KENDİNDEN DÜZENLENEN HARİTALAR İLE DOKÜMAN SINIFLANDIRMA

Yılmaz ALPDOĞAN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAYIS 2007
ANKARA

Yılmaz ALPDOĞAN tarafından hazırlanan KENDİNDEN DÜZENLENEN HARİTALAR İLE DOKÜMAN SINIFLANDIRMA adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Yrd.Doç.Dr. Hasan Şakir BİLGE
Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans tezi olarak kabul edilmiştir.

Başkan: : Prof.Dr. M.Cengiz TAPLAMACIOĞLU

Üye : Doç.Dr. Şeref SAĞIROĞLU

Üye : Yrd.Doç.Dr. Hasan Şakir BİLGE

Üye :

Üye :

Tarih : 02/05/2007

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Yılmaz ALPDOĞAN

KENDİNDEN DÜZENLENEN HARİTALAR İLE
DOKÜMAN SINIFLANDIRMA
(Yüksek Lisans Tezi)

Yılmaz ALPDOĞAN

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Mayıs 2007

ÖZET

İnternet üzerinde web sayfalarının sayısı, büyük bir hızla artmaktadır. Artık otomatik arama motorları, arama sorgularına isabetli cevaplar vermekte yetersiz kalmaktadırlar. Dizin siteleri, bütün web sayfalarını değerlendirmeye yetişememektedir, dolayısıyla dizinlerin kalitesi ve kapsamı azalmaktadır. Ayrıca, bağlantılar güncelliğini kaybetmektedir. Öte yandan, bilgisayarlarda saklanan dokümanların sayısı ve hiyerarşisi de artmaktadır. Sonuç olarak web sayfalarının ve dokümanların otomatik olarak sınıflandırılması daha fazla önem kazanmaktadır.

Bu çalışmanın amacı, dokümanları içeriklerine göre otomatik olarak sınıflandırmaktır. Bu amaçla, özellikle yüksek boyutlu verilerde başarılı olan ve danışmansız öğrenme özelliğine sahip Kendinden Düzenlenen Haritalar (SOM) algoritması kullanılarak bir sınıflandırma sistemi geliştirilmiştir. Kendinden düzenlenen haritalar algoritması ile elde edilen sonuçlar etkin bir sınıflandırma yöntemi olan hiyerarşik sınıflandırma ile karşılaştırılmıştır. Her iki algoritmada da dokümanı ayırt edici kelimelerin ön plana çıkarılması için uygun bir etiketleme yöntemi uygulanmıştır.

Sınıflandırma işleminden önce dokümanlardaki durak kelimelerinin temizlenmesi, çok ve az tekrar eden kelimelerin temizlenmesi, kelimelerin indekslenmesi, ağırlık vektörlerinin bulunması, ağırlık vektörlerinin aynı boyuta getirilmesi, normalizasyon işlemleri yapılmıştır.

Deneysel çalışmalarda 2 farklı doküman kütüphanesi ele alınmıştır. İlk çalışmada bir İnternet haber sitesinden rastgele alınmış haber içerikleri sınıflandırılırken, ikinci çalışmada ise üniversitelerin web sayfalarından alınan ders içerikleri başarılı bir şekilde sınıflandırılmıştır. Geliştirilen sistemin farklı içeriklere sahip dokümanlarda da başarılı olarak çalışması beklenmektedir.

Bilim Kodu : **902.1.014**
Anahtar Kelimeler : **Doküman sınıflandırma, istatistiksel öğrenme, danışmansız yapay sinir ağları, hiyerarşik sınıflandırma, kendinden düzenlenen haritalar, veri madenciliği**
Sayfa Adedi : **82**
Tez Yöneticisi : **Yrd. Doç. Dr. Hasan Şakir BİLGE**

DOCUMENT CLASSIFICATION WITH
SELF-ORGANIZING MAPS
(M.Sc. Thesis)

Yılmaz ALPDOĞAN

GAZİ UNIVERSITY
INSTITUTE OF SCIENCE AND TECHNOLOGY
May 2007

ABSTRACT

The number of internet web pages are growing at a high rate. Automated search engines are becoming insuffienct in returning appropiate results to the search queries. The directory sites can't keep up with evaluation of all web pages, therefore the quality and scope of their directories are decreased. Furthermore, links are becoming out of date. On the other hand, the number of the documents saved in computers are increasing. As a result, automatic classification of the web pages and documents takes more attention.

In this study, it is aimed to classify the documents according to their contents. For this purpose, a classification system is developed that is based on the Self-Organizing Map (SOM) algorithm, which is an effective unsupervised artificial neural network method for high-dimensional data. The results obtained from self-organizing maps are compared with hierarchical classification, an effective classification method. For both methods, the significant and distinctive words within each document are found by using a labeling algorithm.

Before the classification process, some preprocessing steps are applied, these are stopword removal, removing very low and very high frequently used words,

indexing words, calculating weight vectors, equalizing the dimension of the weight vectors, and normalization.

In experimental studies, two different document libraries are being used. The first library is prepared by collecting random news abstracts from an online news site and the second library is prepared by gathering different course contents from web pages of different universities. Both of libraries are being successfully classified. Furthermore, documents with different contents can also be classified by using this developed system.

Science Code : 902.1.014
Key Words : Document classification, statistical learning, unsupervised neural network, hierarchical classification, Self-Organizing Map (SOM), data mining
Page Number: 82
Adviser : Assist. Prof. Dr. Hasan Şakir BİLGE

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren Hocam Yrd.Doç.Dr. Hasan Şakir BİLGE'ye ve manevi destekleriyle beni hiçbir zaman yalnız bırakmayan çok değerli eşime teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

KISALTMALAR

Bu çalışmada kullanılmış bazı kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltmalar	Açıklama
SOM	Self-Organizing Map
YSA	Yapay Sinir Ağları
DMOZ	The Open Directory Project
WWW	World Wide Web
SVD	Singular-Value Decomposition
SVM	Support Vector Machine
URL	Uniform Resource Locator
TF-IDF	Term Frequency–Inverse Document Frequency
SSOM	Scalable Self-Organizing Map

1. GİRİŞ

İçinde bulunduğumuz bilgi çağında, içerik patlaması da diyebileceğimiz bir durumla karşı karşıya gelmiş bulunuyoruz. Hızla büyüyen bilgi hacmi nedeniyle bilim ve teknoloji alanlarında yeni çalışma alanları ortaya çıkmıştır. Yaşadığımız dünyada çok fazla miktarda metin, ses ve görüntü üretilmektedir. Bütün bu verilerin sayısı çok olduğu için etkin kullanımı sağlanamamaktadır. Bu verilerin etkin bir şekilde kullanılabilmesi için çeşitli çalışmalar yapılmaktadır. Bununla birlikte daha birçok çalışmaya ihtiyaç duyulmaktadır. Üretilen bu kadar çok verinin günlük hayatta yaygın bir şekilde etkin olarak kullanılabilmesi için uzun zamana ihtiyaç olduğu tahmin edilmektedir. İnternet ortamında bulunan içeriklerin büyüme hızı bunların arasından işe yarayacak veriler bulma hızından çok daha fazladır. İnternet kullanıcılarının artan içeriğin arasından aradıklarını bulması gittikçe güçleşmektedir.

İnternet üzerinde bulunan içeriği aramak için arama motoru denilen mekanizmalar kullanılır [1]. Bunlar üç bileşenden oluşur: web robotu, arama indeksi ve kullanıcı arabirimi. Web robotu internet üzerinde bulunan web sitelerini, sitelerin birbirlerine verdiği bağlantıları kullanarak otomatik olarak gezer ve bulduğu sayfa içeriklerini depolar. Bu içerik hızlı bir şekilde aranabilmek üzere daha sonra indekslenir. Kullanıcı arabirimi ise bu oluşturulan indeksin kullanıcılar tarafından kolayca aranmasını sağlar.

Büyük arama motorları, klasik ilişkisel veritabanları yerine ters indeks yapısını kullanırlar. Normal indeks yapısında bir dokümanın içindeki kelimeler döküman anahtarı ile indekslenir; bir dokümanı verip içindeki kelimelere ulaşılır. Ters indeks yapısında ise, dokümanlar kelime anahtarı ile indekslenir; böylece bir kelime verilip bunun geçtiği dokümanlar bulunur [1].

Arama motorları sadece belirli kelimelerin geçtiği dokümanları karşımıza getirmektedir. Girilen birkaç anahtar kelimeye karşılık binlerce sayfa gelebilmektedir. Bu kadar çok sayfanın tek tek okunup istenilen web sayfasına

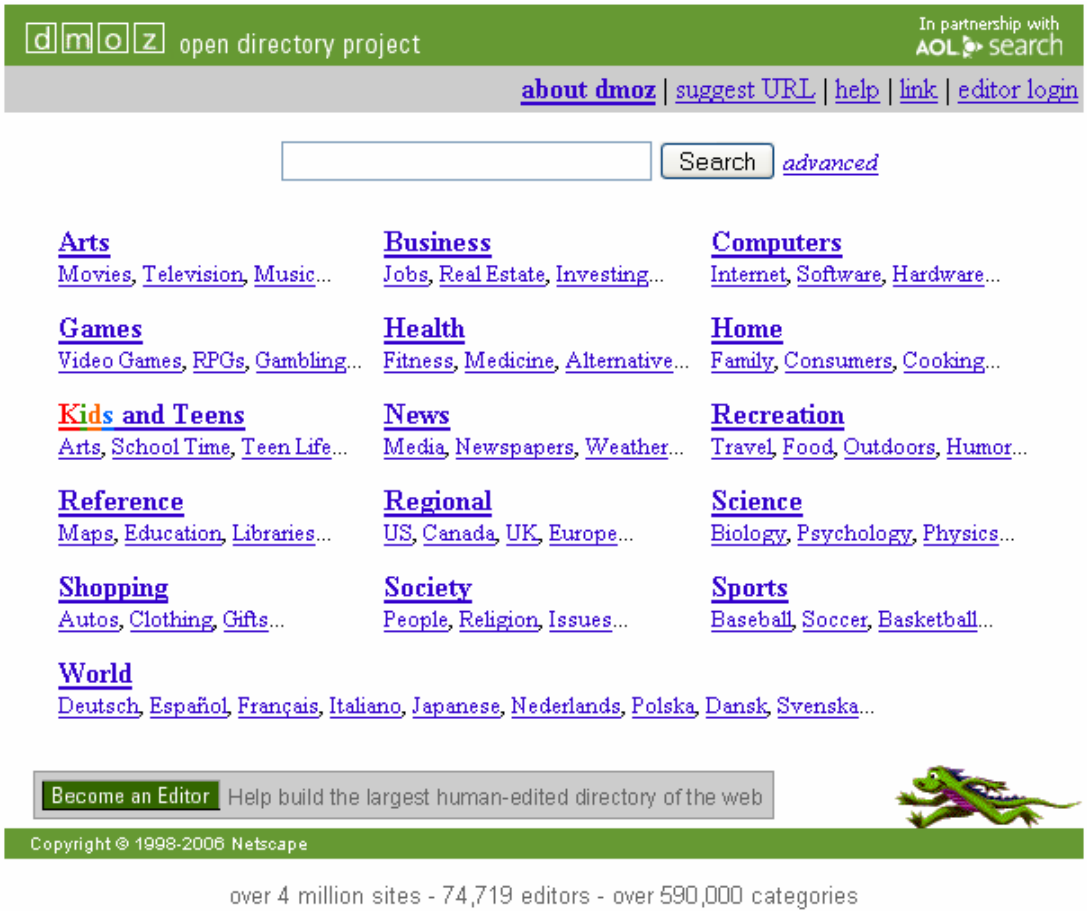
ulaşmak çok zaman almaktadır. Daha da kötüsü ilgisiz birçok web sayfası ziyaret edilmek zorunda kalınmaktadır. Bu durum, İnternet'i kullanarak çalışanların iş verimini düşürmektedir. Dolayısıyla basit bir kelime taraması artık yeterli görülmemektedir. Bunun yerine verileri daha anlamlı bir şekilde kategorize etmiş yapılara ihtiyaç vardır.

İnternet'teki web sayfalarını içeriklerine göre anlamlı gruplandırarak kullanıcıya sunan yapılardan birisi açık dizinlerdir. Açık dizin, insanlar tarafından sınıflandırılmış web içeriğini barındıran en geniş kapsamlı veritabanıdır. Açık dizin veri yapısı bir ağaç yapısı şeklindedir. Ağacın her bir düğümü bir sınıfı veya altsınıfı temsil etmektedir. Ağacın yaprakları ise sınıflandırılmış dokümanların web adreslerini göstermektedir. Açık dizinlerin İnternet kullanıcılarından oluşan bir editör kadrosu vardır. Bu kişiler İnternet'teki kaynakların keşfini sağlayan kolektif beyni oluşturur. Açık dizin, İnternet'in en büyük ve en yaygın kullanılan arama motorlarının ve portallarının dizin hizmetlerini güçlendirmektedir. Bunlara örnek olarak Netscape Search, AOL Search, Google, Lycos, HotBot, DirectHit ve daha yüzlercesi gösterilebilir.

Dünyadaki en büyük açık dizin projesi, "The Open Directory Project" (DMOZ) diye anılan çalışmadır. Bu açık dizin insanlar tarafından elle oluşturulmaktadır. Bu proje, insanlar tarafından oluşturulan dünyadaki en büyük açık dizin projesidir. Bu projeye http://www.dmoz.org adresinden erişilebilmektedir. Bu projenin ana sayfası Şekil 1.1'de görülmektedir. Burada ana kategori olarak sanat, iş, bilgisayar, oyun, sağlık, çocuklar, haberler, kaynaklar, bilim, alışveriş, toplum ve spor gibi başlıklar alınmıştır. Her ana kategorinin altında alt kategoriler vardır. Örneğin bilgisayar ana kategorisinin altında İnternet, yazılım, donanım gibi alt kategoriler bulunmaktadır. Başka bir örnek olarak bilim ana kategorisinin altında biyoloji, psikoloji, fizik gibi alt kategoriler yer almaktadır.

DMOZ projesinde şimdiye kadar 4 milyondan fazla web sitesi sınıflandırılmıştır. Bu işlem için yaklaşık 75 bin gönüllü çalışmıştır. Bu çalışma sonucunda 590.000'nden

fazla kategori oluşturulmuştur. DMOZ projesinin istatistiklerine göre projenin İnternet'in ancak binde birini sınıflandırabildiği belirtilmektedir.



Şekil 1.1. Sınıflandırma örneği

İnternet'te mevcut dokümanların farklı birçok konuda olabilmesi ve arama motorlarına girilen kelimelerin iyi seçilmemesi, Yahoo ve DMOZ gibi açık dizinlere talebi arttırmaktadır. Artan bu talep karşısında elle sınıflandırma yetersiz kalmaktadır. Bu nedenle otomatik sınıflandırma sistemlerine ihtiyaç vardır. Otomatik sınıflandırma sistemleri, bilgi yönetiminin geleceği için kritik yöntemlerdendir [2].

Doküman sınıflandırma ile ilgili birçok çalışma yapılmış ve değişik teknikler kullanılmıştır. Bu tekniklerden en yaygın kullanılanı yapay sinir ağı yaklaşımlarıdır. Yapay sinir ağlarında danışmanlı ve danışmansız olmak üzere iki türlü öğrenme

yaklaşımı vardır. Danışmanlı öğrenme, doküman sınıflandırmada giriş verilerinin çok olması ve her bir düğümün bir vektör olarak gösterilmesi nedeniyle uygun görülmemektedir. Bunun yerine danışmansız öğrenme yaklaşımı tercih edilmektedir. Doküman sınıflandırmada kullanılan danışmansız öğrenme tekniklerinden biri kendinden düzenlenen haritalardır. Kendinden düzenlenen haritaların en önemli avantajı ağın kendi kendini eğitmesidir. Bu ağların eğitiminde dışarıdan bir danışman müdahalesine gerek duyulmamaktadır. Ağ tamamen otomatik olarak eğitilmektedir. Yani tek yapmamız gereken ağa giriş verilerini vermek ve ağın kendi kendini eğitmesini sağlamaktır.

Kendinden düzenlenen haritalar, bir veri kümesindeki var olan anlamsal (semantik) benzerlikleri başarılı bir şekilde ortaya çıkarmaktadır [3]. Bu ağların başarısını ve hızını arttırmak için bazı önişlemler uygulanır. Bu işlemlerin en önemlileri kelime indeksleme, durak kelimelerinin temizlenmesi, normalleştirmedir. Bu kavramlar tez içerisinde daha detaylı bir şekilde açıklanmaktadır.

Kendinden düzenlenen haritalar, anlamsal olarak benzerlikleri ortaya çıkarmakla birlikte, dokümanların etkin bir şekilde sınıflandırması konusunda daha değişik teknikler de kullanılabilmektedir. Bu tekniklerden bir tanesi hiyerarşik kümelemedir. Hiyerarşik kümeleme doküman sınıflandırma için kullanılan yöntemlerden birisidir. Hiyerarşik kümeleme algoritmaları yüksek boyutta verilerin sınıflandırılmasındaki başarısından dolayı tercih edilmektedir [4].

Bu tezde, otomatik doküman sınıflandırma konusunda bir çalışma yapılmıştır. Bu çalışmada kendinden düzenlenen haritalar algoritması kullanılmıştır. Kendinden düzenlenen haritalar algoritması ile elde edilen sonuçlar etkin bir kümeleme yöntemi olan hiyerarşik kümeleme ile karşılaştırılmıştır. Çalışmada 2 farklı doküman kütüphanesi ele alınmıştır. İlk çalışmada bir İnternet haber sitesinden rastgele alınmış haber örnekleri sınıflandırılırken, ikinci çalışmada çeşitli üniversitelerin ders içerikleri web sayfalarından toplanarak başarılı bir şekilde sınıflandırılmıştır. Bu

yöntem, farklı içeriklerin sınıflandırılmasında da başarılı bir şekilde kullanılabilecektir.

Kendinden düzenlenen haritalarla ilgili olarak birçok çalışma yapılmıştır. Kendinden düzenlenen haritalar, 1991 yılında, Lin tarafından bilimsel dokümanların başlıklarının haritalanması için kullanılmıştır [5]. Scholtes, kendinden düzenlenen haritalar tabanlı bir nöron süzgeci ve bilgi erişimi için bir ilgi haritası geliştirmiştir [6]. Merkl, yazılım kütüphanesi bileşenlerini, metin tanımlanmalarını kümelemek için kendinden düzenlenen haritalar kullanmıştır [7]. Segal ve Kephart, e-postaların otomatik sınıflandırılması için uyarlanır bir sınıflandırma geliştirmiş ve %80-%90 oranında kullanıcı tatmini elde etmiştir [8]. Segal ve Kephart, bu çalışmalarında TF-IDF sınıflandırmasını kullanmışlardır.

Son 10-15 yılda metin sınıflandırma alanında çok ilerleme kaydedilmektedir. Bu konuda özellikle istatistiksel sınıflandırma ve makina öğrenme metotları kullanılmaktadır [9]. Bunlar arasında çok değişkenli regresyon modelleri [9-11], en yakın komşuluk [12], karar ağaçları [13], Bayes olasılık yaklaşımları [13, 14], yapay sinir ağları [9, 15-17], sembolik kural öğrenme [18-20] ve tümevarım öğrenme algoritmaları [20] sayılabilir.

SOM algoritması ilk olarak T. Kohonen tarafından geliştirilmiş olmakla birlikte bu konuda daha sonraları birçok çalışma yapılarak değişik yaklaşımlar ortaya atılmıştır. Kohonen 80 farklı Usenet grubundan elde ettiği 1 milyondan fazla mesajı SOM kullanarak sınıflandırmıştır [21]. Bu çalışma benzer konulardaki mesajları gruplandırma konusunda başarılı olmuştur.

SOM algoritması, genel olarak kümeleme ve veri boyutunu azaltma konusunda başarılı olarak kullanılmaktadır. Kümeleme, giriş verilerinin kategorizasyonu için önemlidir. Diğer yandan, verinin boyutunu küçültmek ise verinin görselleştirilmesi, aranması, depolanması ve işlenmesi gibi aşamaların daha verimli yapılabilmesi açısından önemlidir. Kohonen tarafından geliştirilen ve internet ortamındaki

dokümanların sınıflandırılmasını sağlayan WEBSOM metodu otomatik kategorizasyon yapmamaktadır. Bu durumda kullanıcının veri haritasını anlamlı kategorilere ayırması beklenmektedir. Bu açığı farkeden araştırmacılardan Roussinov ve Chen, makalelerinde SOM kullanarak dokümanları otomatik olarak belirli kategorilere dahil edebilmişlerdir [22]. SSOM (Ölçeklenebilir SOM) adını verdikleri bu çalışmada öncelikle hiyerarşik bir SOM elde edilmektedir. Her düğüm için en ayırtedici kelime etiket olarak seçilmekte ve kazanan terim olarak kabul edilmektedir. Daha sonra komşu bölgelerde kazanan terime sahip dokümanlar bu bölgeyle birleştirilmektedir. Chen ve arkadaşları, algoritmalarını farklı 3 veri kümesinde kalite testine sokmuşlar ve başarılı sonuçlar elde etmişlerdir [22, 23].

Veri kümesi seçimi metin sınıflandırmanın geçerliliği ve verimliliği için oldukça önemlidir. Metin sınıflandırma çalışmaları için kabul edilmiş ortak bir veri kümesi bulunmamaktadır. Ancak literatürde araştırmacıların yaygın olarak Reuters, Associated Press, Usenet gibi çok büyük veri hacmine sahip organizasyonların metinlerini kullandıklarını söyleyebiliriz. Özellikle Yang, metin sınıflandırmanın etkin öğrenmesine odaklanarak şu sorulara cevap aramıştır [24] :

- Hangi eğitim kümesi metin sınıflandırma performansını en iyi şekilde ortaya koyar?
- Belirli bir kategori öğrenimi için gereken en az örnek metin sayısı nedir?
- Gerçek bir problem için alınması gereken örnek veri kümesinin büyüklüğü ne olmalıdır?

Merkl ve Rauber, doküman kategorilerinin tek bir harita şeklinde gösterilmesinin yetersiz olduğunu iddia ederek çalışmalarını hiyerarşik SOM üzerine yoğunlaştırmıştır. Bu amaçla coğrafik haritalarda olduğu gibi başlangıçta genel bir harita ile başlanarak kullanıcının istediği noktalarda detay seviyelere inmesini sağlayacak bir algoritma üzerinde çalışmışlardır. Bunu yaparken yapay sinir ağlarının aslında belirli katmanlardan oluştuğunu ve her bir katmanın dokümanlar açısından hiyerarşik olarak bir detay seviyesi olabileceğini dikkate almışlardır.

Dolayısıyla her bir katmanın tekil bir SOM olabileceği üzerinde durmuşlardır. Bu sayede doküman arşivinde bulunan içeriğin istenilen detayda kategorize edilebileceğini göstermişlerdir. Tüm bu çalışmalarını danışmansız yapay sinir ağları üzerinde yapmış olan Merkl, 1990 yılına ait CIA Word Factbook'ta yer alan ülkelerle ilgili 245 adet dokümanı kullanarak değişik detay seviyelerinde bilgiler sunabilmiştir. Merkl, dokümanların gösteriminde vektör uzayı modeli ve TF-IDF terim ağırlıklandırmasını kullanmıştır [25]. Hiyerarşik SOM konusunda detaylı çalışmalar yapan diğer bir araştırmacı ise Dittenbach'tır [26].

Merkl, "NIH Class Library" adlı kütüphanenin kullanım klavuzu sayfalarını kullanan bir SOM oluşturmuştur [27]. "NIH Class Library", karmaşık veri yapılarını depolayan ve istenilen verilere erişim sağlayan bir C++ kütüphanesidir. SOM sonucunda ilişkili fonksiyonları başarılı bir şekilde gruplandırmıştır. Bu çalışma SOM haritalarının çok teknik dokümanlar için de başarıyla kullanılabileceğini göstermiştir.

SOM algoritması çok yüksek boyuttaki verilere uygulamak için çok uygun olmakla birlikte haritanın eğitilerek istenilen sonucu verebilmesi için çok yüksek çalıştırma sürelerine ihtiyaç duymaktadır. Daha etkileşimli bir cevap süresi elde edebilmek için yapılan çalışmalardan birisi de parSOM (paralel SOM)'dur [28]. Bu çalışmada yazılım tabanlı bir paralel işleme gerçekleştirilerek yüksek boyutlu verilerle ağın eğitilmesinde geçen cevap verme süresinde büyük ölçüde azalma elde edilebilmiştir. SOM ağlarının eğitilmesinde geçen uzun süreyi kısaltmak için donanım tabanlı çalışmalar da gerçekleştirilmiş olmakla birlikte istenilen seviyede başarı elde edilememiştir [29, 30].

Bu tezin 2. bölümünde sınıflandırma konusu ele alınmaktadır. Doküman sınıflandırma metotları ve önceki yıllarda yapılan çalışmalar bu bölümde anlatılmaktadır. 3. bölümde doküman modelleme metotları başta vektör modeli olmak üzere anlatılmaktadır. 4. bölüm veri kümeleme algoritmalarına ayrılmıştır. Bu çalışmada kullanılıyor olmasından dolayı hiyerarşik kümeleme, İç Anadolu

bölgemizdeki şehirler kullanarak detaylı bir şekilde anlatılmaktadır. 5. bölüm Yapay Sinir Ağlarının teorik altyapısını açıklamaktadır. 6.bölümde kendinden düzenlenen ağlar detaylı bir şekilde incelenmektedir. 7. bölüm bu çalışmada geliştirilen uygulamayı anlatmaktadır. 8. bölüm ise sonuç ve ileriye dönük çalışma önerilerine ayrılmıştır.

2. DOKÜMAN SINIFLANDIRMA

Bilişim alanında doküman sınıflandırma önemli bir konudur. Doküman sınıflandırmanın temel amacı bir elektronik dokümanı içeriğine göre bir veya birden fazla kategoriye dahil etmektir. Doküman sınıflandırma, benzer dokümanları bir araya getirerek gruplandırmaktır. Örneğin, bir metinde geçen 2 kelime, diğer kelimelere göre daha fazla tekrar ediyorsa, bu iki kelimenin aynı kavrama ait olduğunu söylemek mümkündür. Böylece sınıflandırma aramalarını hızlandıran bir gruplama sağlanmaktadır.

Sınıflandırma konusunda birçok çalışma yapılmış ve değişik teknikler kullanılmıştır [9]. En yaygın kullanılan doküman sınıflandırma metotları aşağıda kısaca anlatılmaktadır.

2.1. Karar Ağaçları

Karar ağacı metotları eğitim dokümanlarının elle doğru/yanlış sorguları şeklinde bir ağaç yapısında sınıflandırılması ile oluşturulur. Ağacın düğümleri soruları yaprakları ise ilgili doküman sınıfını temsil eder. Ağacı bu şekilde oluşturduktan sonra yeni bir dokümanın kök düğümden başlayarak kolaylıkla ilgili yaprağa doğru yönlendirilmesi sağlanabilmektedir [31].

Karar ağaçlarının en büyük avantajı çıktı ağacının model hakkında bilgisi olmayan kullanıcılar tarafından bile kolaylıkla yorumlanabilmesi gerçeğidir.

Karar ağacı uygulamaları aşırı uygunluk denilen bir riski taşırlar. Bir ağacın eğitim verisi ile aşırı uygun olması, ileride çok daha iyi bir sınıfa dahil olabilecek bir eğitim verisinin daha kötü bir sınıflandırmaya tabii tutulmasıdır [31]. Bu durumda eğitim dokümanları için doğru bir sınıflandırma elde edilmekle birlikte sonradan gelen dokümanlar için mevcut sınıflandırma yeterince uygun olmayabilmektedir. Bu riski azaltmak için eğitim dokümanlarının mümkün olabildiğince bu durumları dikkate

alarak hazırlanması gerekir. Bu metodun diğer bir riski ise ağacın aşırı büyümesidir. Ancak bu riski aşmak için ağacın maksimum derinliği ayarlanabilir.

2.2. Karar Kuralları

Karar kuralı algoritmalarında her bir kategori için kategorinin profilini gösteren bir kural kümesi oluşturulur. Genellikle, kategori ismini ve temel bir özelliği gösteren tekil kurallar oluşturulduktan sonra farklı kuralların mantıksal işleçlerle birleştirmesiyle kural kümeleri elde edilir. Mantıksal işleç olarak "VEYA" kullanılır. Dokümanları sınıflandırmak için tüm kurallar kullanılmaz. Bunun yerine sezgisel metotlara ihtiyaç duyulur. Burada amaç her bir dokümanın sınıflandırılmasını etkilemeden kullanılacak karar kümelerinin azaltılmasıdır [18].

Karar kurallarının en büyük avantajı özellik çıkarma fazında kategori başına lokal sözlüklerin oluşturulmasıdır. Örneğin eşsesli bir kelime olan "çay" kelimesi anlamları farklı olmakla birlikte telaffuzları aynı olan iki farklı anlama gelir. "çay" kelimesinin ilk anlamı içilen bir bitki olması diğer anlamı da akan küçük su, deredir. Genel bir sözlükte "çay" kelimesi sadece bir defa listelenir. Böylece her iki farklı anlam için bir özellik vektörü kullanılmak zorunda kalınır. Bununla birlikte lokal sözlükler eşsesli kelimeleri ayırt edebilmektedir. Bu durumda "çay" kelimesi her bir lokal sözlükte ayrı ayrı listelenir ve farklı anlamların farklı doküman kategorilerine dahil edilmesi sağlanmış olur.

Bu metodun dezavantajı bir dokümanın farklı kategorilere ait kuralların uygulanabilirliğinden dolayı yalnızca bir kategoriye dahil edilememesidir. Bu durumda bir doküman genellikle birden çok kategoriye girebilmektedir.

2.3. K-En Yakın Komşuluk

Karar ağaçları ve karar kuralları metotlarında sınıflandırma için birincisi öğrenme fazı olmak üzere en az 2 faz uygulanmaktadır. K-en yakın komşuluk metodu ise öğrenme fazını atlayarak anlık bir sınıflandırma uygulaması gerçekleştirir.

Dokümanları özellik uzayındaki en yakın K sayıda örneklerine göre sınıflandıran bir danışmanlı öğrenme tekniğidir. Nesneler arasındaki uzaklık hesabı için genellikle Öklit uzaklığı kullanılır.

Bu metodun en büyük avantajı basit olmasıdır. Bu metot, kategori spesifik dokümanlar birden fazla küme oluştursa bile iyi bir performans gösterir [31].

K-en yakın komşuluğun dezavantajı sınıflandırma için harcanan sürenin ortalamanın üzerinde olmasıdır. Bu sürenin uzun olmasının sebebi olarak herhangi bir ön hazırlık veya öğrenme fazı uygulanmaması söylenebilir. Daha da kötüsü her bir kategorideki doküman sayılarının birbirinden farklı olmasına rağmen k-en yakın komşuluk ile birbirine çok benzemeyen dokümanların da aynı kategoriye dahil edilmek zorunda kalınmasıdır.

2.4. Bayes Yaklaşımları

Bayes teorisine dayanan olasılıklı bir sınıflandırma tekniğidir. Oldukça karmaşık bir hesaplama tekniği vardır.

Doküman sınıflandırmada kullanılan 2 grup Bayes yaklaşımı vardır: tecrübesiz (naive) ve tecrübeli (non-naive) Bayes yaklaşımı. Tecrübesiz yaklaşımda kelimenin (yani özelliğin) bağımsızlığı söz konusudur. Yani kelimelerin sıralamasının önemsiz olduğu ve bir kelimenin varlığının başka bir kelimenin varlığını veya yokluğunu etkilemediği kabul edilir [32]. Bazı tecrübeli yaklaşımlar bu kabulü dikkate almaz [33].

Tecrübesiz Bayes yaklaşımlar çok daha önceleri geliştirilmiş olmakla birlikte doküman sınıflandırma konusu önem kazanmadan önce veri madenciliğinde çok yoğun bir şekilde kullanılmıştır. Bu metot karmaşık olmakla birlikte doküman sınıflandırmasında iyi bir performans vermektedir [34]. Bayes yaklaşımların dezavantajı sadece ikili özellik vektörlerini işleyebilmeleri ve dolayısıyla ilişkili bilgileri atlayabilmesidir [33].

2.5. Bağlanım Tabanlı Yaklaşımlar

Bu metotta eğitim verisi bir giriş ve bir çıkış olmak üzere bir matris çifti olarak gösterilir. Giriş matrisi (A) özellikleri gösterirken, çıkış matrisi (B) giriş matrisindeki dokümanların kategori ilişkilerini göstermektedir. Böylece B matrisi, A matrisindeki satır sayısı (m) kadar satıra ve toplam kategori sayısını gösteren (c) kadar sütuna sahiptir. Bu metotta amaç, A matrisini B´ matrisine dönüştüren bir F matrisini bulmaktır (B´ = A*F). F matrisi çok değişkenli bağlanım teknikleri kullanılarak elde edilmeye çalışılır [35].

Bu metodun önemli bir avantajı, dil bilgisine bağımlı önişlemlere ihtiyaç duymamasıdır. Böylece farklı diller için kolaylıkla uygulanabilmektedir. Ancak bağlanım tabanlı metotlar sınıflandırma konusunda çok tercih edilmez. Dolayısıyla bu metodun diğer metotlarla karşılaştırması konusunda yeterince araştırma yapılmamıştır.

2.6. Vektör Tabanlı Metotlar

Vektör tabanlı metotların en çok kullanılan 2 türü vardır : kitle merkezi ve destek vektör makinesi (Support Vector Machine, SVM) [36].

En basit sınıflandırma metotlarından birisi kitle merkezi metodudur. Öğrenme aşamasında her bir kategori için ortalama özellik vektörü bulunur ve ilgili kategori için kitle merkezi olarak belirlenir. Yeni bir doküman dokümanın özellik vektörüne en yakın kitle merkezi vektörü bulunarak kolaylıkla sınıflandırılabilmektedir. Uzaklık hesabı için Öklit formülü kullanılır. Doküman kategori sayısının çok fazla olması durumunda bu metot verimli sonuçlar üretememektedir.

Destek vektör makinesi ise sınıflandırma ve bağlanım için kullanılan bir danışmanlı öğrenme metotları kümesidir. Çoğunlukla, doğrusal sınıflandırma tekniklerinin doğrusal olmayan sınıflandırma problemlerine uygulanması için kullanılır [3].

Destek vektör makinesi (SVM), diğer metotlara göre daha güçlü olduğunu gösteren birçok çalışma yapılmıştır [9,37-39].

2.7. Yapay Sinir Ağları

Yapay sinir ağları (YSA), ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem elemanlarından (nöronlar) oluşan matematiksel sistemlerdir. YSA, bir sisteme ilişkin tek veya çoklu parametrelere bağlı olarak tanımlanabilen çıkışları arasında ilişki kurabilme yeteneğine sahiptir. Bu ilişkinin doğrusal bir formda olması zorunlu değildir. YSA'lar, çıkış değerleri bilinmeyen tanımlanmış sistem girişlerine uygun çıkışlarda üretebilirler [15].

Doküman sınıflandırma problemi için farklı sinir ağı yaklaşımları uygulanmıştır. Bunlardan bazıları YSA'nın en basit formu olan (perceptron) giriş ve çıkış katmanlarından oluşurken [40] bazıları gizli katmanlar da kullanarak daha karmaşık yapılar oluşturmuşlardır [41]. Bu çalışmada T. Kohonen tarafından geliştirilen SOM algoritması detaylı olarak anlatılmaktadır.

3. DOKÜMAN SINIFLANDIRMADA İSTATİSTİK MODELLER

Her bir doküman için bir vektör tanımlanır. Bu vektörün elemanları doküman içinde geçen kelimelerdir. Elde ettiğimiz her bir vektöre bir model olarak bakabiliriz. Ancak kullanacağımız model giriş vektörünün bir kopyası olmak zorunda değildir. Örneğin modeller kelimelerin ağırlık histogramları olarak alınabilmektedir. Doküman sınıflandırmada kullanılan veri modeli çok yüksek boyutlarda olabildiğinden bu modellerin küçültülmesine ihtiyaç vardır. Veri modellerinin küçültülmesi amacıyla değişik teknikler önerilmiştir [21]. Bunlardan bazıları aşağıda açıklanmaktadır.

3.1. Temel Vektör Uzayı Modeli

Temel Vektör Uzayı Modelinde [43], dokümanlar gerçek vektörler olarak gösterilir. Vektörün her bir elemanı, her bir kelimenin dokümandaki tekrarlanma sayısını (frekans) gösterir. Model veya doküman vektörü bir ağırlıklı kelime histogramı olarak düşünülebilir. Kelimelerin ağırlıklandırılması için doküman sınıfları üzerinde Shannon entropisi veya ters doküman frekansı kullanılabilir. Vektör uzayı modelinin temel problemi serbest metinli dokümanlarda büyük bir sözlük ortaya çıkması ve dolayısıyla vektör modellerinin büyük ölçekte boyutlardan oluşmasıdır.

Vektör modelleri, bilgi erişiminde ve metin sınıflandırmasında başarılı bir şekilde kullanılmaktadır. Bu yaklaşım, herhangi bir dokümanın bir vektör olarak gösterilebileceğini kabul eder. Dokümandaki kelimelerin sırası ve dilbilgisi kuralları gibi konular göz ardı edilir ve doküman faydalı bilgi elde edilebilecek hale getirilir.

Çizelge 3.1’de dokümanlara ait bir vektör modelinin nasıl oluşturulabileceği basit bir örnekle gösterilmeye çalışılmıştır. Çizelge 3.1’de verilen dokümanlar Çizelge 3.2’de bir vektör modeli olarak gösterilmiştir. Bu örnekte farklı üç doküman (D1, D2, D3) alınmış ve bu dokümanların 4 farklı sınıfa (S1, S2, S3, S4) ait olduğu kabul

edilmiştir. Her bir dokümanın içerisinde geçen metin örneği ve dokümanın ait olduğu sınıflar yanında verilmiştir.

Çizelge 3.1. Vektör modeli için örnek doküman içerikleri ve sınıfları

Doküman	**Doküman içeriği**	**Sınıflar**
D1	“Fen Bilimleri Enstitüsü”	S1,S2
D2	“Fen Bilimleri ve Fizik Laboratuvarı”	S1,S3
D3	“Gazi Üniversitesi Laboratuvarı ile	S1,S3,S4

Bu dokümanlarda geçen kelimeleri tek tek indekslediğimizde her bir dokümana ait kelimeleri aşağıdaki çizelgede olduğu gibi kolaylıkla görebiliriz. Örneğin “Fen” kelimesi D1 ve D2’de geçmekte olup D3’te geçmemektedir.

Çizelge 3.2. Vektör modeli oluşturma örneği

	D1	D2	D3
Fen	1	1	0
Bilimleri	1	1	0
Enstitü	1	0	0
Fizik	0	1	0
Laboratuvar	0	1	1
Gazi	0	0	1
Üniversite	0	0	1

	D1	D2	D3
S1	1	1	1
S2	1	0	0
S3	0	1	1
S4	0	0	1

Çizelgenin sağ tarafında ise doküman ve sınıf eşlemesi görülmektedir. S1 sınıfına ait dokümanlar D1, D2, D3’tür. S2 sınıfına ait doküman D1’dir. S3 sınıfına ait dokümanlar D2 ve D3’tür. S4 sınıfına ait doküman ise D3’tür.

Dokümanlardaki farklı kelime sayısını azaltabilmek için bazı kelimeler temizlenir. Bu kelimeler kendi başına çok anlamı olmayan kelimeler ve bağlaçlardır. Yukarıdaki verilen cümlelerde geçen “ve”, “ile” gibi kelimeler yok sayılacak kelimelere örnek olarak verilebilir. Bu sayede dokümanların uzunluğu %30-40 oranında azaltılabilmektedir. Başka bir önişlem ise kelimelerdeki çoğul ekleri kaldırmaktır.

Bir dokümanı vektörel olarak gösterebilmek için bir a_{ij} terimlerinden oluşan bir A matrisini kullanabiliriz. Matristeki terimlerin ağırlıklarını bulmak için farklı metotlar kullanılabilir [42]. Bunlardan en yaygın olanları şunlardır:

- İkili ağırlıklandırma: Çizelge 3.1' deki ağırlandırma buna bir örnek olarak düşünülebilir. A matrisine ait a_{ij} terimleri i. dokümanda j. terimin değeriyle gösterilir:

$$a_{ij} = \begin{cases} 0 : \text{j. terim i. dokümanda yok} \\ 1 : \text{j. terim i. dokümanda var} \end{cases} \quad (3.1)$$

- Terim frekans ağırlıklandırma (TF)

$$a_{ij} = TF_{ij} = j.\text{ teriminin } i.\text{ dokümanda tekrar sayısı.} \quad (3.2)$$

- Ters doküman frekans ağırlıklandırma (IDF)

$$a_{ij} = IDF_{ij} = \log\left(\frac{toplam\_dokuman\_sayisi}{j.\_terimi\_iceren\_dok\_say}\right) + 1 \quad (3.3)$$

Ağırlık hesabı için,

$$a_{ij} = f(TF_{ij}) \times g(IDF_{ij}) \quad (3.4)$$

çarpımı da kullanılabilmektedir. Ağırlık vektörleri genellikle, kosinüs normalizasyonu olarak adlandırılan

$$\sqrt{\sum (a_{ij}^{\ 2})} \quad (3.5)$$

fonksiyonu ile normalize edilirler.

Ağın eğitimi için vektör ağırlıklarının 0 ile 1 aralığına dönüştürülerek normalize edilmesine ihtiyaç vardır. Burada yapılan bir istatistiksel normalizasyondur. Ağırlıkların büyüklük olarak $d^{\min}$ ve $d^{\max}$ aralığında olduğunu ve [0,1] aralığında olmadığını kabul edelim. Ağırlıkları [0,1] aralığına dönüştürmek için Eş. 3.6 kullanılabilir.

$$\delta = \frac{d - d^{\min}}{d^{\max} - d^{\min}} \tag{3.6}$$

Doküman vektörlerinin boyut sayısı gereksiz kelime ve eklerin çıkarılmasından sonra kalan kelimelere bağlıdır. Boyutlar genellikle çok büyük değerlere ulaştığından dolayı boyut azaltma tekniklerine ihtiyaç duyulur. Boyut azaltmak için dokümandaki gürültü temizlenir. Çok sık tekrarlanan ve çok az geçen kelimeler gürültü olarak kabul edilir. Ayrıca durak kelimeler de gürültü olarak kabul edilir. Gürültü azaltıldıktan sonra doküman vektörleri daha anlamlı olmaktadır ve sınıflandırma daha doğru yapılabilmektedir. Doküman vektörlerinin boyutlarının küçültülmesi, çalışma zamanını (algoritmanın karmaşıklığını) azaltır.

3.2. Terim Frekansı ve Ters Doküman Frekansı (Tf-Idf)

Kelimelerin dokümandaki önemini istatistiksel olarak hesaplamaya çalışan bir tekniktir. Kelimenin önem derecesi dokümanda kaç kez tekrar edildiğine bağlı olarak artarken bir doküman grubu içerisindeki tekrar sayısına göre dengelenir. Tf-idf arama motorları tarafından yaygın olarak kullanılan bir metottur [42].

3.3. Gizli Semantik İndeksleme

Bu metotta, doküman vektörlerinin boyutunu küçültmek için bir matris hazırlanır. Bu matriste her bir kolon bir dokümanın kelime histogramından oluşur. Her bir kolon vektörünün genel uzaydaki çarpanını bulmak için tekil değer ayrıştırması denilen bir hesaplama tekniği kullanılır [44]. Bu hesaplama sonucunda matristeki en az etkisi

olan çarpanlar ihmal edilir. Sonuçta geride kalan çarpanlara ait histogram değerleri ile daha küçük boyutta bir doküman vektörü elde edilmiş olunur.

3.4. Rastgele İzdüşümü Alınmış Histogramlar

Doküman vektörlerinin boyutları, rastgele izdüşümü alma metoduyla dokümanlar arasındaki farklılığı kaybetmeden ciddi oranda küçültülebilmektedir [46]. Orijinal doküman vektörünü (ağırlıklı histogram), $n_i \in \Re^n$ ve her bir kolonundaki elemanları normal dağıtılmış rastgele bir dikdörtgensel matrisi $\Re$ ile gösterelim. Doküman vektörünü $x_i \in \Re^m$ şeklinde tekrar oluşturalım.

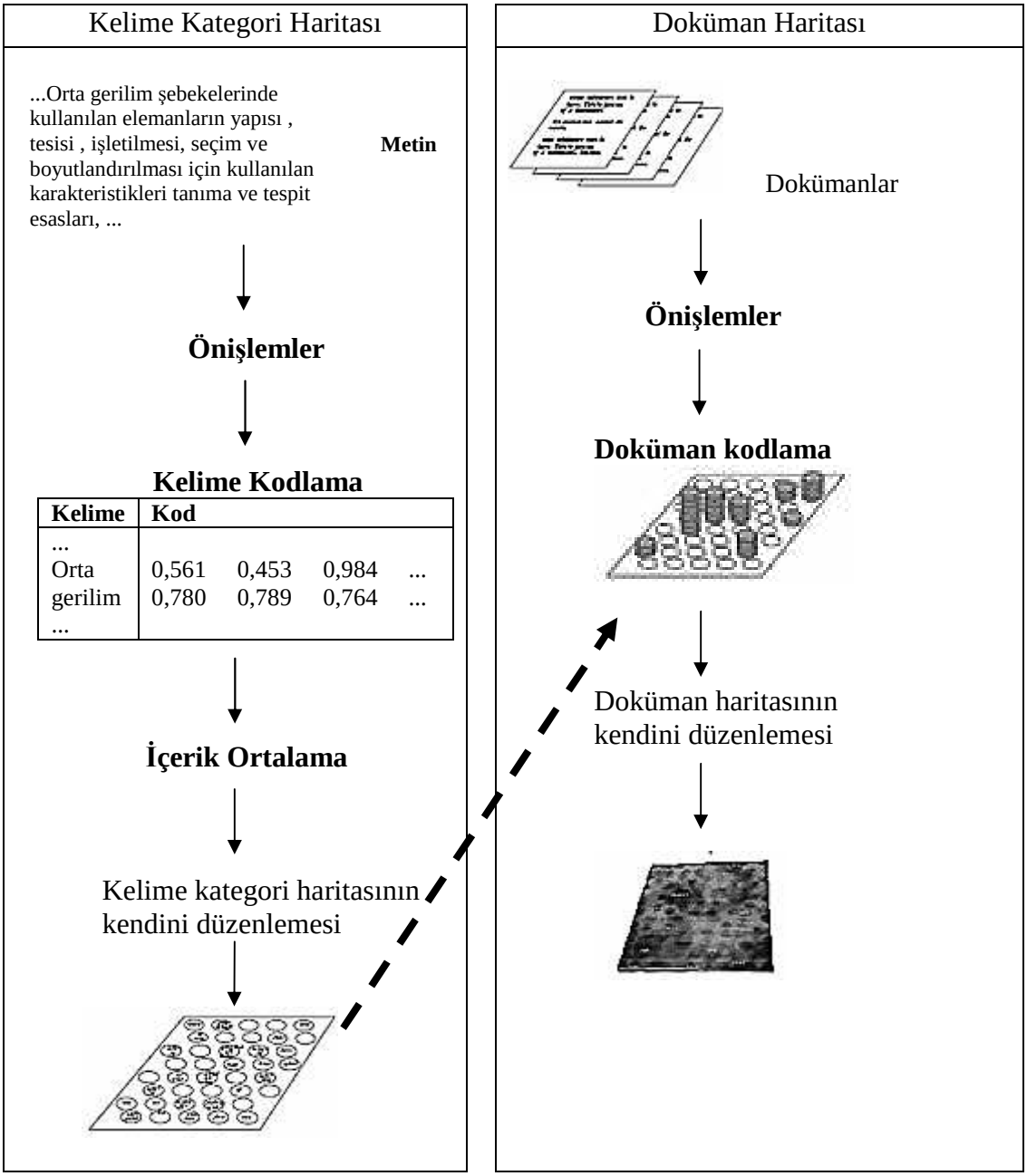
$$x_i = Rn_i, \quad m << n \tag{3.7}$$

Bu metotla doküman vektörünün boyu en az 100 katı oranında küçültülebilmektedir.

3.5. Kelime Kategori Haritası Histogramları

Kelime kategori haritası, doküman kodlama için kullanılmakta ve kelime benzerliklerini dikkate almaktadır. Kelime dizisindeki *i.* kelime *n* boyutlu bir gerçek vektör olan x_i ile gösterilir. Kavramsal olarak birbiriyle ilgili kelimeler aynı veya komşu düğümlere düşer. Böylece düğümler kelime kategorileri olarak düşünülebilir [15].

Kelime kategori haritası ve bu haritalama ile elde edilen doküman haritası mimarisinin gösterimi Şekil 3.1'de verilmiştir [47].



Şekil 3.1. Kendinden düzenlenen haritalar metodunun temel yapısı

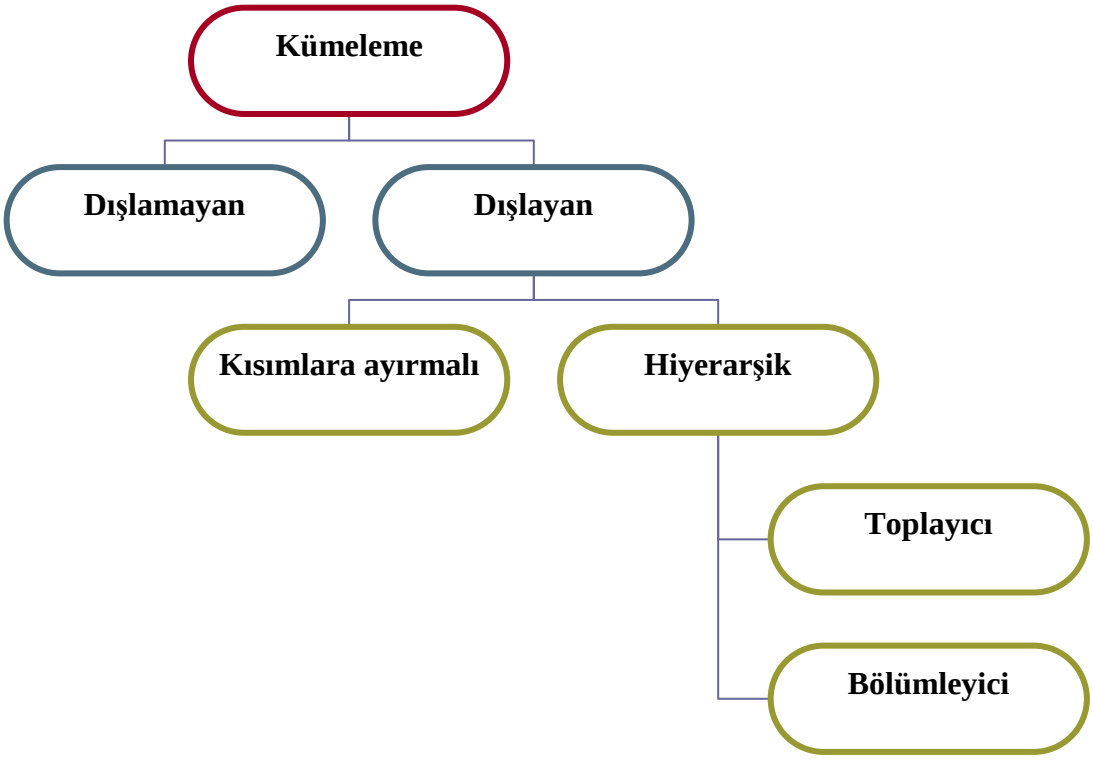
4. VERİ KÜMELEME

Kümeleme, en basit tanımıyla benzer özellik gösteren veri elemanlarının kendi aralarında gruplara ayrılmasıdır. Kümelemenin amacı benzer verileri gruplandırarak veri miktarını azaltmaktır. Bu gruplama insanların bilgiyi işleme şekline uygundur ve çok yaygın olarak kullanılır. Verinin otomatik olarak gruplandırılması ise kategoriler ve taksonomilerin oluşturulmasını kolaylaştırarak, süreçteki insan faktörünün müdahalesini minimuma indirir [48].

Literatürde kümeleme analizini açıklayan bir çok tanım bulunmaktadır [48-50]. Bu tanımlara göre her küme temsil ettiği nesneleri en iyi şekilde ifade edecek şekilde düzenlenir. Kümeleme işleminin uygulandığı veri setindeki her bir veriye nesne adı verilir. Bu nesneler iki boyutlu düzlem üzerinde noktalarla gösterilir. Kümeleme analizi, veri indirgeme veya nesnelerin doğal sınıflarını bulma gibi çeşitli amaçlarla kullanılmaktadır. Kümeleme analizinin kullanıldığı sayısız uygulama alanı bulunmaktadır. Bu alanlardan en çok gündemde olanlar desen tanıma, veri analizi, resim tanıma, pazarlama, metin madenciliği, doküman toplama, istatistik araştırmaları, makine öğrenimi, şehir planlama, coğrafik analizler (deprem, meteoroloji, yerleşim alanları), uzaysal veritabanı uygulamaları, Web uygulamaları, müşteri ilişkileri yönetimi, sağlık ve biyoloji alanında yapılan araştırmalardır.

Kümeleme analizini gerçekleştirmek için birçok kümeleme metodu geliştirilmiştir. Şekil 4.1'de farklı kümeleme metotlarını bir ağaç şeklinde göstermektedir.

Kümeleme metotlarının ilk ayrımı dışlayan (exclusive) olup olmamalarına göre yapılır [51]. Dışlayıcı kümeleme, küme formlarının ayıran özelliğine sahiptir. Yani nesneler belirli bir kümeye aittir. Bu metotta, bir nesne ölçüm kriterlerine göre iki farklı kümeye atanabiliyor olsa dahi belirli bir kümeye atanmak zorundadır. Çok az miktarda kümeleme analizi metotları nesnelerin birden fazla kümeye atanabilmesine (non-exclusive) izin verir [52].



Şekil 4.1. Kümeleme metotları

En çok kullanılan kümeleme metotları hiyerarşik ve kısımlara ayırmalı kümelemedir. Her bir metodun kendine özgü üstünlükleri, alt tipleri ve kümeleme için kullanılan farklı algoritmaları vardır. Hiyerarşik kümeleme bir sonraki kümeyi bulmak için bir önceki kümeleri kullanır. Kısımlara ayırmalı kümeleme ise nesnelerin daha önceden belirlenmiş k sayıda kümeye ayrılmasıdır. Bu kümeler düz kümeler olup hiyerarşik kümelerde olduğu gibi dendrogram bir yapıları yoktur. Her bir nesne tekil olarak bir kümeye atanmıştır. Hiyerarşik kümeleme nesneler arasındaki uzaklıktan elde edilen bir benzerlik matrisi kullanırken kısımlara ayırmalı kümeleme özellik vektör matrisi kullanır.

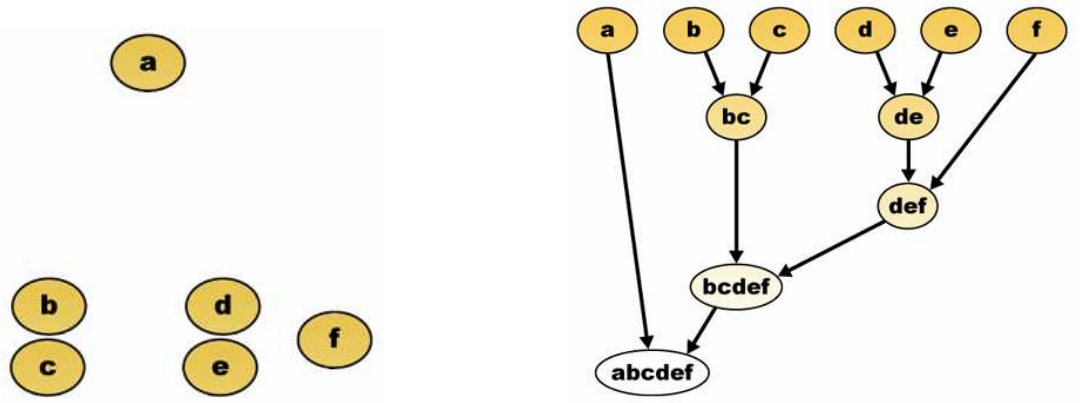
Kümeleme metotlarından önemli olanlar aşağıda açıklanmaktadır.

4.1. Hiyerarşik Kümeleme

Hiyerarşik kümeleme, küçük kümelerden daha büyük kümeler yaparak ilerler (toplayıcı) veya büyük kümeleri bölerek (bölümleyici) ilerler. Bu kümeleme

metodunda önemli olan hangi küçük kümelerin birleştirileceği veya hangi büyük kümenin bölüneceğidir. Algoritmanın sonunda kümeler arasındaki ilişkiyi gösteren ve dendrogram denilen bir küme ağaç yapısı elde edilir. Dendrogramı istenilen seviyelerden keserek veri kümelerinin birbirinden bağımsız gruplara dönüşmesi sağlanabilir.

Hiyerarşik kümeleme için ilk temel adım uzaklık ölçüsünün belirlenmesidir. En yaygın uzaklık ölçüsü olarak Öklit uzaklığı kullanılmaktadır. Öklit uzaklığı, her bir değişkeninin birbirine olan uzaklığının karelerinin toplanarak karekök alınması ile hesaplanır. 2 değişken (2 boyutlu) için bu hesaplama bir dik üçgende hipotenüsün bulunması için kullanılan Pisagor teoremine benzemektedir. Şekil 4.2'de bir toplayıcı hiyerarşik kümeleme örneği verilmiştir.



(a) (b)

Şekil 4.2. Hiyerarşik kümeleme, a) öncesi ve b) sonrası

Şekil 4.2.b'de verilen nesnelere dikkat edilirse birbirine en yakın ilk iki nesne olan {b} ve {c} birleşerek {b c} kümesini oluşturur. Benzer şekilde {d} ve {e} nesneleri de {d e }kümesini oluşturur. Bu elemanlar 1.seviye elemanlarıdır. {d e} kümesi ile {f} nesnesi birleşerek {d e f} kümesini oluşturmaktadır. 2. seviyede sadece bu küme vardır. 1. seviyede elde edilen {b c} kümesi ile 2.seviyede elde edilen {d e f} kümesi birleşerek 3. seviyenin {b c d e f} kümesini oluşturmaktadır. Son olarak {a} nesnesi ile {b c d e f} kümesi birleşerek {a b c d e f} kümesini oluşturur. Son seviye olan 4.seviyede dışarıda hiçbir elemen kalmadığı görülmektedir.

Şekil 4.2.b'de verilen ağacı belirlenen yükseklikten yatay olarak keserek istenilen hassasiyette kümeler elde edilebilir. Örneğin ağaç 2.satırdan sonra kesilirse {a}, {b c}, {d e}, {f} kümeleri elde edilirken, 3.satırdan sonra kesilirse {a}, {b c}, {d e f} kümeleri elde edilecektir.

4.2. Hiyerarşik Kümeleme Algoritmaları

N adet eleman ve N*N lik bir uzaklık matrisinden oluşan bir veri kümesinin temel hiyerarşik kümeleme adımları şu şekildedir [53]:

1. Adım: Her bir eleman bir kümeye atanarak başlanır. Böylece, N eleman olduğuna göre başlangıçta birer elemanı olan N adet küme olacaktır. Ayrıca kümeler arasındaki uzaklıklar da başlangıçta elemanlar arasındaki uzaklığa eşit olacaktır.

2. Adım: Birbirine en yakın (en benzer) küme çifti bulunarak bir kümede birleştirilir. Böylece toplam küme sayısı 1 adet eksilmiş olur.

3. Adım: Yeni küme ile eski kümelerdeki her bir eleman arasındaki uzaklık hesaplanır.

4. Adım: 2. ve 3. adımlar bütün elemanlar N elemanlı tek bir küme içerisinde kümelenene kadar tekrarlanır.

3. Adım farklı metotlarla yapılabilir. Bunlardan en yayın olanları : Tekil Bağlantı, Tam Bağlantı ve Ortalama Bağlantı'dır.

Tekil Bağlantı : Minimum metot olarak da isimlendirilen bu metotta bir küme ile diğer kümeler arasındaki uzaklık bir kümedeki herhangi bir eleman ile diğer kümedeki herhangi bir eleman arasındaki minimum uzaklığa eşittir. Veriyi uzaklık yerine benzerlik olarak değerlendirecek olursak bir küme ile diğer küme arasındaki

benzerlik bir kümedeki herhangi bir eleman ile diğer kümedeki herhangi bir eleman arasındaki en çok benzerliğe eşittir. *A* ve *B* kümesi için uzaklık :

$$\min\{d(x,y) : x \in A, y \in B\} \tag{4.1}$$

Tam Bağlantı : Çap ve Maksimum metot olarak ta isimlendirilen bu metotta bir küme ile diğer küme arasındaki uzaklık kümenin herhangi bir elemanı ile diğer kümenin herhangi bir elemanı arasındaki en büyük uzaklığa eşittir. *A* ve *B* kümesi için uzaklık:

$$\max\{d(x,y) : x \in A, y \in B\} \tag{4.2}$$

Ortalama Bağlantı : Bu metotta bir küme ile diğer küme arasındaki uzaklık bir kümenin herhangi bir elemanı ile diğer kümenin herhangi bir elemanı arasındaki uzaklığın ortalamasına eşittir.

Ortalama Bağlantı metodunun bir varyasyonu olan medyan uzaklık ortalama uzaklığa göre daha verimli sonuçlar üretebilmiştir [54]. *A* ve *B* kümesi için uzaklık :

$$\frac{1}{card(A)card(B)} \sum_{x \in A} \sum_{y \in B} d(x,y) \tag{4.3}$$

Kümeleme işlemini durdurmak için genellikle 2 farklı yol tercih edilmektedir:

1) *Uzaklık kriteri ile durdurma:* Her bir kümeleme bir önceki kümelemeye göre daha büyük uzaklıklarla gerçekleşir. Dolayısıyla kümelerin birleştirme işlemi belirli bir uzaklık kriterinin gerçekleşmesine kadar sürdürülebilir.
2) *Küme sayısı kriteri ile durdurma:* Algoritma boyunca üretilen küme sayısı belirli bir kritere ulaştığında kümeleme işlemi durdurulabilir.

Bu çalışmada Tekil Bağlantı kullanıldığı için bu algoritma üzerinde daha detaylı açıklama yapmakta fayda görüyoruz.

Algoritma, toplayıcı (agglomerative) şema yapısında olduğu için eski kümeler yeni kümelerde birleştirilirken, birleştirilen kümelere uzaklık matrisinde satır ve sütunlardan silinmektedir.

$N * N$ boyutunda bir uzaklık matrisi için $D = [d(i, j)]$ dir. Kümeler $0,1,...,(n-1)$ şeklinde sıralı numaralarla gösterilirken k. kümenin seviyesini göstermek için $L(k)$ kullanılır. Sıra numarası m olan bir küme (m) ile gösterilir. (r) ve (s) kümeleri arasındaki uzaklık $d[(r),(s)]$ ile gösterilir. Bu durumda algoritma aşağıdaki adımlardan oluşur :

1. Adım: $L(0) = 0$ ve $m = 0$ olan ayrık bir küme ile başla.

2. Adım: Güncel kümeleme yapısı içerisinde birbirine en çok benzer (r) ve (s) küme çiftini bul.

$$d[(r),(s)] = \min(d[(i),(j)]) \tag{4.4}$$

3. Adım: Sıra numarasını arttır. $m = m + 1$. (r) ve (s) kümelerini yeni bir kümede birleştir.

Bu kümenin seviyesi :

$$L(m) = d[(i),(j)] \tag{4.5}$$

4. Adım: Uzaklık matrisini güncelle. Bunun için (r) ve (s) kümelerinin bulunduğu satır ve sütunları sil. Ayrıca birleştirilen yeni kümeyi matrise ekle. Yeni küme (r,s) olarak gösterilir ve eski küme (k) ile olan uzaklığı şu şekilde bulunur:

$$d[(k),(r,s)] = \min(d[(k),(r)], d[(k),(s)]) \tag{4.6}$$

5. Adım: Eğer tüm elemanlar bir kümede birleşti ise dur. Aksi halde 2.adımdan tekrar başla.

4.3. Hiyerarşik Sınıflandırma Örneği

Bu bölümde dokümanların hiyerarşik sınıflandırmasını daha iyi anlayabilmek için İç Anadolu Bölgemizdeki şehirlerin kilometre olarak birbirlerine olan uzaklıklarının hiyerarşik sınıflandırma ile sınıflandırılması adım adım anlatılacaktır. Şekil 4.3'de İç Anadolu Bölgemizin haritası ve bu bölgedeki şehirler verilmiştir. Kümelerin birbirine olan uzaklıkları için Tekil Bağlantı (Minimum uzaklık) kullanılmıştır.

Şekil 4.3. İç Anadolu Bölgesi

Şehirlerin kısaltmaları şu şekilde yapılmıştır.

Çizelge 4.1. Şehirler ve Kısaltmaları

1	ANK	Ankara
2	AKS	Aksaray
3	ÇAN	Çankırı
4	ESK	Eskişehir
5	KAR	Karaman
6	KAY	Kayseri
7	KON	Konya
8	NEV	Nevşehir
9	SİV	Sivas
10	YOZ	Yozgat

Uzaklık matrisi başlangıçta şu şekildedir (tüm kümeler için L=0) :

Çizelge 4.2. Şehirler arası uzaklıklar

	ANK	**AKS**	**ÇAN**	**ESK**	**KAR**	**KAY**	**KON**	**NEV**	**SİV**	**YOZ**
ANK	0	225	131	233	363	319	258	275	443	219
AKS	225	0	311	443	211	179	148	75	373	221
ÇAN	131	311	0	364	494	348	389	304	442	248
ESK	233	443	364	0	442	542	335	498	376	452
KAR	363	211	494	442	0	317	113	269	511	432
KAY	319	179	348	542	317	0	327	104	194	175
KON	258	148	389	335	113	327	0	223	521	369
NEV	275	75	304	498	269	104	223	0	298	189
SİV	443	373	442	376	511	194	521	298	0	224
YOZ	219	221	248	452	432	175	369	189	224	0

Bu matriste şehirler ilk satır ve kolona yerleştirilmiştir. Her bir şehrin diğer şehirlere uzaklıkları ise matris elemanları olarak ilgili pozisyona yerleştirilmiştir. Örneğin Ankara ile Eskişehir arası 233 km’dir. Her bir şehrin kendisi ile olan uzaklığı 0 olarak belirtilmiştir. Ancak algoritma boyunca minimum uzaklık olarak 0 olan değerler dikkate alınmamaktadır. Matrise dikkat edilirse bitişik (adjacent) olduğu görülecektir.

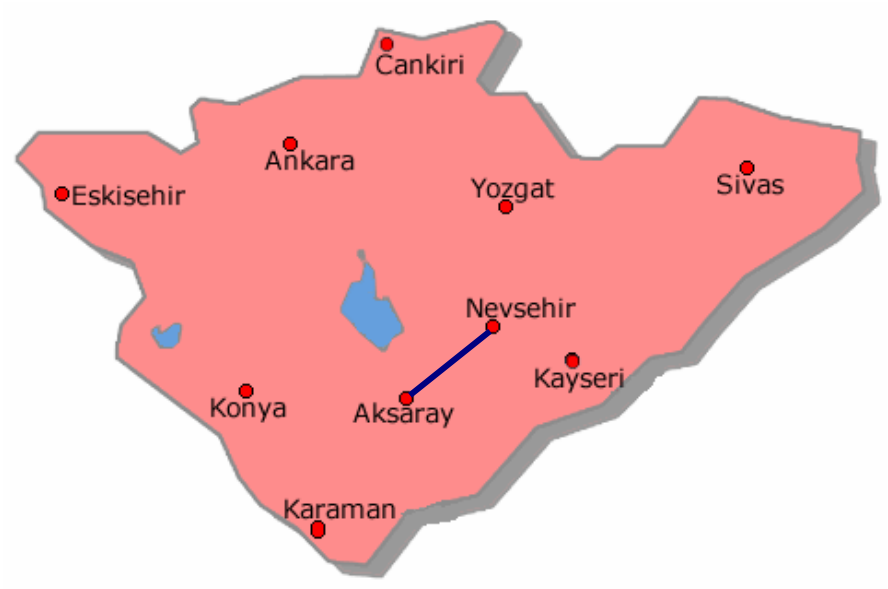
Birbirine en yakın şehir çifti 75 km ile AKS ve NEV dir. Bu iki şehir “AKS/NEV” isimli yeni bir kümede birleştirilir. Yeni kümenin seviyesi L(AKS/NEV) = 75 ve yeni sıra numarası m = 1’ dir.

Daha sonra yeni elde edilen kümenin diğer kümelere olan uzaklığı hesaplanır. Tekil bağlantı kuralına göre yeni kümedeki her bir elemanın diğer küme elemanlarına olan uzaklığı yeni kümedeki elamanlardan uzaklığı en küçük olana eşittir. Bu durumda “AKS/NEV” kümesinin “ANK” a uzaklığı 225 olarak alınır ve bu şekilde devam edilir.

“AKS” ve “NEV” birleştirildikten sonra (Şekil 4.4) uzaklık matrisi şu şekilde olacaktır :

Çizelge 4.3. İlk birleştirmeden sonraki uzaklık matrisi

	ANK	**ÇAN**	**ESK**	**KAR**	**KAY**	**KON**	**SİV**	**YOZ**	**AKS/NEV**
ANK	0	131	233	363	319	258	443	219	225
ÇAN	131	0	364	494	348	389	442	248	304
ESK	233	364	0	442	542	335	376	452	443
KAR	363	494	442	0	317	113	511	432	211
KAY	319	348	542	317	0	327	194	175	104
KON	258	389	335	113	327	0	521	369	148
SİV	443	442	376	511	194	521	0	224	298
YOZ	219	248	452	432	175	369	224	0	189
AKS/NEV	225	304	443	211	104	148	298	189	0



Şekil 4.4. İlk birleştirmeden sonra

min d(i,j) = d(AKS/NEV,KAY) = 104 => AKS/NEV ve KAY yeni bir kümede birleştirilir (Şekil 4.5). Yeni küme AKS/NEV/KAY olarak adlandırılır ve L(AKS/NEV,KAY)=104 ve m=2 dir. Bu durumda yeni uzaklık matrisi Çizelge 4.4'de gösterilmiştir.

Çizelge 4.4. İkinci birleştirmeden sonraki uzaklık matrisi

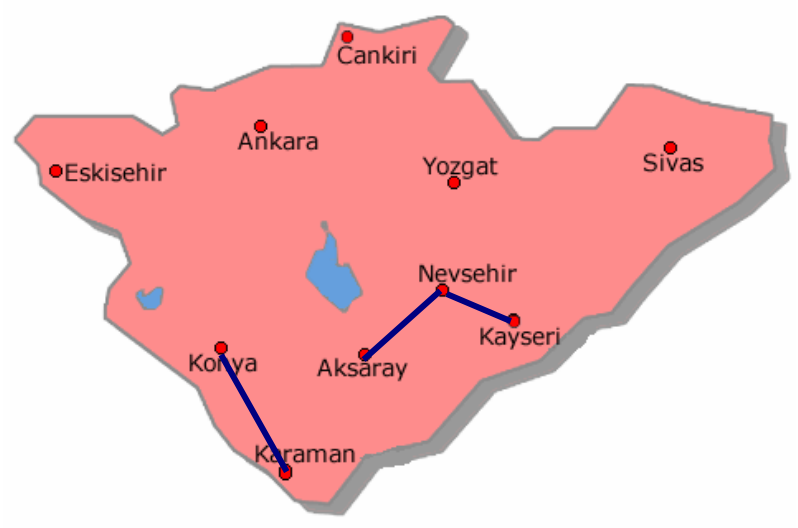
	ANK	**ÇAN**	**ESK**	**KAR**	**KON**	**SİV**	**YOZ**	**AKS/NEV/KAY**
ANK	0	131	233	363	258	443	219	225
ÇAN	131	0	364	494	389	442	248	304
ESK	233	364	0	442	335	376	452	443
KAR	363	494	442	0	113	511	432	211
KON	258	389	335	113	0	521	369	248
SİV	443	442	376	511	521	0	224	194
YOZ	219	248	452	432	369	224	0	175
AKS/NEV/KAY	225	304	443	211	248	194	175	0

Şekil 4.5. İkinci birleştirmeden sonra

min d(i,j) = d(KAR,KON) = 113 => KAR ve KON yeni bir kümede birleştirilir (Şekil 4.6). Yeni küme KAR/KON olarak adlandırılır ve L(KAR,KON)=113 ve m=3 dir. Bu durumda yeni uzaklık matrisi Çizelge 4.5'te gösterilmiştir. Burada Konya ile Karaman'ın birleştirildiği görülmektedir.

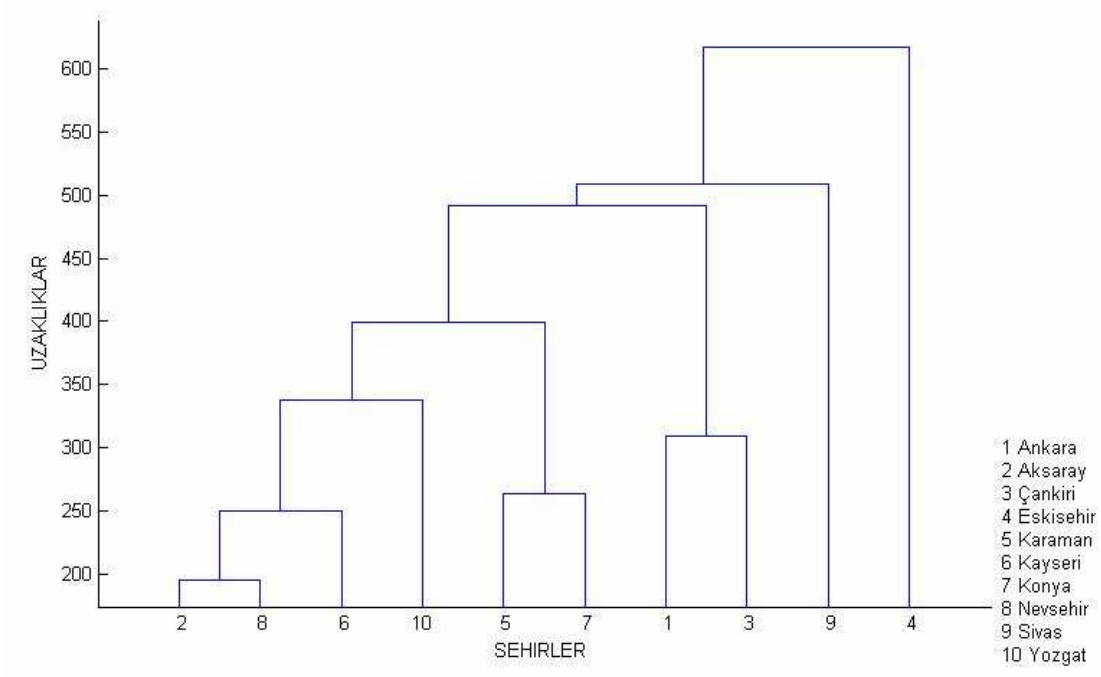
Çizelge 4.5. Üçüncü birleştirmeden sonraki uzaklık matrisi

	ANK	**ÇAN**	**ESK**	**SİV**	**YOZ**	**AKS/NEV/KAY**	**KAR/KON**
ANK	0	131	233	443	219	225	258
ÇAN	131	0	364	442	248	304	389
ESK	233	364	0	376	452	443	335
SİV	443	442	376	0	224	194	511
YOZ	219	248	452	224	0	175	369
AKS/NEV/KAY	225	304	443	194	175	0	211
KAR/KON	258	389	335	511	369	211	0



Şekil 4.6. Üçüncü birleştirmeden sonra

Bu işlemlere küme sayısı 1 tane oluncaya kadar devam edilir. Sonuçta elde edilen hiyerarşi ağacı (dendrogram) Şekil 4.7'deki gibidir.



Şekil 4.7. Şehirlerin birbirlerine uzaklıklarına göre hiyerarşik sınıflandırması

4.4. Kısımlara Ayırmalı Kümeleme

Kısımlara ayırmalı kümeleme ise verileri doğrudan ayrık kümelere ayrıştırmaya çalışır. Bu metotta kümeleme algoritması verinin lokal yapısı üzerine odaklanır. Algoritmalar, tipik olarak her bir kümedeki veri örneklerinin arasındaki benzersizlikleri minimuma indirmeye çalışırken farklı veri kümeleri arasındaki benzersizlikleri maksimuma çıkarmaya çalışır.

Yaygın olarak kullanılan bir kısımlara ayırmalı kümeleme metodu olan K-ortalama kümeleme, Kendinden düzenlenen haritalar algoritmasıyla yakınlık gösterir. K-ortalamanın kendinden düzenlenen haritalar ile çok benzerliğinden dolayı burada bahsetmekte yarar görüyoruz.

4.5. K-ortalama Kümeleme

K-ortalama algoritmasının işlem basamakları şöyledir [55]:

1. Adım: Başlangıç küme merkezleri belirlenir. Bunun için iki farklı yol vardır. Birinci yol nesneler arasından küme sayısı olan k adet rastgele nokta seçilmesidir. İkinci yol ise merkez noktaların tüm nesnelerin ortalaması alınarak belirlenmesidir,

2. Adım: Her nesnenin seçilen merkez noktalara olan uzaklığı hesaplanır. Elde edilen sonuçlara göre tüm nesneler k adet kümeden kendilerine en yakın olan kümeye yerleştirilir,

3. Adım: Oluşan kümelerin yeni merkez noktaları o kümedeki tüm nesnelerin ortalama değeri ile değiştirilir,

4. Adım: Merkez noktalar değişmeyene kadar 2. ve 3. adımlar tekrarlanır.

K-ortalama algoritmasında her bir nesnenin merkez noktalara uzaklığını hesaplamak için kullanılan dört farklı formül aşağıda açıklanmaktadır :

Öklit Uzaklığı - Öklit Uzaklığının Karesi : Öklit uzaklığı ve Öklit uzaklığının karesi formülleri ile standartlaştırılmış verilerle değil, işlenmemiş verilerle hesaplama yapılır. Öklit uzaklıkları kümeleme analizine sıra dışı olabilecek yeni nesnelerin eklenmesinden etkilenmezler. Ancak boyutlar arasındaki ölçek farklılıkları Öklit uzaklıklarını önemli ölçüde etkilemektedir. Öklit uzaklık formülü en yaygın olarak kullanılan uzaklık hesaplama formülüdür. Öklit uzaklık ve Öklit uzaklığının karesi formülleri aşağıda görülmektedir.

Öklit uzaklık formülü :

$$\text{distance}(x, y) = \left\{ \sum_i (x_i - y_i)^2 \right\}^{1/2} \tag{4.7}$$

Öklit uzaklığının karesi formülü :

$$\text{distance}(x, y) = \sum_i (x_i - y_i)^2 \tag{4.8}$$

City-block (Manhattan) Uzaklık Formülü : Manhattan uzaklığı boyutlar arasındaki ortalama farka eşittir. Bu ölçüt kullanıldığında farkın karesi alınmadığı için sıra dışılıkların etkisi azalır. Manhattan uzaklığının formülü aşağıda görülmektedir.

$$\text{distance}(x, y) = \sum_i |x_i - y_i| \tag{4.9}$$

Chebychev Uzaklığı : Chebychev uzaklığı iki nesne arasındaki mutlak maksimum uzaklığa eşittir. Chebychev uzaklığının formülü aşağıda görülmektedir.

$$\text{distance}(x, y) = \max |x_i - y_i| \tag{4.10}$$

distance(x, y): x ve y noktaları arasındaki uzaklık, hata parametresidir. *x, y :* aralarındaki uzaklık hesaplanan nesneleri uzayda temsil eden noktalardır.

K-ortalama algoritmasının en büyük eksikliği k değerini tespit edememesidir. Bu nedenle başarılı bir kümeleme elde etmek için farklı k değerleri için deneme yanılma yönteminin uygulanması gerekmektedir [56].

5. YSA VE DOKÜMAN SINIFLANDIRMA

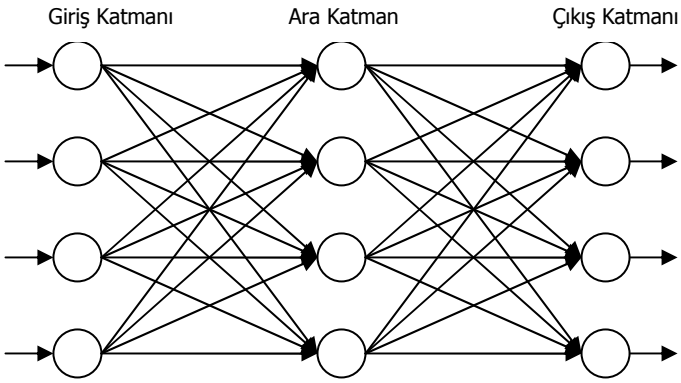
5.1. Yapay Sinir Ağları (YSA)

İnsan beyni, bilinen en gizemli ve karmaşık hesaplayıcıdır. Yapay sinir ağları, insan beyninin işleyişini taklit ederek yeni sistem oluşturulmaya çalışılan yaklaşımlardır. İstinasız tüm YSA yapılarının esin kaynağı biyolojik sinir ağlarının işleyiş yöntemidir [15]. Taklit edilmeye çalışılan 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, YSA'lar, 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.

Biyolojik sistemlerde öğrenme, nöronlar arasındaki sinaptik bağlantıların ayarlanması ile olur. Yani, insanlar doğumlarından itibaren bir yaşayarak öğrenme süreci içerisine girerler. Bu süreç içinde beyin sürekli bir gelişme göstermektedir. Yaşayıp tecrübe ettikçe sinaptik bağlantılar ayarlanır ve hatta yeni bağlantılar oluşur. Bu sayede öğrenme gerçekleşir. Bu durum YSA için de geçerlidir. Öğrenme, eğitme yoluyla örnekler kullanarak olur; başka bir deyişle, gerçekleşme girdi/çıktı verilerinin işlenmesiyle, yani eğitme algoritmasının bu verileri kullanarak bağlantı ağırlıklarını bir yakınsama sağlanana kadar, tekrar tekrar ayarlamasıyla olur. Ağırlıkların değişimi öğrenmeyi ifade eder. YSA'da ağırlık değişimi yok ise öğrenme işlemi de durmuştur [15].

YSA'lar, ağırlıklandırılmış şekilde birbirlerine bağlanmış birçok işlem elemanlarından oluşan matematiksel sistemlerdir. Bu elemanlar farklı formda ifade edilebilen nümerik verileri taşıyan "bağlantılar" veya "ağırlıklar" ile birbirlerine bağlıdırlar [15]. Bu işlem elemanı, diğer işlem elemanlarından sinyalleri alır; bunları

birleştirir, dönüştürür ve sayısal bir sonuç ortaya çıkartır. Genelde, işlem elemanları kabaca gerçek nöronlara karşılık gelirler ve bir ağ içinde birbirlerine bağlanırlar; bu yapı da sinir ağlarını oluşturmaktadır (Şekil 5.1).



Şekil 5.1. Çok katmanlı ileri beslemeli bir yapay sinir ağı modeli

YSA'lar, geleneksel işlemcilerden farklı şekilde işlem yapmaktadırlar. Geleneksel işlemcilerde, tek bir merkezi işlem elemanı her hareketi sırasıyla gerçekleştirir. YSA'lar ise her biri büyük bir problemin bir parçası ile ilgilenen, çok sayıda basit işlem elemanlarından oluşmaktadır. En basit şekilde, bir işlem elemanı, bir girdiyi bir ağırlık kümesi ile ağırlıklandırır, doğrusal ve/veya doğrusal olmayan bir şekilde dönüşümünü sağlar ve bir çıktı değeri oluşturur. İlk bakışta, işlem elemanlarının çalışma şekli yanıltıcı şekilde basittir. Yapay sinir ağlarının hesaplamanın gücü, toplam işlem yükünü paylaşan işlem elemanlarının birbirleri arasındaki yoğun bağlantı yapısından gelmektedir.

Çoğu YSA'da, benzer karakteristiğe sahip işlemci elemanları katmanlar halinde yapılandırılırlar ve transfer fonksiyonları eş zamanlı olarak çalıştırılırlar. Hemen hemen tüm ağlar şu kısımlardan oluşur: girişler, ağırlıklar, toplama fonksiyonu , transfer fonksiyonu ve çıkış.

YSA'nın 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.

YSA üzerinde yapılan birçok çalışmaya Kohonen, Hopfield, Grossberg, Cohen, Anderson, Rosenfeld, DeSieno, Zurada, Hecht-Nieken, Hertz, Pao, Minsky, Haykin, Papert, Amari, Hinton, Sejnowski, Widrow, Albus, Carpenter, Elman, Jordan, Hebb, Fukishama, Kosko, Littmann, Oja, Rumelhart, Spect, Williams, Rosenblatt, McClelland gibi bir çok bilim adamının katkılarıyla bir çok yapı geliştirilmiş ve bugün farklı problemlere başarıyla uygulanmaktadır [15]. Bu tezin konusu olduğu için Kohonen'in geliştirdiği YSA tekniğinden daha detaylı söz etmekte fayda var.

Kohonen ağı, bir giriş tabakası ve bir de çıkış tabakası olmak üzere iki tabakadan oluşur. Bu ağ Şekil 5.2'de gösterilmiştir. Çıkış tabakasındaki işlemci elemanlar genellikle düzenli iki boyutlu aralıklar olarak düzenlenir. Çıkıştaki her işlem elamanı, bütün giriş işlemci elemanlarına bağlıdır. Bağlantıların ağırlıkları verilen çıkış işlemci elemanı ile ilgili olan referans vektörünün elemanlarını oluşturur. İyi öğrenmiş bir Kohonen ağında birbirine yakın çıkış işlemci elemanlarının referans vektörleri vardır. Öğrenmeden sonra bir etiketleme işlemine başlanır [15].

5.2. YSA ile Doküman Sınıflandırma

Doküman sınıflandırma, sisteme giren dokümanla çıkan sınıf arasında bir eşleme problemi olarak düşünülebilir. Doküman sınıflandırma ile ilgili birçok çalışma yapılmış ve değişik teknikler kullanılmaya çalışılmıştır. Bu teknikler arasında birtakım performans farklılıkları oluşmuştur. Yang doküman sınıflandırmayla ilgili birçok farklı yaklaşım uygulamıştır. Bunlar arasında kth Nearest Neighbors (kNN), Linear Least Square Fit (LLSF), destek vektör makinesi (SVM), Naive Bayes (NB) ve Neural Networks (NNet) sayılabilir [38].

Yapay Sinir Ağları, doğrusal olmayan eşlemeleri bir eğitim veri kümesinden öğrenebilmektedir. Bunun için genellikle danışmanlı öğrenme tercih edilmekte ve geri yayılım (backpropagation) tekniği kullanılmaktadır. Geri yayılım tekniğinin en büyük problemi ise ağın önce doğru çıktılarla eğitilmesi gerekliliğidir. Bu eğitim için

ağa bir miktar giriş verisi verilmekte ve birçok iterasyondan sonra ağın doğru sonuçları verebilir duruma gelmesi beklenmektedir.

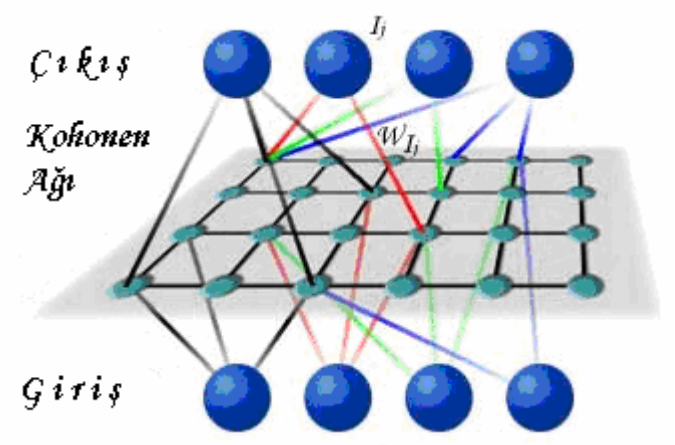
Doküman sınıflandırmada 2 farklı öğrenme yaklaşımı vardır:

– Danışmanlı doküman sınıflandırma: Dokümanı doğru olarak sınıflandırmak için bir dış etkene (örneğin insana) ihtiyaç duyulan sınıflandırma şeklidir.

– Danışmansız doküman sınıflandırma: Harici hiçbir bilgiye ihtiyaç olmadan yapılan doküman sınıflandırma şeklidir.

Doküman sınıflandırma konusunda danışmanlı öğrenme, giriş verisinin çok fazla ve her bir düğümün bir vektör gösterimiyle olması nedeniyle kullanılmamaktadır. Bunun yerine danışmansız öğrenmeli ağların en yaygını olan kendinden düzenlenen haritalar (SOM) kullanılmaktadır. Bu tür ağlarda doğru cevabın ne olduğunu bilmemize gerek yoktur. Tek yapmamız gereken ağa giriş verilerini vermek ve ağın kendi kendini eğitmesini sağlamaktır.

Kendinden düzenlenen haritalar, Finlandiyalı bir bilim adamı olan Teuvo Kohonen tarafından geliştirilmiştir [57]. Bu nedenle Kohonen ağı olarak da adlandırılmaktadır (Şekil 5.2).



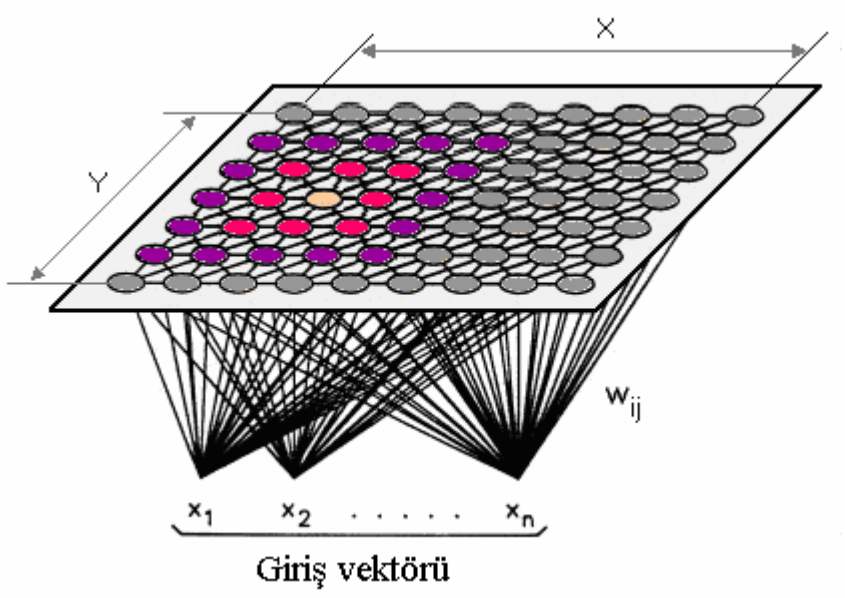
Şekil 5.2. Kohonen Ağı

Kendinden düzenlenen haritalar, doğrusal olmayan bir projeksiyonla, çok yüksek boyutlu verileri az boyutlu (genellikle 2 boyutlu) bir forma getirebilmektedir. 1 boyutlu veri yeterince özellik bilgisi taşıyamaz. 3 boyutlu veriler ise birçok avantajlarına karşın verinin anlaşılabilirliği ve görselliği açısında karmaşık bir yapıdadırlar. Dolayısıyla bu çalışmamızda veriyi 2 boyuta indirgeyeceğiz. Bu işlem sırasında verinin küme yapısı korunmaktadır. Kendinden düzenlenen haritalar için sonuç olarak verinin benzerlik grafiğini verdiğini söyleyebiliriz [57].

6. KENDİNDEN DÜZENLENEN HARİTALAR

6.1. Ağ Yapısı

Kohonen ağının topolojik yapısı Şekil 6.1'de görülmektedir. Bu ağ 2 katman ve işlem elemanlarından oluşur: bir giriş katmanı ve bir de çıkış katmanı. Çıkış katmanı Şekil 6.1'de görüldüğü gibi 2 boyutlu yapıdadır. Giriş katmanı ise, dağıtım katmanı gibi davranır.



Şekil 6.1. Kendinden düzenlenen haritalar gösterimi

İşlem elemanı sayısı bir giriş işlem elemanıyla ilişkilendirilmiş nitelik sayısına eşittir. Çıkış katmanındaki her bir işlem elemanı eşit sayıda niteliğe sahiptir. Başka bir ifadeyle ağırlık vektörlerinin uzunluğu her işlem elemanı için eşittir. Çıkış işlem elemanları rastgele başlangıç değerleriyle üretilir.

6.2. Önişlemler

Bir doküman kümesine kendinden düzenlenen haritalar uygulanmadan önce metin bilgisi olmayan bazı verilerin temizlenmesi gerekir. Temizlenmesi gereken verilerden bazıları:

- ASCII çizim karakterleri : “|”, “-“, ”/”, ”\”, ”_”, vb.
- Çeşitli işaretler : “:”,”?”,”*”,”-“,”+”,”(“,”)”,”[“,”]”, vb.
- Otomatik gelen imza vb bilgiler : “Sayın”, “iyi günler”, “saygılarımla”, “iyi çalışmalar”, vb.
- Sayısal ifadeler : “0”,”1”,”2”,”3”,”4”,”5”,”6”,”7”,”8”,”9” rakamları ve bu rakamlarla ifade edilen tüm sayılar. Örneğin 1970, 2006, 1000011, vb.

Doküman sınıflandırma için kullanılan veriler kelime sayıları dikkate alındığında çok yüksek boyutlara ulaşabilmektedir. Yüksek boyutlu veriler ise ağın eğitim süresinin önemli oranda uzun olmasına neden olmaktadır. Bu süreyi kısaltmak için çok az karşılaşılan terimler veya çok fazla tekrar edilen terimler de metinlerden temizlenir. Bu işlem için literatürde kabul edilen oran %10’dan az ve %90’dan çok geçen terimlerin temizlenmesi şeklindedir.

6.3. Durak Kelimeleri

Her dilde cümlelerin anlam bütünlüğü açısından bazı ara kelimelere ihtiyaç duyulur. Bu kelimelerin dokümanın sınıflandırmasına etkisi olmamakla birlikte oluşturulan vektörün boyutunu ciddi oranda etkileyeceği açıktır. Dolayısıyla bu kelimelerin başlangıçta dokümanlardan temizlenmesi gerekir.

Türkçe’deki durak kelimeleri için geliştirilmiş hazır bir liste bulunmamaktadır. Bu çalışmada http://www.ranks.nl/stopwords/turkish.html adresindeki durak kelimeleri alınmış ve üzerine bazı eklemeler yapılmıştır. Hazırlanan bu liste, bu çalışma için yeterli olmaktadır (Ek.2).

6.4. Kelime Köklerinin Bulunması

Birçok dilde olduğu gibi Türkçe’de de kök kelimelerden anlamsal olarak farklı birçok yeni kelime türetilebilmektedir. Kök bulma konusunda her dilin kendine göre özellikleri vardır. Dolayısıyla bir dilde yapılan kök bulma yöntemi birebir başka bir

dile uygulanamaz. Türkçe için kelime kökü bulma konusunda çeşitli çalışmalar yapılmıştır.

Özellikle Web arama motorları için kök bulma işi önemlidir. “Kitaplar” kelimesini arayan bir kişinin “kitap” kelimesi geçen sayfalarla da ilgileneceği açıktır. Dokümanlardaki kelimelerin doğru bir şekilde indekslenmesi için köklerin bulunması ve bu köklerle eğitim ve sınıflandırma işlemlerinin yapılması, arama işleminin başarısını artırır.

6.5. Kendinden Düzenlenen Haritalar Algoritması

Kendinden düzenlenen haritalar, bir dikdörtgen biçiminde 2 boyutlu düğümlerden oluşan bir ızgara ile gösterilir. Örneğin bir harita 10 sütun ve 12 satırdan oluşuyorsa, bu haritada 120 adet düğüm bulunur. Her bir düğüm bir ağırlık vektörü ile ilişkilendirilmiştir. Bu vektörler veri fihristinde bulunan terimlere karşılık gelir. Her bir vektör elemanının 0 ile 1 arasında olan ilk değeri genellikle rastgele atanır.

Kendinden düzenlenen haritalar, danışmansız öğrenen yapay sinir ağı modellerinden en önemli ve en yaygın kullanılanıdır. Bu model, birim (unit) denilen işlem elemanlarından oluşur. Her bir i birimine, n boyutlu bir ağırlık vektörü, m_i atanmıştır, $\mathrm{m_i} \in \Re^n$. Ağırlık vektörlerinin boyutu ile ağa giriş desenlerinin boyutları birbirine eşittir.

Kendinden düzenlenen haritaların eğitilmesi işlemi kısaca ağa verilen giriş deseni ile ağırlık vektörlerinin uyarlanması şeklinde tanımlanabilir. Her bir eğitim iterasyonu t, rastgele bir giriş deseni $x(t)$ seçimi ile başlar. Seçilen giriş deseni ağa alınır ve her bir birimin, $m_i(t)$, bu desen ile etkileşimi bulunur. Bu etkileşimi bulmak için genellikle her bir ağırlık vektörünün giriş desenine olan Öklit uzaklığı hesaplanır. Bu hesaplamayı YSA’da aktivasyon fonksiyonu olarak düşünebiliriz. Bu hesaplamalar sonucunda en az etkileşime sahip olan birim ilgili eğitim iterasyonunun kazanan

birimi, *c* olarak kabul edilir (Eş. 6.1). Literatürde kazanan birim için BMU (Best Matching Unit) kısaltması çoğunlukla tercih edilmektedir [58-60].

$$c : m_c(t) = \min_i \left\| x(t) - m_i(t) \right\| \tag{6.1}$$

Daha sonra, kazanan birim ve çevresindeki bazı birimlerin ağırlık vektörleri güncellenmektedir. Bu güncelleme (uyarlama) işlemi, giriş deseninin ilgili bileşenleri ile ağırlık vektörünün farkına bir gradyan azaltımı uygulanmasıyla gerçekleştirilir (Eş. 6.2).

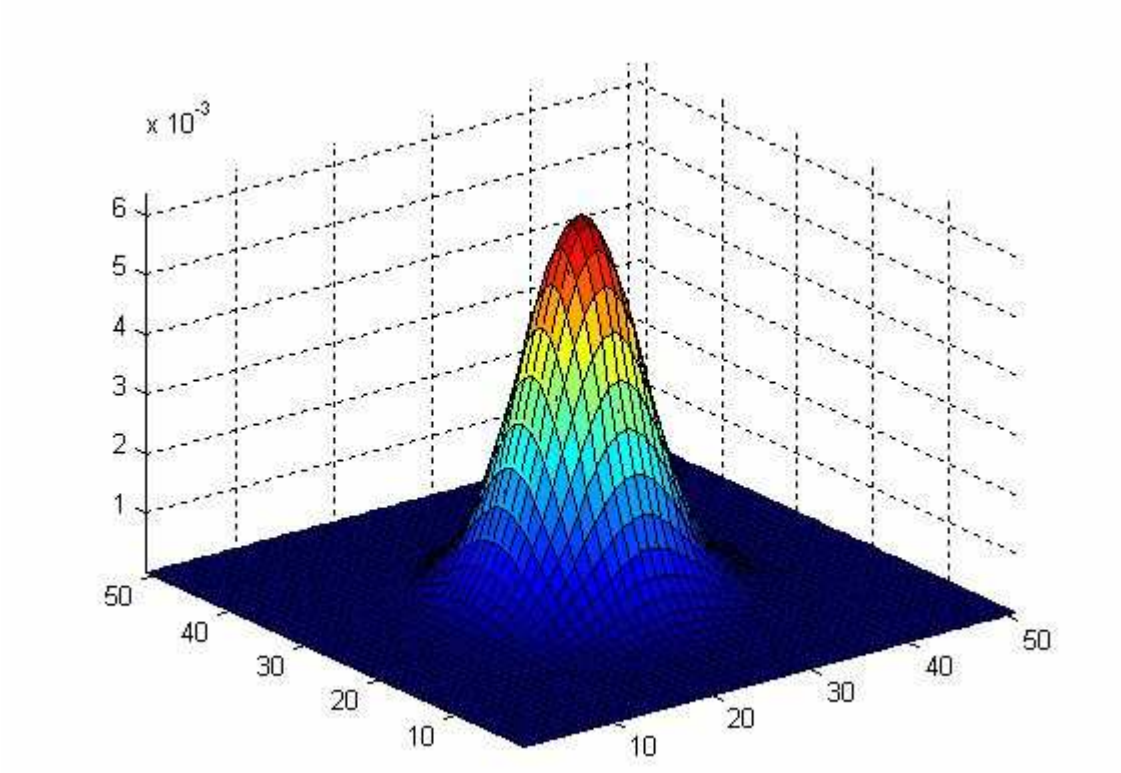
$$m_i(t+1) = m_i(t) + \alpha(t).h_{ci}(t).\left[x(t) - m_i(t)\right] \tag{6.2}$$

Bu fonksiyon iteratif bir fonksiyon olup, t iterasyon adımını ifade eder. Genel olarak formül bir düğüm için yeni ağırlığı, $m_i(t+1)$, mevcut ağırlığın, $m_i(t)$, bir fonksiyonu olarak göstermektedir. Formüldeki *x(t)*, *t*. iterasyondaki giriş desenini göstermektedir.

Eğitim boyunca, güncellenen birimlerin ağırlık vektörleri giriş desenine bir miktar yaklaştırılmış olmaktadır. Ağırlık vektörlerinin değişim hızı öğrenme oranı denilen $\alpha(t)$ ile belirlenir ve bu oran zamanla azaltılarak en sonunda 0 yapılır.

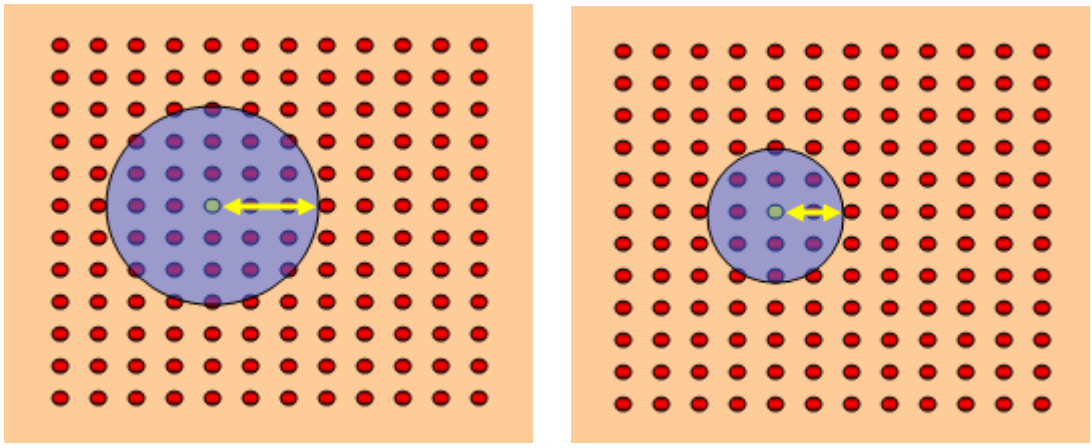
Etkileşime dahil edilecek birimler, komşuluk fonksiyonu denilen h_{ci} ile belirlenir. Etkileşime dahil edilen bu birimlerin sayısı da zamanla azalır ve eğitim işleminin sonuna doğru sadece kazanan birim etkileşime girer. Tipik olarak, komşuluk fonksiyonu tek tepeli bir fonksiyon olup kazanan birimin bulunduğu yerin çevresinde simetrik ve kazanandan uzaklaştıkça tekdüze azalan bir yapıdadır. Komşuluk fonksiyonunu modellemek için bir Gauss fonksiyonu kullanılabilir (Eş. 6.3). Şekil 6.2'de tek tepeli bir Gauss fonksiyonunun grafiği 3-boyutlu olarak gösterilmiştir.

$$h_{ci}(t) = \exp\left(-\frac{\left\|r_c - r_i\right\|^2}{2\sigma^2(t)}\right) \tag{6.3}$$



Şekil 6.2. Tek tepeli bir Gauss fonksiyonunun grafiği

Bu eşitlikte, r_i, *i* biriminin ızgaradaki yerini gösteren 2 boyutlu bir vektördür. Eşitlikteki $\|r_c - r_i\|$ ise aktif eğitim iterasyonundaki kazanan birim *c* ile çıkış uzayındaki *i* birimi arasındaki uzaklığı göstermektedir. Yapılan çalışmalar eğitimin başında çıkış uzayının geniş bir alanının etkileşime dahil olduğunu göstermektedir. Etkileşime giren birimlerin uzaysal genişliği zamanla azalmaktadır. Bu strateji ile başlangıçta büyük kümelerin (cluster) oluşması ve eğitimin sonuna doğru çok daha küçük tanecikli ayrımların oluşması sağlanmış olmaktadır (Şekil 6.3) [60]. Etkileşimin uzaysal genişliği, zamanla değişen σ parametresi ile belirlenir.



(a) (b)

Şekil 6.3. Komşuluk yarıçapı zamanla küçülür.
(a) Başlangıçtaki yarıçap (b) Bir adım sonraki yarıçap

Ağırlık vektörlerinin hareketiyle giriş deseni ve ağırlık vektörü arasındaki Öklit uzaklığı sürekli azalır ve sonuçta ağırlık vektörleri giriş desenine çok benzer hale gelir. Böylece ilgili birimin sonraki iterasyonlarda kazanma olasılığı artmaktadır. Sadece kazanan birimin değil bu birime komşu diğer birimlerin de kazananla birlikte etkileşime dahil edilmesi neticesinde birbirine benzer desenlerin uzaysal kümelenmesi sağlanmaktadır. Böylece n boyutlu bir giriş uzayında bulunan giriş desenlerinden benzer olanları kendinden düzenlenen haritalar ile 2 boyutlu çıkış uzayında komşu olmaktadır. Çıkış uzayında benzer olan desenlerin coğrafik olarak birbirine yakın olacak şekilde kümelenmesi kendinden düzenlenen haritaların eğitim süreci ile sağlanmış olmaktadır.

Kendinden düzenlenen haritalar algoritması :

1. Adım: Çıkış işlem elemanlarının ilk değerlerini belirle,
2. Adım: Eğitim kümesinden rastgele bir girişi seç,
3. Adım: Kazanan çıkış işlem elemanını belirle (Seçilen giriş desenine en yakın ağırlık vektörüne sahip işlem elemanıdır. Ağırlık vektörü ile giriş vektörü arasındaki uzaklık için genellikle Öklit uzaklığı kullanılır.),
4. Adım: Kazanan işlem elemanının ve çevresindeki komşularının ağırlık vektörlerini güncelleştir. Bu güncelleme ile ağırlık vektörleri giriş vektörüne yaklaştırılır. Bu

yaklaştırma kazanan işlem elemanı için en fazla ve bu işlem elemanından uzaklaştıkça daha azdır. Öğrenme ilerledikçe komşuların sayısı azalmakta ve öğrenme sonunda sadece kazanan işlem elemanının ağırlık vektörü ayarlanmaktadır.

5. Adım: İterasyon sayısınca 2. adımdan itibaren tekrarla.

Öğrenmeyi başarmış bir kendinden düzenlenen haritada birbirine benzeyen ağırlık vektörlerine sahip düğümler birbirine olabildiğince yakınlaşmışlardır. Doküman sınıflandırmada bunu aynı kavramdan bahseden dokümanların öğrenme bittikten sonra birbirlerine komşu düğümlerde olması şeklinde düşünebiliriz.

6.6. Doküman Etiketleme

Kendinden düzenlenen haritalar algoritması ile dokümanların bir eşleme haritasıyla anlaşılabilir bir şekilde görüntülenmesi için uygun bir şekilde etiketlenmesi gerekmektedir. Etiketleme işlemi genellikle dokümandaki en karakteristik kelimeler ile yapılır. Bu konuda çeşitli yöntemler bulunmaktadır. Bunlardan en yaygın olarak kullanılanı LabelSOM yöntemidir [61].

Temel olarak bir dokümanı en iyi karakterize eden kelimelerle ilgileniriz. Bu kelimeler belirli bir dokümanın özeti gibi düşünülebilir. Etiketleme için her bir dokümandaki kelimelerin tekrar sayılarını tutan bir desen analiz edilmelidir. Bunun için ağırlık vektörü bileşenleri ile ilgili bileşenlerin giriş vektörleri arasındaki standart sapma dikkate alınır. D_i bir birimdeki doküman kümesini göstersin. Bu durumda belirli bir vektör bileşeni (kelime), k için standart sapma, δ_{i_k}, şu şekilde ifade edilebilir :

$$\delta_{i_k} = \sum_{x \in D_i} \sqrt{(\mu_{i_k} - \xi_k)^2} \tag{6.4}$$

Bu eşitliğe göre standart sapması belirli bir eşik değerinin, T_1 altında kalan kelimeler etiketlemeye aday olan kelimelerdir.

Ancak çok sayıda büyük boyutlu dokümanlarda bilgi analizi yaptığımızdan dolayı ikinci bir kriter daha kullanmamız gerekmektedir. Dokümanların içerdikleri anahtar kelimeler incelendiğinde birçok kelimenin doküman içerisindeki tekrar sayısının az olmasından dolayı ağırlık değerinin 0 veya 0'a yakın olduğu görülür. Bu tür kelimelerin Eş. 6.4'e göre standart sapma değerleri de oldukça küçük çıkacaktır. Dolayısıyla bu tür kelimelerin dokümanı temsil edemeyeceği açıktır. Sadece dokümanı temsil edebilecek kelimelerle ilgilendiğimiz için 2. bir eşik değeri, T_2 kullanırız. Bu durumda, T_2 kelimenin ağırlık vektöründeki değerini gösterir. Sonuç olarak ağırlık vektöründe yüksek değere sahip olan kelimeler belirli bir birimde dokümanı temsil etmeye aday kelimeler olarak alınmış olmaktadır.

Çizelge 6.1, normalize edilmemiş rakamlarla her bir dokümanda kelimelerin kaç kez tekrar edildiğini göstermektedir.

Çizelge 6.1. Dokümanlarda geçen kelime sayıları (kelime histogramı)

	D1	D2	D3	D4	D5	D6	D7
Köpek	3	1	2		3		
Kedi	1	3		4	2	2	1
Kartal	6		5	6	5		6
At	3		4	6			
Kurt		5		2		1	1
İnek	2		4		2	1	3
Kaplan		1	1	3		2	
Zebra	2	2	1		3		1

Eş. 6.4'ün hesaplanması her bir doküman vektörü (D1..D7) ve her bir kelime için şu şekilde yapılır ($T_1 = 3$, $T_2 = 2$);

D1 doküman vektörü için;

- Köpek : $\sqrt{(3-1)^2+(3-2)^2+(3-3)^2} = \sqrt{5}$ $< T_1$
- Kartal : $\sqrt{(6-5)^2+(6-6)^2+(6-5)^2+(6-6)^2} = \sqrt{2}$ $< T_2$
- At : $\sqrt{(3-4)^2+(3-6)^2} = \sqrt{10}$ $> T_1$
- İnek : $\sqrt{(2-4)^2+(2-2)^2+(2-1)^2+(2-3)^2} = \sqrt{6}$ $< T_1$
- Zebra : $\sqrt{(2-2)^2+(2-1)^2+(2-3)^2+(2-1)^2} = \sqrt{7}$ $< T_1$

D1 dokümanında Kedi kelimesi ($T_2 = 2$), 2'den daha az tekrar edildiğinden hesaba katılmaz. Kartal kelimesi için bulunan standart sapma değeri $\sqrt{2}$ olup T_2 'den küçük olduğundan etiketleme kelimesi olarak alınmaz. Bu durumda ($T_1 = 3$) olduğundan D1 dokümanını temsil edebilecek kelimeler şunlardır : Köpek, İnek, Zebra.

7. UYGULANAN YÖNTEM

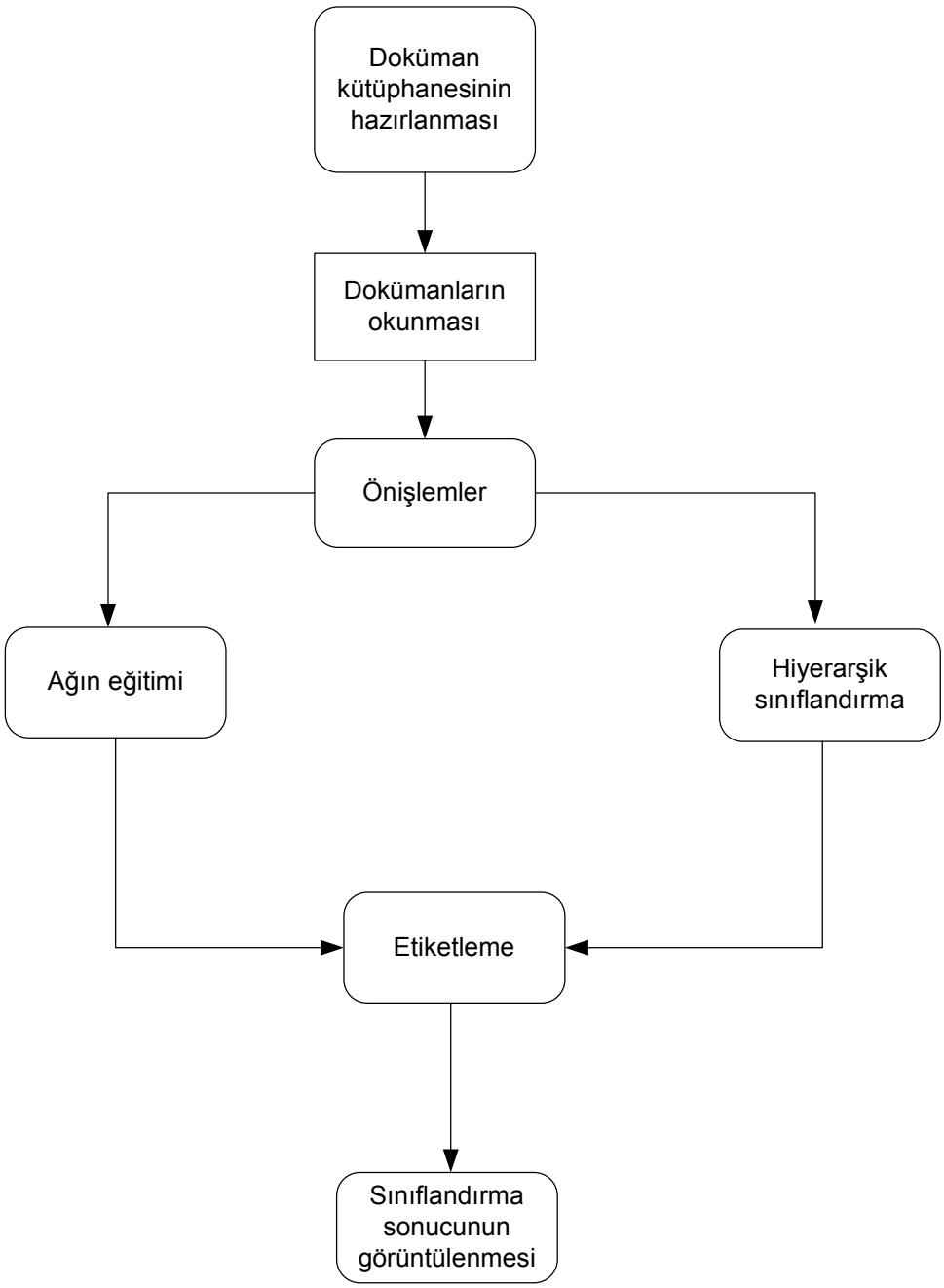
Yazılım ve geliştirme ortamını seçerken en başta gelen kriterimiz öğrenmesi ve kullanımı kolay olmasıdır. Bu uygulama Java programlama dili kullanılarak geliştirilmiştir. Bu tercihteki temel nedenler şunlardır :

- Öğrenmesi ve kullanım kolaylığı.
- Hatalı kod yazmayı önleyici özelliklere sahip olması,
- Görsellik ve web ortamına tam uyumluluk,
- Grafik tabanlı geliştirme ihtiyaçları için hazır sınıflara (class) sahip olması,
- Platform bağımsız çalışma. Yazılımı geliştirdikten sonra başka donanım veya yazılım ortamlarına taşıyabilmek için hiç bir çaba gerektirmemesi,
- Kendinden düzenlenen haritalarda en çok ihtiyacımız olan ağırlık vektörleri için hazır ve güçlü veri yapılarına karşılık gelen Vector, Hashtable, ArrayList gibi sınıflara (class) sahip olması,
- Nesne tabanlı bir programlama dili olması. Dolayısıyla uygulamada yapılacak değişiklik ve eklemelere son derece uyumlu bir yapıda olması,
- Yaygın kullanımı nedeniyle geliştirme sırasında karşılaşılan problemlere çok kolaylıkla çözüm bulunabilmesi,
- Ücretsiz olması.

Java programlama diliyle yazılım geliştirme ortamları sunan birçok araç vardır. Bunlardan en çok kullanılan Eclipse, uygulamamızı geliştirmek için şu nedenlerden dolayı tercih edilmiştir :

- Ücretsiz olması (GNU lisansı)
- Grafik tabanlı geliştirme ihtiyaçları için hazır araçlara sahip olması. Bu amaçla kullanılan araç : Swing Designer Free Edition'dır.
- Yaygın kullanımı.

Uygulama kapsamında takip edilen adımlar genel başlıklarıyla Şekil 7.1'de verilmiştir. Bu adımlar sırasıyla aşağıda açıklanmaktadır.



Şekil 7.1. Uygulama adımları

7.1. Doküman Kütüphanesinin Hazırlanması

Doküman kütüphanesi elde etmek için 2 farklı çalışma yapılmıştır. 1. çalışmada doküman kütüphanesi olarak bir internet sitesindeki 100'den fazla haber özeti toplanmıştır. Haber portalından alınan haber özetlerinin her biri ayrı bir metin

dosyası haline getirilmiştir. 2. çalışmada ise farklı üniversitelerin farklı bölümlerinin web sayfalarından alınan ders içerikleri toplanarak bir doküman kütüphanesi elde edilmiştir. Bu amaçla her bir ders içeriği farklı bir metin dosyasına çevrilmiş ve ders kodları dosya ismi olarak kullanılmıştır. Örneğin BİM101 kodlu ders içeriği BİM101.txt şeklinde bir dosyada saklanmıştır. Bu şekilde elde edilen 100 farklı ders içeriği sınıflandırılmaya çalışılmıştır.

Sınıflandırılacak dokümanların tamamı bir dizine kopyalanır. Bu dizin programda doküman kütüphanesi olarak okunacak ve ağa giriş olarak verilecek dokümanların bulunduğu bir dizindir. Bu dizinler kullanıcı tarafından parametrik olarak seçilebilmektedir.

Programda dokümanlar satır satır okunmakta ve her bir satır kelimelere ayrıştırılmaktadır. Bu işlem sonucunda her bir doküman, içerisinde geçen kelimelerin tekrar sayıları ile birlikte bir vektör olarak elde edilmiş olunmaktadır.

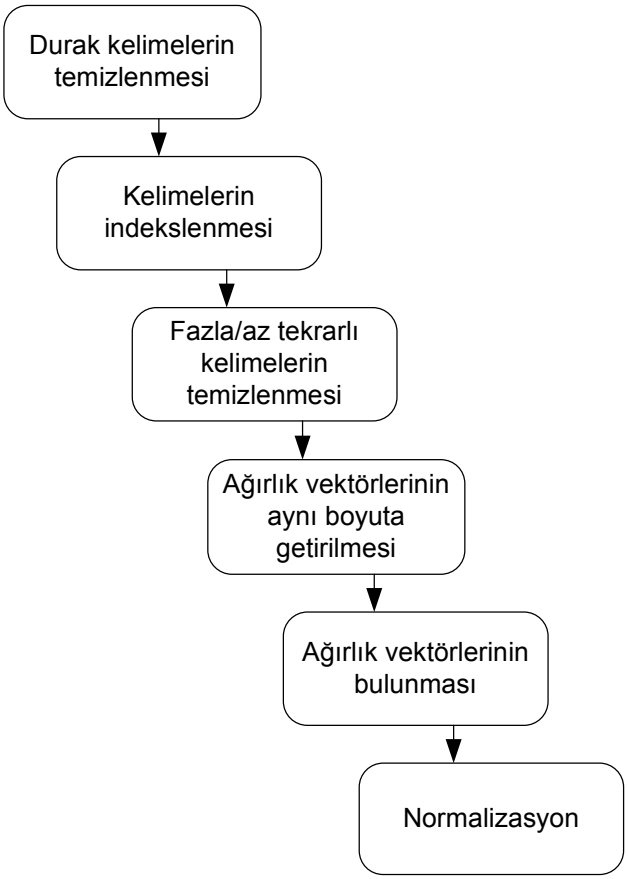
7.2. Önişlemler

Bir doküman kümesine kendinden düzenlenen haritalar algoritması uygulanmadan önce metin bilgisi olmayan ve sınıflandırmada doğrudan etkisi olmayan birtakım verilerin temizlenmesi gerekir. Bu amaçla ASCII çizim karakterleri ve sayısal ifadeler metinden temizlenmelidir. Ağın eğitiminden önce verilerin temizlenmesi dışında yapılması gereken başka işlemler de vardır. Bu işlemler Şekil 7.2'de gösterilmiş ve aşağıda tek tek açıklanmıştır.

7.3. Durak Kelimelerinin Temizlenmesi

Her dilde doküman sınıflandırmaya etkisi olmayan bazı kelimeler, bağlaçlar, harf veya semboller vardır. Dokümanlar analiz edildiğinde bu tür kelimelerin önemli ölçüde çok tekrar ettiği görülmektedir. Bu kelimeleri ağı eğitmeden önce giriş

verilerinden temizlemek sınıflandırmayı doğru yönlendirmek ve daha kısa süren bir hesaplama zamanı kullanabilmek açısından önemlidir.



Şekil 7.2. Önişlemler

Türkçe'deki durak kelimelerin tamamını veren bir çalışma bulunamadığından bu kelimelerin birçoğu bu çalışmada tespit edilmeye çalışılmıştır. Türkçe için bulunan ve bu uygulamada kullanılan durak kelimelerinin listesi Ek.2'de verilmiştir. Ancak bunlar üzerinde daha detaylı çalışarak zenginleştirmek mümkündür.

7.4. Kelimelerin İndekslenmesi

Bir tabloda doküman kütüphanesindeki her bir kelimenin indeksi vardır (Çizelge 7.1). Bu tablo, tüm dokümanlarda geçen kelimelerin ve tekrar sayılarının tutulduğu

bir tablodur. Başka bir ifadeyle her bir kelimeyi ve kelimenin kaç farklı dokümanda kullanılmış olduğu bu tabloda yer almaktadır.

Çizelge 7.1. Her kelimenin geçtiği doküman sayısı

	Kelime	Geçtiği Doküman Sayısı
1	Kitap	15
2	Kalem	6
3	Defter	9
4	Silgi	20
.	..	..
m	..	..

Tüm dokümanlarda en az bir defa tekrar edilen kelimelerin sayısını m ile gösterecek olursak bu tabloyu mx2 şeklinde ifade edebiliriz. Bu durumda 1.sütunda kelimeler 2.sütunda ise bu kelimelere karşılık gelen her bir kelimenin tekrar sayısı yer alır. Elde edilen indeks tablosu çok büyük boyutlara ulaşabildiğinden bazı boyut küçültme teknikleri uygulamak mümkündür.

7.5. Ağırlık Vektörlerinin Aynı Boyuta Getirilmesi

Ağın eğitimi ve eğitim sonrası sınıflandırmanın doğru değerlerle yapılması için dokümanlarda geçen kelimelerin sıralanmasına ihtiyaç vardır. Yani farklı dokümanlarda geçen veya geçmeyen tüm kelimeler aynı sırada dokümanı bir vektör olarak temsil edebilmelidir. Bu işlemin çalışma şekli Çizelge 7.2'de bir örnek üzerinde gösterilmiştir. Bu çizelgede ilk durum üst tarafta, son durum ise alt tarafta verilmiştir. Bu şekilde görüldüğü gibi bir dokümanda geçmeyen diğer dokümanlara ait kelimeler de bu dokümanın ağırlık vektörüne eklenmiştir. Örneğin D1 dokümanında hiç geçmeyen "Masa", "Kağıt" ve "Kalemtıraş" kelimeleri D1 dokümanı için kullanılan vektöre eklenmiştir. Benzer şekilde D2 dokümanında kullanılmayan "Kitap", "Kağıt" ve "Kalemtıraş" kelimeleri D2 doküman vektörüne,

D3 dokümanında kullanılmayan “Kitap”, “Kalem” ve ”Masa” kelimeleri D3 dokümanına eklenmiştir.

Çizelge 7.2. Kelimelerin aynı boyuta getirilmesi

D1	
Kitap	3
Defter	2
Kalem	1
Silgi	6

D2	
Masa	4
Defter	3
Silgi	7
Kalem	1

D3	
Silgi	9
Kağıt	3
Defter	6
Kalemtıraş	8

D1	
Kitap	3
Defter	2
Kalem	1
Silgi	6
Masa	0
Kağıt	0
Kalemtıraş	0

D2	
Kitap	0
Defter	3
Kalem	1
Silgi	7
Masa	4
Kağıt	0
Kalemtıraş	0

D3	
Kitap	0
Defter	6
Kalem	0
Silgi	9
Masa	0
Kağıt	3
Kalemtıraş	8

7.6. Çok ve Az Tekrar Eden Kelimelerin Temizlenmesi

Doküman sınıflandırma için kullanılan veriler kelime sayıları dikkate alındığında çok yüksek boyutlara ulaşabilmektedir. Yüksek boyutlu veriler ise ağın eğitim süresinin önemli oranda yüksek olmasına neden olmaktadır. Bu süreyi kısaltmak için çok az karşılaşılan terimler veya çok fazla tekrar edilen terimler de metinlerden temizlenir. Bu işlem için literatürde kabul edilen oran %10’dan az ve %90’dan çok geçen terimlerin temizlenmesi şeklindedir.

7.7. Ağırlık Vektörlerinin Bulunması

Dokümanlarda geçen kelimelerin ayrıştırma işlemi bittikten sonra her bir doküman vektöründeki terimlerin ağırlıklarından oluşan ağırlık vektörlerinin hesaplanmasına ihtiyaç vardır. Ağırlıkların hesaplanmasında kullanılan formül Eş.3.4 verilmiştir. Bu ağırlıkların hesaplanabilmesi için daha önceki adımlarda elde edilen toplam doküman sayısı, her bir kelimenin her bir dokümandaki ve tüm dokümanlardaki toplam tekrar

sayıları kullanılmıştır. Hesaplama sonucunda elde edilen ağırlık vektörleri ilgili dokümanla bağlantılı olarak bir vektör matrisinde saklanmıştır.

Eş.3.4'teki formüle göre Çizelge 7.2'de verilen D1 dokümanı için doküman vektörlerinin ağırlıkları hesaplandıktan sonraki durumu Çizelge 7.3'te görülmektedir.

Çizelge 7.3. Ağırlık vektörlerinin hesaplanması

D1			
	TF	*IDF*	*TFxIDF*
Kitap	3	1,477	4,431
Defter	2	1	2
Kalem	1	1,176	1,176
Silgi	6	1	6
Masa	0	1,477	0
Kağıt	0	1	0
Kalemtıraş	0	1	0

Bu hesaplama D1 dokümanı için şu şekilde yapılmıştır :

TDS (Toplam doküman sayısı) = 3,
TS = Kelimenin kaç farklı dokümanda tekrar edildiği,
TF = Kelimenin D1 dokümandaki tekrar sayısı,

Kitap :
TF = 3,
IDF = $\log(TDS/TS_{Kitap})+1 = \log(3/1)+1 = 1,477$
w = TFxIDF = 3x1,477 = 4,431
Defter:
TF = 2,
IDF = $\log(TDS/TS_{Defter})+1 = \log(3/3)+1 = 1$
w = TFxIDF = 2x1 = 2

Kalem :

TF = 1,

IDF = log(TDS/TS_{Kalem})+1 = log(3/2)+1 = 1,176

w = TFxIDF = 1x1,176 = 1,176

Silgi :

TF = 6,

IDF = log(TDS/TS_{Silgi})+1 = log(3/3)+1 = 1

w = TFxIDF = 6x1 = 6

Daha sonra, ağın iyi bir performans verebilmesi için diğer YSA tekniklerinde olduğu gibi kendinden düzenlenen haritalar için de normalizasyon işlemi gerçekleştirilmiştir. Bu amaçla kelime ağırlıkları 0-1 arasında normalize edilir. Normalizasyon için Eş.3.6'da verilen formül kullanılmıştır.

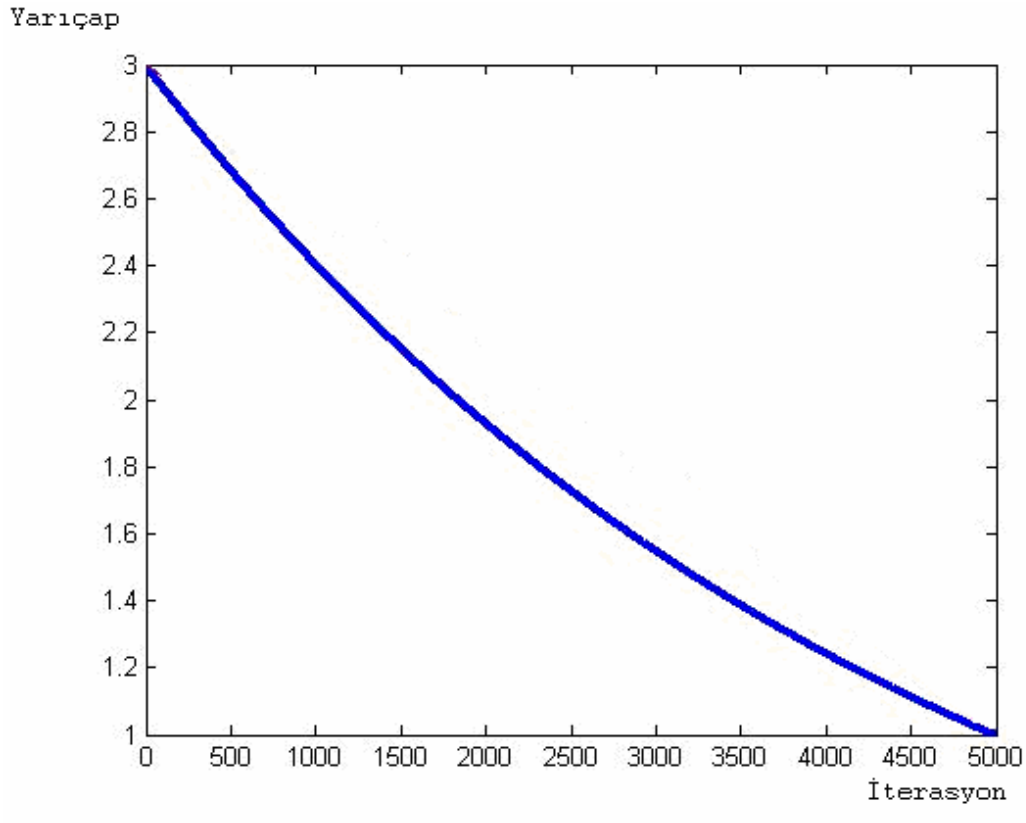
7.8. Ağın Eğitimi

Kendinden düzenlenen haritaların eğitimi için 6.5'de anlatılan algoritma kullanılır. Eğitimin başında, yarıçap uzunluğunun başlangıç değeri bulunur. Bunun için; doküman matrisinin en (w) ve boy (h) değerinden büyük olanının yarısı alınır. Örneğin, 120 dokümanlı bir doküman kütüphanesi için, w=10, h=12 alınması durumunda başlangıç yarıçapı olarak 6 değeri kabul edilecektir. Başlangıçta alınan bu yarıçap değeri eğitim boyunca azaltılır. 49 doküman ve 5000 iterasyonlu bir çalıştırmada yarıçapın iterasyonlar boyunca değişimi Şekil 7.3'de verilmiştir.

İterasyon boyunca komşuluk yarıçapı bulunurken, bir zaman sabitine ihtiyaç vardır. Bu zaman sabiti şu şekilde hesaplanır :

$$\text{zaman\_sabiti} = \frac{\text{toplam\_iterasyon\_sayisi}}{\log(\text{baslangic\_yariçapi})} \tag{7.1}$$

Komşuluk fonksiyonunun hesaplanmasında Eş. 6.3'de verilen formül kullanılır.



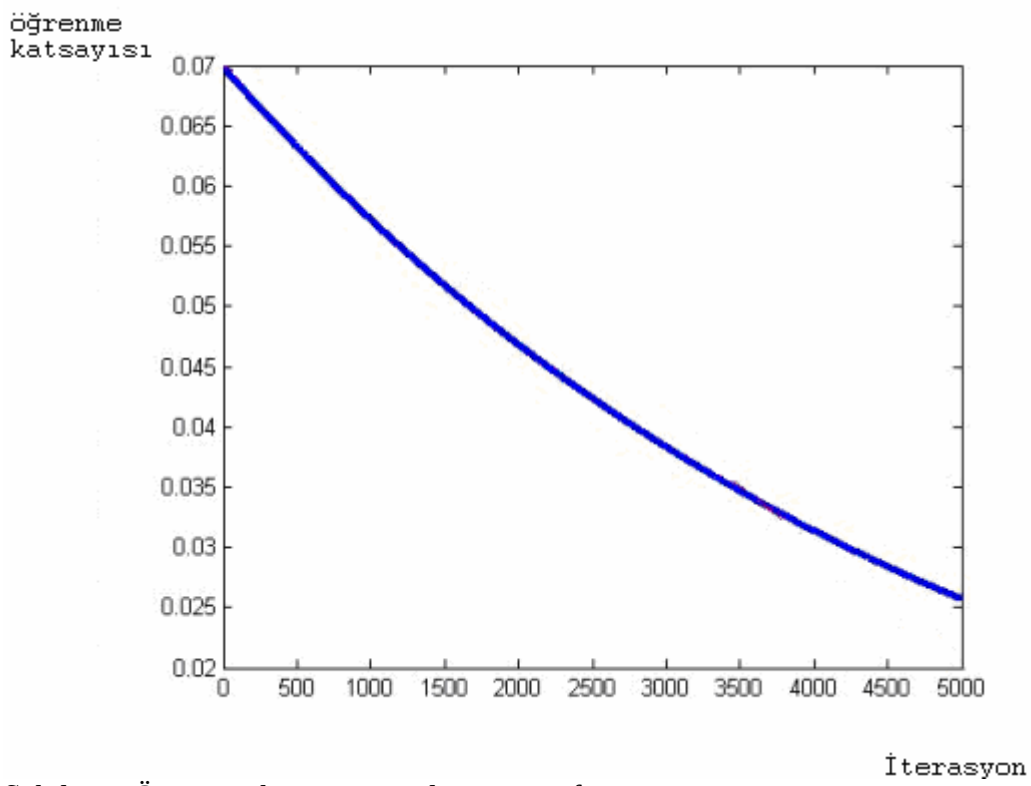
Şekil 7.3. Yarıçapın iterasyona göre değişim grafiği

Kendinden düzenlenen haritalar algoritmasının kullandığı öğrenme katsayısı başlangıçta aldığı bir değerle eğitime devam ederken sürekli bu katsayının başlangıç değerini değiştirmektedir. Bu değer her bir iterasyonda üstel olarak azaltılmaktadır. Şekil 7.4.'de bu katsayının değişimi görülmektedir.

Kendinden düzenlenen haritalar algoritmasında her bir iterasyonun başında rastgele alınan bir düğüme en benzer düğümü bulmak için değişik uzaklık formülleri kullanılmıştır. Kullanılan uzaklık formülleri şunlardır:

1) Öklit (Bknz. Eş. 4.8),
2) City-block (Bknz. Eş. 4.9),
3) Chebychev (Bknz. Eş. 4.10)

Bu uzaklık formülleri hiyerarşik sınıflandırma algoritmasının uygulanmasında da kullanılmıştır.



Şekil 7.4. Öğrenme katsayısının değişim grafiği

7.9. Hiyerarşik Sınıflandırma

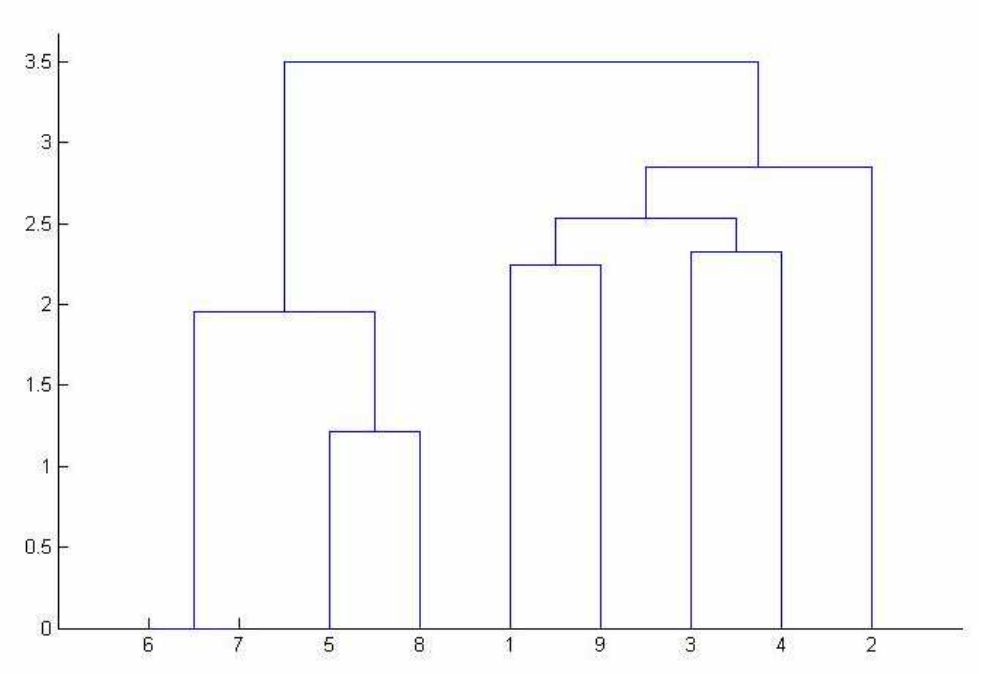
Hiyerarşik sınıflandırma yapmak için en başta bir başlangıç uzaklık matrisi oluşturulur. Başlangıç uzaklıkları olarak her bir dokümanın kendinden düzenlenen ağlar için hazırlanan ağırlık vektörleri kullanılmaktadır. Dolayısıyla iki doküman arasındaki uzaklık bu ağırlıkların öklit uzaklığından bulunmuştur. Dokümanların hiyerarşik sınıflandırması için bölüm 4.2'de anlatılan algoritma kullanılmıştır.

Çizelge 7.4'te listesi verilen dokümanların hiyerarşik sınıflandırma ile elde edilen dendrogramı Şekil 7.5.'de görülmektedir. Hiyerarşik sınıflandırma için kullanılan dokümanlar içerikleri ile birlikte Ek.3'te verilmiştir. Hiyerarşik sınıflandırma sonucuna dikkat edilirse öncelikle D6 ve D7 dokümanlarının birleştirildiği görülmektedir. Ek.3'te görüldüğü gibi D6 ve D7 dokümanları içerik olarak birebir

aynıdır. Daha sonra sırasıyla D5, D8, D1, D9, D3, D4 ve D2 dokümanları birleştirilmiştir. Hiyerarşik sınıflandırmanın doküman sınıflandırmada kendinden düzenlenen haritalar kadar başarılı olmadığı görülmüştür.

Çizelge 7.4. Ders listesi (Hiyerarşik sınıflandırma için)

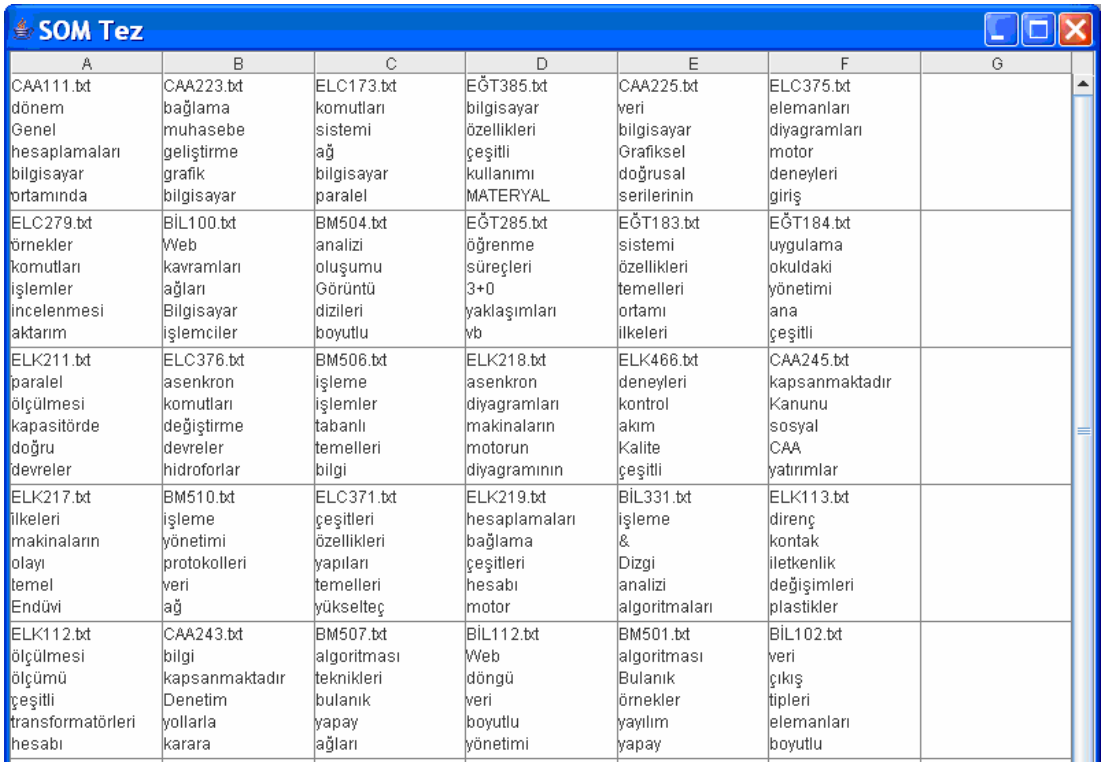
Doküman Numarası	**Doküman Adı**
D1	BM206_Sayısal_Çözümleme.txt
D2	BM301_Mikroişlemciler.txt
D3	BM303_Bilgisayar_Organizasyonu.txt
D4	BM306_Bilgisayar_Mimarisi.txt
D5	EM308_Nümerik_Analiz.txt
D6	ENF102_C_Programlama_Dili.txt
D7	K_ENF102_C_Programlama_Dili.txt
D8	MM313E_Sayısal_Analize_Giriş.txt
D9	İM343_Sayısal_Çözümleme.txt



Şekil 7.5. Hiyerarşik sınıflandırılma sonucu elde edilen dendrogram yapı

7.10. Etiketleme

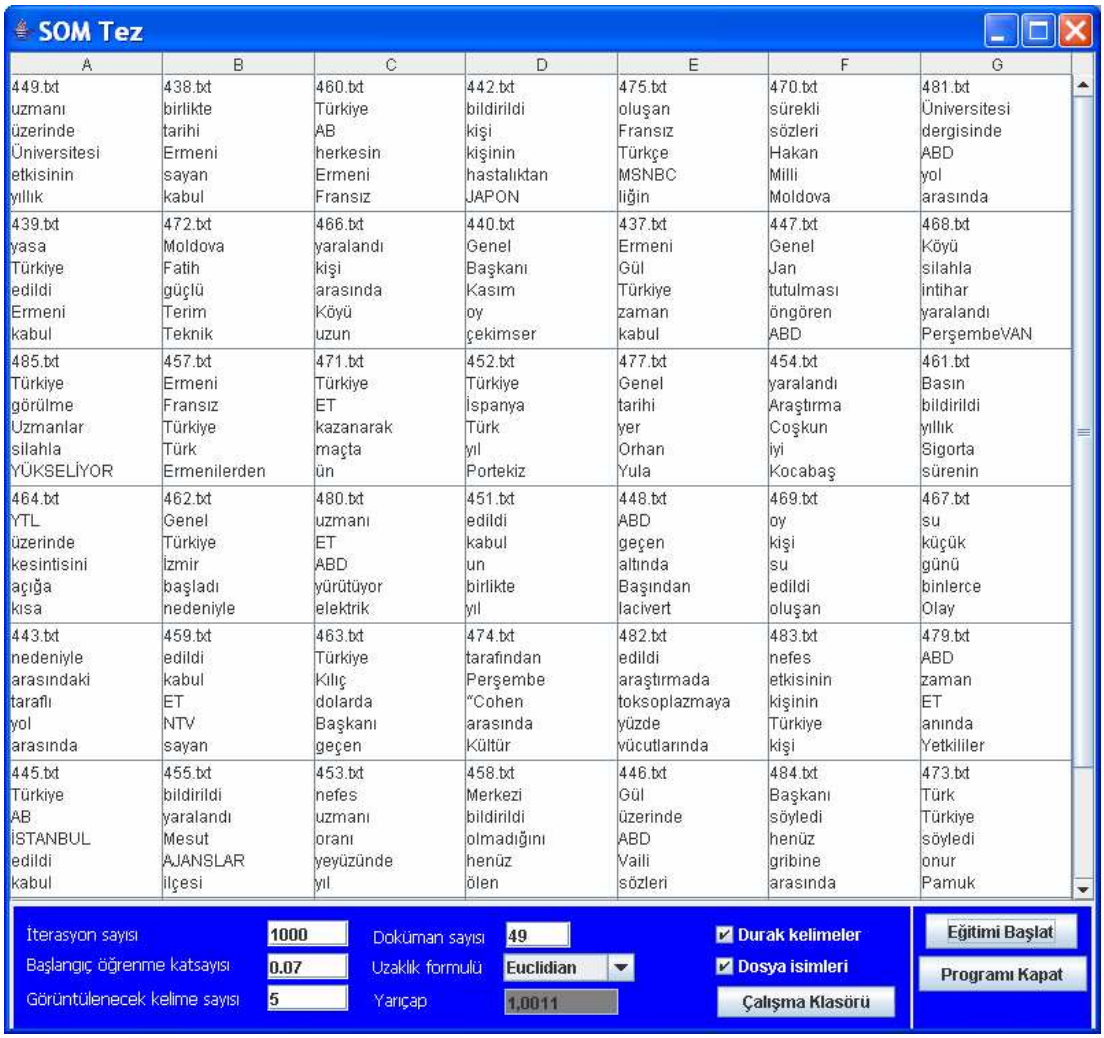
Sınıflandırma işlemi bittikten sonra dokümanları karakterize eden kelimeleri bulabilmek için etiketleme denilen işlem gerçekleştirilir. Etiketleme işlemi bölüm 6.6'de anlatıldığı gibi yapılmaktadır. Etiketleme işlemi sonucunda her bir doküman kendisini en iyi temsil edebilen kelimelerle birlikte ızgara görünümünde sunulur. Çıktı ekranındaki her bir karenin ilk satırında dokümanın adı ve hemen altındaki satırlarda etiket kelimeleri gösterilmektedir (Şekil 7.6).



Şekil 7.6. Her karenin ilk kelimesi doküman adını, diğerleri etiketi gösterir

7.11. Deneysel Sonuçlar

49 farklı haber özetinin kendinden düzenlenen haritalar algoritması ile eğitildikten sonraki doküman örüntüsündeki etiket kelimeleri Şekil 7.7'da görülmektedir. Bu sınıflandırmada iterasyon sayısı, 1000, başlangıç öğrenme katsayısı, 0.5, ve uzaklık formülü olarak Öklit kullanılmıştır.



Şekil 7.7. Internet haber özetlerinin sınıflandırıldığı bir uygulama çıktısı

Yapılan çalışmanın başarılı olup olmadığını kesin olarak söyleyebilmek için şu şekilde bir uygulama yapılmıştır. Öncelikle 3 adet doküman alınarak bu dokümanların ikişer adet kopyaları elde edilmiştir. Bu 3 doküman aşağıda verilmiştir:

- D1 (BM301_Mikroişlemciler): mikroişlemcilere ilişkin temel mantıksal kavramlar. bellek öğeleri. çalışma ilkeleri ve türlerin incelenmesi. adres uzayı ve bellek tasarımı. mikroişlemciler ve g/ç temel kavramları. kesilme yapıları, kesilme önceliği kodlayıcılar. doğrudan bellek erişimi. g/ç arabirimi tasarımı: koşut (8155), ardıl (8251) arabirimlerinin incelenmesi. 8085 işleyicisinin komut zaman

çizeneklerinin incelenmesi. diğer 8/16 bit mikroişlemcilerin incelenmesi. mikroişlemci tabanlı dizge tasarımına giriş.

- D2 (EM308_Nümerik_Analiz): nümerik hata analizi. denklem köklerinin bulunması. doğrusal denklem sistemleri. eğri uydurulması. interpolasyon. sayısal türev ve integrasyon. adi diferansiyel denklemlerin çözümleri. özdeğer ve özvektörler.
- D3 (BM306_Bilgisayar_Mimarisi): bilgisayar sistemlerine bakışlar. diller, düzeyler ve sanal makineler. bilgisayar sistemlerinin düzenlenişi. sayısal mantık düzeyi, mikroprogramlama düzeyi. geleneksel makine dili düzeyi. işletim sistemi düzeyi. ileri bilgisayar mimarileri. azaltılmış komut seti bilgisayarları. paralel mimariler.

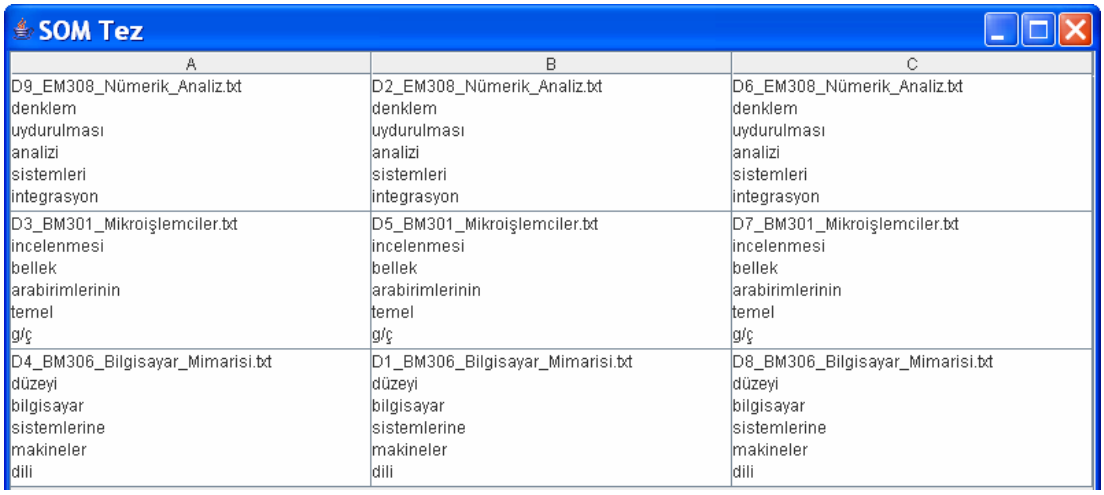
Bu dokümanlardan D1'in kopyası D4 ve D7 olarak D2'nin kopyası D5 ve D8, D3'ün kopyası ise D6 ve D9 olarak alınmıştır. Dolayısıyla dokümanlar arasındaki benzerlik durumu için :

D1 = D4 = D7,
D2 = D5 = D8,
D3 = D6 = D9 söylenebilir.

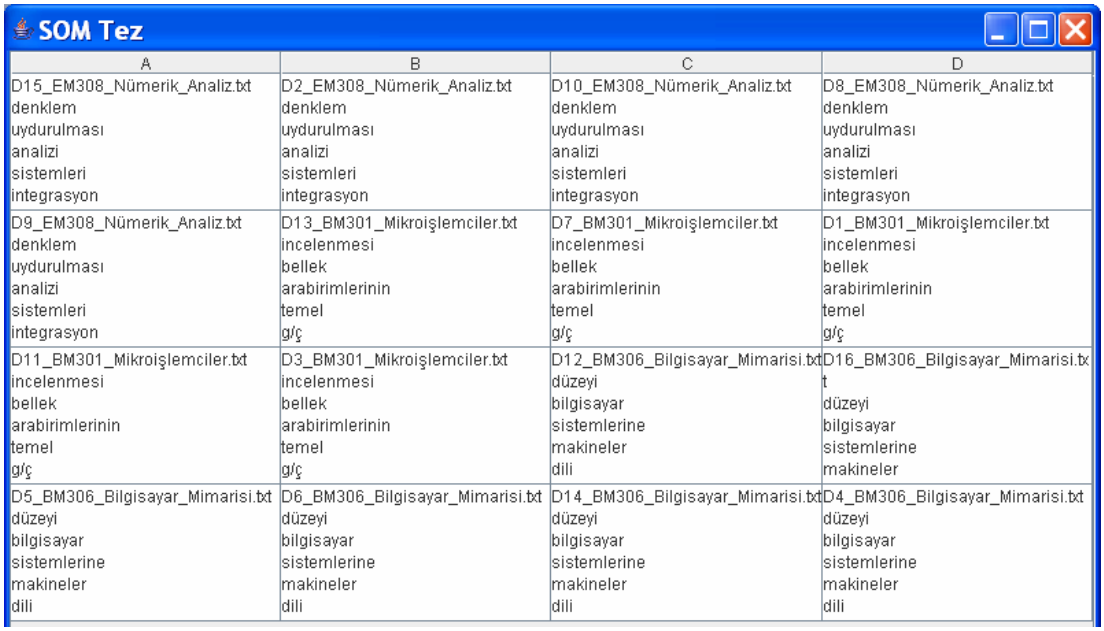
Bu 9 adet dokümanın sınıflandırma çalışması sonucu Şekil 7.8'de verilmiştir. Şekil 7.8'te görüldüğü gibi aynı içeriğe sahip dokümanlar yanyana gelmiştir. Benzer bir uygulamada toplamda 16 doküman olamak üzere dokümanların kopyaları alınmıştır ve Şekil 7.9'de görülen sonuç elde edilmiştir.

Burada, birbirinin kopyası olan dokümanlar şu şekilde oluşturulmuştur :

D1 = D3 = D7 = D11 = D13
D2 = D8 = D9 = D10 = D15
D4 = D5 = D6 = D12 = D14 = D16



Şekil 7.8. Örnek uygulama sonucu (3x3)



Şekil 7.9. Örnek uygulama sonucu (4x4)

Şekil 7.9'da görüldüğü gibi birbirinin kopyası olan dokümanlar yanyana gelmiştir. Uygulamanın farklı içeriklere sahip doküman sayısı arttıkça nasıl davrandığını öğrenmek için birbirinden farklı 9 doküman ele alınarak benzer bir çalışma yapılmıştır. Bu çalışmada ele alınan ders içerikleri Ek.3'te verilmiştir. Derslerin listesi Çizelge 7.5.'de verilmiştir. Kullanılan ders içeriklerinden D2 ve D5 içerik olarak birebir aynı olarak seçilmiştir.

Çizelge 7.5. Ders listesi (kendinden düzenlenen haritalar için)

Doküman Numarası	**Doküman Adı**
D1	MM313E_Sayısal_Analize_Giriş
D2	ENF102_C_Programlama_Dili
D3	İM343_Sayısal_Çözümleme
D4	EM308_Nümerik_Analiz
D5	K_ENF102_C_Programlama_Dili
D6	BM301_Mikroişlemciler
D7	BM306_Bilgisayar_Mimarisi
D8	BM206_Sayısal_Çözümleme
D9	BM303_Bilgisayar_Organizasyonu

Bu derslerin sınıflandırma programıyla 100 iterasyon sonucunda sınıflandırma sonucu Şekil 7.10'da verilmiştir. Görüldüğü gibi benzer içeriğe sahip dersler başarılı bir şekilde yanyana getirilmiştir.

A	B	C
ENF102_C_Programlama_Dili.txt	K_ENF102_C_Programlama_Dili.txt	İM343_Sayısal_Çözümleme.txt
bellek	bellek	diferansiyel
nesneler	nesneler	çözümü
döngüler	döngüler	analizi
diziler	diziler	sayısal
fonksiyonlar	fonksiyonlar	etme
BM206_Sayısal_Çözümleme.txt	EM308_Nümerik_Analiz.txt	MM313E_Sayısal_Analize_Giriş.txt
diferansiyel	bulunması	çözümü
nümerik	nümerik	diferansiyel
eğri	hata	denklem
bulunması	denklem	öz
hata	diferansiyel	farklar
BM303_Bilgisayar_Organizasyonu.txt	BM301_Mikroişlemciler.txt	BM306_Bilgisayar_Mimarisi.txt
yönetimi	bellek	bilgisayar
bellek	adres	ileri
donanım	tasarımı	mimarileri
mekanizması	dizge	sayısal
mikroprogramlama	giriş	paralel

Şekil 7.10. Farklı ders içerikleri sınıflandırıldığı bir uygulama çıktısı (3x3)

Uygulama aynı içeriğe sahip dokümanlar için etiket kelimeleri olarak aynı kelimeleri bulmuştur. 1. grupta yer alan ENF102_C_Programlama_dili.txt ve K_ENF102_C_Programlama_Dili.txt dokümanları birbirinin kopyası olduğu için her ikisinde de etiket kelimeleri olarak “bellek”, ”nesneler”, ”döngüler”, ”diziler”, ”fonksiyonlar” kelimeleri bulunmuştur. 2. grupta yer alan İM343_Sayısal_Çözümleme, BM206_Sayisal_Çözümleme, EM308_Nümerik_Analiz ve MM313E_Sayısal_Analize_Giriş dokümanları benzer içeriklerinden dolayı biraraya getirilmiştir. Bu dokümanlarda yer alan “diferansiyel”, “nümerik”, “denklem” gibi kelimelerin etiketlemede kullanıldığı görülmektedir. 3. grupta yer alan BM303_Bilgisayar_Organizasyonu, BM301_Mikroişlemciler ve BM306_Bilgisayar_Mimarisi dokümanları da benzer içeriklere sahip oldukları için yanyana gelmişlerdir.

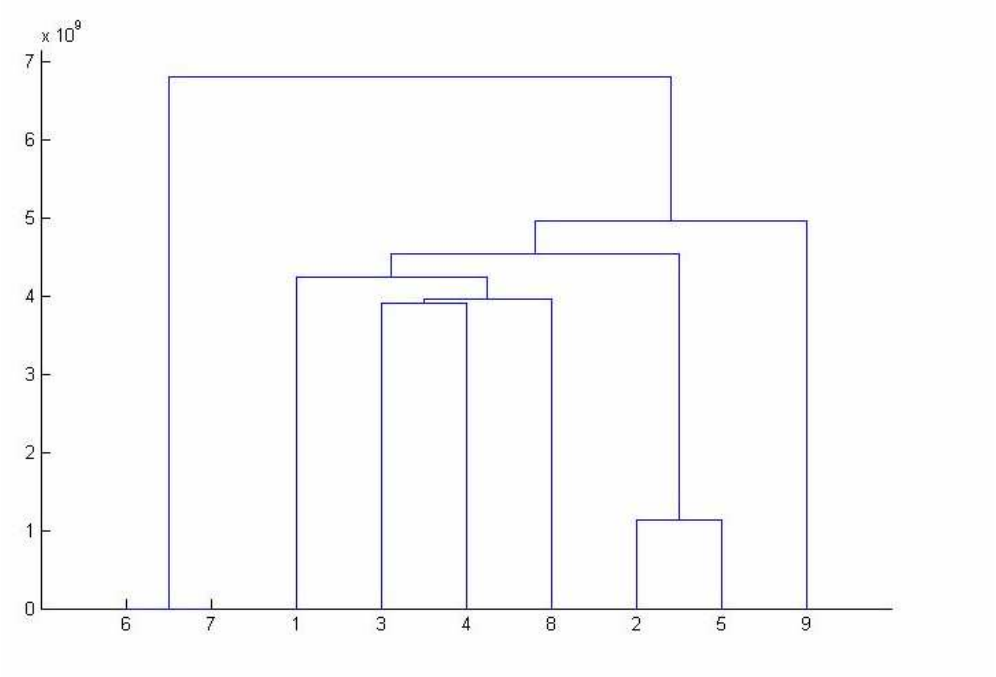
Çizelge 7.6. Sınıflandırılmış ders listesi

Doküman Numarası	**Doküman Adı**	**Doküman Sınıfı**
D2	ENF102_C_Programlama_Dili	S1
D5	K_ENF102_C_Programlama_Dili	S1
D3	İM343_Sayısal_Çözümleme	S2
D4	EM308_Nümerik_Analiz	S2
D8	BM206_Sayısal_Çözümleme	S2
D1	MM313E_Sayısal_Analize_Giriş	S2
D9	BM303_Bilgisayar_Organizasyonu	S3
D6	BM301_Mikroişlemciler	S3
D7	BM306_Bilgisayar_Mimarisi	S3

Sınıflandırma çalışmasının sonucunda oluşan durum Çizelge 7.6’da verilmiştir. D2 ve D5 programlama dili ile ilgili aynı içeriğe sahip iki ders olup birbirine en benzer doküman olmaları nedeniyle en başta sınıflandırılmıştır. Ardından sayısal analiz ile

ilgili olarak D3, D4, D8 ve D1 dokümanları yanyana getirilmiştir. Benzer şekilde bilgisayar mimarisi ile ilgili olan D9, D6 ve D7 dokümanları da yanyana getirilmiştir. Yanyana gelen dokümanlar S1, S2, ve S3 sınıflarıyla gösterilmiştir.

Hiyerarşik kümeleme sırasında elde edilen uzaklık matrisine ait hiyerarşi ağacı dendrogram kullanılarak çizilmiş ve Şekil 7.11'de gösterilmiştir.



Şekil 7.11. Dokümanların uzaklıklarına göre hiyerarşik sınıflandırması

Sınıflandırma çalışması için geçen süreler Çizelge 7.7'da verilmiştir.

Çizelge 7.7. Çalıştırma süreleri

Veri kümesi	**Harita**	**İterasyon**	**Süre**
I	3x3	1000	3 sn.
		5000	12 sn.
II	5x5	1000	6 sn.
		5000	26 sn.
III	7x7	1000	29 sn.
		5000	129 sn.

8. SONUÇ VE ÖNERİLER

Günümüzde elektronik dokümanların sayısı büyük bir hızla artmaktadır. Bilgisayarlarda çok miktardaki dosyalar konularına göre elle oluşturulan çeşitli dizinlerin altında saklanmaktadır. Dosya sayısının artmasıyla yapılan gruplandırmalar nitelik kaybına uğramaktadır. İnternet üzerinde ise milyonlarca web sayfası bulunmakta ve bu sayfaların konularına göre elle sınıflandırılması giderek daha da zorlaşmaktadır. Bu nedenle otomatik sınıflandırma sistemlerine ihtiyaç vardır. Otomatik sınıflandırma sistemlerinin, bilgi yönetiminin geleceği açısından çok kritik olduğu değerlendirilmektedir.

Bu çalışmada, elektronik dokümanların içeriklerine göre uygun bir şekilde otomatik olarak sınıflandırılması için bir sistem gerçekleştirilmiştir. Çalışmada özellikle yüksek boyutlu verilerde başarılı bir şekilde sınıflandırılma yapan kendinden düzenlenen haritalar (SOM) yöntemi kullanılmıştır. Bu yöntem danışmansız olarak çalıştığı için, otomatik sınıflandırma için çok ugundur. Kendinden düzenlenen haritalar algoritması ile elde edilen sonuçlar, hiyerarşik sınıflandırma ile karşılaştırılmıştır. Sınıflandırılan dokümanların bir eşleme haritasıyla anlaşılabilir bir şekilde görüntülenmesi için, LabelSOM yöntemi ile dokümanların etiketlenmesi yapılmıştır. Bu yöntemle bir dokümanı temsil edebilecek en belirgin (karakteristik) kelimeler seçilmiştir.

Uygulanan yöntemin aşamaları kısaca şu şekilde sıralanabilir: doküman kütüphanesinin hazırlanması, dokümanların okunması, önişlemler, durak kelimelerin temizlenmesi, kelimelerin indekslenmesi, ağırlık vektörlerinin aynı boyuta getirilmesi, çok ve az tekrar eden kelimelerin temizlenmesi, ağırlık vektörlerinin bulunması, normalizasyon, ağın eğitimi veya hiyerarşik sınıflandırma, etiketleme ve sınıflandırma sonucunun görüntülenmesi. Uygulanan yöntemin başarısı, 2 farklı tipte doküman kütüphanesi ele alınarak incelenmiştir. İlk olarak bir İnternet haber sitesinden rastgele alınmış haber içerikleri sınıflandırılmıştır. İkinci olarak ise, üniversitelerde verilen derslerin içerikleri başarılı bir şekilde sınıflandırılmıştır.

Ders içerikleri sınıflandırılırken uygulanan sistemin başarısı değişik senaryolarla detaylıca irdelenmiştir. Öncelikle 3 adet orijinal doküman alınarak bu dokümanların ikişer adet kopyaları daha çıkarılmıştır. Böylece elde edilen 9 adet dokümanın başarılı bir şekilde sınıflandırıldığı görülmüştür; aynı içeriğe sahip dokümanlar harita üzerinde yanyana gelmiştir. Benzer bir uygulamada ise, 3 adet orijinal dokümandan çeşitli sayılarda kopyalar üretilerek toplam 16 adet doküman elde edilmiştir. Bu durumda da beklendiği gibi, birbirinin kopyası olan dokümanlar harita üzerinde yanyana gelmiştir. Farklı içeriklere sahip doküman sayısı arttıkça, sistemin nasıl davrandığını öğrenmek için birbirinden farklı 9 doküman ele alınarak benzer bir çalışma yapılmıştır. 100 iterasyon sonucunda benzer içeriğe sahip dersler başarılı bir şekilde yanyana getirilmiştir.

Bu çalışmada uygulanan yöntem ile Web sayfaları ve haber gruplarındaki yazılar gruplanabileceği gibi elektronik posta mesajları kişinin özel ilgilerine göre otomatik olarak sınıflandırılabilir. Ayrıca resmi yazılar, kişisel dosyalar, tam metin veritabanları kolaylıkla sınıflandırılabilir. Bir işletmeye gelen yazı, makale vb. metinlerin ilgili kişilere otomatik dağıtımı yapılabilir. Örneğin bir ürün geliştirici ile pazarlama elemanının ilgileri birbirinden farklı olacaktır. İçerik patlamasının yaşandığı İnternet dünyasında bilginin otomatik sınıflandırılmasına olan ihtiyacın sürekli arttığı dikkate alınırsa, bu tür uygulamalara olan ihtiyaç daha iyi anlaşılacaktır.

Çalışmanın geliştirilmesi, daha verimli sonuçlar elde etmek açısından önemlidir. Bu konuda daha detaylı olarak aşağıdaki konularda çalışmalar yapmakta yarar görülmektedir :

- Doğal dil işleme metotları geliştirilerek Türkçe kelimelerin eklerden arındırılıp kök kelimelere ulaşılması sağlanabilir. Bu konuda başlamış ancak yarım kalmış projeler vardır.

- Durak kelimeleri üzerinde çalışarak, Türkçe'deki tüm durak kelimeleri çıkarılabilir.
- Sınıflandırma sonucunda elde edilen doküman örüntüsünde yer alan her bir doküman için bir kategori belirlenerek daha sonradan gelen yeni dokümanların otomatik olarak ilgili kategoriye otomatik olarak dahil edilmesi sağlanabilir.
- Doküman sayısı ve büyüklükleri arttıkça veri çok yüksek boyutlara ulaşabilmektedir. Bu nedenle boyut küçültme teknikleri ile daha hızlı sonuçlar elde etmek mümkün olabilecektir.

KAYNAKLAR

1. İnternet : Wikipedia, the free encyclopedia, "Arama motorları" http://tr.wikipedia.org/wiki/Arama_motoru (2007).

2. İnternet : Wikipedia, the free encyclopedia "Document classification" http://en.wikipedia.org/wiki/Document_classification#Techniques (2007).

3. Merkl, D., "Content-based document classification with highly compressed input data", ***Proceedings of 5th International Conference on Artificial Neural Networks (ICANN'95)***, Paris, 2: 239-244 (1995).

4. Dasgupta, S., Long, P.M., "Performance guarantees for hierarchical clustering", ***Journal of Computer and System Sciences***, 70(4): 555-569 (2005).

5. Lin, X., Soergel, D., Marchionini, G., "A self-organizing semantic map for information retrieval", ***Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval***, Chicago, Illinois, USA, 262-269 (1991).

6. Scholtes, J.C., "Unsupervised learning and the information retrieval problem", ***1991 IEEE International Joint Conference on Neural Networks (IJCNN'91)***, Seattle, WA, USA, 1: 95-100 (1991).

7. Merkl, D., Tjoa, A.M., "The representation of semantic similarity between documents by using maps: Application of an artificial neural network to organize software libraries", ***Proceedings of General Assembly Conference and Congress of the International Federation for Information and Documentation (FID'94)***, Saitama, Japan, 2: 145-149 (1994).

8. Segal, R.B., Kephart, J., O., "MailCat: An Intelligent Assistant for Organizing E-mail", ***Proceedings of the Third International Conference on Autonomous Agents***, Seattle, Washington, United States, 276-282 (1999).

9. Brücher, H., Knolmayer, G., Mittermayer, M.A., "Document classification methods for organizing explicit knowledge", ***Third European Conference on Organizational Knowledge, Learning and Capabilities***, Athens, 123-148 (2002).

10. Yang, Y., "Noise reduction in a statistical approach to text categorization", ***Proceedings of the 18th Ann. Int. ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)***, Seattle, Washington, United States, 256-263 (1995).

11. Fuhr, N., Hartmann, S., Lustig, G., Schwantner, M., Tzeras, K., "Air/x - a rule-based multistage indexing system for large subject fields", ***Proceedings of RIAO'91***, 606-623 (1991).

12. Yang, Y., "Expert network: effective and efficient learning from human decisions in text categorization and retrieval", ***Proceedings of the 17th annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'94)***, Dublin, Ireland, 13-22 (1994).

13. Lewis, D.D., Ringuette, M., "A comparison of two learning algorithms for text categorization", ***Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR'94)***, Las Vegas, USA, 81-93 (1994).

14. Tzeras, K., Hartman, S., "Automatic indexing based on bayesian inference networks", ***Proceedings of the 16th Ann. Int. ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)***, Pittsburgh, Pennsylvania, United States, 22-34 (1993).

15. Sağıroğlu, Ş., Beşdok, E., Erler, M., "Mühendislikte yapay zeka uygulamaları-I: yapay sinir ağları", ***Ufuk Kitap Kırtasiye Yayıncılık***, Kayseri, 23-116 (2003).

16. Merkl, D., Rauber A., "Document classification with unsupervised artificial neural networks", Soft Computing in Information Retrieval: Techniques and Applications, 50, Editors: F. Crestani and G. Pasi, Eds. Heidelberg, ***Physica-Verlag***, Germany, 102-121 (2000).

17. Wiener, E.D., Pedersen, J.O., Weigend, A.S., "A neural network approach to topic spotting", ***Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR'95)***, Las Vegas, US, 317-332 (1995).

18. Apte, C., Damerau, F., Weiss, S.M., "Towards language independent automated learning of text categorization models", ***Proceedings of the 17th Annual ACM/SIGIR Conference on Research and Development in Information Retrieval***, 23-30 (1994).

19. Moulinier, I., Raskinis, G., Ganascia, J., "Text categorization: a symbolic approach", ***Proceedings of the Fifth Annual Symposium on Document Analysis and Information Retrieval***, 87-99 (1996).

20. Cohen, W.W., Singer, Y., "Context-sensitive learning methods for text categorization", ***Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval***, 307-315 (1996).

21. Kohonen, T., “Self-organization of very large document collections: State of the art”, ***Proceedings of the 8th International Conference on Artificial Neural Networks***, Skovde, Sweden, 1: 65-74 (1998).

22. Roussinov, D.G., Chen, H., “A scalable self-organizing map algorithm for textual classification: a neural network approach to thesaurus generation”, ***Communication and Cognition - Artificial Intelligence,*** 15(1-2): 81-111, (1998).

23. Chen, H., Schuffels, C., Orwig, R., “Internet categorization and search: a self-organizing approach”, ***Journal of Visual Communication and Image Representation***, 7(1): 88-102 (1996).

24. Yang, Y., “An evaluation of statistical approaches to text categorization”, ***Information Retrieval***, 1: 69-90 (1999).

25. Merkl, D., “Document classification with self-organizing maps”, Kohonen Maps, ***Elsevier Science***, 183-197 (1999).

26. Dittenbach, M., Rauber, A., Merkl, D., “Uncovering hierarchical structure in data using the growing hierarchical self-organizing map”, ***Neurocomputing***, 48(1): 199-216 (2002).

27. Merkl, D., “Exploration of text collections with hierarchical feature maps”, ***Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval***, Philadelphia, 186-195 (1997).

28. Rauber, A., R., Tomsich, P., Merkl, D., “parSOM: a parallel implementation of the self-organizing map exploiting cache effects: making the SOM fit for interactive high-performance data analysis”, ***Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)***, Como, Italy, 6: 177-182 (2000).

29. Hammerstrom, D., “A VLSI architecture for high-performance, low-cost, on-chip learning”, ***In Proceedings International Joint Conference on Neural Networks***, IEEE Press, II:537-543 (1990).

30. Liu, J., Brooke, M., “A fully parallel learning neural network chip for real-time control”, ***In Intl. Joint Conf. on Neural Networks (IJCNN99)***, Washington, DC, 1:35-44 (1999).

31. Gerstl, P., Hertweck, M., Kuhn, B., “Text Mining: Grundlagen, Verfahren und Anwendungen”, ***Praxis der Wirtschaftsinformatik- Business Intelligence***, 39(222): 38-48 (2001).

32. Domingos, P., Pazzani, M., “On the Optimality of the Simple Bayesian Classifier under Zero-One Loss”, ***Machine Learning***, 29(2-3) : 103-130 (1997).

33. Lam, W., Low, K. F., Ho, C. Y., “Using a Bayesian Network Induction Approach for Text Categorization”, ***Proceedings of the 15th nternational Joint Conference on Artificial Intelligence***, 745-750 (1997).

34. Agrawal, R., Bayardo, R., Srikant, R., “Athena: mining-based interactive management of text databases”, ***Proceedings of the 7th International Conference on Extending Database Technology: Advances in Database Technology,*** Konstanz, Germany, 365-379 (2000).

35. Yang, Y., Chute, C., “An Example-Based Mapping Method for Text Categorization and Retrieval”, ***ACM Transactions on Information Systems***, 12(3) : 253-277 (1994).

36. Joachims, T., “Text Categorization with Support Vector Machines: Learning with Many Relevant Features”, ***Proceedings of the 10th European Conference on Machine Learning***, 137-142 (1998).

37. Hearst, M. A., Schoelkopf, B., Dumais, S., Osuna, E., Platt, J., “Trends and Controversies - Support Vector Machines”, ***IEEE Intelligent Systems***, 13(4) : 18-28 (1998).

38. Yang, Y., Liu, X.,”A Re-Examination of Text Categorization Methods”, ***Proceedings of the 22nd Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval,*** 42-49 (1999).

39. Siolas, G., D'Alché-Buc, F., “Support vector machines based on a semantic kernel for text categorization”, ***Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)***, Como, Italy, 5: 205-209 (2000).

40. Ng, H. T., Goh, W. B., Low, K. L., “Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization”, ***Proceedings of the 20th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval***, 67-73 (1997).

41. Ruiz, M. E., Srinivasan, P., “Automatic Text Categorization Using Neural Network”, ***Proceedings of the 8th ASIS SIG/CR Workshop on Classification Research***, 59-72 (1998).

42. Salton, G., Buckley, C., “Term-weighting approaches in automatic text retrieval”, ***Information Processing and Management***, 24(5) : 513-523 (1988).

43. Salton, G., McGill, M.J., “Introduction to modern information retrieval”, ***McGraw-Hill***, New York, 1-400 (1986).

44. Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R., “Indexing by latent semantic analysis”, ***Journal of the American Society for Information Science***, 41(6): 391-407 (1990).

45. Ritter, H., Kohonen, T., “Self-organizing semantic maps”, ***Biological Cybernetics***, 61(4): 241-254 (1989).

46. Kaski, S., “Dimensionality reduction by random mapping: fast similarity computation for clustering”, ***The 1998 IEEE International Joint Conference on Neural Networks***, Anchorage, Alaska, USA, 1: 413-418 (1998).

47. Honkela, T., Kaski, S., Lagus, K., Kohonen, T., “WEBSOM - self-organizing maps of document collections”, ***Proceedings of Workshop on Self-Organizing Maps***, Espoo, Finland, 310-315 (1997).

48. Berkhin, P., “Survey of Clustering Data Mining Techniques”, ***Teknik Rapor, Accrue Software***, 1-56 (2002).

49. Han, J., Kamber, M., Tung, A. K. H. “Spatial clustering methods in data mining: A survey”. ***Geographic Data Mining and Knowledge Discovery, Taylor and Francis*** , 188-217 (2001).

50. Jain, A., Dubes, R.. “Algorithms for clustering data”. ***Prentice-Hall***, Englewood Cliffs, NJ, 149-180 (1988).

51. Zahn, C. T., “Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters”, ***IEEE Transactions on Computers***, 20(1): 68-86 (1971).

52. Shepard, R. N., Arabie, P., “Additive clustering: Representation of Similarities as Combinations of Discrete Overlapping Properties”, ***Psychological Review***, 86: 87-123 (1979).

53. Johnson, S.C., “Hierarchical clustering schemes”, ***Psychometrika***, 2:241-254 (1967).

54. D'Andrade, R.G. , “U-Statistic hierarchical clustering”, ***Psychometrika***, 43(1):59-67 (1978).

55. MacQueen, J., “Some methods for classification and analysis of multiattribute instances”, ***The Fifty Berkeley Symposium on Mathematics, Statistics and Probability***, 1: 281-296 (1967).

56. Alsabti, K., Ranka, S., & Singh, V, “An efficient k-means clustering algorithm”, ***IPPS/SPDP Workshop on High Performance Data Mining***, IEEE Computer Society Press, 125-130 (1998).

57. Kohonen, T., “Self-organizing maps”, Springer Series in Information Sciences, ***Springer-Verlag***, New York, 30:1-426 (1997).

58. Koikkalainen, P., Oja, E., “Self-organizing hierarchical feature maps”, ***IJCNN International Joint Conference on Neural Networks***, 2: 279-284 (1990).

59. Vesanto, J., “SOM-based data visualization methods”, ***Intelligent Data Analysis***, 3(2): 111-126 (1999).

60. Kohonen, T., Kaski, S., Lagus, K., Salojarvi, J., Honkela, J., Paatero, V. Saarela, A., “Self organization of a massive document collection”, ***IEEE Transactions on Neural Networks,*** 11(3):574-585 (2000).

61. Rauber, A., Merkl, D., “Automatic labeling of self-organizing maps: Making a treasure-map reveal its secrets”, ***The Third Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining***, Beijing, China, 228-237 (1999).

EKLER

EK-1 Türkçe-ingilizce terim karşılıkları

Türkçe	**İngilizce**
Aşırı uygunluk	overfitting
Ayrıştırma	parsing
Bağlanım	regression
Danışmanlı doküman sınıflandırma	supervised document classification
Danışmansız doküman sınıflandırma	unsupervised document classification
Danışmansız yapay sinir ağları	unsupervised neural network
Destek vektör makinesi	Support vector machine
Doküman sınıflandırma	document classification
Gelişmemiş semantik indeksleme	latent semantic indexing
Hiyerarşik sınıflandırma	hierarchical classification
İkili ağırlıklandırma	binary weighting
İlgi haritası	Interest map
İstatistiksel öğrenme	statistical learning
Karar ağaçları	decision trees
Karar kuralları	decision rules
Kelime köklerinin bulunması	stemming
Kendinden düzenlenen haritalar	self-organizing maps
k-komşuluk	k-nearest neighbor
Normalizasyon	Normalization

EK-1 (Devam) Türkçe-ingilizce terim karşılıkları

Nöron süzgeci	Neural filter
Ortalama Bağlantı	average linkage
Öklit Uzaklığı	euclidean distance
Öklit Uzaklığının Karesi	Squared Euclidean Distance
Özellik çıkarma	feature extraction
Sınıf	Class
Tam Bağlantı	complete linkage
Tek tepeli	Unimodal
Tekil bağlantı	single linkage
Tekil değer ayrıştırması	singular-value decomposition
Terim frekansı	Term frequency
Ters doküman frekansı	Inverse document frequency
Veri kümeleme	clustering
Veri madenciliği	data mining
Yapay sinir ağları	artificial neural network

EK-2 Durak kelimeleri

acaba	bunda	iki	olduğunu	şu
altı	bundan	ile	olsa	şuna
altmış	bunu	ilgili	on	şunda
ama	bunun	ise	ona	şundan
ancak	çok	kadar	ondan	şunu
aynı	çünkü	katrilyon	onlar	tek
az	da	kez	onlardan	trilyon
bana	daha	kırk	onların	tüm
bazı	dahi	ki	onlari	üç
belki	de	kim	onu	var
ben	dedi	kimden	otuz	ve
bence	defa	kime	önce	veya
benden	diye	kimi	sanki	www
beni	doksan	mı	sekiz	ya
benim	dokuz	mi	seksen	yani
beş	dört	milyar	sen	yaptığı
bin	eden	milyon	sence	yedi
bir	elli	mu	senden	yetmiş
biri	en	mü	seni	yirmi
birkaç	etti	nasıl	senin	yüz
birkez	fazla	ne	siz	
birşey	gibi	neden	sizden	
birşeyi	hem	nerde	sizi	
biz	hep	nerede	sizin	
bizden	hepsi	nereye	son	
bizi	her	niçin	şey	
bizim	hiç	niye	şeyden	
bu	için	olarak	şeyi	
buna	içinde	olduğu	şeyler	

EK-3 Çalışmada kullanılan dersler ve içerikleri

BM206_Sayısal_Çözümleme

nümerik analizin mühendislikteki yeri. hatalar. sonlu fark işlemcileri. ileri fark, geri fark, merkezi farklar tabloları oluşturulması ve hata bulunması. enterpolasyon kavramı. newton-gregory ileri ve geri farklar enterpolasyon formülleri. lagrange enterpolasyonu. eğri uydurma (curve fitting) ve en küçük kareler yöntemi. sayısal integral yöntemleri. adi türevli diferansiyel denklemlerin (ordinary dif. equations) yaklaşık çözüm yöntemleri. iterasyon yöntemleri.

BM301_Mikroişlemciler

mikroişlemcilere ilişkin temel mantıksal kavramlar. bellek öğeleri. çalışma ilkeleri ve türlerin incelenmesi. adres uzayı ve bellek tasarımı. mikroişlemciler ve g/ç temel kavramları. kesilme yapıları, kesilme önceliği kodlayıcılar. doğrudan bellek erişimi. g/ç arabirimi tasarımı: koşut (8155), ardıl (8251) arabirimlerinin incelenmesi. 8085 işleyicisinin komut zaman çizeneklerinin incelenmesi. diğer 8/16 bit mikroişlemcilerin incelenmesi. mikroişlemci tabanlı dizge tasarımına giriş.

BM303_Bilgisayar_Organizasyonu

bilgisayar yönetimi ve tasarımı, işlemler, kod çözme ve çalıştırma, merkezi işlem ünitesi kontrol ve programlaması. mikroprogramlama ile kontrol ve donanım bazında kontrol. matematiksel işlem ünitesi ve çalışma mekanizması. veri girişi ve veri alma, taşıma yolu yapıları, çok zamanlı veri işleme ve taşıma. bellek kontrol ve adresleme teknikleri.

BM306_Bilgisayar_Mimarisi

bilgisayar sistemlerine bakışlar. diller, düzeyler ve sanal makineler. bilgisayar sistemlerinin düzenlenişi. sayısal mantık düzeyi, mikroprogramlama düzeyi. geleneksel makine dili düzeyi. işletim sistemi düzeyi. ileri bilgisayar mimarileri. azaltılmış komut seti bilgisayarları. paralel mimariler.

EM308_Nümerik_Analiz

nümerik hata analizi. denklem köklerinin bulunması. doğrusal denklem sistemleri. eğri uydurulması. interpolasyon. sayısal türev ve integrasyon. adi diferansiyel denklemlerin çözümleri. özdeğer ve özvektörler.

ENF102_C_Programlama_Dili

nesneler, fonksiyonlar, operatörler, kontrol deyimleri, döngüler, diziler, göstericiler, dinamik bellek yönetimi, yapılar.

EK-3 (Devam) Çalışmada kullanılan dersler ve içerikleri

İM343_Sayısal_Çözümleme

hata analizi. kök bulma yöntemleri. gauss yok etme yöntemi. matris tersinin bulunması. gauss-seidel yaklaştırma yöntemi. alt ve üst matrisleri bulma yöntemleri. en küçük kareler metodu. sayısal türev ve integral hesabı. adi diferansiyel denklemlerin sayısal çözümü. özdeğer problemi. eliptik ve parabolik kısmi diferansiyel denklemlerin sonlu farklarla çözümü.

K_ENF102_C_Programlama_Dili

nesneler, fonksiyonlar, operatörler, kontrol deyimleri, döngüler, diziler, göstericiler, dinamik bellek yönetimi, yapılar.

MM313E_Sayısal_Analize_Giriş

sonlu farklar, interpolasyon ve ekstrapolasyon, lineer olmayan denklemlerin çözümü, sayısal integrasyon ve türev, lineer denklem sistemleri ve matrisler, en küçük kareler metodu, adi diferansiyel denklem çözümleri, sınır değer problemlerine giriş, özdeğer ve öz vektörler.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : ALPDOĞAN, Yılmaz

Uyruğu : T.C.

Doğum tarihi ve yeri : 01.02.1970 Gaziantep

Medeni hali : Evli

Telefon : 0 (532) 483 47 77

e-mail : alpdogan@gmail.com

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Yıldız Teknik Ünv./ Bilgisayar Müh.	1992
Lise	Atatürk Lisesi	1988

İş Deneyimi

Yıl	Yer	Görev
1992-1995	Coşkunöz Holding A.Ş.	Uygulama Geliştirme Sor.
1996-1998	Siemens Business Services A.Ş.	SAP HR uzmanı
1999-2005	Havelsan A.Ş.	Teknik Proje Yönetici
2005-2007	Bimsa A.Ş.	SAP İş Zekası Danışmanı

Yabancı Dil

İngilizce

Hobiler

Kitap okumak, Futbol, Doğa aktiviteleri.