

KIRIKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
YÜKSEK LİSANS TEZİ

KESİK ULV AYRIŞIMI İLE GİZLİ ANLAMSAL DİZİNLEME

Fatih VARÇIN

AĞUSTOS 2016

Bilgisayar Mühendisliği Anabilim Dalında Fatih VARÇIN tarafından hazırlanan KESİK ULV AYRIŞIMI İLE GİZLİ ANLAMSAL DİZİNLEME adlı Yüksek Lisans Tezinin Anabilim Dalı standartlarına uygun olduğunu onaylarım.

Prof. Dr. Erdem Kamil YILDIRIM
Anabilim Dalı Başkanı

Bu tezi okuduğumu ve tezin **Yüksek Lisans Tezi** olarak bütün gereklilikleri yerine getirdiğini onaylarım.

Prof. Dr. Hasan ERBAY
Danışman

Jüri Üyeleri

Başkan : Yrd. Doç. Dr. Sait SAN _____
Üye (Danışman) : Prof. Dr. Hasan ERBAY _____
Üye : Yrd. Doç. Dr. B. Gürsel EMİROĞLU _____

.../.../...

Bu tez ile Kırıkkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu Yüksek Lisans derecesini onaylamıştır.

Prof. Dr. Mustafa YİĞİTOĞLU
Fen Bilimleri Enstitüsü Müdürü

ÖZET

KESİK ULV AYRIŞIMI İLE GİZLİ ANLAMSAL DİZİNLEME

VARÇIN, Fatih

Kırıkkale Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı, Yüksek Lisans Tezi

Danışman: Prof. Dr. Hasan ERBAY

Ağustos 2016, 50 sayfa

Günümüzde bilgisayar ve ağ teknolojilerinin hızlı gelişimiyle birlikte internet ortamında aşırı bir doküman yığını oluşmuştur. İstenilen bilgiye erişim zorluğu da bu doküman miktarıyla doğru orantılı olacak şekilde artmıştır. Hali hazırda kullanılan birçok bilgi erişim sistemi kullanıcıyı doğru dokümana ulaştırma konusunda garanti verememektedir. Kullanıcının yanlış dokümana yönlendirilmesinin en büyük sebebi bu sistemlerin kullanıcının girdiği sorgu ile sonuç olarak döndürülen doküman arasında sözcüksel eşleştirme yapmasıdır. Bu sorunu çözmek için Gizli Anlamsal Dizinleme (LSI) metodu kullanılır.

LSI, sözcüksel eşleştirme problemini çözmek için terim ve dokümanların gizli anlamsal yapısını kullanan matematiksel bir metottur. Terimler ve dokümanlar arasındaki ilişkiyi Tekil Değer Ayrışımı (SVD) olarak adlandırılan matris ayrışımını kullanarak ortaya çıkarır. Ancak, SVD'nin maliyeti gizli anlamsal analizde kullanılabilir alternatif metotların yolunu açmıştır. Bu çalışmada, örnek bir doküman yığını üzerinden terimler ve dokümanlar arasındaki gizli anlamsal yapı Kesik ULV ayrışımı ile keşfedilmiş ve performansı SVD ile karşılaştırılmıştır. Ayrıca daha önce oluşturulan gizli anlamsal yapı folding-in ve kesik ULV ayrışımını tekrar hesaplama metotlarıyla güncellenmiş ve yeni elde edilen gizli anlamsal yapılar karşılaştırılmıştır.

Anahtar Kelimeler: Bilgi Eriřim Sistemi, Gizli Anlamsal Dizinleme, Tekil Deęer Ayrıřımı, Kesik ULV Ayrıřımı



ABSTRACT

LATENT SEMANTIC INDEXING VIA TRUNCATED ULV DECOMPOSITION

VARÇIN, Fatih

Kırıkkale University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering, M. Sc. Thesis

Supervisor: Prof. Dr. Hasan ERBAY

August 2016, 50 pages

Nowadays, an excessive collection of documents is occurred by rapid development of the internet and computer network technology. The difficulty of accessing the desired information is increasing proportional to the amount of the collection of documents. Most of the existing information retrieval systems can't guarantee to the users for accessing the right documents. The biggest reason for this is that these systems use lexical matching method for documents and query. Latent Semantic Indexing (LSI) is used for solving this problem.

The LSI is a mathematical method which tries to overcome the problem of lexical matching by using latent semantic structure of words and documents. The LSI discovers the relationships between terms and documents using a matrix decomposition such as the Singular Value Decomposition (SVD). However the computational cost of the SVD has paved the way of alternative methods for LSI. In this study, the Truncated ULV Decomposition based LSI is used for discovering the relationships between terms and documents of a sample collection of documents and compared with the SVD based LSI. Also, the existing latent semantic structure is updated by using folding-in and recomputing the Truncated ULV Decomposition methods and new latent semantic structures are compared with each other.

Key Words: Information Retrieval System, Latent Semantic Indexing, Singular Value Decomposition, Truncated ULV Decomposition



TEŐEKKÜR

Tezimin hazırlanması esnasında yardımlarını esirgemeyen ve büyük destek olan tez yöneticisi hocam, Sayın Prof. Dr. Hasan ERBAY'a, üzerimde büyük emeđi olan hocam, Sayın Yrd. Doç. Dr. Sait SAN'a, tezimin düzenlemelerini yaparken yardımlarını esirgemeyen Arş. Gör. Enes AYAN, Arş. Gör. Fahrettin HORASAN ve Arş. Gör. Emre DENİZ'e teşekkürlerimi sunarım.

Maddi ve manevi her zaman yanımda olan desteklerini hiçbir zaman esirgemeyen aileme, özellikle gösterdiği sabır ve verdiği destekten ötürü sevgili eşim, Duygu VARÇIN'a teşekkür ederim.

İÇİNDEKİLER DİZİNİ

	<u>Sayfa</u>
ÖZET	i
ABSTRACT	iii
TEŞEKKÜR	v
İÇİNDEKİLER DİZİNİ	vi
ŞEKİLLER DİZİNİ	viii
ÇİZELGELER DİZİNİ	ix
1. GİRİŞ	1
2. METERYAL VE YÖNTEM	7
2.1. Tekil Değer Ayrışımı (SVD).....	7
2.2. Kesik ULV Ayrışımı.....	9
2.3. Gizli Anlamsal Dizinleme.....	9
2.3.1. Dokümanın Parçalanması ve Noktalama İşaretlerinin Temizlenmesi.....	10
2.3.2. Etkisiz Kelimelerin Filtrelenmesi.....	10
2.3.3. Kelimeleri Köklerine Ayrıştırma.....	11
2.3.4. Terim-Doküman Matrisini Oluşturma.....	11
2.3.4.1. Yerel Ağırlıklandırma Yöntemleri.....	12
2.3.4.2. Genel Ağırlıklandırma Yöntemleri.....	14
2.3.5. Matris Ayrışımının Uygulanması.....	17
2.3.6. Rank-k Yaklaşımı.....	17
2.3.7. Vektör Uzayının Elde Edilmesi.....	19
2.3.8. Sorgu.....	20
2.3.9. Benzerlik Ölçümü.....	21
2.3.10. Performans Değerlendirmesi.....	23
2.3.11. Gizli Anlamsal Yapının Güncellenmesi.....	25
2.3.11.1. Folding-in Metodu.....	25
2.3.11.2. Kesik ULV Ayrışımını Tekrar Hesaplanma Metodu.....	27

3. ARAŐTIRMA BULGULARI	28
3.1. Gizli Anlamsal Dizinleme.....	28
3.2. Sorgu OluŐturma.....	36
3.3. Anlamsal Yapının G¼ncellenmesi.....	41
4. TARTIŐMA VE SONUÇ	45
KAYNAKLAR	48



ŞEKİLLER DİZİNİ

<u>ŞEKİL</u>	<u>Sayfa</u>
2.1. Kesik ULV Ayrışımı Gösterimi.....	17
2.2. MED Doküman Yığınının Ait Tekil Değerler	18
2.3. Terim-Doküman Matrisinin Rank-k Yaklaşımı.....	20
2.4. Doküman Yığınının Kümesi.....	24
2.5. Folding-in Metoduyla Vektör Uzayına Doküman Ekleme	26
2.6. Folding-in Metoduyla Vektör Uzayına Terim Ekleme	27
3.1. A terim-doküman matrisinin rank -2 yaklaşımı.....	33
3.2. Kesik ULV Ayrışımıyla Elde Edilen Terim-Doküman Grafiği	35
3.3. SVD İle Elde Edilen Terim-Doküman Grafiği	36
3.4. Sorgu Eklenmiş Terim-Doküman Grafiği.....	38
3.5. Folding-in Metoduyla Terim Ekleme Grafiği.....	42
3.6. Kesik ULV Ayrışımını Tekrar Hesaplama İle Terim Ekleme Grafiği.....	42
3.7. Folding-in Metoduyla Doküman Ekleme Grafiği.....	43
3.8. Kesik ULV Ayrışımını Tekrar Hesaplama İle Doküman Ekleme Grafiği	44

ÇİZELGELER DİZİNİ

<u>ÇİZELGE</u>	<u>Sayfa</u>
2.1. SVD İle Elde Edilen Matrislerin Hesaplama Karmaşıklığı.....	8
2.2. Logaritma Yerel Ağırlıklandırma Örneği.....	14
3.1. Doküman Yığını.....	28
3.2. Doküman Yığına Ait Terim Doküman Matrisi.....	29
3.3. Farklı k Değerleri İçin Sorgu ile Dokümanların Benzerliği.....	39
3.4. SVD İle Farklı k Değerleri İçin Sorgu ile Dokümanların Benzerliği	40



1. GİRİŞ

Bilgisayar ve internet teknolojilerinin gelişmesiyle birlikte bilgilerin tutulduğu alanlar da değişmiştir. Günümüzde neredeyse her türlü veri elektronik ortamlarda saklanmakta ve insanların hizmetine sunulmaktadır. Gerekli veya gereksiz devasa veri yığınları insanlar tarafından bu elektronik ortama kesintisiz bir şekilde aktarılmaktadır. Ancak, bu kadar veri birikimi beraberinde çözülmesi gereken problemleri de getirmektedir. Bu problemlerden biri de kullanıcıyı doğru bilgiye götüren bilgi erişim sistemleri geliştirmektir.

Bilgi erişim sistemi, devasa doküman yığınları içinden sorgu vasıtasıyla bilgi ihtiyacını karşılayan yapılandırılmamış doğal bir dokümanın materyaline ulaşmak olarak nitelendirilebilir [1]. Zamanla kullanıcıyı bu karmaşık doküman yığını içerisinde hedef dokümanlara ulaştıracak farklı bilgi erişim sistemleri ortaya çıkmıştır. Bunlardan biri de vektör uzay modelidir. Vektör uzay modelinde dokümanlar belirlenen ağırlıklandırma fonksiyonuna göre bir terim uzayında vektör olarak temsil edilir. Bir sorgu yapıldığı zaman sistem sorguyu tıpkı bir doküman gibi ağırlıklandırıp sorgu vektörü olarak vektör uzayına ekler. Daha sonra doküman vektörleri seçilen yöntemle bağlı olarak sorgu vektörüne olan benzerlikleri tespit edilip sıralanır ve kullanıcıya geri döndürülür [2]. Vektör uzay modeli doküman sınıflandırma ve doküman kümeleme işlemlerinde sıkça kullanılsa da bazı dezavantajları vardır. Bunlardan en önemlisi, büyük doküman yığnında yer alan birçok terimin birçok dokümanda bulunmamasıdır. Bunun sonucu olarak da oluşturulan vektör uzayının boyutu çok büyük olur. Dolayısıyla bu durum sistemin zaman ve kaynak maliyetinin yüksek olmasına sebep olur. Ayrıca, birçok bilgi erişim sisteminde olduğu gibi bu modelde, sistemde kullanılan dilin iç farklılıkları ile ilgili problemler yaşamaktadır.

Dil, kavramları ifade etmemizde bize birden fazla seçenek sunar. Bu durum, dil açısından güzel bir özellik olsa da bilgi erişim sistemleri açısından önemli bir sorundur. Bunun sebebi geleneksel bilgi erişim sistemlerinin çoğunun bir veri tabanında tutulan kelimelerle kullanıcıdan alınan kelimeler arasında direkt olarak sözcüksel eşleme yapmasıdır. Ancak, bir kavramı ifade etmenin birden fazla yolu

vardır. Arařtırmalara gre iki bireyin gnlk hayatta ok sık rastlanan objeleri bile ifade etmede aynı kelimeleri kullanma oranı %20 den dřktr [3]. Dolayısıyla bu durum bilgi eriřim sistemlerine de yansımaktadır. Buna ilaveten, aynı kkten treyen kelimelerin farklı kelimeler olarak deęerlendirilmesi de bilgi eriřim sistemlerinin performansını olumsuz etkiler. Bu olumsuz etkiyi azaltmak iin kelime kkn bulma (stemming) yntemi geliřtirilmiřtir. Bu yntem kelimeleri kklerine dnřtrerek farklı kullanımlarını normalize eder ve bu sayede deęerlendirilecek olan kelime sayısı azalır. Kelime kkn bulma yntemi her ne kadar performansı hızlandırsa da anlamsal olarak aynı morfolojik olarak iliřkisiz olan kelimelerde iře yaramaz. Kontroll szlkler sayesinde bu problem ařılabilir ancak bu iřlemler elle yapıldığı iin bu sistem zaman karmařası aısından maliyetlidir [4].

Bilgi eriřim sistemlerinin maruz kaldığı bir dięer problem ise kullanıcı tarafından girilen sorgudaki kelimelerin birden ok anlama gelmesidir. Hatta kelimelerin birlikte kullanımından dolayı yan anlamlarında da ortaya ıkma olasılıęı vardır. Bu problemler bilgi eriřim sistemlerinde kullanıcı tarafından girilen sorguyla iliřkisiz kaynaklara eriřime neden olur, ayrıca arama sonularının hassasiyetini de dřrr. Bahsedilen bu problemlerin stesinden gelebilmek iin kelime eřleme yerine kelimeler arasındaki anlamsal iliřkileri kullanan gizli anlamsal dizinleme yntemi ortaya atılmıřtır [5]. Gizli anlamsal dizinleme, anlamsal iliřkilerin keřfinde gizli anlamsal analiz yntemini kullanır.

Gizli anlamsal dizinleme geleneksel bilgi eriřim sistemlerinin kısıtlarının stesinden gelen otomatik bir yntemdir. Bir dokmanın ierdiği anahtar kelimelere dayalı dokman eriřimine ek olarak gizli anlamsal dizinleme var olan dokmanların anlamsal iliřkilerinin derecesini istatistiksel olarak analiz eder. Gizli anlamsal dizinlemeyi kullanan bir bilgi eriřim sistemi kullanıcı tarafından girilen anahtar kelimelerin hi birini iermeyen; ancak anlamsal olarak bu kelimelerle iliřkili dokmanları sonu olarak dndrebilir.

Giriř blmnn bundan sonraki kısmında gizli anlamsal analiz ve bu alanda kullanılan matris ayrıřımları hakkındaki literatr kısaca zetlenmektedir.

Kullanıcıyı doğru dokümanlara ulaştırma amacıyla sözcük eşleme tekniği tabanlı geliştirilen metotların aksine Gizli Anlamsal Analiz (Latent Semantic Analysis - LSA) dokümanlar arasında yapısal bir ilişkinin varlığına dayanır [5]. Yapılan akademik çalışmalar ve bu çalışmada yer alan testler LSA'nın indeksleme ve bilgi erişimi açısından diğer metotlardan daha başarılı olduğunu göstermektedir [6]. Matematiksel tabanlı bir yöntem olan LSA, satırların terimleri (kelimeleri) sütunların ise dokümanları temsil ettiği terim-doküman matrisi olarak adlandırılan bir girdi matrisi kullanır. Bu matrisin elemanlarını oluşturmak için doküman yığnında yer alan terim ve dokümanları kullanan farklı metotlar geliştirilmiştir. Bir terimin ağırlığını bulmak için yerel ve genel olarak iki farklı türde ağırlıklandırma yöntemi kullanılmaktadır. Dumais 'e [7] ait bir çalışmada, yerel ağırlıklandırma yöntemi üç genel ağırlıklandırma yöntemi ise altı farklı ağırlıklandırma tipine ayrılmıştır. Daha sonra bu ağırlıklandırma tiplerinin performansları CRAN doküman yığını üzerinde test edilmiştir. Bu testler sonucunda en başarılı ağırlıklandırma kombinasyonu elde edilmiştir. Daha ayrıntılı ve hassas ağırlıklandırma yöntemleriyle birlikte doküman yığnındaki gizli ilişkileri göstermedeki başarı da artmıştır.

LSA yönteminin temel adımlarından biri, $X \in \mathbb{R}^{m \times n}$ terim-doküman matrisi olmak üzere en iyi rank- k yaklaşımı olan $X_k \in \mathbb{R}^{m \times n}$ matrisini

$$\min_{\text{rank}(X_k)=k} \|X - X_k\|_F \quad (1.1)$$

probleminin çözümü olarak bulmasıdır. Bu problemi çözmeye en etkili yöntem Tekil Değer Ayrışımı(Singular Value Decomposition-SVD)'dir ve aynı zamanda LSA modellemesindeki hesaplama karmaşası en yüksek olan adımdır. Diğer bir deyişle LSA yönteminin asimptotik sınırlar dâhilinde hesaplama karmaşıklığını belirler.

Zamanla doküman yığnındaki doküman sayısının artmasından dolayı mevcut gizli anlamsal yapı değişen doküman yığını iyi temsil edemez. Bu yüzden mevcut gizli anlamsal yapının güncellenmesi gerekir. Güncelleme işleminde farklı yaklaşımlar kullanılmaktadır. Birinci yaklaşım, yeni doküman yığnının terim-doküman matrisinin yeniden oluşturulup Denklem (1.1)'deki problemi çözmektir. Bu yüksek maliyetli bir

yaklaşımıdır. İkinci yaklaşım ise folding-in metodudur. Bu yöntemin kaynak ve hesaplama maliyeti düşük olmasına karşın bu metot mevcut anlamsal yapıya dayanır yani eklenecek yeni terim ve/veya doküman var olan terimlerin ve dokümanların temsilcilerine etkisi olmadığından hatalı anlamsal yapı üretebilir. Buna ek olarak folding-in metodunun ortogonal matrislerin yapısını bozma potansiyeli de güncel anlamsal yapıyı bozabilir. Üçüncü yaklaşım ise güncelleme işlemidir yani mevcut SVD'nin ayrışım bilgisini kullanarak yeni SVD'nin hesaplanmasıdır. O'brien'in yaptığı çalışmada gizli anlamsal yapıyı güncellemek için "SVD-güncelleme" isimli yeni bir metot sunulmuş ve bu metodu hali hazırda kullanılan yöntemlerle belli bir veri tabanı üzerinde karşılaştırılmıştır. Karşılaştırma hafıza kullanımı, zaman karmaşıklığı ve çıkarım performansı çatısı altında yapılmıştır. Önerilen yöntemin, hafıza kullanımında SVD'nin tekrar hesaplanması yöntemine göre daha avantajlı; ancak folding-in metoduna göre daha fazla hafıza kullandığı saptanmıştır. Zaman karmaşıklığında da aynı sıralama göze çarparken çıkarım performansında ise SVD-güncelleme metodunun, doğru gizli anlamsal yapıyı oluşturma da folding-in metodundan daha başarılı ve SVD'nin tekrar hesaplanması metoduna da iyi bir alternatif olduğu gösterilmiştir. Fakat bu ve buna benzer diğer çalışmalar, SVD'yi güncelleme işleminin SVD'nin sıfırdan hesaplanması kadar maliyetli olduğunu göstermektedir [8].

SVD'nin sahip olduğu bu dezavantaj araştırmacıları LSA'da kullanılabilecek farklı matris ayrışımına yöneltmiştir. Bu yönde yapılan çalışmaların birinde, Berry ve Fierro tarafından hem ilk hesaplanması hem de güncellenmesi SVD'ye oranla daha az maliyetli olan ULV tabanlı bir algoritma önerilmiştir [9]. MEDLINE veri tabanı kullanılarak elde edilen sonuçlar önerilen algoritmanın SVD kadar doğru çıktılar ürettiğini göstermektedir. Çalışmada verilen algoritmanın avantajı özellikle güncelleme işleminde ortaya çıkmaktadır. Bu yeni algoritma, LSA güncelleme işlemlerinde sıklıkla kullanılan folding-in ve matris ayrışımını tekrar hesaplama metotları ile kıyaslanmıştır. Yapılan testlerde doğru dokümanlara erişim performansında önerilen algoritmanın folding-in metodundan daha başarılı, matris ayrışımını tekrar hesaplama metoduna da yaklaştığı gözlemlenmiştir.

Kolda ve O'Leary [10] tarafından yapılan diğerk bir alıřmada ise SVD yerine daha nceden grnt sıkıřtırma tekniđi olarak kullanılan ve depolama alanını nemli lde azaltmaya yarayan SDD (Semi Discrete Decomposition) ayrıřımı kullanılmıřtır. SVD ve SDD ayrıřımlarından elde edilen LSA modellemeleri hesaplama maliyeti ve gncelleme hızları kapsamında karřılařtırılmıř ve SDD ynteminin SVD kadar iyi sonular verdiđi grlmřtr.

SVD'ye alternatif yaklařımlardan bir diğeri Alexandrov ve arkadaşlarının [11] yaptıđı bir alıřmada ele alınmıřtır. SVD yerine, bir matristeki istatistiksel varyansları ifade eden lineer cebir tekniđiyle yakın iliřkili olan tekil deđerleri nermiřlerdir. Bu alıřmada tam boyutlu terim-dokman matrisi yerine daha kk bir matris kullanılması sebebiyle daha ok hesaplama karmařasını dřrmek hedeflenmektedir. Hesaplama karmařasıyla ilgili bir bařka alıřmada ise Goa ve Zhang [12] SVD yerine seyrek kavram ayrıřımı olarak adlandırılan SCD yntemini kullanmıřlardır. SCD toplam veri depolama maliyetini azaltmak iin terim-dokman matrisinin seyrek blmleri yerine matematiksel yaklařımlar kullanır. SCD sayesinde genel hesaplama maliyetini ve depolama alanını azaltılmıř SVD'ye oranla daha iyi bir eriřim performansı elde edilmiřtir.

LSA'nın bařarılı uygulamalarına rađmen daha tam olarak cevaplanamayan birok problem vardır. Jessup ve Martin'e ait bir alıřmada bu problemlerden bazıları ele alınmıřtır [13]. Bu alıřmada vektr uzay modelinde LSA'nın ıkarım performansının rank seimine gre nasıl deđiřtiđi deneysel alıřmalarla gzlenmiř, ayrıca LSA'nın performansının eř anlamlılık problemine ne denli bađlı olduđu da tartıřılmıřtır. Ayrıca alıřmada boyut indirgemek iin SVD' den farklı bir matris ayrıřımı uygulaması denenmiř ve performansları karřılařtırılmıřtır.

Gulcin' e [14] ait bir alıřmada LSA'nın dokmanlardaki anlamsal yapıyı bulma zelliđini kullanarak LSA tabanlı iki farklı zet ıkarma algoritması geliřtirilmiřtir. Sunulan algoritmalar Trke ve İngilizce veri setleri zerinde alıřtırılmıř ve ROUGE deđerleri kullanarak bazı zet ıkarma sistemleri ile karřılařtırılmıřtır. Sonular incelendiđinde sunulan LSA tabanlı algoritmaların zellikle kısa dokmanlarda diğerk zet ıkarma sistemleri kadar bařarılı olmadıđı gzlenmiřtir. Ancak nerilen

algoritmaların diđer yaklaşımlarda kullanılan eğitim seti gibi dış bilgileri kullanmadığına, sadece verilen dokümandan elde edilen anlamsal bilgiyle özetleme yaptığına dikkat çekilmiştir.

LSA'yı başka bir alanda kullanan bir diđer çalışma da Duman'a aittir [15]. Bu çalışmada LSA kullanılarak web sayfalarını sınıflandıran bir sistem oluşturulmuştur. Elde edilen doküman vektörlerine destek vektör makinesi metoduyla eğitim ve test işlemleri uygulanmıştır. Terim ağırlıklandırma yöntemi olarak yerel ağırlıklandırma yöntemlerinden terim frekansı, genel ağırlıklandırma yöntemlerinden ise ters doküman frekansı kullanılmıştır. Seçilen bu ağırlıklandırma kombinasyonunun diđerlerine göre daha başarılı olduğu saptanmış ve sistemin başarısı daha yüksek seviyelere çekilmiştir.

Bu çalışmada, LSA kullanılarak oluşturulan LSI bilgi erişim sisteminde matris ayrışımı olarak SVD yerine kesik ULV ayrışımı kullanılmış ve aynı şartlar altında bu SVD tabanlı bilgi erişim sistemiyle karşılaştırılmıştır. Ayrıca, kesik ULV ayrışımı kullanarak oluşturulan gizli anlamsal yapı folding-in ve kesik ULV ayrışımını tekrar hesaplama yöntemleriyle güncellenmiştir.

Tezin geriye kalan kısmı aşağıdaki şekilde organize edilmiştir.

Tezin ikinci bölümünde, ilk olarak çalışmada kullanılan SVD ve kesik ULV matris ayrışimleri hakkında bilgi verilmiştir. Devamında, LSA ve kesik ULV ayrışımı tabanlı LSI sisteminin adımları anlatılmıştır. Ayrıca, oluşturulan gizli anlamsal yapıyı güncelleme metotları da gösterilmiştir.

Tezin üçüncü bölümünde, kesik ULV ayrışımı tabanlı LSI sistemi ile SVD tabanlı LSI sisteminin performansları bir örnek üzerinde sayısal sonuçlar verilerek karşılaştırılmıştır. Buna ek olarak, kesik ULV ayrışımı kullanılarak oluşturulan anlamsal yapı folding-in ve kesik ULV ayrışımını tekrar hesaplama metotlarıyla güncellenmiş ve karşılaştırılmıştır.

Tezin son bölümünde ise tez ile ilgili elde edilen sonuçlara ve değerlendirmelere değinilmiştir.

2. METERYAL VE YÖNTEM

Bu kısımda, tez boyunca kullanacağımız materyal ve yöntemlerden bahsedeceğiz. LSA yöntemini uygulayabilmek için öncelikle doküman yığınınından elde edilen bir $m \times n$ boyutlu terim-doküman matrisine ihtiyaç vardır. Terimler ve/veya dokümanlar arasında var olduğu kabul edilen gizli yapıları keşfetmek amacıyla terim-doküman matrisinin rank- k yaklaşımı hesaplanır. Burada $k \ll \min(m, n)$ şartı sağlanmalıdır ve SVD bu yaklaşımı hesaplamada kullanılan en popüler araçtır. Daha sonra terim-doküman matrisinin en büyük k tekil değeri ve onlara karşılık gelen sağ ve sol tekil vektörler kullanılarak doküman ve terimler düşük ranklı olarak temsil edilir. Son olarak oluşturulan sorgu vektörünün k vektör uzayında düşük ranklı temsilcisi elde edilir ve dokümanları temsil eden vektörlerle benzerlik ilişkisi incelenir.

2.1. Tekil Değer Ayrışımı (SVD)

Tekil değer ayrışımı matris rankı belirlemede, lineer bağımsız en küçük kareler probleminin çözümünde ve standart korelasyon analizinde yaygın olarak kullanılan bir metottür [16]. $m \geq n$ olmak üzere verilen $m \times n$ boyutlu ve r ranklı terim-doküman matrisi A 'nın tekil değer ayrışımı,

$$A = U\Sigma V^T \quad (2.1)$$

olacak şekilde üç matrisin çarpımıdır. Burada U ve V ortogonal matrislerdir yani

$$U^T U = U U^T = I_m \text{ ve } V^T V = V V^T = I_n \quad (2.2)$$

U ve V ortogonal matrislerinin sütunları sırasıyla sol ve sağ tekil vektörler olarak anılır.

Ayrıca, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \dots, \sigma_n)$ bir köşegen matristir ve köşegen elemanları A 'nın tekil değerleri olarak adlandırılır. Bu tekil değerler,

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_n = 0 \quad (2.3)$$

biçimindeki eşitsizliğini sağlar ve AA^T 'nin öz değerlerinin pozitif kareköküne karşılık gelir [17]. Fakat sayısal hesaplamalardaki temsil ve yuvarlama hatalarından dolayı rank kavramı esnetilerek A matrisine sayısal rank atanır. A matrisinin sayısal rankı k ise,

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > \delta \geq \sigma_{k+1} \geq \dots \geq \sigma_n \geq 0 \quad (2.4)$$

olur. Burada δ eşik değeridir ve yukarıdaki eşitsizliğin tatmin edici olması için σ_k ve σ_{k+1} arasında anlamlı bir boşluk olması gerekir [18].

LSA'da kelime kullanımındaki farklılıklar sebebiyle oluşturulan $m \times n$ boyutlu terim-doküman matrisi büyük boyutlu bir matristir. Bu matrisin tekil değer ayrışımının hesaplama karmaşıklığı $O(m^2n)$ dir ve matrisin ayrışım sonucu oluşan bileşenlerinin zaman karmaşıklığı Çizelge 2.1'de verilmiştir [17].

Çizelge 2.1. SVD İle Elde Edilen Matrislerin Hesaplama Karmaşıklığı

Matris	Golub-Reinsch	R-SVD
Σ	$4mn^2 - \frac{4n^3}{3}$	$2mn^2 + 2n^3$
Σ, V	$4mn^2 + 8n^3$	$2mn^2 + 11n^3$
Σ, U	$4m^2n - 8n^2$	$4m^2n + 13n^3$
Σ, U, V	$4m^2n + 8mn^2 + 9n^3$	$4m^2n + 22n^3$

Bu durum, SVD'nin LSA uygulamalarında oldukça maliyetli olduğu anlamına gelir. SVD'nin yüksek maliyetinden dolayı onun yerine kullanılabilir alternatif yaklaşımlar giderek önem kazanmıştır. Bu çalışma kapsamında da bir LSA uygulamasında SVD yerine daha az maliyetli olan ve matrislerin ortogonal yapısını koruyan kesik ULV ayrışımı kullanılmıştır.

2.2. Kesik ULV Ayrışımı

Sayısal rankı r olan terim-doküman matrisi $A \in \mathbb{R}^{m \times n}$ 'nin kesik ULV ayrışımı

$$A = ULV^T + E \quad (2.5)$$

biçimindedir. Burada $U \in \mathbb{R}^{m \times r}$ ve $V \in \mathbb{R}^{n \times r}$ sol ortogonal matrislerdir, yani

$$U^T U = I_m \text{ ve } V^T V = I_n \quad (2.6)$$

$L \in \mathbb{R}^{r \times r}$ tekil olmayan alt üçgensel matris ve $E \in \mathbb{R}^{m \times n}$ ise hata matrisidir [19]. Ayrıca L matrisinin tekil değerleri, A matrisinin tekil değerlerine yaklaşıp [20].

2.3. Gizli Anlamsal Dizinleme

LSA, doküman yığınındaki gizli anlamsal ilişkileri keşfetmede kullanılan istatistiksel bir yöntemdir. LSA ilk olarak özellik çıkarımı bağlamında kullanılmış ancak çeşitli metin tabanlı uygulamalarda da başarılı sonuçlar üretmiştir [21]. Metinsel uygulamalarda kelimeler en temel bileşendir ve kullanıcıdan alınan sorgu ile dokümanlar arasındaki anlamsal ilişkiyi keşfeder. LSA sözdizimsel ve gramer yapısı temizlenen her doküman yığınına uygulanabilir [5]. Eğer LSA tabanlı bir bilgi erişim sistemi yapılmak isteniliyorsa oluşturulan sistem Gizli Anlamsal Dizinleme (Latent Semantic Indexing-

LSI) olarak adlandırılır. LSI, dokümanları inceleyerek anlamlarını bilmediği kavramların birbirleriyle olan ilişkilerini bularak bir erişim yaklaşımı sunmaktadır. Örneğin, ‘doktor’ ve ‘hekim’ kelimelerinin kullanıldığı dokümanlar yeterince ‘hasta’, ‘ilaç’, ‘reçete’ gibi ortak kelimeler içeriyorsa ‘doktor’ ve ‘hekim’ kelimelerinin anlamsal olarak birbirine yakın olduğu sonucuna ulaşılır. Bu sayede ‘doktor’ kelimesini içeren bir sorgulamada ‘doktor’ kelimesinin geçmediği ama ‘hekim’ kelimesinin geçtiği dokümanlarda kullanıcıya geri döndürülür. LSI yönteminin temel adımları aşağıdaki belirtilmiştir.

2.3.1. Dokümanın Parçalanması ve Noktalama İşaretlerinin Temizlenmesi

Kullanılan doküman yığınındaki dokümanlar kendi içlerinde kelimelere ayrılır. Dokümanda bulunan aynı kelimelerin harf büyüklüğü veya küçüklüğü sebebiyle farklı kelimeler olarak anlaşılması ve sorguda bulunan kelimelerle eşleme işlemindeki duyarlılık için tüm harfler küçük veya büyük harfe çevrilir. Böylece standart bir yapı elde edilmiş olur. Ayrıca doküman yığınındaki noktalama işaretleri temizlenerek hem doküman sadeleştirilmiş hem de ekstra maliyetlerden kaçınılmıştır.

2.3.2. Etkisiz Kelimelerin Filtrelenmesi

Doküman yığınlarında çok sık geçen ancak bilgi erişiminde faydası olmayan veya çok az faydası olan birçok kelime vardır. Bu kelimeleri işleme dâhil etmek, terim-doküman matrisinin satır sayısının büyük olmasına dolayısıyla da hesaplama maliyetinin artmasına neden olur. Ayrıca, bu kelimeler çıkarım performansını da kötü etkiler. Bu problemi çözmek için duraklama kelime listesi olarak adlandırılan listeler kullanılır ve doküman yığını bu listelerde yer alan kelimelerden temizlenir. Bu sayede zaman ve kaynak maliyeti açısından bir avantaj elde edilmekte ve dokümanlarda geçen bu tür kelimelerin çıkartılmasıyla anlamsal yapıdaki sapmaların da önüne geçilmektedir. Evrensel bir duraklama listesi olmadığından her dil için özel olarak bir liste hazırlanmaktadır. Bu çalışmada, kullanılan doküman yığını İngilizce seçildiğinden bu dil için oluşturulan duraklama kelime listesi kullanılmıştır [22].

2.3.3. Kelimeleri Köklerine Ayırıştırma

Doküman yığımındaki herhangi bir kelime dokümanlar içerisinde farklı formatlarda bulunabilir. Örneğin bir kelimenin, bir dokümanda tekil hali bulunurken bir diğerinde çoğul hali bulunabilir. Ayrıca, farklı kip ekleri alarak da değişik şekillerde karşımıza çıkabilir. Her ne kadar farklı görünseler de genellikle bu kelimeler aynı anlamı taşırlar. Ancak, bu şekil farklılığı yüzünden aynı kelime sanki farklı bir anlam taşıyormuş gibi ele alınır ve dolayısıyla da sorgulama, değerlendirme ve özellik çıkarım performansını olumsuz etkiler. Bu problemi çözmek için kelimeye köklerine ayırıştırma işlemi (stemming) uygulanır. Bu işlem sayesinde kelime orijinal haline döndürülür ve birçok farklı kelime tek bir formata indirildiği için kelimelerin kapladığı hafıza da azaltılmış olur. Buna ek olarak, terim-doküman matrisinin boyutu ve seyrekliği düşürülür. Porter Stemmer, Lovins Stemmer ve Paice Stemmer başlıca kelime kökü bulma yöntemleridir. Porter Stemmer bunların içinde en çok kullanılan yöntemdir [23]. Kelimeleri köklerine ayırıştırma işleminin, bilgi erişimindeki avantajlarına karşın bazı riskleri de vardır. Bunlardan biri farklı anlamları olan kelimeleri aynı köke indirgemektir. Örneğin, "experience" ve "experiment" kelimeleri farklı anlamlar taşıdığından dolayı farklı köklere indirgenmelidir ancak kök bulma işlemi sonucunda bu kelimelerin kökleri "experi" olarak bulunur [24]. Bir diğer problem ise düzensiz fiillerle ilgilidir. Örneğin, "dig" ve "dug" kelimelerinin kökleri aynı olması gerekirken bunlar farklı kelimeler olarak algılanıp farklı köklere indirgenir. Dolayısıyla kök bulma işlemi doküman yığımındaki kelimelerin sadeleştirmesini sağlarken bazı ekstra maliyetlere de sebep olabilir. Ayrıca, yöntem dil bağımlı olduğu için eğer doküman yığını içinde farklı bir dilde kelime veya kelimeler varsa bu durum sistem için problem teşkil eder.

Bu tezde, kelimelerin köklerini bulmak için Porter Stemmer metodu kullanılmıştır.

2.3.4. Terim-Doküman Matrisini Oluşturma

Daha önceden belirtildiği gibi LSA, terim-doküman matrisi A 'yı girdi olarak alır. Terim-doküman matrisini,

$$A = [a_{ij}] \quad (2.7)$$

şeklinde tanımlarsak a_{ij} değeri i 'nci terimin j 'inci dokümandaki ağırlığını gösterir.

Doküman yığını m tane terim ve n tane doküman içeriyorsa A matrisi $m \times n$ boyutludur. Ayrıca, her terim her dokümanda bulunmayacağı için A matrisi genellikle seyrek bir matristir [8]. Burada yapılan ağırlıklandırma işleminin belirlenmesi erişim performansı açısından oldukça önemlidir. Örneğin, bir terimin her hangi bir dokümandaki ağırlığı büyük fakat doküman yığınındaki ağırlığı küçükse bu terim o doküman için iyi bir temsilciyken doküman yığını için iyi bir temsilci olmayabilir. Bu durumun tersi de olabilir. Bu yüzden a_{ij} elemanının değeri i 'nci terimin j 'inci dokümandaki ve tüm doküman yığınındaki etkisini dikkate alan

$$a_{ij} = L(i, j) \times G(i) \quad (2.8)$$

formülü aracılığıyla hesaplanır. Burada $L(i, j)$, j 'inci dokümandaki i 'nci teriminin yerel ağırlığını, $G(i)$ ise i 'nci teriminin genel ağırlığını temsil eder [7]. Literatürde tanımlı farklı yerel ve genel ağırlıklandırma yöntemleri vardır.

2.3.4.1. Yerel Ağırlıklandırma Yöntemleri

Denklem (2.8)'deki $L(i, j)$ çarpanı yerel ağırlıklandırma fonksiyonudur ve i 'nci terimin j inci dokümandaki ağırlığını gösterir. Yerel ağırlıklandırma yöntemi, $L(i, j)$ nin tanımına göre değişir. $L(i, j)$ fonksiyonunun tanımı da i 'nci terimin j 'inci dokümandaki frekansını gösteren tf_{ij} 'ye bağlıdır. En popüler yerel ağırlıklandırma yöntemleri şunlardır:

- **İkili Ağırlıklandırma Yöntemi**

İkili ağırlıklandırma yönteminde verilen terimin ilgili dokümandaki varlığına ya da yokluğuna bakılır. Eğer terim dokümanda bulunuyorsa terim ağırlığı bir değerini aksi halde ise sıfır değerini alır. Diğer bir deyişle bu yöntemde,

$$L(i, j) = \begin{cases} 1 & tf_{ij} > 0 \\ 0 & tf_{ij} = 0 \end{cases} \quad (2.9)$$

Bu ağırlıklandırma yöntemi maliyet açısından uygun olsa da tf_{ij} 'nin büyüklüğünü dikkate almadığından doğruluk ve hassasiyet açısından çok tercih edilen bir yöntem değildir.

- **Terim Frekansı Ağırlıklandırma Yöntemi**

Terim frekansı yönteminde verilen bir terim verilen dokümandaki geçme sıklığına göre ağırlıklandırılır yani $L(i, j) = tf_{ij}$ biçiminde tanımlanır. Bir terimin doküman içindeki frekansı anlamsal benzerlik açısından önemlidir ancak doğru orantılı değildir. Terim frekansı ağırlıklandırma yöntemi, basit bir yöntem olup maliyet açısından avantajlıdır fakat hassasiyet açısından yeterli değildir.

- **Logaritma Ağırlıklandırma Yöntemi**

Bu yöntemde i 'nci terimin j 'inci dokümandaki ağırlığı,

$$L(i, j) = \begin{cases} \log(tf_{ij} + 1) & tf_{ij} > 0 \\ 0 & tf_{ij} = 0 \end{cases} \quad (2.10)$$

biçiminde hesaplanır [25]. Ancak bazı çalışmalarda,

$$L(i, j) = \begin{cases} \log(tf_{ij}) + 1 & tf_{ij} > 0 \\ 0 & tf_{ij} = 0 \end{cases} \quad (2.11)$$

şeklinde de kullanılmıştır [1]. Bu çalışmada ise Denklem (2.11)'de verilen tanım kullanılmıştır.

Çizelge 2.2. Logaritma Yerel Ağırlıklandırma Örneği

Terim	1	2	3	4	5	6
Terim Frekansı, tf_{ij}	0	1	2	10	20	30
Yerel Ağırlık, $L(i, j)$	0	1	1,3	2	2,3	2,4

Çizelge 2.2'deki örnek incelenirse terim frekansları arasındaki farklar büyük olmasına rağmen ağırlıklar arasındaki farklar büyük değildir. Bu sayede terim frekanslarındaki büyük farkların anlamsal yapıya olan etkisi azaltılmıştır. Ancak bu yöntemin maliyeti terim frekansı yöntemine göre daha yüksektir.

2.3.4.2. Genel Ağırlıklandırma Yöntemleri

Denklem (2.8)'deki $G(i)$ genel ağırlıklandırma fonksiyonudur ve i 'inci teriminin tüm doküman yığımındaki ağırlığını verir.

Genel ağırlıklandırma yöntemi, $G(i)$ 'nin tanımına göre değişir. Literatürde farklı genel ağırlıklandırma yöntemleri tanımlanmıştır. Bu yöntemlerin bazılarında $G(i)$ fonksiyonunun tanımında tf_{ij} (i 'inci terimin j 'inci dokümandaki frekansı) kullanılırken bazılarında df_i (i 'inci terimin bulunduğu doküman sayısı) bazılarında

ise gf_i (genel doküman frekansı, i 'inci terimin tüm doküman yığımındaki sayısı) veya bunların kombinasyonu kullanılır. Genel ağırlıklandırma yöntemlerinin en çok bilinenleri şunlardır:

- **Normal Ağırlıklandırma Yöntemi**

Normal ağırlıklandırma yönteminde, verilen i teriminin her bir dokümandaki terim frekanslarının tf_{ij} , $j=1,2\dots n$ karelerinin toplamının bir bölüsünün karekökü alınarak i 'nci terimin genel ağırlığına ulaşılır. Diğer bir deyişle,

$$G(i) = \sqrt{\frac{1}{\sum_j tf_{ij}^2}} \quad (2.12)$$

Dikkat edelim ki bu yöntemde verilen terim, doküman yığımındaki dokümanlarda çok fazla geçiyorsa bu terimin dokümanlar arasındaki ayırıcı özelliği düşüktür.

- **Ters Doküman Frekansı (IDF) Ağırlıklandırma Yöntemi**

Ters doküman frekansı yönteminde i 'nci terimin genel ağırlığı doküman yığımındaki toplam doküman sayısının i 'nci terimin geçtiği doküman sayısına oranının logaritmasına eşittir. Matematiksel olarak ifade edersek,

$$G(i) = \log\left(\frac{N}{df_i}\right) \quad (2.13)$$

Formülde kullanılan N , doküman yığımındaki toplam doküman sayısını gösterir. Sonuç olarak bulunan değer verilen terimin ters doküman frekansı olarak da anılır. Bu yöntemde, ilgili terimin geçtiği doküman sayısı fazla ise bu terim bir dokümanlar için önemli gibi gözükse de doküman yığımındaki anlamsal ilişkiler açısından ayırıcı özelliği düşüktür.

- **GFIDF Ağırlıklandırma Yöntemi**

GFIDF ağırlıklandırma yönteminde i 'nci terimin genel ağırlığı, i 'nci terimin doküman yığımındaki frekansının i 'nci terimin geçtiği doküman sayısına oranına eşittir yani,

$$G(i) = \frac{gf_i}{df_i} \quad (2.14)$$

- **Entropy Ağırlıklandırma Yöntemi**

Bu yöntemde, verilen terimin bir dokümandaki frekansı tüm doküman yığımındaki sayısına bölünerek terimin ilgili dokümandaki bulunma olasılığı elde edilir. Daha sonra bu değer ile logaritması alınmış hali çarpılır. Bu işlem verilen terim için doküman yığımındaki tüm dokümanlara uygulanır ve toplamları elde edilir. Bu toplam değeri doküman sayısının logaritmasına bölünür ve bir eklenir [26]. Bu yöntemin matematiksel olarak ifade edersek,

$$p_{ij} = \frac{tf_{ij}}{gf_i} \quad (2.15)$$

olmak üzere

$$G(i) = 1 + \sum_j \frac{p_{ij} \log_2(p_{ij})}{\log_2(N)} \quad (2.16)$$

biçimindedir.

Entropy ağırlıklandırma yöntemi teorik bir mantığa dayanır ve doküman yığımındaki terimlerin dağılışını dikkate alan gelişmiş bir yöntemdir [7].

Bu çalışma kapsamında terim doküman matrisi oluşturulurken yerel ağırlıklandırma

yöntemlerinden logaritma, genel ağırlıklandırma yöntemlerinden ise entropy yöntemi kullanılmıştır.

2.3.5. Matris Ayrışımının Uygulanması

Uygun ağırlıklandırma metodu belirlenip oluşturulan $m \times n$ boyutlu A terim-doküman matrisine belirlenen matris ayrışımı uygulanır. Bu çalışmada, hafıza ve zaman maliyeti açısından SVD yaklaşımına göre daha uygun olan kesik ULV ayrışımı kullanılmıştır. A terim-doküman matrisinin kesik ULV ayrışımı Denklem (2.5)'deki gibidir. Daha önce belirtildiği gibi $U \in \mathbb{R}^{m \times n}$ ve $V \in \mathbb{R}^{n \times n}$ matrisleri sol ortogonal matrisler olup Denklem (2.6) sağlanır. Yine, $L \in \mathbb{R}^{n \times n}$ tekil olmayan alt üçgensel matris ve $E \in \mathbb{R}^{n \times r}$ ise hata matrisidir. Hata matrisi E 'nin gizli anlamsal yapıya herhangi bir katkısı olmadığı için bundan sonraki sürece dâhil edilmez [27].

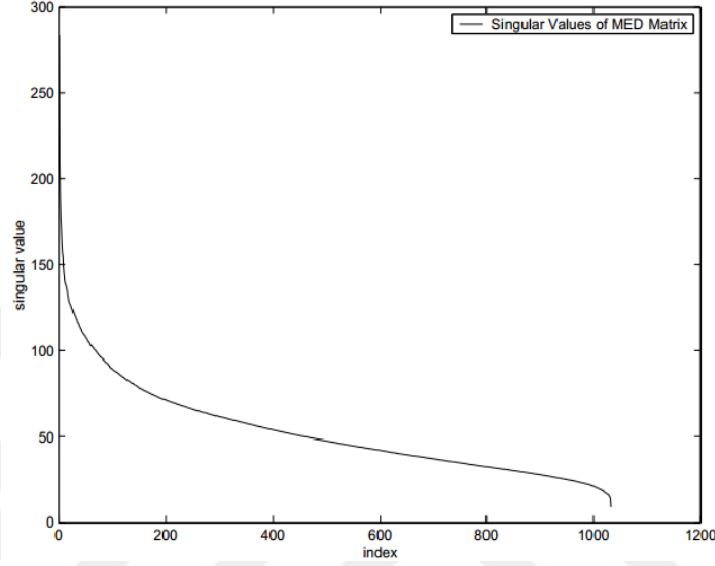
$$\begin{array}{cccc}
 \boxed{A} & = & \boxed{U} & \boxed{L} & \boxed{V^T} \\
 m \times n & & m \times n & n \times n & n \times n
 \end{array}$$

Şekil 2.1. Kesik ULV Ayrışımı Gösterimi

2.3.6. Rank-k Yaklaşımı

Doküman yığımında bir terimin dokümanların çoğunda bulunmadığından dolayı terim-doküman matrisinin genellikle seyrek olduğunu söylemiştik. Özellikle terimlerin seçilmesinde herhangi bir değerlendirilmenin yapılmadığı durumlarda terim-doküman matrisi çok büyük olur ve buda işlem yükünü artırır. Ayrıca, anlamsal yapıya katkısı

olmayan veya aynı anlama gelen terimlerin kullanılması doğru bir anlamsal yapının oluşturulmasını engeller. Bunun sonucu olarak da bilgi erişim sistemi kullanıcının girdiği sorguyla ilişkisiz dokümanları geri döndürülebilir. Bahsedilen bu problemlerin üstesinden gelebilmek için matris ayrışımını uyguladıktan sonra bir boyut düşürme işlemi olan rank- k yaklaşımı uygulanmalıdır.



Şekil 2.2. MED Doküman Yığınının Ait Tekil Değerler

A terim-doküman matrisinin tekil değerleri,

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k > \delta > \sigma_{k+1} = \dots = \sigma_n = 0 \quad (2.17)$$

şeklinde sıralı olup σ_k ve σ_{k+1} arasında anlamlı bir fark var ise A matrisine kesik ULV ayrışımı uyguladıktan sonra rank- k yaklaşımı uygulanır ve

$$A_k = U_k L_k V_k^T + E \quad (2.18)$$

denklemini elde edilir.

Buradaki A_k matrisi, A matrisinin en uygun rank- k yaklaşımı olarak adlandırılır ve

bu yaklaşım sayesinde gizli anlamsal yapıyı bozan ve “gürültü” olarak adlandırılan kısım yok edilir [5]. Bu, A ve A_k arasındaki farkın frobenius normunun minimum olduğu anlamına gelmektedir. Ancak burada k 'nın belirlenmesi kolay bir işlem değildir. Bunun sebebi tekil değerler arasında anlamlı bir farkı bulmanın her zaman mümkün olmamasıdır.

Örneğin, MED doküman yığını kullanılarak oluşturulan bir terim-doküman matrisinin Şekil 2.2’de gösterildiği gibi tekil değerleri arasında anlamlı bir boşluk veya belirgin bir kırılma noktası yoktur [13]. Dolayısıyla hangi tekil değerlerin ihmal edileceğini belirlemek zordur. Ancak, yapılan deneysel çalışmalar terim-doküman matrisinin boyutuna bakılmaksızın k değerinin genel olarak 100 ile 300 arasında olduğunu göstermektedir [7,28,29].

2.3.7. Vektör Uzayının Elde Edilmesi

Rank- k yaklaşımı uygulandıktan sonra elde edilen L_k matrisinin tekil değerlerini bulabilmek için tekil değer ayrışımı uygulanır.

$$L_k = X_k S_k Y_k^T \quad (2.19)$$

Bu eşitlik Denklem (2.18) de yerine koyulursa,

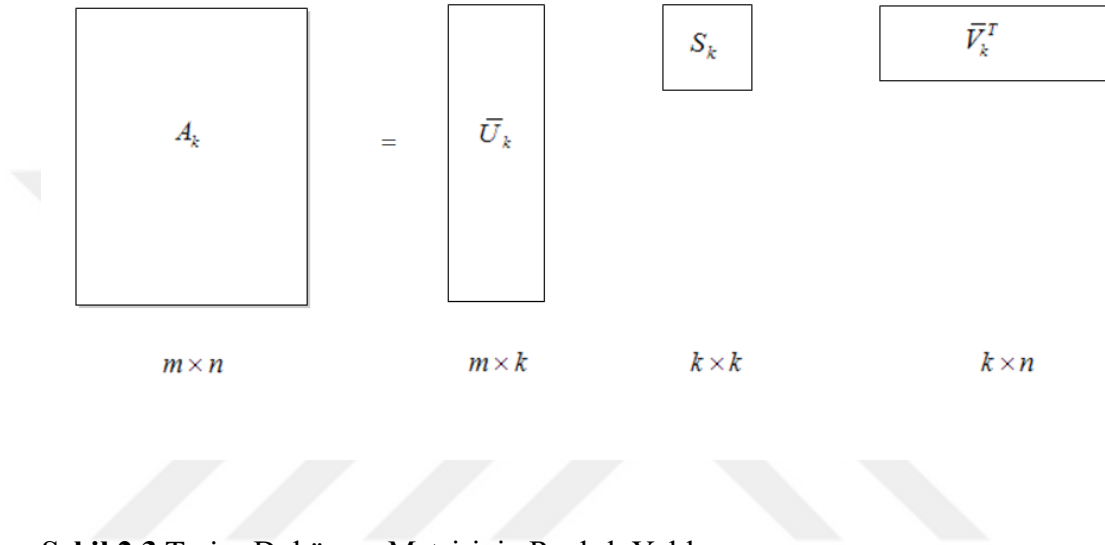
$$A_k = U_k X_k S_k Y_k^T V_k^T \quad (2.20)$$

elde edilir. Burada $\bar{U}_k = U_k X_k$ ve $\bar{V}_k^T = Y_k^T V_k^T$ olarak alınırsa,

$$A_k = \bar{U}_k S_k \bar{V}_k^T \quad (2.21)$$

denkleminde ulařılır ve k boyutlu vektör uzayı elde edilir. \bar{U}_k , k boyutlu terim vektörlerini \bar{V}_k^T ise k boyutlu doküman vektörlerini gösterir.

Bu adımda L_k matrisine uygulanan tekil deęer ayrışımının maliyeti $k \ll \min(m, n)$ olduęu için küçüktür.



Şekil 2.3. Terim-Doküman Matrisinin Rank-k Yaklaşımı

2.3.8. Sorgu

Bir sorgu kelimelerden oluşur ve vektör uzayında temsil edilen bir doküman olarak düşünülür [9]. Kullanıcı tarafından girilen bir sorgu esasında $m \times 1$ boyutlu q vektörüdür ve elemanları terim-doküman matrisini oluştururken kullanılan ağırlıklandırma yöntemleri kullanılarak elde edilir. q sorgu vektörü mevcut vektör uzayında

$$\hat{q} = q^T \bar{U}_k S_k^{-1} \quad (2.22)$$

formülü kullanılarak temsil edilir. Burada \hat{q} , q sorgu vektörünün k boyutlu vektör uzayındaki temsilcisidir. Sorgu vektörünün vektör uzayında temsil edilmesi sayesinde vektör uzayında var olan bütün doküman vektörleri, sorgu vektörüyle karşılaştırılıp benzerlik derecelerine göre sıralanabilir.

2.3.9. Benzerlik Ölçümü

Kullanıcı tarafından girilen sorgunun k boyutlu vektör uzayında temsil edilmesinden sonra dokümanlarla olan anlamsal ilişkisini bulmak için sorgu vektörü ile doküman vektörü arasında benzerlik ölçümü yapılır. Bu ölçüm sayesinde herhangi bir dokümanın sorguyla anlamsal ilişkisini belirlemek ve anlamsal olarak sorguya yakın olan dokümanlar arasında bir sıralama yapmak mümkündür. Doğru benzerlik ölçüm metodunun belirlenmesi, dokümanları sınıflandırma ve bilgi erişim performansı açısından oldukça önemlidir [28].

Benzerlik ölçümünde kullanılacak metotların bazıları aşağıda açıklanmıştır. Bu metotlarda kullanılan m terim sayısını gösterirken D_1 ve D_2 ise $m \times 1$ boyutlu doküman vektörlerini göstermektedir.

- **Jaccard Katsayısı Metodu**

Tanimoto katsayısı olarak da adlandırılan bu metot dokümanlar arasındaki benzerliği dokümanlardaki ortak terimlerin ağırlıkları toplamını, dokümanın birinde bulunan fakat diğerinde bulunmayan terimlerin ağırlıkları toplamıyla karşılaştırarak hesaplar [28]. Bu yöntemin formülü,

$$SIM_J(D_1, D_2) = \frac{D_1 \cdot D_2}{\|D_1\|_2^2 + \|D_2\|_2^2 - D_1 \cdot D_2} \quad (2.23)$$

şeklinindedir. Formülden elde edilen Jaccard katsayısı 0 ve 1 arasında değer alır. 1'e doğru gittikçe karşılaştırılan dokümanların benzerliği artar eğer 1 değerini

alırsa bu durum iki dokümanın tamamen aynı olduğu anlamına gelmektedir. 0 a doğru gittikçe ise dokümanların benzerliği azalır ve 0 değerini alırsa bu durum da iki dokümanın tamamen farklı dokümanlar olduğu anlamına gelmektedir.

- **Öklid (Euclidean) Benzerlik Metodu**

Öklid benzerlik ölçüm metodu iki nokta arasındaki uzaklığın belirlenmesine dayanır [29]. Bu metod doküman sınıflandırma problemlerinde yaygın olarak kullanılır [28]. Bir doküman yığınınındaki iki doküman arasındaki benzerliği öklid metodu ile hesaplamak için aşağıdaki formül kullanılır.

$$SIM_E(D_1, D_2) = \left(\sum_{i=1}^m |D_{1,i} - D_{2,i}|^2 \right)^{\frac{1}{2}} \quad (2.24)$$

Öklid yöntemi vektörlerin uzunluklarını dikkate aldığı için çok kullanılan bir yöntem değildir. Örneğin, X dokümanının iki kopyasından yeni bir \hat{X} dokümanı oluşturulsun. Bu iki dokümanın öklid uzaklıkları farklı olduğu için anlamsal olarak aynı olmasına rağmen farklı dokümanlar olarak algılar buda bilgi erişim sisteminin performansını olumsuz etkiler.

- **Kosinüs Benzerliği Metodu**

Kosinüs benzerliği metodu bilgi erişim uygulamalarında, doküman sınıflandırmada ve dokümanlar arasındaki benzerliği bulmada kullanılan en yaygın benzerlik ölçüm metodudur [30]. Vektör uzayındaki doküman vektörleri ve sorgu vektörü arasındaki ilişkiyi kullanarak seçilen dokümanların birbirleriyle veya sorguyla olan benzerliğini hesaplar. Burada vektörler arasındaki ilişki ile kastedilen şey vektörlerin arasındaki açının kosinüs değeridir. Bu metotta verilen iki dokümanın benzerliği aşağıdaki formül ile hesaplanır.

$$SIM_c(D_1, D_2) = \frac{D_1 \cdot D_2}{\|D_1\|_2 \cdot \|D_2\|_2} \quad (2.25)$$

Kosinüs benzerliği metodu önceden belirlenen bir eşik değeri kullanır. Bir doküman ile sorgunun kosinüs benzerlik değeri bu eşik değerinden büyükse bu doküman ile sorgu ilişkili aksi durumda ise ilişkisiz olarak kabul edilir. Bu yöntemin bir diğer özelliği öklid yönteminin aksine dokümandaki terim sayısından bağımsız olmasıdır. Örneğin, D dokümanının iki kopyasından yeni bir \hat{D} dokümanı oluşturulsun. Bu dokümanları temsil eden vektörlerin kosinüs benzerliği 1 olduğundan bu iki doküman özdeş sayılır. Ayrıca, başka bir dokümanın bu iki dokümanla olan benzerliği de birbirine eşittir. Başka bir ifadeyle aynı terimlerin farklı sayılarından oluşan dokümanlar benzer anlamsal yapıya sahiptir.

Bu çalışmada, kullanıcı tarafından girilen sorguyla dokümanlar arasındaki anlamsal benzerlik kosinüs benzerliği metodu ile tespit edilmiş ve sıralanmıştır.

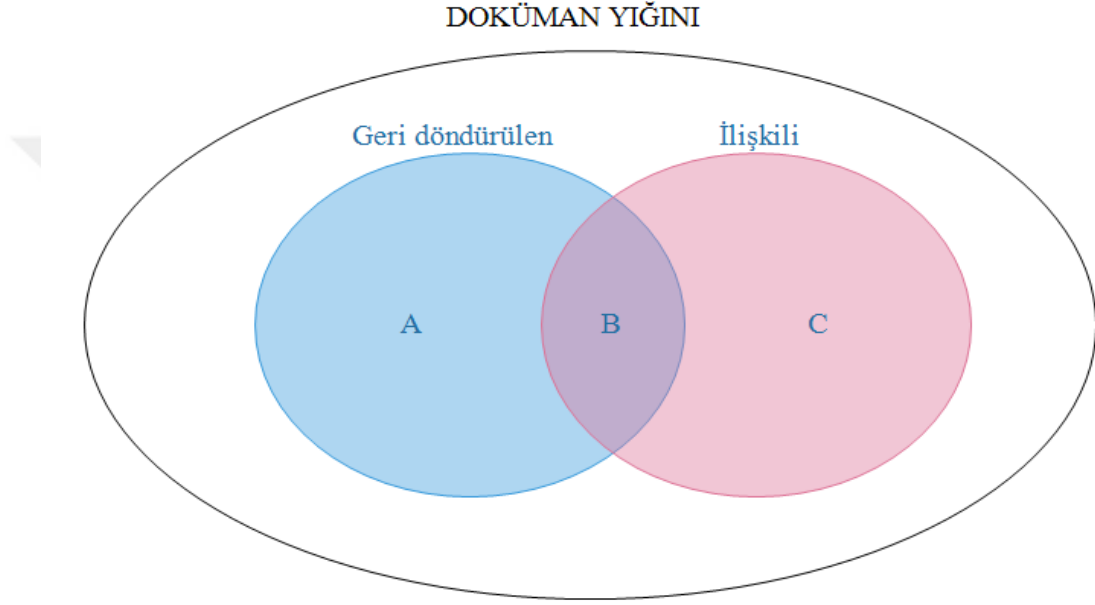
2.3.10. Performans Değerlendirmesi

Kullanılan yöntemin performansını ölçmek için sorgulama sonucunda sistemin sorguyla ilişkili dokümanları geri döndürüp döndürmediğine bakılır. Burada geri çağırma ve hassasiyet olarak adlandırılan iki ölçüt kullanılır [13]. Öncelikle doküman yığını ilişkili dokümanlar ve ilişkisiz dokümanlar olarak ikiye bölünür. İlişkili dokümanlar sistem tarafından geri döndürülmesi beklenen dokümanlardır. Yine doküman yığını geri döndürülen dokümanlar ve geri döndürülmeyen dokümanlar olmak üzere ikiye bölünür. Geri çağırma, sorgulama sonucu geri döndürülen ve ilişkili doküman sayısının toplam ilişkili doküman sayısına oranını gösterirken hassasiyet ise sorgulama sonucunda geri döndürülen ve ilişkili doküman sayısının toplam geri döndürülen doküman sayısına oranıdır. Şekil 2.4'deki doküman yığını kümesinde $s(X)$, X kümesinin eleman sayısını gösterebilir. Bu durumda,

$$\text{geri çağırma} = \frac{s(B)}{s(B \cup C)} \quad (2.26)$$

$$\text{hassasiyet} = \frac{s(B)}{s(A \cup B)} \quad (2.27)$$

şeklinde ifade edilebilir.



Şekil 2.4. Doküman Yığını Kümesi

Örneğin, bir doküman yığnında kullanıcıdan alınan sorguyla ilişkili 8 doküman bulunsun. Sorgu sonucu 10 doküman geri döndürülsün ve bunlardan 2 tanesi sorguyla ilişkili olsun. Bu durumda,

$$\text{Geri çağırma} = \frac{2}{8} = \%25$$

$$\text{Hassasiyet} = \frac{2}{10} = \%20$$

değerlerine ulaşılır.

Bir bilgi erişim sisteminin %100 başarılı olması için doküman yığınındaki ilişkili dokümanların tümü geri döndürülmelidir yani geri çağırma ve hassasiyet ölçütleri 1 değerini almalıdır [31].

2.3.11. Gizli Anlamsal Yapının Güncellenmesi

Dijital ortama sürekli olarak veri akışı olduğundan doküman yığınları da sabit kalmayıp sürekli yeni terimler ve/veya dokümanlar eklenmektedir. Doğal olarak daha önceden oluşturulan anlamsal yapı doküman yığınının iyi temsil edemez ve bunun sonucu olarak da bilgi erişim sistemi kullanıcıyı istediği dokümanlara eriştiremez. Bu problemi çözmek için mevcut anlamsal yapının güncellenmesi gerekir. Bu güncelleme işleminde genellikle folding-in veya kullanılan matris ayrışımının tekrar hesaplanması yöntemleri kullanılır [8]. Bu tezde matris ayrışımı olarak kesik ULV ayrışımı kullanıldığı için anlamsal yapının güncellenme işlemi folding-in ve kesik ULV ayrışımının tekrar hesaplanması yöntemleriyle yapılmıştır.

2.3.11.1. Folding-in Metodu

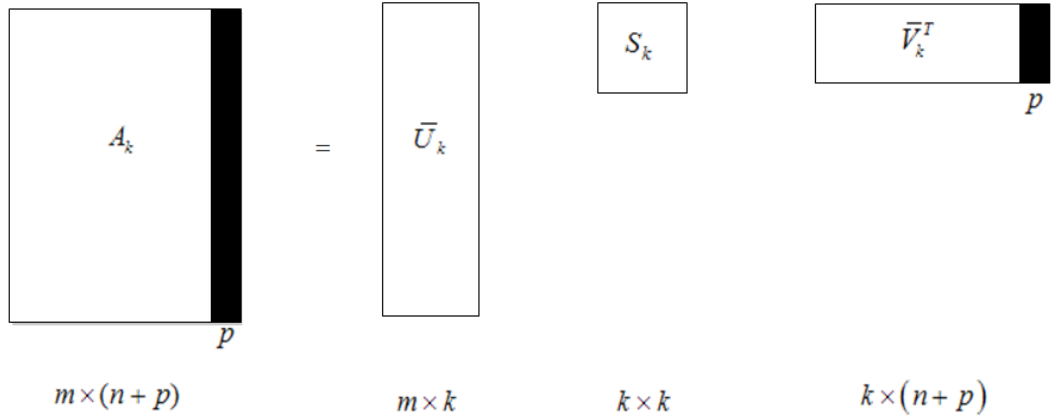
Mevcut doküman yığına yeni terim ve doküman eklendiğinde terim-doküman matrisi de değişeceğinden var olan anlamsal yapının da değişmesi olası durumdur. Yeni anlamsal yapıyı bulmanın bir yolu da folding-in metodudur. Folding-in yöntemiyle k boyutlu vektör uzayına doküman vektörü eklemek temelde daha önce bahsedilen sorgu vektörü eklemeye benzer. Eklenecek doküman vektörleri onları oluşturan terim vektörlerinin ağırlıkları toplamı olarak temsil edilir. Terim-doküman matrisinde kullanılan ağırlıklandırma yöntemleriyle oluşturulan $m \times 1$ boyutlu d doküman vektörünü k boyutlu vektör uzayına folding-in metoduyla eklemek için

$$\hat{d} = d^T \bar{U}_k S_k^{-1} \quad (2.28)$$

denklemini kullanılır. Burada \hat{d} vektörü, d doküman vektörünün vektör uzayındaki temsilcisidir. Benzer şekilde, belirlenen ağırlıklandırma yöntemleriyle hesaplanan $1 \times n$ boyutlu t terim vektörünü k boyutlu vektör uzayına folding-in metoduyla eklemek için

$$\hat{t} = t \bar{V}_k S_k^{-1} \quad (2.29)$$

denklemini kullanılır. Burada \hat{t} vektörü t terim vektörünün vektör uzayındaki temsilcisidir [9].



Şekil 2.5. Folding-in Metoduyla Vektör Uzayına Doküman Ekleme

Dikkatli incelenirse yeni eklenen terim veya doküman vektörleri var olan anlamsal yapıyı herhangi bir şekilde etkilemediği görülür. Folding-in metodu gerek kaynak kullanımı gerekse hesaplama karmaşası yönüyle etkin olmasına karşın ortogonalliği bozma potansiyeli sebebiyle hatalı anlamsal yapı üretebilir [26,28].

2.3.11.2. Kesik ULV Ayrışımını Tekrar Hesaplanma Metodu

Daha önce de bahsedildiği gibi doküman yığınını temsil eden anlamsal yapıyı güncelleştirme yöntemlerinden biri de kullanılan matris ayrışımının tekrar hesaplanması yöntemidir. Bu çalışmada kesik ULV ayrışımı kullanıldığından eklenen q adet terim ve p adet dokümanı dikkate alarak oluşturulan $A \in \mathbb{R}^{(m+q) \times (n+p)}$ yeni terim-doküman matrisinin kesik ULV ayrışımı hesaplanır. Diğer adımlar da aynı şekilde uygulanır ve anlamsal yapı güncellenir. Bu yöntemin avantajı yeni eklenen terim ve dokümanların anlamsal yapıyı nasıl değiştirdiğini net bir şekilde göstermesidir. Ancak çok büyük boyutlu matrislerin kesik ULV ayrışımını hesaplamının maliyeti de çok büyüktür hatta bazı durumlarda bellek yetersizliğinden dolayı imkânsızdır. Bu tür dezavantajlarına rağmen bir güncelleme metodunun doğru anlamsal yapı üretmedeki başarısını görmek için genellikle bu yöntemle karşılaştırılır.

$$\begin{array}{ccccccc} \boxed{A_k} & = & \boxed{U_k} & \boxed{S_k} & \boxed{V_k^T} \\ \begin{array}{c} (m+q) \times n \\ q \end{array} & & \begin{array}{c} (m+q) \times k \\ q \end{array} & \begin{array}{c} k \times k \end{array} & \begin{array}{c} k \times n \end{array} \end{array}$$

Şekil 2.6. Folding-in Metoduyla Vektör Uzayına Terim Ekleme

3. ARAŞTIRMA BULGULARI

Bu bölümde bir doküman yığnında gizli anlamsal dizinleme işlemi yaptıktan sonra oluşturulan vektör uzayına yeni dokümanları folding-in ve kesik ULV ayrışımının tekrar hesaplanması yöntemleriyle ekleyerek gizli anlamsal yapının nasıl değiştiğini inceleyeceğiz.

3.1. Gizli Anlamsal Dizinleme

Bellcore teknik bildirisinden alınan ve Çizelge 3.1’de gösterilen 9 adet konu başlığını doküman yığını olarak kabul edersek bu dokümanları iki gruba ayrılabiliriz. “C” etiketli dokümanlar insan-bilgisayar etkileşimi ile ilgiliyken “M” etiketli dokümanlar ise grup teorisi ile ilgilidir

Çizelge 3.1. Doküman Yığını

DOKÜMAN NO	DOKÜMANLAR
C1	<u>Human Machine Interface</u> for Lab ABC <u>Computer Applications</u> .
C2	A <u>Survey of User Opinion of Computer Systems Response Time</u> .
C3	The <u>EPS User Interface Management Systems</u> .
C4	<u>Systems and Human Systems Engineering Testing of EPS2</u> .
C5	Relation of <u>User-Perceived Response Time</u> to Error Measurement.
M1	The Generation of Random, Binary, Unordered <u>Tree</u> .
M2	Intersection <u>Graph</u> of Paths in a <u>Tree</u> .
M3	<u>Graph Minors IV: Tree-Width and Well-Quasi-Ordering</u> .
M4	<u>Graph Minors – A Survey</u> .

Burada altı çizili kelimeler ise birden fazla dokümanda geçtiği için önemli görülen terimler olarak alınmıştır.

Terim-doküman matrisini oluşturmadan önce sistemin hesaplama maliyetini azaltmak için Bölüm 2’de bahsedilen ön işlemler yapılmalıdır. Doküman yığınındaki noktalama işaretleri temizlenip dokümanlar terimlerine parçalanır. Daha sonra bu terimler duraklama kelime listesine göre kontrol edilir ve listede bulunan terimler varsa elenir. Örneğin, Çizelge 3.1’de bazı dokümanlarda bulunan “for” ve “the” gibi kelimeler duraklama listesinde bulunduğundan dolayı sonraki işleme dâhil edilmez yani elenir. Bu filtreleme işleminden geçen terimlere Porter Stemmer algoritması uygulanarak köklerine indirgenir ve terimlerin farklı kullanımları normalize edilmiş olur. Örneğin “systems” teriminden “s” eki atılarak terimin kökü olan “system” haline indirgenir.

Çizelge 3.2. Doküman Yığınına Ait Terim Doküman Matrisi

TERİMLER	DOKÜMANLAR								
	C1	C2	C3	C4	C5	M1	M2	M3	M4
computer	0,47	0,47	0	0	0	0	0	0	0
eps	0	0	0,47	0,47	0	0	0	0	0
graph	0	0	0	0	0	0	0,35	0,35	0,35
human	0,47	0	0	0,47	0	0	0	0	0
interface	0,47	0	0,47	0	0	0	0	0	0
minors	0	0	0	0	0	0	0	0,47	0,47
response	0	0,47	0	0	0,47	0	0	0	0
survey	0	0,47	0	0	0	0	0	0	0,47
systems	0	0,37	0,37	0,58	0	0	0	0	0
time	0	0,47	0	0	0,47	0	0	0	0
tree	0	0	0	0	0	0,35	0,35	0,35	0
user	0	0,35	0,35	0	0,35	0	0	0	0

Doküman yığınının ön işleme süreci bittikten sonra yerel ağırlıklandırma fonksiyonu olarak logaritma, genel ağırlıklandırma fonksiyonu olarak da entropy kullanılarak A terim-doküman matrisi hesaplanmış ve Çizelge 3.2' de gösterilmiştir.

$$U = \begin{bmatrix} 0,30 & -0,20 & 0,10 & -0,58 & 0,17 & 0,03 & 0,34 & 0,23 & -0,42 \\ 0,42 & 0,27 & -0,24 & 0,37 & -0,08 & 0,09 & -0,09 & -0,05 & 0,10 \\ -0,09 & -0,26 & -0,51 & -0,04 & -0,12 & -0,03 & -0,02 & -0,67 & -0,44 \\ 0,36 & 0,22 & -0,16 & -0,33 & 0,17 & -0,64 & -0,19 & -0,18 & 0,30 \\ 0,35 & 0,17 & -0,10 & -0,46 & -0,23 & 0,53 & -0,26 & -0,07 & 0,14 \\ -0,09 & -0,30 & -0,54 & 0 & 0,17 & -0,02 & -0,49 & 0,58 & -0,08 \\ 0,20 & -0,44 & 0,26 & 0,07 & -0,17 & -0,23 & -0,17 & -0,04 & 0,04 \\ 0,10 & -0,44 & -0,14 & 0,05 & 0,56 & 0,29 & 0,26 & -0,20 & 0,53 \\ 0,53 & 0,08 & -0,16 & 0,40 & 0,13 & -0,08 & 0,33 & 0,18 & -0,32 \\ 0,20 & -0,44 & 0,26 & 0,07 & -0,17 & -0,23 & -0,17 & -0,04 & 0,04 \\ -0,07 & -0,15 & -0,37 & -0,11 & -0,61 & -0,15 & 0,50 & 0,21 & 0,35 \\ 0,29 & -0,24 & 0,13 & 0,14 & -0,30 & 0,30 & -0,18 & -0,01 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 1,30 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0,22 & -1,10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0,02 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -0,01 & 0,01 & -0,04 & 0,72 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0,05 & 0,61 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0,01 & -0,01 & 0,52 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -0,11 & -0,41 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0,22 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0,16 \end{bmatrix}$$

$$V^T = \begin{bmatrix} 0,37 & 0,52 & 0,51 & 0,52 & 0,23 & -0,02 & -0,04 & -0,08 & -0,02 \\ -0,16 & 0,60 & -0,24 & -0,36 & 0,41 & 0,05 & 0,14 & 0,27 & 0,40 \\ 0,07 & -0,23 & 0,17 & 0,28 & -0,31 & 0,13 & 0,32 & 0,59 & 0,52 \\ -0,89 & -0,01 & 0,22 & 0,37 & 0,14 & -0,04 & -0,06 & -0,05 & 0,03 \\ 0,01 & 0,21 & -0,32 & 0,21 & -0,42 & -0,35 & -0,42 & -0,28 & 0,50 \\ -0,07 & 0,02 & 0,71 & -0,59 & -0,23 & -0,10 & -0,13 & -0,14 & 0,23 \\ 0,16 & -0,45 & 0,07 & 0,01 & 0,60 & -0,41 & -0,38 & 0,19 & 0,23 \\ 0,05 & -0,15 & -0,01 & 0,04 & 0,17 & -0,33 & 0,71 & -0,52 & 0,25 \\ -0,06 & 0,20 & 0,01 & -0,05 & -0,21 & -0,75 & 0,18 & 0,41 & -0,38 \end{bmatrix}$$

Elde edilen 12×9 boyutlu A terim-doküman matrisine Denklem (2.5)' deki gibi kesik ULV ayrışımı uygulanır ve U , L , V^T matrisleri elde edilir. Daha önce belirtildiği gibi burada da $U \in \mathbb{R}^{12 \times r}$ ve $V \in \mathbb{R}^{9 \times r}$ sol ortogonal matrislerdir.

Daha sonra U , L ve V^T matrislerine $k = 2$ olmak üzere rank-2 yaklaşımı uygulanır ve U_2 , L_2 , V_2^T matrisleri elde edilir. Burada rank-2 yaklaşımının uygulamamızın amacı terim ve doküman vektörlerini kartezyen koordinat sisteminde temsil edebilmektir. Bu sayede 2 boyutlu bir grafik üzerinde terim ve dokümanların konumları incelenebilir. Büyük boyutta rank yaklaşımları uygulanırsa terim ve dokümanların konumunu grafik üzerinden incelemek mümkün olmaz. Ancak, gizli anlamsal ilişki çıktıları sayesinde aralarındaki bağ açıklanabilir.

$$U_2 = \begin{bmatrix} 0,30 & -0,20 \\ 0,42 & 0,27 \\ -0,09 & -0,26 \\ 0,36 & 0,22 \\ 0,35 & 0,17 \\ -0,09 & -0,30 \\ 0,20 & -0,44 \\ 0,11 & -0,44 \\ 0,53 & 0,08 \\ 0,20 & -0,44 \\ -0,07 & -0,15 \\ 0,29 & -0,24 \end{bmatrix}$$

$$L_2 = \begin{bmatrix} 1,30 & 0 \\ -0,22 & -1,10 \end{bmatrix}$$

$$V_2^T = \begin{bmatrix} 0,37 & 0,52 & 0,51 & 0,52 & 0,23 & -0,02 & -0,04 & -0,08 & -0,02 \\ -0,16 & 0,60 & -0,24 & -0,36 & 0,41 & 0,05 & 0,14 & 0,27 & 0,40 \end{bmatrix}$$

Rank-2 yaklaşımı sonrasında elde edilen L_2 matrisinin SVD'si aşağıdaki şekilde hesaplanır.

$$L_2 = X_2 S_2 Y_2^T$$

Buradaki X_2 , S_2 ve Y_2^T matrisleri

$$X_2 = \begin{bmatrix} -0,90 & 0,44 \\ 0,44 & 0,90 \end{bmatrix}$$

$$S_2 = \begin{bmatrix} 1,35 & 0 \\ 0 & 1,05 \end{bmatrix}$$

$$Y_2^T = \begin{bmatrix} -0,93 & -0,36 \\ 0,36 & -0,93 \end{bmatrix}$$

biçimindedir. $\bar{U}_2 = U_2 X_2$ ve $\bar{V}_2^T = Y_2^T V_2^T$ olarak alınırsa

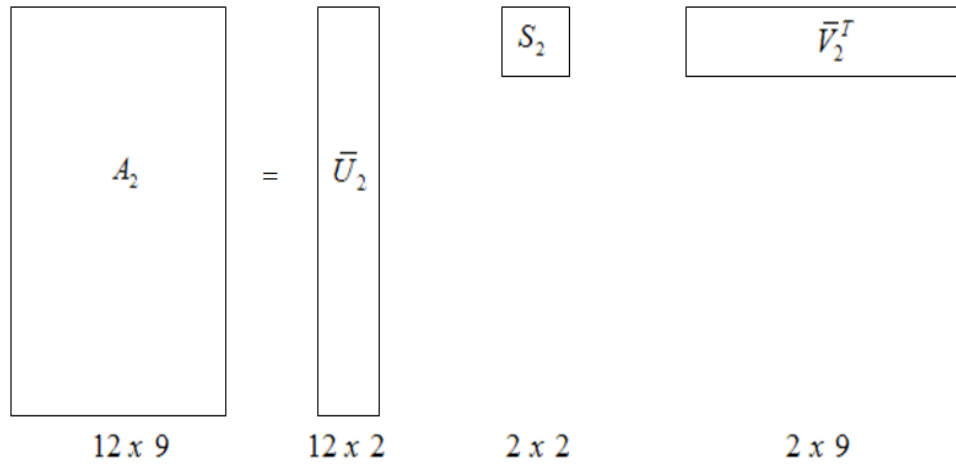
$$\bar{U}_2 = \begin{bmatrix} -0,35 & -0,05 \\ -0,26 & 0,42 \\ -0,03 & -0,27 \\ -0,23 & 0,36 \\ -0,24 & 0,31 \\ -0,05 & -0,31 \\ -0,37 & -0,30 \\ -0,29 & -0,34 \\ -0,44 & 0,30 \\ -0,37 & -0,30 \\ 0 & -0,16 \\ -0,37 & -0,09 \end{bmatrix}$$

$$\bar{V}_2^T = \begin{bmatrix} -0,29 & -0,70 & -0,39 & -0,36 & -0,36 & 0 & -0,03 & -0,02 & -0,13 \\ 0,28 & -0,37 & 0,41 & 0,52 & -0,30 & -0,05 & -0,14 & -0,28 & -0,38 \end{bmatrix}$$

olarak hesaplanır.

$$A_2 = \begin{bmatrix} 0,12 & 0,35 & 0,17 & 0,15 & 0,19 & 0 & 0,01 & 0,03 & 0,08 \\ 0,22 & 0,08 & 0,31 & 0,35 & -0,01 & -0,02 & -0,06 & -0,12 & -0,13 \\ -0,07 & 0,14 & -0,10 & -0,13 & 0,10 & 0,02 & 0,04 & 0,08 & 0,12 \\ 0,20 & 0,08 & 0,27 & 0,30 & 0 & -0,02 & -0,05 & -0,10 & -0,11 \\ 0,19 & 0,11 & 0,26 & 0,28 & 0,02 & -0,02 & -0,04 & -0,08 & -0,08 \\ -0,07 & 0,16 & -0,11 & -0,15 & 0,12 & 0,02 & 0,05 & 0,09 & 0,13 \\ 0,06 & 0,47 & 0,07 & 0,02 & 0,28 & 0,02 & 0,05 & 0,10 & 0,19 \\ 0,01 & 0,41 & 0 & -0,05 & 0,25 & 0,02 & 0,06 & 0,11 & 0,19 \\ 0,26 & 0,30 & 0,36 & 0,38 & 0,12 & -0,02 & -0,04 & -0,07 & -0,05 \\ 0,06 & 0,47 & 0,07 & 0,02 & 0,28 & 0,02 & 0,05 & 0,10 & 0,19 \\ -0,05 & 0,07 & -0,07 & -0,09 & 0,05 & 0,01 & 0,02 & 0,05 & 0,07 \\ 0,12 & 0,39 & 0,16 & 0,13 & 0,21 & 0 & 0,02 & 0,04 & 0,10 \end{bmatrix}$$

A matrisinin rank-2 yaklaşımı olan A_2 matrisi tüm çarpanları elde edildikten sonra Şekil 3.1'deki gibi elde edilir.



Şekil 3.1. A terim-doküman matrisinin rank -2 yaklaşımı

Oluşturulan 2 boyutlu vektör uzayında terim vektörleri $U_2 S_2$ matrisinin satırları, doküman vektörleri ise $S_2 V_2^T$ matrisinin sütunlarıdır.

$$\bar{U}_2 S_2 = \begin{bmatrix} -0,48 & -0,05 \\ -0,35 & 0,45 \\ -0,05 & -0,29 \\ -0,31 & 0,38 \\ -0,32 & 0,32 \\ -0,06 & -0,32 \\ -0,51 & -0,32 \\ -0,39 & -0,36 \\ -0,60 & 0,32 \\ -0,51 & -0,32 \\ 0 & -0,17 \\ -0,50 & -0,09 \end{bmatrix}$$

$$S_2 \bar{V}_2^T = \begin{bmatrix} -0,39 & -0,95 & -0,52 & -0,48 & -0,48 & 0 & -0,01 & -0,04 & -0,17 \\ 0,30 & -0,39 & 0,43 & 0,55 & -0,32 & -0,06 & -0,15 & -0,30 & -0,40 \end{bmatrix}$$

Doküman yığınınındaki terimleri ve dokümanları 2 boyutlu vektör uzayında temsil eden terim ve doküman vektörleri aşağıdaki şekildedir.

$$\text{computer} = \begin{bmatrix} -0,48 \\ -0,05 \end{bmatrix}, \text{eps} = \begin{bmatrix} -0,35 \\ 0,45 \end{bmatrix}, \text{graph} = \begin{bmatrix} -0,05 \\ -0,29 \end{bmatrix}, \text{human} = \begin{bmatrix} -0,31 \\ 0,38 \end{bmatrix}$$

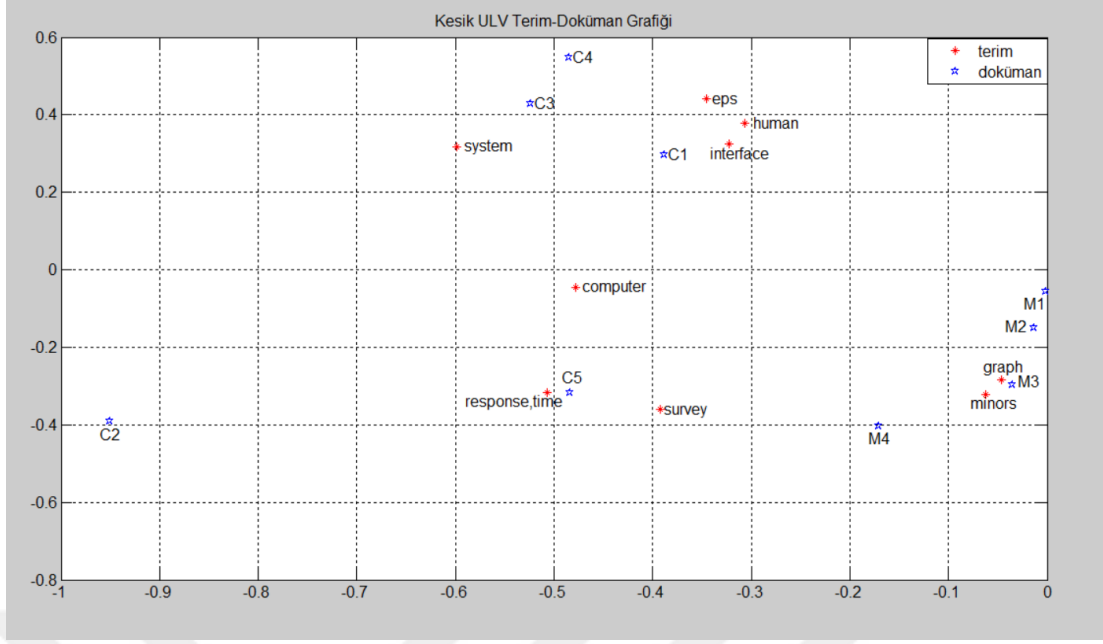
$$\text{interface} = \begin{bmatrix} -0,32 \\ 0,32 \end{bmatrix}, \text{minors} = \begin{bmatrix} -0,06 \\ -0,32 \end{bmatrix}, \text{response} = \begin{bmatrix} -0,51 \\ -0,32 \end{bmatrix}, \text{survey} = \begin{bmatrix} -0,39 \\ -0,36 \end{bmatrix}$$

$$\text{system} = \begin{bmatrix} -0,60 \\ 0,32 \end{bmatrix}, \text{time} = \begin{bmatrix} -0,51 \\ -0,32 \end{bmatrix}, \text{tree} = \begin{bmatrix} 0 \\ -0,17 \end{bmatrix}, \text{user} = \begin{bmatrix} -0,50 \\ -0,09 \end{bmatrix}$$

ve

$$C1 = \begin{bmatrix} -0,39 \\ 0,30 \end{bmatrix}, C2 = \begin{bmatrix} -0,95 \\ -0,39 \end{bmatrix}, C3 = \begin{bmatrix} -0,52 \\ 0,43 \end{bmatrix}, C4 = \begin{bmatrix} -0,48 \\ 0,55 \end{bmatrix}, C5 = \begin{bmatrix} -0,48 \\ -0,32 \end{bmatrix}$$

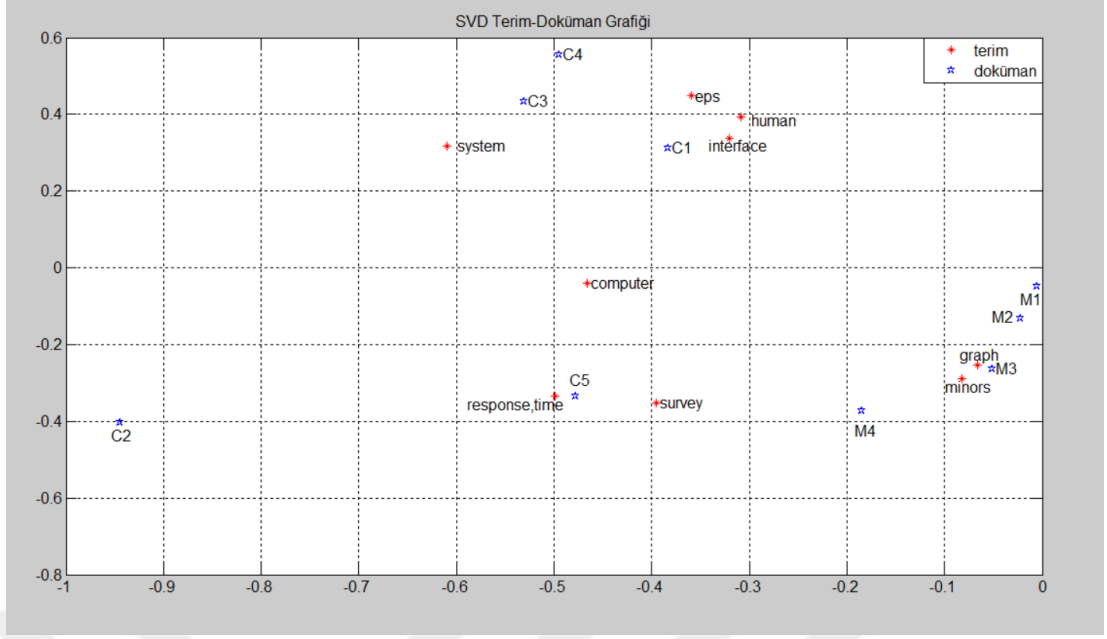
$$M1 = \begin{bmatrix} 0 \\ -0,06 \end{bmatrix}, M2 = \begin{bmatrix} -0,01 \\ -0,15 \end{bmatrix}, M3 = \begin{bmatrix} -0,04 \\ -0,30 \end{bmatrix}, M4 = \begin{bmatrix} -0,17 \\ -0,40 \end{bmatrix}$$



Şekil 3.2. Kesik ULV Ayrışımıyla Elde Edilen Terim-Doküman Grafiği

Şekil 3.2'de terim ve doküman vektörlerinden elde edilen doküman yığına ait terim-doküman grafiği verilmiştir. Grafik incelendiğinde terim ve dokümanların konumları ile anlamsal ilişkileri bağ olabileceği düşünülebilir. Örneğin, C1 ve C3 dokümanları yakın anlamlı olarak gözükmemektedir. Ancak bu tarz çıkarımların temeli öklid uzaklığına dayandığı için çoğu zaman sağlıklı değildir. Örneğin, C5 dokümanı, diğer C etiketli dokümanlara nazaran M etiketli dokümanlara daha yakın durmaktadır. Fakat anlamsal olarak C etiketli dokümanlarla ilişkili olduğunu biliyoruz. Dolayısıyla daha önce de belirttiğimiz gibi benzerlik ilişkisini ölçerken, terim ve doküman vektörlerinin öklid uzaklığına değil aralarındaki açının kosinüs değerine dikkat edilmesi daha doğrudur.

Şekil 3.3'de ise terim-doküman matrisinin SVD'si kullanılarak elde edilen terim-doküman grafiği verilmiştir. Bu iki terim-doküman grafiği incelendiğinde aralarında bir farkın olmadığı görülür. Bunun ana sebebi L matrisinin tekil değerlerinin A matrisinin tekil değerlerine yaklaşmasıdır. Ayrıca rank-k yaklaşımındaki k 'nın 2 seçilmesinde yani vektör 2 boyutlu olmasının da bunda payı vardır. Eğer k daha büyük seçilirse kesik ULV ve SVD ayrışimlarıyla oluşturulan anlamsal yapılar arasında küçük farklar olur fakat bunlar ciddi boyutta değildir.



Şekil 3.3. SVD İle Elde Edilen Terim-Doküman Grafiği

3.2. Sorgu Oluşturma

Doküman yığınının anlamsal yapısı 2 boyutlu vektör uzayı aracılığıyla keşfedilmiş ve oluşturulan sistemde artık terimler indekslenmiş durumdadır. Bu sayede kullanıcı tarafından girilen bir sorgu, vektör uzayına dâhil edilip dokümanlar ile benzerliği incelenebilir; ancak girilen sorguda bulunan fakat indekslenmeyen terimler işleme dâhil edilmez. Benzerlik ölçümünde daha önceden bahsettiğimiz kosinüs benzerlik metodu kullanılır ve benzerlik oranı belli bir eşik değerinin üstünde olan dokümanlar geri döndürülür. Eşik değerinin değişmesi durumunda geri döndürülen dokümanların sayısı da değişebilir.

Örneğin, kullanıcının girdiği sorgu “human computer interaction” olsun ve q sorgu vektörüyle temsil edilsin. Sorgu cümlesindeki “interaction ” kelimesi indekslenen terimler arasında olmadığından dikkate alınmaz. Burada sorgu vektörü bir doküman vektörü olarak düşünülüp terim-doküman matrisinde kullanılan ağırlıklandırma metotlarına göre hesaplanır.

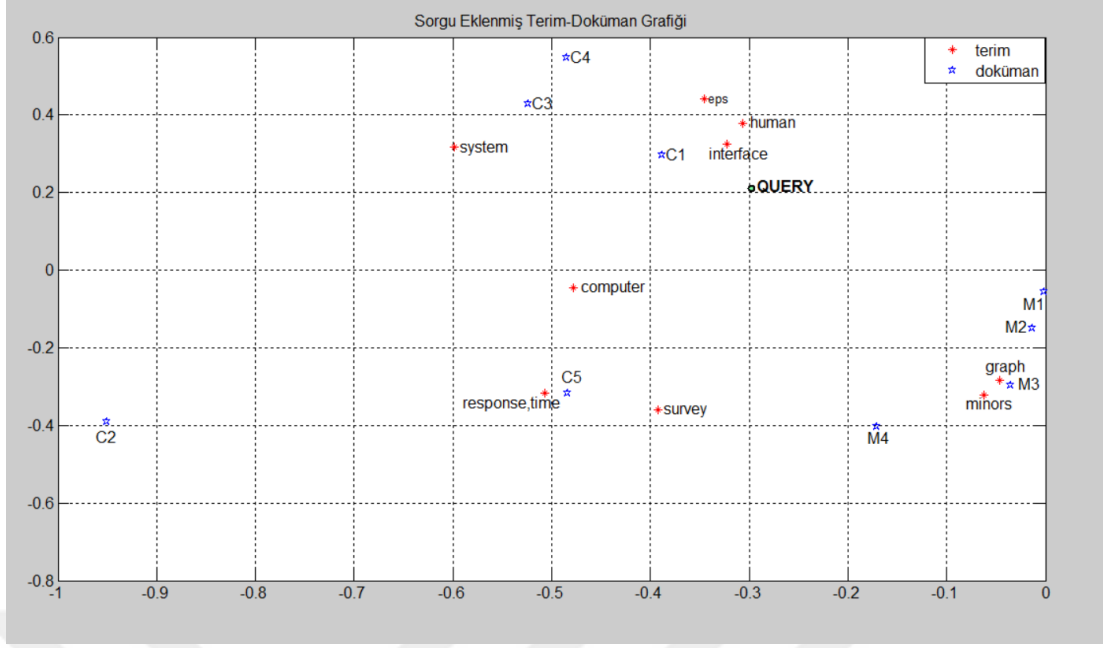
$$q = \begin{bmatrix} 0,69 \\ 0 \\ 0 \\ 0,69 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Daha sonra elde edilen q sorgu vektörü 2-boyutlu vektör uzayına Denklem (2.22) kullanılarak eklenir ve \hat{q} vektörü elde edilir.

$$\hat{q} = \begin{bmatrix} 0,69 \\ 0 \\ 0 \\ 0,69 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} -0,35 & -0,05 \\ -0,26 & 0,42 \\ -0,03 & -0,27 \\ -0,23 & 0,36 \\ -0,24 & 0,31 \\ -0,05 & -0,31 \\ -0,37 & -0,30 \\ -0,29 & -0,34 \\ -0,44 & 0,30 \\ -0,37 & -0,30 \\ 0 & -0,16 \\ -0,37 & -0,09 \end{bmatrix} \begin{bmatrix} 1,35 & 0 \\ 0 & 1,05 \end{bmatrix}^{-1}$$

$$\hat{q} = \begin{bmatrix} -0,30 \\ 0,21 \end{bmatrix}$$

\hat{q} sorgu vektörünün 2 boyutlu vektör uzayına dâhil edilmiş hali Şekil 3.4'de gösterilmiştir.



Şekil 3.4. Sorgu Eklenmiş Terim-Doküman Grafiği

Sorgu vektörü ile doküman vektörlerinin benzerliğinin k değerine bağlı değişimi Çizelge 3.3’de gösterilmiştir. Dokümanların sorguyla olan benzerliğini bulduktan sonra geri döndürülecek dokümanları belirlemek için bir eşik değerine ihtiyacımız vardır. Örneğin, $k = 2$ için eşik değeri 0,3 alalım. Bu durumda sistem, sırasıyla C1 C3, C4, C2 ve C5 dokümanlarını kullanıcıya geri döndürür. Eğer $k = 9$ için eşik değerini 0,1 olarak belirlersek, sistem sırasıyla C1, C4 ve C2 dokümanlarını sonuç olarak geri döndürür.

Oluşturduğumuz bilgi erişim sisteminde $k = 2$ ve eşik değer 0,3 alındığında C etiketli dokümanların hepsi geri döndürüldü. Eğer sözcüksel eşlemeye dayanan bir metot kullanmış olsaydık sadece C1, C2 ve C4 dokümanlarının sonuç olarak döndürülmesi gerekirdi. Ancak, LSI sorguyla ortak kelime içermeyen fakat anlamsal olarak yakın olan C3 ve C5 dokümanlarını da sonuç olarak geri döndürdü. Bu durum, LSI’nın yalnızca sözcükleri değil aynı zamanda anlamsal ilişkileri de dikkate aldığına göstergesidir ve erişim performansını da pozitif olarak etkiler.

Çizelge 3.3. Farklı k Değerleri İçin Sorgu ile Dokümanların Benzerliği

$k = 2$		$k = 4$		$k = 9$	
C1	0,99	C1	0,97	C1	0,68
C2	0,53	C2	0,20	C2	0,18
C3	0,99	C3	0,17	C3	-0,22
C4	0,97	C4	0,06	C4	0,27
C5	0,37	C5	-0,04	C5	-0,07
M1	-0,56	M1	0,17	M1	-0,08
M2	-0,50	M2	0,07	M2	0
M3	-0,48	M3	0,01	M3	0,05
M4	-0,21	M4	-0,04	M4	-0,07

LSI'nın seçilen doküman yığımındaki performans değerlendirmesini yaparken k 'yi 2 eşik değerini ise 0,9 olarak kabul edelim. Bu durumda 3 doküman sonuç olarak döndürülür ve değerlendirme ölçütleri

$$\text{Geri çağırma} = \frac{3}{5} = \%60$$

$$\text{Hassasiyet} = \frac{3}{3} = \%100$$

şeklindedir. Eğer LSI sisteminde matris ayrışımı yöntemi olarak SVD'yi kullanırsak sorgu ile dokümanlar arasındaki benzerlik oranları k 'ya bağlı olarak Çizelge 3.4'deki gibi olur.

Çizelge 3.4. SVD İle Farklı k Değerleri İçin Sorgu ile Dokümanların Benzerliği

$k = 2$		$k = 4$		$k = 9$	
C1	0,99	C1	0,96	C1	0,68
C2	0,50	C2	0,21	C2	0,18
C3	0,99	C3	0,09	C3	-0,22
C4	0,98	C4	0,08	C4	0,27
C5	0,30	C5	-0,12	C5	-0,07
M1	-0,51	M1	-0,09	M1	-0,08
M2	-0,47	M2	-0,06	M2	0
M3	-0,44	M3	-0,04	M3	0,05
M4	-0,19	M4	0,04	M4	-0,07

Çizelge 3.3 ve Çizelge 3.4 incelenirse her iki sistemde de, oluşturulan sorgunun dokümanlara benzerlik oranlarının çok yakın olduğu gözlemlenebilir.

Örneğin, SVD tabanlı LSI sisteminin $k = 2$ ve eşik değerinin 0,3 olduğu durumda geri döndürdüğü dokümanlar, kesik ULV tabanlı sistemde olduğu gibi sırasıyla C1, C3, C4, C2 ve C5 şeklindedir. SVD tabanlı LSI sisteminde performans değerlendirmesini yaparken yine k 'yı 2 ve eşik değerini 0,9 olarak kabul edelim. Bu durumda, performans ölçütleri aşağıdaki şekilde olur.

$$\text{Geri çağırma} = \frac{3}{5} = \%60$$

$$\text{Hassasiyet} = \frac{3}{3} = \%100$$

3.3. Anlamsal Yapının Güncellenmesi

Çizelge 3.1’de gösterilen doküman yığınınına yeni bir doküman veya bir terim eklemek istersek doküman yığınınını temsil eden anlamsal yapıyı da güncelleştirmemiz gerekmektedir. Örneğin, M1 etiketli dokümanda bulunan fakat daha önce indekslenmeyen “random” terimi folding-in metoduyla Denklem (2.29) kullanılarak anlamsal yapıya dâhil edilebilir. Burada, öncelikle yeni eklenen terimin vektörü olan t terim-doküman matrisinde kullanılan ağırlıklandırma yöntemlerine göre oluşturulur.

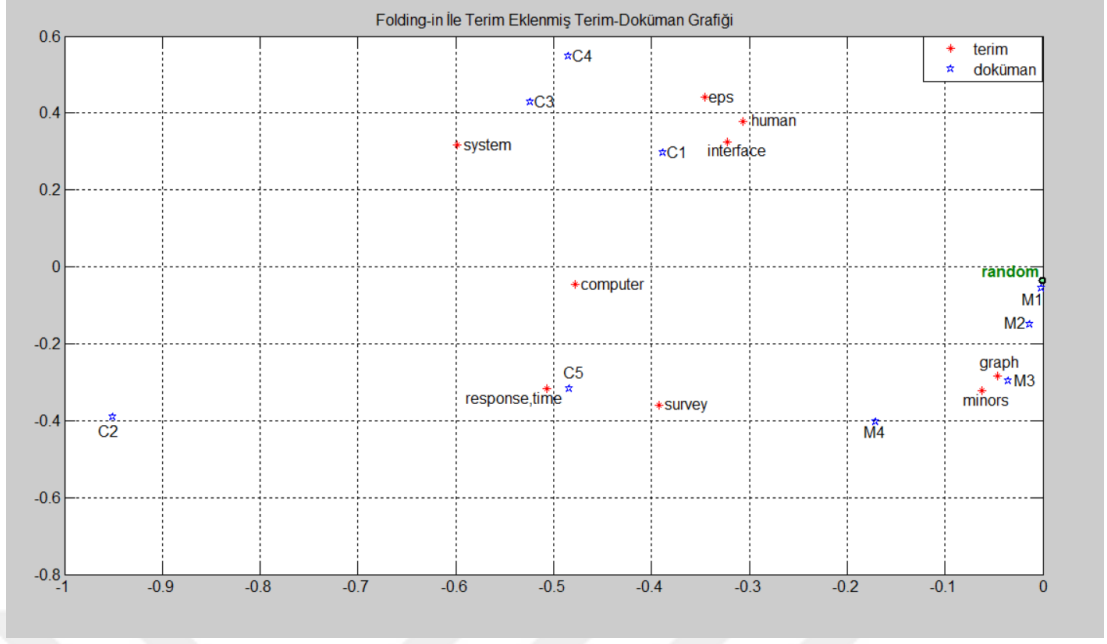
$$t = [0 \ 0 \ 0 \ 0 \ 0 \ 0,69 \ 0 \ 0 \ 0]$$

Daha sonra \bar{V}_k ve S_k^{-1} matrisleri kullanılarak t terim vektörünün 2 boyutlu vektör uzayındaki temsilcisi olan \hat{t} vektörü elde edilir.

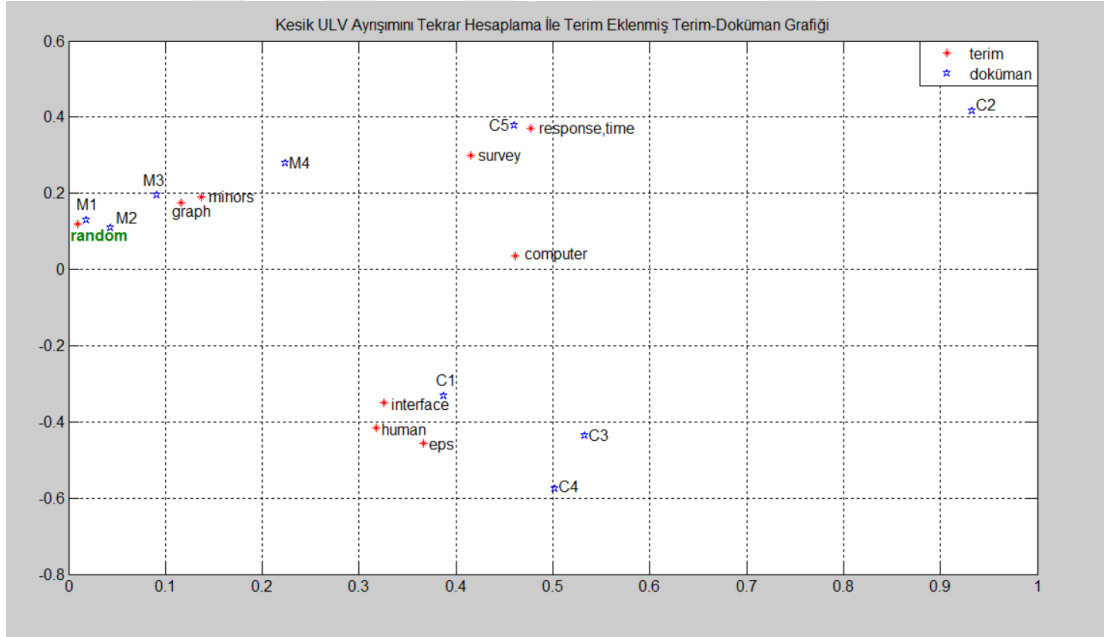
$$\hat{t} = [0 \ 0 \ 0 \ 0 \ 0 \ 0,69 \ 0 \ 0 \ 0] \begin{bmatrix} -0,29 & 0,28 \\ -0,70 & -0,37 \\ -0,39 & 0,41 \\ -0,36 & 0,52 \\ -0,36 & -0,30 \\ 0 & -0,05 \\ 0 & -0,14 \\ -0,03 & -0,28 \\ -0,13 & -0,38 \end{bmatrix} \begin{bmatrix} 1,35 & 0 \\ 0 & 1,05 \end{bmatrix}^{-1}$$

$$\hat{t} = \begin{bmatrix} 0 \\ -0, \end{bmatrix}$$

Şekil 3.5’de “random” terim vektörünün terim doküman grafiğine folding-in metoduyla eklenmiş hali gösterilmektedir. Daha öncede teorik olarak bahsedildiği üzere, yeni eklenen “random” terimin mevcut terim ve dokümanlar üzerinde herhangi bir etkisi olmamıştır.



Şekil 3.5. Folding-in Metoduyla Terim Ekleme Grafiği

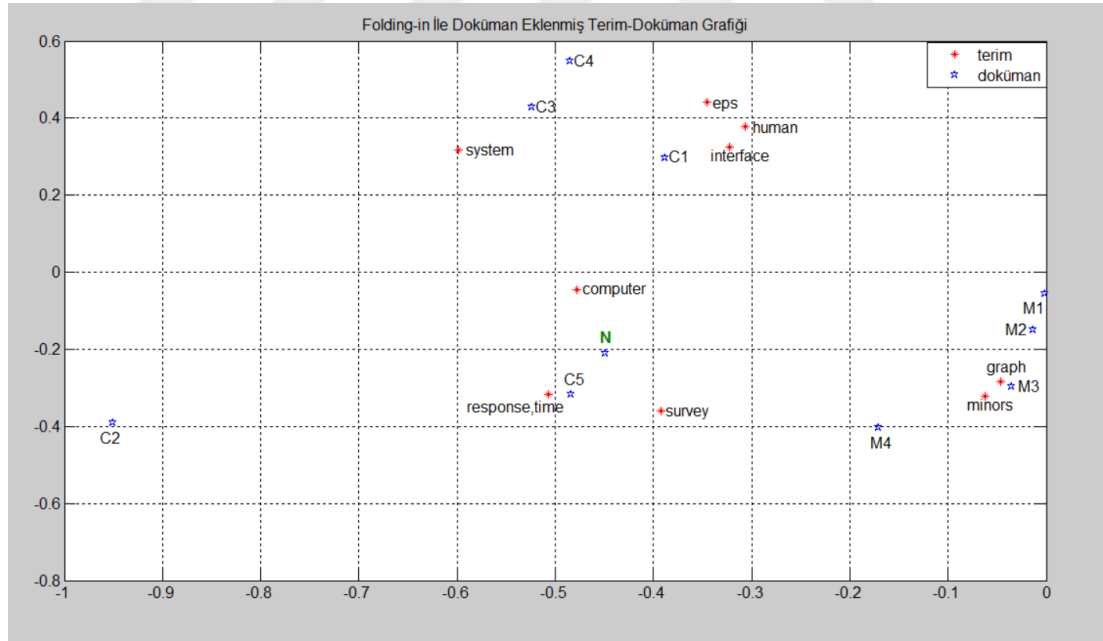


Şekil 3.6. Kesik ULV Ayrışımını Tekrar Hesaplama İle Terim Ekleme Grafiği

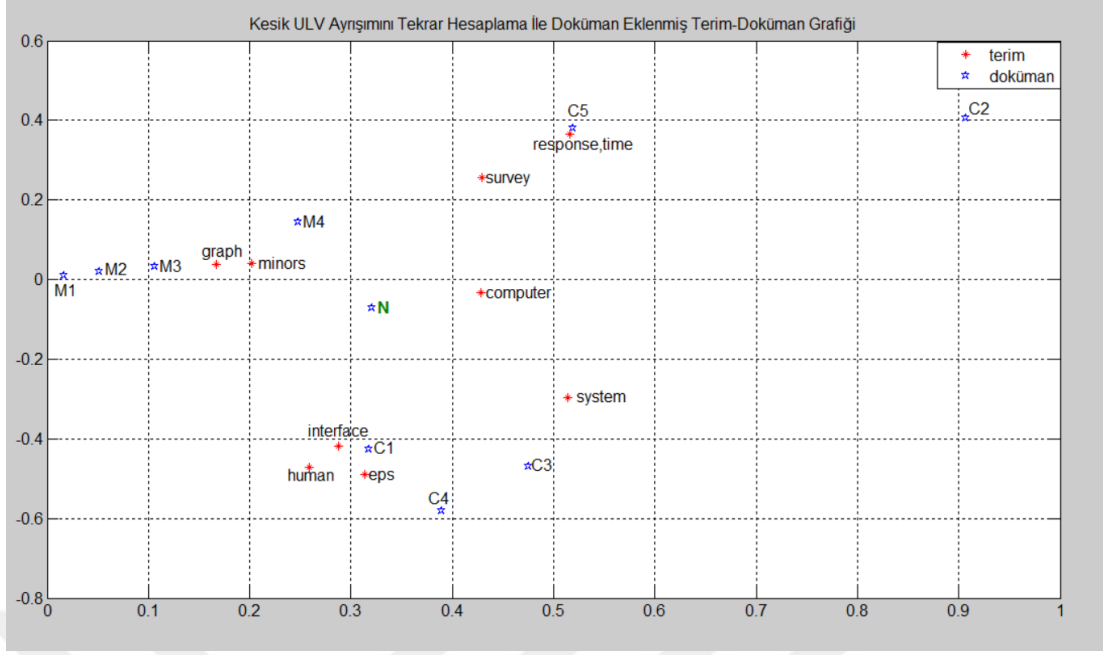
Yeni eklenen "random" teriminin anlamsal yapıya etkisi, güncelleme işleminde kesik ULV ayrışımının tekrar hesaplanması metodunun kullanılmasıyla daha net bir şekilde

görülür. Bu yöntemde “random” terimi önemli terim kümesine dâhil edilir ve yeni terim-doküman matrisi oluşturulur. Bundan sonraki adımlar aynı şekilde uygulanarak Şekil 3.6’daki gibi güncellenen anlamsal yapı elde edilir.

Eğer doküman yığınınına N etiketli “Graph Minors Implemented on Computer Systems” dokümanı eklenirse anlamsal yapı, Denklem (2.28) kullanılarak folding-in veya kesik ULV ayrışımını tekrar hesaplama metotlarıyla güncellenebilir. Şekil 3.7’de doküman yığınınına dâhil edilen N dokümanının folding-in metoduyla anlamsal yapıya eklenmiş halini gösterilmektedir. Şekil 3.8’de ise yine N dokümanının anlamsal yapıya kesik ULV ayrışımını tekrar hesaplama metoduyla eklenmiş hali gösterilmektedir. İki şekilde incelendiği zaman folding-in metodunun daha önce de bahsedildiği gibi tam olarak güncel anlamsal yapıyı üretmediği görülür. Kesik ULV ayrışımını tekrar hesaplama metodu ise mevcut anlamsal yapıyı doğru bir şekilde güncelleyerek yeni anlamsal yapıyı üretir. Ancak büyük boyutlu terim-doküman matrisleri için hafıza ve zaman açısından oldukça maliyetlidir.



Şekil 3.7. Folding-in Metoduyla Doküman Ekleme Grafiği



Şekil 3.8. Kesik ULV Ayrışımını Tekrar Hesaplama İle Doküman Ekleme Grafiği

4. TARTIŞMA VE SONUÇ

Günümüzde teknolojinin hızlı gelişimiyle birlikte dijital ortamdaki doküman yığınlarının da sayısı aşırı derecede artmıştır. Bu artışla doğru orantılı olarak insanların aradıkları bilgi veya dokümanlara erişimi de zorlaşmıştır. Bu devasa doküman yığınları arasında istenilen dokümanı bulabilmek için farklı yöntemler geliştirilmiştir. Bilgi erişim sistemleri de bu problemi çözebilmek için geliştirilen yöntemlerden biridir. Bu sistemler sayesinde kullanıcılar aradıkları dokümanlara daha kolay ulaşmaya başlamışlardır.

Başlıca bilgi erişim sistemleri mantıksal modeller, vektör uzay modelleri ve olasılıklı modellerdir. Mantıksal erişim modelleri klasik küme teorisine ve mantık cebriyle dayanan basit bir yöntemdir. Bir mantıksal erişim modelinde sorgu terimleri mantıksal operatörlere bağlıdır ve sistem bu mantıksal operatörlerin sınırladığı dokümanları geri döndürür. Olasılıklı modeller ise sorguda geçen terimlerin dokümanlarda dağılımını inceler. Bu sayede sorgu ile dokümanların benzerliği hesaplanır. Vektör uzayı modelinde ise dokümanlar ve sorgular terim vektörlerinden oluşturulan bir uzayda temsil edilir. Sorgu ile dokümanların ilişkisi öklid uzaklığı veya kosinüs benzerliği gibi yöntemlerle belirlenir.

Bilgi erişim sistemlerinin başarısı geri çağırma ve hassasiyet ölçütlerine göre belirlenir. Bu ölçütlerin değerinin 1 olması durumunda sistemin başarısının %100 olduğu anlaşılır. Yani, kusursuz bir bilgi erişim sisteminde yapılan sorgu sonucu geri döndürülen tüm dokümanlar sorguyla ilişkili olmalıdır. Ayrıca, geri döndürülmeyen ama sorguyla ilişkili doküman kalmamalıdır.

Geleneksel bilgi erişim sistemleri direkt olarak kelime eşleme mantığıyla çalıştıkları için kullanılan dilin iç farklılıklarıyla ilgili sıkıntı yaşar. Örneğin, sorgularda bulunan eş anlamlı ve çok anlamlı kelimeler bu sistemlerin erişim başarısını düşürmektedir. Bu ve buna benzer problemleri çözebilmek için kelime eşleme tabanlı olmayan ve doküman yığınındaki anlamsal yapıyı keşfeden LSA yöntemi ortaya atılmıştır. LSA çağrışım ve anlamsal benzerlik gibi insana ait bilişsel olguları kullanarak terim ve

dokümanların anlamlarını tanımlar. LSA kullanılarak oluşturulan bilgi erişim sistemi ise LSI olarak adlandırılır.

LSI doküman yığımından elde edilen terim-doküman matrisine SVD uygulanır. Bu sayede elde edilen sıralı tekil değerler incelenerek bir k kırılma noktası belirlenir ve bu noktadan bir kırılma işlemi uygulanır. Aynı zamanda sağ ve sol tekil vektörlerinde belirlenen kısmı atılır. Bu işlem sonucunda doküman yığımını temsil eden anlamsal yapıyı bozan gürültü temizlenmiş olur. Tekil vektörler ve tekil değerler kullanılarak k boyutlu vektör uzayı oluşturulur ve kullanıcıdan alınan sorgunun vektörü elde edilerek k boyutlu vektör uzayına dâhil edilir. Sorgu vektörü ile doküman vektörlerinin kosinüs benzerliği hesaplanarak belirlenen dokümanlar kullanıcıya geri döndürülür. Geri döndürülen dokümanların durumlarına göre geri çağırma ve hassasiyet ölçütleri hesaplanır ve sistemin performansı değerlendirilir.

LSI, matris ayrışımı olarak genellikle SVD'yi kullanır; ancak SVD'nin maliyeti çok yüksektir. Örneğin, $m \times n$ boyutlu bir matrisin SVD'sinin hesaplama karmaşıklığı $O(mn^2)$ dir. Bu hesaplama karmaşıklığını düşürmek ve doğru anlamsal yapı üreten matris ayrışımını LSI'da kullanmak zamanla önemli bir çalışma konusu haline gelmiştir.

Bu çalışmamızda, LSI'da SVD yerine hesaplama karmaşıklığı özellikle güncelleme işlemlerinde düşük olan kesik ULV ayrışımı kullanılmıştır. Seçilen doküman yığımını temsil eden terim-doküman matrisi logaritma ve entropy ağırlıklandırma yöntemlerine göre oluşturulmuştur. Terim doküman matrisine uygulanan SVD ve kesik ULV ayrışimleri ile elde edilen 2 boyutlu vektör uzayındaki anlamsal yapılar neredeyse aynıdır. Yine bu ayrışimleri kullanarak oluşturulan LSI sistemlerinde de sorgu ile dokümanlar arasındaki benzerlik oranları çok yakın çıkmıştır. Bu iki sistemde $k = 2$ ve eşik değerinin 0,3 olduğu durumda, girilen sorguya karşılık sırasıyla C1, C3, C4, C2 ve C5 dokümanlarını geri döndürmüştür.

SVD tabanlı LSI sisteminin performansı $k = 2$ ve 0,9 eşik değeri alınarak ölçülmüştür. Bunun sonucunda, geri çağırma da %60 hassasiyet de ise %100 başarı oranı

yakalanmıştır. Benzer şekilde kesik ULV ayrışımı tabanlı LSI sisteminin performansı yine aynı rank yaklaşımı ve eşik değeri altında incelenmiştir. Sonuç olarak SVD tabanlı sistemde olduğu gibi geri çağırma da %60 hassasiyet de ise %100 başarı oranı yakalanmıştır.

Kesik ULV ayrışımını kullanarak oluşturulan anlamsal yapı folding-in ve kesik ULV ayrışımını tekrar hesaplama yöntemlerine göre güncellenmiştir. Bu metotlara göre güncellenen anlamsal yapılar incelendiğinde kesik ULV ayrışımını tekrar hesaplama yönteminin folding-in yönteminden daha başarılı olduğu gözlemlenmiştir.

İlerleyen çalışmalarda ise doküman yığını temsil eden mevcut anlamsal yapıyı doküman yığına yeni eklenen terim ve dokümanların etkisine göre güncellenmesi planlanmaktadır. Bunun için, var olan terim-doküman matrisine ait kesik ULV ayrışımını kullanarak yeni terim-doküman matrisinin kesik ULV ayrışımı blok güncelleme algoritması ile hesaplanması hedeflenmektedir.

KAYNAKLAR

- [1] Manning, C.D., Raghavan, P., Schütze, H., An Introduction to Information Retrieval. Cambridge University Press, Cambridge, 2009.
- [2] Raghavan, V. V., Wong, S. M., A critical analysis of vector space model for information retrieval. Journal of the American Society for information Science. 37(5): 279, 1986.
- [3] Furnas, G. W., Landauer, T. K., Gomez, L. M., Dumais, S. T., The vocabulary problem in human-system communication. Communications of the ACM. 30 (11): 964-971, 1987.
- [4] Dumais, S. T., Latent semantic analysis. Annual review of information science and technology. 38 (1): 188-230, 2004.
- [5] 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, 1990.
- [6] Dumais, S. T., Latent semantic indexing (LSI) and TREC-2. NIST SPECIAL PUBLICATION SP, 1994.
- [7] Dumais, S. T., Improving the retrieval of information from external sources. Behavior Research Methods, Instruments, & Computers. 23 (2): 229-236, 1991.
- [8] O'Brien, G. W., Information management tools for updating an SVD-encoded indexing scheme. MS Thesis. University of Tennessee, Knoxville, 1994.
- [9] Berry, M.W., Fierro, R. D., Low-rank orthogonal decompositions for information retrieval applications. Numerical linear algebra with applications. 3 (4): 301-327, 1996.
- [10] Kolda, T. G., O'leary, D. P., A semidiscrete matrix decomposition for latent semantic indexing information retrieval. ACM Transactions on Information Systems (TOIS). 16 (4): 322-346, 1998.
- [11] Alexandrov, V. N., Dimov, I. T., Karaivanova, A., Tan, C. J., Parallel Monte Carlo algorithms for information retrieval. Mathematics and Computers in Simulation. 62 (3): 289-295, 2003.

- [12] Gao, J., Zhang, J., Text retrieval using sparsified concept decomposition matrix. International Conference on Computational and Information Science, December 2004, Berlin, s. 523-529, 2004.
- [13] Jessup, E. R., Martin, J. H., Taking a new look at the latent semantic analysis approach to information retrieval. Computational information retrieval. 121-144, 2001.
- [14] Ozsoy, M. G., Cicekli, I., Alpaslan, F. N., Text summarization of turkish texts using latent semantic analysis. 23rd international conference on computational linguistics. Association for Computational Linguistics, August 2010, Beijing, s. 869-876, 2010.
- [15] E. Duman, Web Sayfalarının Gizli Anlam Analizi Yaklaşımıyla Otomatik Olarak Sınıflandırılması. Yüksek Lisans Tezi. Kırıkkale Üniversitesi, Kırıkkale, 2013.
- [16] Berry, M. W., Large-scale sparse singular value computations. International Journal of Supercomputer Applications. 6 (1): 13-49, 1992.
- [17] Golub, G. H., Van Loan, C. F., Matrix computations. JHU Press, 2012.
- [18] Björck, A., Numerical methods for least squares problems. Siam, 1996.
- [19] Barlow, J., Erbay, H., Modifiable low-rank approximation to a matrix. Numerical Linear Algebra with Applications. 16 (10): 833-860, 2009.
- [20] Erbay, H., Varcin, F., Horasan, F., Alternate Low Rank Approximation In Latent Semantic Analysis. 21th International Conference Mathematical Modelling and Analysis, June 2016, Tartu, 2016.
- [21] Landauer, T. K., Foltz, P. W., Laham, D., An introduction to latent semantic analysis. Discourse processes. 25 (2-3): 259-284, 1998.
- [22] Anonim, Stop Words List - English, <http://www.ranks.nl/resources/stopwords.html> (Erişim tarihi: 16.01.2016)
- [23] Porter, M., Porter Stemmer. <http://www.tartarus.org/~martin/PorterStemmer>. (Erişim tarihi: 20.01.2016)
- [24] Ozsoy, M. G., Gizil Anlamsal Analiz Yöntemi İle Doküman Özeti Çıkarma. Yüksek Lisans Tezi. Orta Doğu Teknik Üniversitesi, Ankara, 2011.

- [25] Berry, M. W., Browne, M., Understanding search engines: mathematical modeling and text retrieval, SIAM, 2005.
- [26] Landauer, T. K., McNamara, D. S., Dennis, S., Kintsch, W., Handbook of latent semantic analysis, Psychology Press, 2013.
- [27] Varcin, F., Erbay, H., Horasan, F., Latent semantic analysis via truncated ULV decomposition. 24th Signal Processing and Communication Application Conference (SIU), Mayıs 2016, Zonguldak, s. 1333-1336, 2016.
- [28] Berry, M. W., Dumais, S. T., O'Brien, G. W., Using linear algebra for intelligent information retrieval. SIAM review. 37 (4): 573-595, 1995.
- [29] Lochbaum, K. E., Streeter, L. A., Comparing and combining the effectiveness of latent semantic indexing and the ordinary vector space model for information retrieval. Information Processing & Management. 25 (6): 665-676, 1989.
- [30] Huang, A., Similarity measures for text document clustering. Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), April 2008, New Zealand, s. 49-56, 2008.
- [31] Voorhees, H., Poggio, T., Computing texture boundaries from images. Nature. 6171(333): 364-367, 1988.
- [32] Baeza-Yates, R., Riberio-Neto, B., Modern information retrieval, New York: ACM press, 1999.
- [33] Grossman, D. A., Frieder, O., Information retrieval: Algorithms and heuristics, Springer Science & Business Media, 2012.
- [34] Tapan, P., Routray, A., Kabi, B., Comparative Evaluation of Symmetric SVD Algorithms for Real-Time Face and Eye Tracking. Matrix information geometry, Berlin, s. 323-340, 2013.