HİYERARŞİK KÜMELEME MODELİ KULLANAN WEB TABANLI BİR ÖDEV DEĞERLENDİRME SİSTEMİ

Erdinç UZUN[1], Cihat ERDOĞAN[2], Ahmet SAYGILI[3]

ÖZET

Ödevlerin öğrencilerin öğretim sürecinde önemli bir yeri bulunmaktadır. Klasik bir ödev değerlendirme sürecinde ödevin sadece doğru olup olmadığı değerlendirilmektedir. Ancak, ödevin öğretime daha iyi katkı verebilmesi için öğrencilerin yapmış oldukları intihallerin de göz önüne alınması gerekmektedir. İntihalleri ve intihallerin oranını tespit etme son derece zor bir ödev değerlendirme prosedürüdür. Bu çalışmada, bu prosedürü kolaylaştıracak doküman benzerliği ölçütlerini hiyerarşik kümeleme modeli ile bütünleştirebilen web tabanlı bir uygulama tanıtılacaktır. Bu uygulama, kimlerin benzer ödev yaptığını ve ödevlerin hangi oranda benzerliğe sahip olduğunu değerlendirme imkânı vermektedir. Bu uygulamanın doküman benzerliği hesaplanmasında Cosine, Jaccard ve Dice benzerlik ölçütleri denenmiştir. Diğer taraftan hiyerarşik kümeleme tarafında Tek Bağlantı, Tam Bağlantı ve Ortalama Grup olmak üzere üç farklı algoritma incelenmiştir. Önceki yıllara ait iki öğretim dönemini kapsayan 6 farklı öğretim üyesinin 18 farklı dersine ait 54 ödevi içeren bir test verisi oluşturulmuştur. Her ödev için doküman benzerlik ölçütlerinin ve kümeleme algoritmalarının çaprazlanmasından 9 farklı sonuç elde edilmiş ve hiyerarşik kümeleme algoritmalarının ne kadar iyi olduğunu test etmek için cophenetic korelasyon katsayıları hesaplanmıştır. Sonuçlar analiz edildiğinde, doküman benzerliğinde Jaccard ölçütü ve hiyerarşik kümelemede Ortalama Grup algoritmasının en uygun ödev değerlendirme çifti olduğu görülmüştür.

Anahtar Kelimeler: Hiyerarşik kümeleme, Doküman benzerliği, Yerel İntihal Tespiti, Yazılım Geliştirme

A WEB-BASED ASSIGMENT EVALUATION APPLICATION USING HIERARCHICAL CLUSTERING MODEL

ABSTRACT

Assignments are one of the most important parts of education process of students. In the classical assignment evaluation process, an assignment can be evaluated whether it is correct or not. However, for the assignments to give better contribution to education, plagiarisms committed by students should be considered. Detection of plagiarism and its extent are extremely difficult assignment evaluation procedures. In this study, in order to facilitate this procedure, a web-based application, which can combine document similarity measures with hierarchical clustering model, is introduced. This application gives the opportunity to evaluate which students submit similar assignments and the assignments' similarity degree. Cosine, Dice and Jaccard similarity measures have been investigated in terms of document similarity calculation of this application. On the other hand, three different algorithms including Single Linkage, Complete Linkage and Average Group are examined in hierarchical clustering side. Test data which covers two education period of previous years and contains 54 different assignments of 18 different courses of 6 lecturers, are created. By using document similarity methods and hierarchical clustering algorithms, 9 different cophenetic correlation coefficients are obtained for each assignment and cophenetic correlation coefficients are calculated to test how well hierarchical clustering algorithms are . When the results were analyzed, it was discovered that Jaccard measure in document similarity and Average Group algorithm in hierarchical clustering is the best matching assignment evaluation pair.

Keywords: Hierarchical Clustering, Document similarity, Plagiarism Detection, Software Development

* Bu çalışma, NKUBAP.00.17.AR.13.15 protokol nolu 'Akıllı Ders Yönetim Sistemi ile Programlama Ödevleri için İntihal Tespiti Uygulaması' başlıklı projemiz kapsamında Namık Kemal Üniversitesi - BAP tarafından desteklenmiştir.

[1] Yrd.Doç.Dr., Namık Kemal Üniversitesi, erdincuzun@nku.edu.tr

[2] Arş. Gör., Namık Kemal Üniversitesi, cerdogan@nku.edu.tr

[3] Arş. Gör., Namık Kemal Üniversitesi, asaygili@nku.edu.tr

Giriş

Ödev, eğitim kalitesinin arttırılmasında öğretimin en önemli öğelerindendir. İyi yapılan bir ödevin öğrencinin konu hakkındaki kişisel gelişimini arttırması kaçınılmazdır. Ancak, bazı öğrenciler zaman darlığı, sınavlara bağlı not sistemi ve dersi gereksiz görme gibi sebeplerden dolayı ödev hazırlamada intihal yoluna gitmeyi tercih etmektedirler (Arda ve diğerleri, 2007). Bu durumda ödevi değerlendiren kişinin sadece ödevlerin doğruluğunu değerlendirilmesi yeterli olmayacaktır. Ödevi değerlendiren kişi intihal yapan öğrencileri bularak ödev değerlendirme işlemini daha sağlıklı bir hale getirebilir. Ancak, intihal belirleme süreci ödevin doğruluğunun belirlenmesi işleminden çok daha zor ve zaman alıcı bir iştir. Bu çalışmada, bu zor ve zaman alıcı işin çok kısa sürelerde gerçekleştirilmesini sağlayacak web tabanlı bir uygulama ve bu uygulamanın çıktıları anlatılacaktır.

İntihal tespiti, sadece ödev alanında değil akademik alanda tez veya makale değerlendirilmesi (Arda, 2003) ve diğer taraftan internet ortamındaki alıntı haberlerin veya web sayfalarının bulunması gibi birçok farklı alanda kullanılmaktadır. Ödev intihali, sınıf içinde arkadaştan yapılan ödev intihali ve başka kaynaklardan yapılan ödev intihali olmak üzere ikiye ayrılabilir. Tez, makale, haber siteleri ve web sayfaları gibi kaynaklar üzerinde intihalleri / alıntıları bulmak için verilerin depolanması ve hızlı olarak işlenebilmesi için büyük sunuculara ihtiyaç vardır. Büyük kaynaklara sahip olan ve farklı değerlendirmelerin yapılabilmesine olanak sağlayan internet üzerinde birçok yazılım göze çarpmaktadır[4]. Örneğin bu yazılımlardan ithenticate.com 17 bilyon web sayfası ve 86 milyon araştırma makalesi gibi büyük bir kaynağa sahiptir. Ancak, bu yazılımlar ticari, algoritmaları gizli ve çok büyük kaynaklara ihtiyaç duymaktadırlar. Bu noktada, günümüzde yaygınlaşan açık kaynak kodlu yazılımlar düşünülebilir. Bunlardan biri WCopyfind[5] yazılımıdır. Bu yazılım sadece iki dokümanı inceleyip benzer kısımların altını çizerek bir html raporu üretmektedir. Diğer bir uygulama ise 1994 yılında Stanford Üniversitesi'nde geliştirilmeye başlanan ve halen güncellemeleri çıkan MOSS (for a Measure of Software Similarity) uygulamasıdır (Schleimer, Wilkerson ve Aiken, 2003). Bu uygulama, yazılım kodları arasındaki benzerlikleri için bir değer üreten ve intihalleri tespit etmekte kullanılan bir programdır. Ancak, sadece üyelik sistemi üzerinden kullanılabilen ve web servisi bazında çalışan bir uygulamadır. Bu web servisi ticari olarak satılmaktadır. Bu uygulama benzer olarak özellikle metin benzerliklerinin araştırılmasına yönelik birçok algoritma literatürde tanıtılmıştır (Donaldson, Green ve Sposato, 1981; Wise, 1992; Heintze, 1996; Lyon, Malcolm ve Dickerson, 2001; Yerra ve Ng, 2005; Samuel ve Zelda, 2006; Lingxiao, Su ve Chiu, 2007). Liu, Xu ve Ouyang (2015) bilgisayar bilimi eğitiminde bilgisayar kodlarında ufak değişiklikler yaparak öğrencilerin klasik intihal tespit sistemlerini atlatabildiği belirtmişlerdir. Geliştirdikleri "Gelişmiş Longest Common Subsequence" algoritması ile ufak değişiklikleri göz ardı ederek intihalleri tespit edebilen yeni bir algoritma önermişlerdir. Ceglarek (2013) doğal dil işlemeyi algoritması katarak geliştirdiği SHAPD2 algoritmasının etkinliğini göstermiştir. Ancak geliştirilen algoritmalar genelde iki metin arasındaki benzer kısımlarının veya benzerlik oranının bulunması üzerinedir. Bu çalışmada geliştirilen uygulama dendrogram denilen bir ağaç grafiği şeklinde tüm öğrencilerin ödevleri arasındaki yakınlığı görsel olarak görmemize olanak sağlayacaktır. Bu sayede ikişerli gruplar halinde ödevler karşılaştırılması yerine tüm ödevlerin bir grafik üzerinden incelenebilmesi mümkün olacaktır. Bu grafiğin elde edilebilmesi için en uygun yapı hiyerarşik kümeleme modelidir.

Hiyerarşik kümeleme tıp, ekonomi ve bunlar gibi birçok farklı alan problemlerinin çözümünde kullanılmaktadır. Örneğin, Fung ve diğerleri (2004) hiyerarşik doküman kümelemede büyük miktarda veride ve anlamlı kümeleme etiketi oluşturmada karşılaşılan zorlukları aşmayı anlatmışlardır. Çiftçi (2011) beyin bölgelerini işlevsel olarak gruplandırmada için kullanmıştır. Karabulut ve diğerleri (2008) ise çalışmalarında Türkiye'deki illerin sosyo-ekonomik benzerliklerini bu kümeleme metodu üzerinden incelemişlerdir. Yaptığımız bu çalışma, Fung ve diğerlerine (2004) benzer şekilde doküman kümeleme üzerine olacak fakat onların belirttiği gibi kelimelerin bir kısmı değil öğrencinin ödev içinde kullandığı tüm kelimeler dikkate alınacaktır. Genelde bu tür çalışmalarda kümeleme veya dendrogram çizimi aşamalarında Matlab veya Phyton içindeki hazır kütüphaneler tercih edilir. Bu uygulamada, ödevler web üzerinden öğretim üyesi tarafından verilmekte, öğrenciler ödevi internetten öğretim üyesinin belirlediği süre içinde göndermekte ve ödev gönderme süreci bittiğinde uygulama tüm ödevleri alıp öğretim üyesi için benzerlikleri bulmaktadır. Uygulamada geliştirilen tüm modüller .Net Framework platformunda C# programlama dili ile yazılmıştır. Geliştirilen uygulamanın tüm modülleri tarafımızca kodlandığı için farklı doküman benzerlik ölçütleri ve hiyerarşik kümeleme algoritmalarının getirileri de incelenebilmiştir.

[4] www.ithenticate.com, www.turnitin.com, www.jplag.de ve www.plagiarism.com gibi ticari olarak intihal tespit eden sistemler vardır. (Son erişim 20 Eylül 2016 tarihinde yapılmıştır.)

[5] plagiarism.bloomfieldmedia.com (Son erişim 20 Eylül 2016 tarihinde yapılmıştır.)

İkinci bölümde geliştirilen intihal tespit uygulaması hakkında bilgi verilecek ve kelime frekanslarının elde edilmesi, doküman benzerliği, hiyerarşik kümeleme ve dendrogram çizimi örnek veriler üzerinden açıklanacaktır. Üçüncü bölümde test verisi tanıtılacak ve kümelemenin ne kadar iyi olduğunu test etmek için cophenetic korelasyon katsayısına bakılacaktır. Dördüncü bölümde test sonuçları verilecek ve dendrogram üzerinden ödev göndermedeki iyileşmeler gösterilecektir. Son bölümde sonuçlar ve gelecek çalışmalar yer almaktadır.

Uygulama Hakkında

Geliştirilen intihal tespit uygulaması bölümümüzde geliştirilen ADYS[6] (Akıllı Ders Yönetim Sistemi) üzerinde çalışmaktadır. ADYS'nin başlıca özellikleri:

- Web üzerinden öğrencilerin ders notlarına ulaşması ve ödev gönderebilmesi
- Öğretim üyelerinin kolayca dünya veya sadece kendi öğrencileri ile ders notu paylaşabilmeleri
- Öğretim üyelerinin ödev verebilmesi ve ödev süresini kolayca belirleyebilmesi
- Öğretim üyelerinin gönderilen ödevler arasındaki benzerlikleri kolayca tespit edebilmesi

Bu çalışmada, ADYS içindeki birçok kısımdan sadece gönderilen ödevlerin intihal tespiti için hazırlanması ve hiyerarşik kümeleme algoritması kısmı anlatılacaktır.

Ödev değerlendirme süreci ödev bitiş süresi bittikten sonra başlar ve 4 ana bölümden oluşur.

- Gelen ödev dokümanları içinden kelimelerin ve kelime frekanslarının çıkarılması
- Metinler üzerinden dokümanların benzerliklerinin bulunması
- Bu benzerlik değerleri üzerine Hiyerarşik kümelemenin uygulanması
- Dendrogram üretimi ve dendrogram üzerinden öğretim üyesinin küme sayısının belirlenmesi

Kelimelerin ve Kelime Frekanslarının Elde Edilmesi

Genelde bilgisayar ortamında hazırlanan ödevler bir çok dosya tipinde (örneğin .c, .cs, .java, .py, .php, .docx, .doc, .xls, .xlsx, .ppt, .pptx, .pdf, .zip ve .rar gibi) gönderilmektedir. Gönderilen ödevler MySQL veritabanında BLOB formatında depolanmaktadır. Ödevlere veritabanında depolandığı için yetkilendirme ve ödevleri görebilme imkânı kullanıcı tabanlı verilebilmektedir. Ödevler sadece öğretim üyesi ve ödevi gönderen öğrenci ulaşabilmektedir. Ödev süresi dolduktan sonra sunucu üzerinde çalışan zamanlayıcı bir uygulama sayesinde değerlendirme süreci başlar. Ödev sürecinde öncelikle her öğrencinin ödevi bir doküman havuzuna atılır.

$$D = \{d_1, d_2, d_3, ..., d_n\}$$

D öğrenci ödevleri olmak üzere *d*'ler her bir öğrencinin dokümanını temsil etsin. Değerlendirme sürecinde öncelikle *d* içinden metinler elde edilir. *d* sıkıştırılmış (.zip veya .rar) dosya ise bir klasör içersine tüm dosyalar açılır, eğer açılan klasörde sıkıştırılmış dosya var ise açma işlemi rekürsif bir şekilde sıkışmış dosya kalmayıncaya kadar devam eder. Tüm dosyalarda yer alan metinlerden bir kelime havuzu oluşturulur.

$$d_i = \{w_1, w_2, w_3, ..., w_n\}$$

d_i bir öğrencinin dosyası/dosyaları olmak üzere *w* gibi kelimelerden oluşur. İki öğrencinin ödevindeki kelimelerin karşılaştırılması için iç içe döngüye ihtiyaç vardır. Birinci döngüden elde edilen kelimeler ikinci döngüde elde edilen kelimelerle karşılaştırılır ve örtüşen kelime sayısı hesaplanır. Bu karşılaştırma işlemi $O(N^2)$ karmaşıklığa sahiptir. Bu karmaşıklığı azaltmak için Hash fonksiyonları kullanılabilir. Hash fonksiyonlar anahtar ve kelime çiftlerinden oluşur ve arama karmaşıklığı O(1)'dir. Bunun için kelime havuzu hazırlanırken veriler hash fonksiyonuna atılır.

$$hd_i = \{(w_1, f_1), (w_2, f_2), (w_3, f_3), ..., (w_n, f_n)\}$$

[6] ADYS uygulamasına tüm üniversite tarafından http://bilgmuh.nku.edu.tr/adys/ adresi üzerinden erişilebilmektedir.

Burada *w* anahtar hash değeri ve *f* kelimenin geçme frekansı olmak üzere kelime havuzu aranır. Bu sayede iki dokümanı karşılaştırma maliyeti O(N)'e düşürülmüş olur. Doküman havuzları hazırlandıktan sonra doküman benzerliklerinin bulunmasına geçilir.

Doküman Benzerliği

Metinler elde edildikten sonra ikinci aşamada tüm öğrenci dokümanları birbiri ile karşılaştırılıp bir sayısal değer üretilmelidir. Bu sayısal değer üretimi için Bilgi Erişimi (Information Retrieval) alanında kullanılan Cosine Benzerliği (Cosine Similarity), Jaccard Benzerlik Katsayısı (Jaccard Similarity Coefficient) ve Dice'in Katsayısı (Dice's Coefficient) olmak üzere üç farklı benzerlik ölçütü denenmiştir. Bu üç benzerlik ölçütü, metin madenciliğinde dokümanların karşılaştırılmasında ve veri madenciliğinde ise kümeler arasındaki uyumu ölçmede sıklıkla kullanılmaktadır (Tan, Steinbach ve Kumar, 2005).

Cosine benzerlik metodunda, iki vektör arasındaki kosinüs açısının ölçümünden elde edilen değer benzerlik için kullanır. Bu değer aşağıdaki formül yardımıyla hesaplanmaktadır.

$$\cos\_sim(d_X, d_Y) = 1 - \frac{d_x \cdot d_Y}{|dx||d_Y|} = 1 - \frac{x_i x_j + y_i y_j}{\sqrt{x_i^2 + x_j^2}\sqrt{y_i^2 + y_j^2}}$$

Burada, d_X ve d_Y farklı iki doküman olmak üzere $x_i x_j$ - $y_i y_j$ benzer kelime sayısını ve x_i - x_j - y_i - y_j dokümanlardaki tüm kelime sayısını vermektedir. Jaccard Benzerlik Katsayısı, benzer eleman sayısının toplam kelime sayısına oranıdır.

$$jac_{sim}(d_X, d_y) = 1 - \frac{d_X \cap d_y}{d_X \cup d_y} = 1 - \frac{x_i}{x_j + y_j - x_i}$$

Burada, x_i - y_i benzer kelime sayıları eşit olduğu için sadece x_i alınmıştır. Başka bir deyişle x_i iki dokümandaki benzer kelime sayıdır. Dice katsayısı, "Lee Raymond Dice" tarafından farklı bir alan için önerilmiş fakat doküman benzerliğinde kullanılan bir ölçüttür (Dice, 1945).

$$dice_{sim}(d_X, d_y) = 1 - \frac{2|d_X \cap d_y|}{|d_X| + |d_y|} = 1 - \frac{2x_i}{x_j + y_j}$$

Burada, benzer kelime sayısı toplam kelime sayısından çıkarma işlemi yerine 2 ile çarpılmaktadır. Tüm ölçütlerde [0,1] arasında bir değer üretilmektedir. Değer 0'a yakın çıktığında benzerlik fazla iken değer 1'e yaklaştığında benzerliğin az olduğu anlamına gelmektedir. Testler bölümünde tüm ölçütlerin incelemesi yapılacak ve ödev değerlendirme işlemi için en uygun ölçüt belirlenecektir.

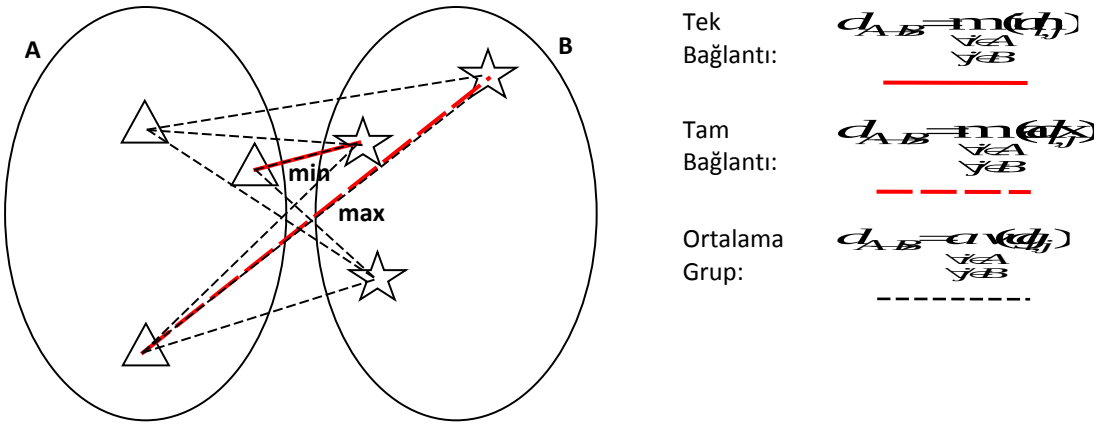
Hiyerarşik Kümeleme

Problemimizin en zor yönlerinden biri ödevdeki benzerlik miktarına karar verip öğrencinin benzer ödev yapıp yapmadığının tespitidir. Örneğin öğrenci bir ödev için %60 benzer ödev yaptıysa kopya çekmemiş olabilir ancak başka bir ödev için kopya çekmiş olabilir. Bazı programlama ödevlerinde hazır fonksiyonlar ve kodların bir kısmı öğrenciye verilmektedir. Öğrenci boş fonksiyonları hazırlayıp göndermektedir. Bu durumda öğrenci ödev benzerlikleri %80'e kadar çıkabilmektedir. Ödev sistemini verilen ödeve göre esnek olarak yapabilmek ve görsel olarak sonuçlardan benzerlik kümelerine karar verebilmek için en uygun yapı hiyerarşik kümelemedir.

Hiyerarşik kümeleme, diğer kümeleme yöntemlerine göre daha bilgilendirici bir çıktı üretir. Hiyerarşik kümelemede, küme sayısının önceden belirlenmesi gerekmemektedir. Hiyerarşik kümeleme, birleştirici ve ayrıştırıcı hiyerarşik kümeleme yöntemleri olmak üzere iki türe sahiptir. Birleştirici hiyerarşik kümelemede başlangıçta her bir değer bağımsız bir küme olarak değerlendirilir ve bu değerler çeşitli algoritmalarla birleştirilip her aşamada bir üst küme oluşturulur. Ayrıştırıcı kümelemede ise başlangıçta tüm değerler bir küme olarak değerlendirilip yine çeşitli ayrıştırma algoritmalarıyla alt kümeler elde edilir. Genelde literatürde, bu çalışmada olduğu gibi birleştirici kümeleme yöntemi tercih edilmektedir (Manning, Raghavan ve Schütze, 2008).

Hiyerarşik kümelemede, öncelikle özellikler arası uzaklıklar hesaplanır. Bu çalışmada uzaklık değeri olarak doküman benzerliği kullanılmıştır. Doküman benzerliğinden elde edilen sayısal değerler üzerinden farklı kümeleme algoritmalarına göre kümeler üretilebilir. Bu noktada Tek Bağlantı, Tam Bağlantı ve Ortalama Grup algoritmalarından biri kullanılabilir. Tek bağlantı kümeleme, her adımda en küçük mesafeye sahip en yakın üyeleri kullanarak iki kümenin birleştirilmesidir. Bu algoritmaya, en yakın komşu algoritması da denir. Tam bağlantı kümeleme, tek bağlantının tam tersine her adımda en büyük mesafeye sahip en uzak üyeler kullanılarak iki kümenin birleştirilmesidir. Bu algoritmaya bu sebepten dolayı en uzak komşu algoritması da denir. Ortalama grup algoritması ise iki karşılıklı küme arasındaki tüm mesafelerin ortalamasıdır. Bu algoritma ile tek bağlantı ve tam bağlantı algoritmalarında elde edilen değerlerin arasında bir değer elde edilir (Manning, Raghavan ve Schütze, 2008).

Şekil 1'de, A ve B adında iki küme verilmiştir. Öncelikle, iki kümenin her elemanının mesafeleri hesaplanır. Sonra istenilen algoritmaya göre min, max veya avg değerleri hesaplanır. Şimdi, ödev değerlendirme için geliştirilen hiyerarşik kümeleme algoritmasını inceleyelim. Aşağıda bu algoritmanın kaba kodu gösterilmektedir.



Şekil 1: Tek Bağlantı, Tam Bağlantı ve Ortalama Grup Algoritmaları

1. Küme Sayısı: Öğrenci Sayısı;

2. Benzerlik Matrisi: Öğrenci Ödev Benzerlik oranlarını içeren matris

3. while Küme Sayısı = 1 **do**

4. Benzerlik Matrisi içindeki minimum değeri bul

5. Minimum değere ait satır ve sütunu matriste birleştir

6. Matriste işlem yapılan satır ve sütun değerini güncelle (min, max veya avg)

7. Küme sayısı bir azalt

8. end

Algoritma 1: Ödev Değerlendirme için Hiyerarşik Kümeleme Algoritması

Algoritma 1, küme sayısı tek oluncaya kadar devam eder. Başlangıç aşamasında (öğrenci sayısı X öğrenci sayısı) kadar bir matris yaratılıp her döngü sonunda matris boyutu bir azalır. Döngü işlemi başlarken öncelikle matris içindeki minimum değer bulunur. Bu Minimum değer tek bağlantı (min) algoritması ile karıştırılmamalıdır. Minimum değere ait satır ve sütun bilgilerinden istenilen üç algoritmadan birisi kullanılarak yeni matris oluşturulur. Şimdi, hiyerarşik kümeleme algoritması daha iyi anlaşılması için sayısal bir örnek üzerinden anlatılacaktır. Bu örnekte 6 öğrencinin ödevleri arasında doküman benzerliğinden elde edilen değerler kullanılacaktır.

Başlangıç Durumu: Ödev Benzerlik Oranları

	1	2	3	4	5	6
1	0.000	0.458	0.177	0.419	0.400	0.161
2	0.458	0.000	0.486	0.263	0.263	0.462
3	0.177	0.486	0.000	0.000	0.300	0.004
4	0.419	0.263	0.300	0.000	0.000	0.276
5	0.419	0.263	0.300	0.000	0.000	0.276
6	0.161	0.462	0.004	0.276	0.276	0.000

Birinci Döngü Minimum Seçimi

	1	2	3	4	5	6
1						
2	0.458					
3	0.177	0.486				
4	**0.419**	**0.263**	**0.300**			
5	**0.419**	**0.263**	**0.300**	**0.000**		
6	0.161	0.462	0.004	**0.276**	**0.276**	

Şekil 2: Hiyerarşik kümeleme algoritması başlangıç durumu - 1

Şekil 2'de, matris incelendiğinde matristeki değerlerin yarısının tekrar olduğu görülmektedir. Bu yüzden i < j türünden bir şart ile matrisin sadece bir kısmını incelemek yeterlidir. Kalan elemanlar arasında en küçük değer 4 ve 5. öğrenci ödevlerinin benzerlik oranını gösteren 0 değeridir. Başka bir deyişle bu iki öğrencinin ödevleri tamamen aynıdır. 4 ve 5'ten yeni bir küme üretilebilir. Şimdi algoritmadaki 5. ve 6. adımları uygulayalım. (Şekil 3)

5. Adım

	1	2	3	4, 5	6
1					
2	0.458				
3	0.177	0.486			
4,5	**???**	**???**	**???**	**????**	
6	0.161	0.462	0.004	**???**	

???: Algoritma tarafından belirlenecek değerler

6.Adım

	1	2	3	4, 5	6
1					
2	0.458				
3	0.177	0.486			
4, 5	**0.419**	**0.263**	**0.300**		
6	0.161	0.462	0.004	**0.276**	

Şekil 3: Hiyerarşik kümeleme algoritması başlangıç durumu - 2

(4, 5) kümesi yapıldıktan sonra yeni değerler hesaplanmalıdır. Bu hesap sırasında Şekil 1'deki matris bilgileri kullanılır. Bu aşamada min, max veya avg algoritmalarından biri kullanılır. Bu örnekte min değerini kullanılacaktır. Min yani Tek Bağlantı hesabı:

$$d_{(4,5)1} = \min(d_{41}, d_{51}), \quad d_{(4,5)2} = \min(d_{42}, d_{52}), \quad d_{(4,5)3} = \min(d_{43}, d_{53}),$$
$$d_{(4,5)6} = \min(d_{46}, d_{56})$$

Tek bağlantıya göre bu kümelerden en küçük değerler seçilir. İki ödev birbiriyle aynı olduğu için tüm benzerlikler aynı çıkmıştır. Şimdi ikinci döngüye geçelim. Yeni oluşan matriste en küçük değer 0.004'tür. Başka bir deyişle 3 ve 6'ıncı öğrencilerden yeni bir küme oluşturulacaktır. (Şekil 4)

2. Döngü Başlangıç

	1	**2**	**3**	**4, 5**	**6**
1					
2	0.458				
3	**0.177**	**0.486**			
4, 5	0.419	0.263	**0.300**		
6	**0.161**	**0.462**	**0.004**	**0.276**	

2. Döngü Sonu

	1	**2**	**3,6**	**4, 5**
1				
2	0.458			
3, 6	**0.161**	**0.462**		
4, 5	0.419	0.263	**0.276**	

Şekil 4: Hiyerarşik kümeleme algoritması 2. Döngü

Her döngünün sonunda küme sayısı 1 azalır ve küme sayısı 1 oluncaya kadar aynı işlemler tekrar eder. Her adım sonunda bir birleşim işlemi olur. Tablo 1'de adım adım oluşan küme, seçilen minimum değer ve küme ilişkisi verilmiştir.

Tablo 1: Algoritma Adımları ve Sonuçlar

Adım	**Kümeler**	**Minimum**
0.	1, 2, 3, 4, 5, 6	-
1.	4, 5	0.000
2.	3, 6	0.004
3.	1, (3, 6)	0.161
4.	2, (4, 5)	0.263
5.	(1, (3, 6)), (2, (4, 5))	0.276

Algoritma içindeki minimum değer dendrogram çiziminde ve test aşamasındaki cophenetic korelasyon matrisinin oluşturulmasında kullanılacaktır. Bu korelasyon matrisi sayesinde kullanılan doküman benzerliği ölçütlerinin ve hiyerarşik kümeleme algoritmalarının etkinliği test edilecektir.

Dendrogram ve Küme Sayısı

Hiyerarşik kümelemenin en güzel taraflarından biri bir uzman tarafından belirlenen kesme değerine göre kümelerin belirlenmesidir. Bu uygulamada uzman öğretim üyesi olmaktadır. Bu belirleme işleminde hiyerarşik kümeleme çıktılarını en iyi şekilde yorumlamak için dendrogram kullanılır. Öğretim üyesi bu dendrograma bakıp ve benzerliği yakın öğrencilerin ödevlerini inceleyip bir kesme değerine karar verir. Bu kesme değerine göre de kümeler şekillenir. Uygulamamızda kesme işlemine göre öğrenciler farklı renklerle gruplanmaktadır. Öğretim üyesi benzer grupların ödevlerini indirip rahat bir şekilde inceleyebilmektedir. Dendrogram çizimi Algoritma 2'deki gibidir.

Uygulamamızda geliştirdiğimiz C# ortamında dendrogram çizimi için hazır bir kütüphaneye yoktur. Bunun için C# System.Drawing kütüphanesini kullanan resim üretici bir kütüphane geliştirilmiştir. Üretilen resim sunucudan istemciye gönderilir. İstemci tarafında öğretim üyesi kesme değerini değiştirdiğinde çizim işlemi tekrar sunucu da yapılır ve istemciye gönderilir. Bu çizim işleminin hızlı şekilde yapılabilmesi için Tablo 1'deki değerler kullanılır. Algoritma 2'de 5. satırda dersin öğretim üyesi kesme değeri belirlemektedir. Bu kesme değeri oluşacak küme sayısını belirler. Oluşan dendrogram Şekil 5'de gösterilmiştir.

1. Öğrenci sayısına uygun beyaz bir çizim alanı oluştur.

2. Maksimum değer: Kümeleme sonuçlarındaki maksimum sayı.

3. Maksimum değere göre üst ve alt cetvelleri x koordinatında oluştur.

4. y koordinatına öğrenci numaralarını kümeleme sonuçlarına göre yerleştir.

5. Dersin öğretim üyesinin belirlediği kesme değerini çiz

6. while tüm kümeleme sonuçları bitinceye kadar **do**

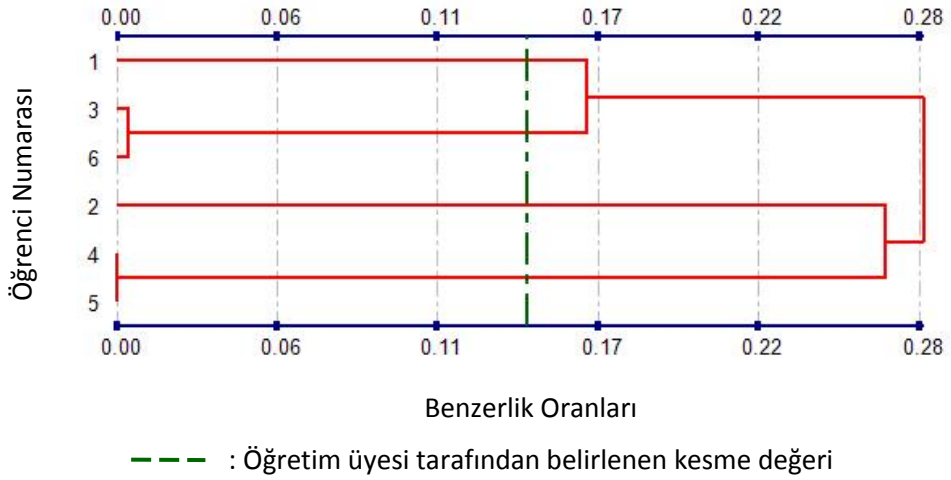
7. Kümeleme sonucundaki minimum değere göre uygun koordinatları hesapla

8. Bulunan koordinata kadar iki kümeleme sonucundan x ekseninde birer doğru çiz

9. İki doğrunun sonunu birleştir

10. end

Algoritma 2: Dendrogram çizimi Algoritması



Şekil 5: Örnek veriler için oluşturulan dendrogram

Tablo 1'deki en büyük değer 0.28 olarak alınmıştır. Öncelikle (4, 5)'in 0 değeri çizilmiş ardından (3, 6)'nın 0.004 değeri çizilmiştir. Bu şekilde çizim algoritması tüm kümeleme sonuçlarının çizimini yapar. Dendrogram sayesinde öğretim üyesi çok rahat bir şekilde yapılan ödevleri yorumlayabilir. Örneğin 4 ve 5 nolu öğrencilerin aynı ödevi gönderdiği 3 ile 6 nolu öğrencilerin ise çok az bir farklılık içerdiğini görmektedir. 1 nolu öğrenci ise 3 ve 6 nolu öğrencilere benzer bir ödev göndermiştir. Bu benzerliğin öğretim üyesi tarafından intihal oluşturacağı düşünülürse 0.161'den büyük bir kesme değeri seçilebilir. Eğer farklı bir ödev yaptığı düşüncesinde ise şekildeki gibi 0.15 kesme değerini seçebilir. Bu kesme değeri kullanıldığında 1, (3, 6), 2, (4, 5) olmak üzere 4 adet küme oluşmaktadır.

Test Verisi ve Kriterleri

Uygulamadaki doküman benzerlik ölçütlerinden ve hiyerarşik kümeleme algoritmalarından hangi çiftin en iyi sonuç verdiği bulmak için 54 adet ödevden oluşan bir veri seti hazırlanmıştır. Bu veri seti, 2012-2013 ve 2013-2014 öğretim dönemini kapsayan 6 öğretim üyesine ait 18 farklı dersten oluşmaktadır. Genelde, Nesneye Yönelik Programlama, İşletim Sistemleri, Web Programlama, C Programlama Dili ve Sayısal Analiz gibi kod tabanlı dersler içermektedir. Bu kodlar, C#, Java, HTML, PHP ve Javascript gibi dillerle yazılmıştır.

Hiyerarşik kümelemenin ne kadar iyi olduğunu test etmek için hiyerarşik kümelemeden çıkan sonuçlar ile başlangıç değerlerinin cophenetic korelasyon katsayısına bakılır.

Burada, $y \in Y$ başlangıç matrisi ve $z \in Z$ hiyerarşik kümelemeden çıkan sonuçlardan elde edilen matristir. *ks* matris içinde *i<j* bölümünde bulunan eleman sayısıdır. Bu değerler üzerinden cophenetic korelasyon katsayısı (*c*) hesaplanır. Hiyerarşik kümeleme işlemindeki örnek veriler üzerinden cophenetic korelasyon katsayısını bulalım. (Şekil 6)

Y matrisi

	1	2	3	4	5	6
1						
2	0.458					
3	0.177	0.486				
4	0.419	0.263	0.300			
5	0.419	0.263	0.300	0.000		
6	0.161	0.462	0.004	0.276	0.276	

Z Matrisi

	1	2	3	4	5	6
1						
2	0.276					
3	**0.161**	0.276				
4	0.276	0.263	0.276			
5	0.276	0.263	0.276	0.000		
6	**0.161**	0.276	0.004	0.276	0.276	

Şekil 6: Cophenetic korelasyon katsayısına için kullanılan matrisler

Şekil 6'teki Y matrisi başlangıç değerlerini içerir. Z matrisi oluşturulurken ise Tablo 1'deki değerler göz önüne alınır. Örneğin 1, (3, 6) değeri 0.161 Z matrisindeki (1,3) ve (1,6)'ya yazılır. Bu örnek için c değeri %87.02'dir. Hiyerarşik kümelemede ölçütlerin ve algoritmaların uygunluğu c değerinin yüksek çıkması ile ölçülebilir.

Testler

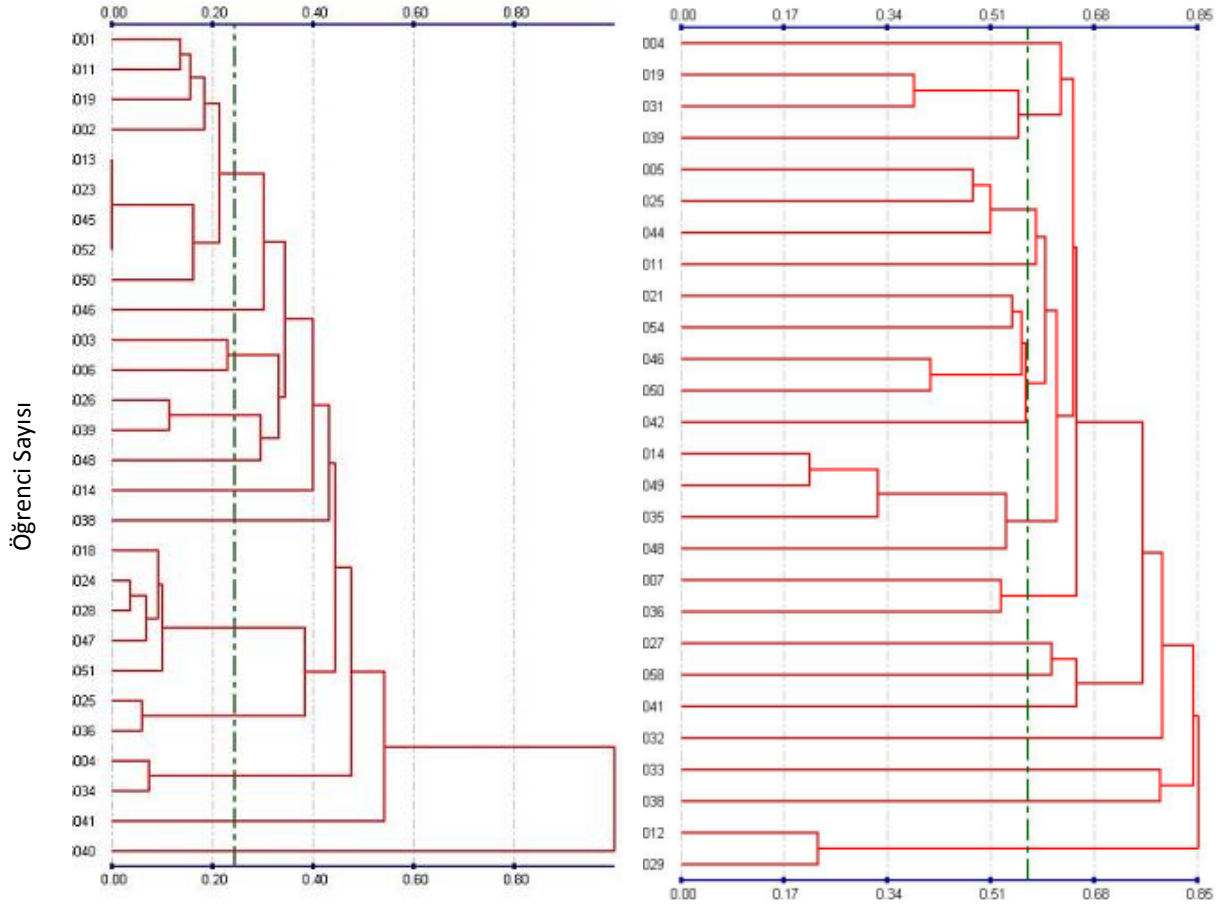
Sonuçların değerlendirilmesi açısından iki ayrı test yapılmıştır. Birinci test, en iyi doküman benzerliği metodu ve hiyerarşik kümeleme algoritmasını seçimi üzerinedir. Bu seçim için cophenetic korelasyon katsayısına kullanılacaktır. İkinci test, uygulamadan çıkan dendrogramların karşılaştırılması üzerine olacaktır. Tablo 2'de, doküman benzerliği ölçütlerinden Cosine Benzerliği (cos), Jaccard Benzerlik Katsayısı (jac), Dice'in Katsayısı (dic) ve hiyerarşik kümeleme algoritmalarından Tek Bağlantı (min), Tam Bağlantı (max), Ortalama Grup (avg) algoritmalarının oluşturduğu cophenetic korelasyon katsayıları ve standart sapma değerleri verilmektedir.

Tablo 2: Doküman benzerliği ölçütlerinin ve hiyerarşik kümeleme algoritmalarının cophenetic korelasyon katsayıları ve standart sapmaları

	cos	**jac**	**dic**
min	0.852 ± 0.135	0.958 ± 0.058	0.949 ± 0.063
max	0.869 ± 0.119	0.959 ± 0.055	0.948 ± 0.063
avg	0.910 ± 0.086	**0.977 ± 0.033**	0.970 ± 0.039

Tablo 2'deki sonuçlara göre Jaccard doküman benzerliği metodu ve Ortalama Grup algoritmasının oluşturduğu çift en iyi sonucu vermiştir. Cosine hiyerarşik kümeleme algoritması her üç durumda en kötü sonuçları üretmektedir. Jaccard ve Dice doküman benzerliği ölçütleri birbirine yakın sonuçlar üretirken Jaccard ölçütü tüm hiyerarşik kümeleme algoritmalarında en iyi sonucu verdiği görülmüştür. Diğer bir taraftan hiyerarşik kümeleme algoritmaları açısından üç doküman benzerliği metodunda ortalama grup algoritması en iyi sonuçları verirken tek bağlantı ve tam bağlantı algoritmaları birbirine yakın sonuçlar üretmektedir. İkinci testte, Şekil 7'de 2012-2013 eğitim-öğretim yılında Nesneye Yönelik Programlama dersine ait iki farklı ödevde öğrencilerin intihal durumlarını gösteren iki dendrogram görülmektedir. Her iki dendrogramda Jaccard doküman benzerliği metodu ve Ortalama Grup algoritması kullanılmıştır.

Şekil 7'de Nesneye Yönelik Programlama konusunda verilen birinci ödevde öğrencilerin gruplar halinde birbirine yakın ödev yaptıkları hatta 4 öğrencinin tamamen aynı ödevi gönderdikleri görülmektedir. Son üç rakamı "040" olan öğrenci en orijinal ödevi göndermiştir. İkinci ödevde öğrencilerin ödevlerinin bir intihal tespit sisteminde geçtiği öğrencilere aktarılmıştır. Böyle bir sistem sayesinde dendrogramda öğrencilerin hiçbirinin tamamen benzer ödev göndermedikleri görülmektedir. Kesme değeri ise ödevin türüne, ödevin büyüklüğüne ve sınıfın intihal durumuna göre büyük farklılıklar gösterir. Bu yüzden kesme değeri için kesin bir sonuç söylemek çok zordur. Web ortamında öğretim üyesi kesme değerini üretilen dendrogramı inceleyerek kolayca belirleyebilir. Kesme değeri belirlendikten sonra öğrenci isimleri renklendirilmiş olarak öğretim üyesine gösterilir.



(a) Nesneye Yönelik Programlama 1. Ödev (b) Nesneye Yönelik Programlama 2. Ödev

Şekil 7: Ödevlerde intihal durumları gösteren dendrogramlar

İntihal tespiti uygulamasının kullanılmadığı düşünen öğrencilerin ödevde orijinallikten uzaklaştıkları görülmektedir. İntihal tespit uygulaması sayesinde intihal yapan öğrencilerin sonraki ödevlerde daha orijinal ödev yaptıkları görülmüştür. Sonuç olarak, intihal tespiti yapıldığını bilen öğrenciler daha iyi ödev hazırlamaya yönelmektedirler.

Sonuçlar

Bu çalışmada, hiyerarşik kümeleme yardımıyla öğrencilerin birbirlerinden yaptıkları intihalleri kolayca tespit etmemizi sağlayan ve bölümümüzde geliştirilen web tabanlı bir uygulama anlatılmıştır. Bu uygulama tüm üniversite öğretim üyeleri ve öğrenciler tarafından kullanılabilmektedir. Sınıf içi intihal tespiti konusunda ülkemizde pek kontrol yapılmamaktadır. Tezler için yapılan intihalleri kontrol etmek için uluslararası ticari çözümler kullanılmaktadır. Ticari çözümleri sayfa sınırı olan ve tezleri kontrol etmekte kullanılabilecek çözümlerdir. Diğer taraftan öğrencilere verilen bir ödevde bu kontrolün yerel olarak yapılması gerekmektedir. Bu çalışmada doküman

karşılaştırmasındaki dikkat edilecek noktalar, doküman benzerliklerinin kıyaslanması, hiyerarşik kümelemenin sisteme adapte edilmesi, farklı hiyerarşik kümeleme algoritmalarının kıyaslanması ve dendogramın ödev değerlendirme sürecinde kullanılması sağlanmıştır. Uygulama geliştirilirken denenen doküman benzerliği ölçütleri arasından Jaccard Benzerlik Katsayısı ve denenen hiyerarşik kümeleme algoritmaları arasından da Ortalama Grup algoritması çifti en iyi sonuçları vermiştir.

Uygulamanın getirileri iki farklı başlık altında toplanabilir.

- Öğretim üyeleri ödevleri daha kısa sürede ve ödev orijinalliğini daha rahat değerlendirme imkânına kavuşmuşlardır. Bu sayede ödev notu verilirken sadece ödevin doğruluğuna değil ayrıca orijinalliğinin de göz önüne alınma imkânı oluşmuştur.
- Bu sistem sayesinde öğrencilerin hazırladıkları ödevlerdeki intihal oranı azalmıştır ve bunun yanında öğrencilerin ödevleri daha dikkatli hazırladıkları gözlenmiştir.

Açılan üniversite/bölüm sayısının artması öğretimin en önemli parçalarından biri olan ödevin verilmesini ve değerlendirilmesini zorlaştırmaktadır. 8 yıldır öğrenci alan bölümümüzde öğretim üyesi ve asistan sıkıntısı olması sebebiyle ödevlerin genelde doğruluğuna bakılmaktaydı. Ödev açısından sadece doğruluğa bakıldığını fark eden öğrenciler intihal yoluna başvurmaya başlamışlardı. Bölümüzde geliştirilen bu uygulama sayesinde intihal oranının azalmaya başladığı ve öğrencilerin daha orijinal ödev hazırlamaya teşvik edildiği görülmüştür.

Hazırlanan test verisi genelde programlama ödevlerini kapsamaktadır. Bundan sonraki çalışmalarda metin tabanlı ödevlerde test verisine eklenecektir. Mevcut uygulamamızda kelimeler kullanılarak benzerlik karşılaştırması yapılabilir. Ancak, arama motorlarında kullanılan kelimelerin gövdesini bulma işlemi (Can ve diğerleri, 2008; Uzun, 2012) ile farklı doküman benzerliği sonuçları üretilebileceği ve bu durumun mevcut sistemi iyileştirebileceği düşünülmektedir. Ayrıca, mevcut sistemdeki diğer bir sorun ödev sayısı ve ödev boyutu arttıkça değerlendirme sürecinin uzamasıdır. Bu durumu çözmek için HADOOP[7] benzeri birden fazla bilgisayarı aynı anda farklı ödevleri karşılaştırmak için kullanıp karşılaştırma süresi azaltılmaya çalışılacaktır.

Kaynakça

Arda, B. (2003). Üniversitenin araştırma işlevi ve etik. *C. Ü. Tıp Fakültesi Dergisi*, 25 (4), 7-11.

Can, F., Kocberber, S., Balcik, E., Kaynak, C., Ocalan, H. C. ve Vursavas, O. M.(2008). Information retrieval on Turkish texts. *Journal of the American Society for Information Science and Technology*, 59 (3), 407-421.

Çiftçi, K.(2011). Minimum spanning tree reflects the alterations of the default mode network during Alzheimer's Disease. *Annals of Biomedical Engineering*, 39, 1493-1504.

Dice, L. R.(1945). Measures of the Amount of Ecologic Association Between Species. *Ecology*, 26 (3), 297-302.

Donaldson J. L., Green, B. ve Sposato, P. H.(1981). A plagiarism detection system. *Proceedings of the 12th SIGCSE symposium on Computer science education*, 13 (1), 21-25.

Fung, B. K., Wang, K. ve Ester, M.(2004). Hierarchical Document Clustering. *Encyclopedia of Data Warehousing and Mining, Idea Group Reference*, 1 (A-H), 555-559.

Heintze, N.(1996). Scalable document fingerprinting. *In Proceedings of the Second USENIX Workshop on Electronic Commerce*, 191-200.

Karabulut, M., Gürbüz, M. ve Sandal, E. K.(2008). Hiyerarşik Küme Tekniği Kullanılarak Türkiye'deki İllerin Sosyo-ekonomik Benzerliklerinin Analizi. *SDÜ Sosyal Bilimler Enstitüsü Dergisi*, 3 (5), 65-78.

Lingxiao, J., Su, Z. ve Chiu, E.(2007). Context-based detection of clone-related bugs. *Proceedings of the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, 55-64, Dubrovnik, Croatia.

Liu, X., Xu, C. ve Ouyang, B. (2015). Plagiarism Detection Algorithm for Source Code in Computer Science Education. *International Journal of Distance Education Technologies (IJDET)*, 13(4), 29-39.

[7] http://hadoop.apache.org/ (Son erişim 20 Eylül 2016 tarihinde yapılmıştır.)

Ceglarek, D.(2013). Evaluation of the SHAPD2 algorithm efficiency in plagiarism detection tasks. *Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE), 2013 International Conference on. IEEE*, 9-11 May, Konya, Turkey.

Lyon, C., Malcolm, J. ve Dickerson, B.(2001). Detecting short passages of similar text in large document collections. *In Proceedings of Conference on Empirical Methods in Natural Language Processing*, 118-125.

Manning, C. D., Raghavan, P. ve Schütze, H., *Introduction to Information Retrieval*. Cambridge University Press, http://informationretrieval.org/, 2008.

Samuel, M. ve Zelda, F.(2006). Similarity and originality in code: plagiarism and normal variation in student assignments. *Proceedings of the 8th Australian conference on Computing education*, 52, 143-150, Hobart, Australia, Australian Computer Society.

Schleimer, S., Wilkerson, D. S. ve Aiken, A.(2003). Winnowing: local algorithms for document fingerprinting. *In Proceedings of the 2003 ACM SIGMOD - International Conference on Management of Data*, pp. 76-85.

Tan, P. N., Steinbach, M. ve Kumar, V.(2005). *Introduction to Data Mining*. Addison-Wesley, ISBN 0-321-32136-7, Bölüm 8: pp. 500.

Uzun, E.(2012). A fuzzy ranking approach for improving search results in Turkish as an agglutinative language. *Expert Systems with Applications*, 39(5), 5658-5664.

Uzun, E., Karakuş, T., Kurşun, E. ve Karaaslan, H.(2007). Öğrenci gözüyle aşırma (intihal): neden ve çözüm önerileri. *Akademik Bilişim '07*, Bildiri Kitabı (pp. 183-188), Dumlupınar Üniversitesi, Kütahya.

Wise, M.(1992). Detection of similarities in student programs: YAP'ing may be preferable to plague'ing. *SIGCSE '92 Proceedings of the twenty-third SIGCSE technical symposium on Computer science education*, 24 (1), 268-271.

Yerra, R. ve Ng, Y.(2005). A Sentence-Based Copy Detection Approach for Web Documents. *Fuzzy Systems and Knowledge Discovery*, Lecture Notes in Computer Science 557-570, Springer.

Yeşilbudak, M., Kahraman, H.T. ve Karacan H.(2011). Veri madenciliğinde nesne yönelimli birleştirici hiyerarşik kümeleme modeli. *Gazi Üniversitesi Mühendislik Mimarlık Dergisi*, 26 (1), 27-39.