;
       end
     end
 end
```

```
end
for k=1:20
 for i=1:oznitelikSayisi
   for j=1:length(butunveriler.gungor.dosya.sozcukler{k})
    if strcmp(oznitelik_kelime{i,1}, butunveriler.gungor.dosya.sozcukler{k}{j})
        gungor_oznitelik_inputfrekans(i,k)=
butunveriler.gungor.dosya.frekanslar{k}{j};
        break;
     end
   end
 end
end
for k=1:20
 for i=1:oznitelikSayisi
    for j=1:length(butunveriler.mehmet.dosya.sozcukler{k})
      if strcmp(oznitelik_kelime{i,1}, butunveriler.mehmet.dosya.sozcukler{k}{j})
        mehmet_oznitelik_inputfrekans(i,k)=
butunveriler.mehmet.dosya.frekanslar{k}{j};
        break;
     end
    end
 end
end
  for k=1:20
 for i=1:oznitelikSayisi
    for j=1:length(butunveriler.mesut.dosya.sozcukler{k})
     if strcmp(oznitelik_kelime{i,1}, butunveriler. mesut.dosya.sozcukler{k}{j})
       mesut_oznitelik_inputfrekans(i,k)=
butunveriler.mesut.dosya.frekanslar{k}{j};
        break;
     end
    end
 end
 end
 for k=1:20
 for i=1:oznitelikSayisi
    for j=1:length(butunveriler.ridvan.dosya.sozcukler{k})
     if strcmp(oznitelik_kelime{i,1}, butunveriler. ridvan.dosya.sozcukler{k}{j})
        ridvan_oznitelik_inputfrekans(i,k)=
butunveriler.ridvan.dosya.frekanslar{k}{j};
        break;
     end
   end
 end
 end
```

```
P=[deniz_oznitelik_inputfrekans(:,1:50) ergun_oznitelik_inputfrekans(:,1:50)
gungor_oznitelik_inputfrekans(:,1:50) mehmet_oznitelik_inputfrekans(:,1:50)
mesut_oznitelik_inputfrekans(:,1:50)  ridvan_oznitelik_inputfrekans(:,1:50)];
sonuc1=simulation(NET_LVQ_X,P(:,1:50));
sonuc2=simulation(NET_LVQ_X,P(:,51:100));
sonuc3=simulation(NET_LVQ_X,P(:,101:150));
sonuc4=simulation(NET_LVQ_X,P(:,151:200));
sonuc5=simulation(NET_LVQ_X,P(:,201:250));
sonuc6=simulation(NET_LVQ_X,P(:,251:300));
 ToplamSonuc=[sonuc1;sonuc2;sonuc3;sonuc4;sonuc5;sonuc6];
sinif_isimleri={'1','2','3','4','5','6'};
disp('NET_LVQ_X')
Evaluate(ToplamSonuc,sinif_isimleri);
```

EK 8: Değerlendirme (Evaluate) Fonksiyonu

```
function Evaluate(sonucmatris,sinif_isimleri)
 sinifsayisi=size(sonucmatris,1);
  sinifverisayisi =size(sonucmatris,2);
for i=1:sinifsayisi
      TP(i)=sum(sonucmatris(i,:)==i);
      FN(i)= sinifverisayisi-TP(i);
      FP(i)=sum(sum(sonucmatris==i))-TP(i);
      TN(i)=(sinifsayisi*sinifverisayisi-sinifverisayisi) -FP(i);
     if  TP(i)==0
       Precision(i)=0;
     else
      Precision(i)=TP(i)/(TP(i)+FP(i));
     end
      if  TP(i)==0
       Recall(i)=0;
     else
       Recall(i)=TP(i)/(TP(i)+FN(i));
     end
       if  TN(i)==0
       TrueNegativeRate(i)=0;
     else
       TrueNegativeRate(i)=TN(i)/(TN(i)+FP(i)); %
     end
       if  ((TP(i)==0)  && (TN(i)==0))
       Accuracy(i)=0;
     else
       Accuracy(i)=(TP(i)+TN(i))/(TP(i)+TN(i)+FP(i)+FN(i));
      end
     if  ((Precision(i)==0)  || (Recall(i)==0))
       FMeasure(i)=0;
     else
       FMeasure(i)=(2*Precision(i)*Recall(i))/(Precision(i)+Recall(i));
     end
end
toplamAccuracy=0;
toplamFmeasure=0;
toplamPrecision=0;
toplamRecall=0;
toplamTrueNegativeRate=0;
 for i=1:sinifsayisi
toplamPrecision=toplamPrecision+Precision(i);
toplamRecall=toplamRecall+Recall(i);
toplamTrueNegativeRate=toplamTrueNegativeRate+TrueNegativeRate(i);
```

```
toplamAccuracy=toplamAccuracy+Accuracy(i);
toplamFmeasure=toplamFmeasure+FMeasure(i);
end
fprintf('Mean Precision: %4.1f percent\n',100*(toplamPrecision/i));
fprintf('Mean Recall: %4.1f percent\n',100*(toplamRecall/i));
fprintf('Mean TrueNegativeRate: %4.1f
percent\n',100*(toplamTrueNegativeRate/i));
fprintf('Mean Accuracy: %4.1f percent\n',100*(toplamAccuracy/i));
fprintf('Mean FMeasure: %4.1f percent\n\n',100*(toplamFmeasure/i));

end
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : Özcan KOLYİĞİT
Doğum Yeri ve Tarihi : KARAMAN -16.09.1987

EĞİTİM DURUMU

Lisans Öğrenimi : Adnan Menderes Üniversitesi, Fen Edebiyat Fakültesi-Matematik Bölümü
Yüksek Lisans Öğrenimi : Adnan Menderes Üniversitesi-Matematik A.B.D.
Bildiği Yabancı Diller : İngilizce

BİLİMSEL FAALİYETLERİ

Bildiriler : Kolyiğit Ö., Aşlıyan R. ve Günel K. 2012. Türkçe Dokümanlarda Yazar Tanıma, Akademik Bilişim 2012.
Katıldığı Projeler : Aşlıyan R., Günel K.,Yılmaz R., Kolyiğit Ö., Yıldırım Ö., Web Sayfalarının Sınıflandırılması, Bilimsel Araştırma Projeleri Kurulu, ADU, FEF11007, Aydın, Türkiye, 29.09.2012.

İLETİŞİM

E-posta Adresi : okolyigit@gmail.com