FEN BİLİMLERİ ENSTİTÜSÜ

DOKÜMANLARI ÇIKARIMSAL ÖZETLEMEK İÇİN PARAGRAFLARI ÖNEME GÖRE SIRALAMA

Ahmet İlkay KISAYOL

Dr. Öğr. Üyesi Metin TURAN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL - 2018

KABUL VE ONAY SAYFASI

Ahmet İlkay KISAYOL tarafından hazırlanan **"Dokümanları Çıkarımsal Özetlemek İçin Paragrafları Öneme Göre Sıralama"** adlı tez çalışması 09/07/2018 tarihinde aşağıdaki jüri üyeleri önünde başarı ile savunularak, İstanbul Ticaret Üniversitesi Fen Bilimleri Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Danışman	Dr. Öğr. Üyesi Metin TURAN İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	**Prof. Dr.**Abdül Halim ZAİM İstanbul Ticaret Üniversitesi	.....
Jüri Üyesi	Dr. Öğr. Üyesi Murat ORHUN İstanbul Bilgi Üniversitesi	.....

Onay Tarihi : 23/07/2018

Prof. Dr. Necip Şimşek
Enstitü Müdürü

AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında,

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversitede veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

09/07/2018

Ahmet İlkay KISAYOL

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

DOKÜMANLARI ÇIKARIMSAL ÖZETLEMEK İÇİN PARAGRAFLARI ÖNEME GÖRE SIRALAMA

Ahmet İlkay KISAYOL

İstanbul Ticaret Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Metin TURAN
2018, 47 sayfa

Özetleme, bir bakıma metinleri kısaltma işlemidir. Bu kısaltma işlemi metinlerdeki önemli bilgileri içerecek şekilde olmalıdır. Bu çalışmanın amacı da İngilizce dilinde yazılmış makale, haber vs. gibi doküman paragraflarının içerdiği bilgi önemine göre seçilerek özetleme yapılmasıdır.

Çalışmanın ilk aşamasında doküman kümesini temsil edecek önemli kelimeler belirlenmiştir. Bu aşamada tüm dokümanlarda geçen kelimeler kök geçiş sıklıklarına göre büyükten küçüğe göre sıralanır ve belirli sayıda seçilen en sık kelimeler ile paragraf vektörü temsil edilir.

Bir sonraki aşamada, istenilen özet oranına göre paragraflar kümelere ayrıştırılır. Kümeleme algoritması olarak K-Means kullanılmıştır. Kümeler oluşturulurken başlangıç noktalarının belirlenmesi amacıyla iki farklı yöntem kullanılmıştır. İlk yöntemde, geçiş sıklıkları en yüksek ilk 10 kelimeden birinin en fazla görüldüğü paragraflar küme başlangıçları olarak seçilir. İkinci yöntemde, kullanıcının belirlediği özet oranına göre seçilecek anahtar kelime sayısı belirlenir. Daha sonra bu anahtar kelimelerin en çok geçtiği paragraflar başlangıç noktaları olarak belirlenir. Özet oluşturmada çıkarım yöntemi olarak, ayrıştırılan her bir küme içinden kümenin merkez noktasına Jaccard uzaklığı bakımından en yakın olan paragraf seçimi uygulanmıştır. Çıkan sonuçlar kontrol edildiğinde ikinci yöntemin daha başarılı bir sonuç verdiği gözlemlenmiştir. İkinci yönteme göre başarı oranları %20 özet oranı için %40 , %40 özet oranı için %50 ve %60 özet oranı için %71 elde edilmiştir.

Anahtar Kelimeler: çoklu dokümanlarda özetleme, metin öbekleme, özellik çıkarımı, paragraf tabanlı özetleme

ABSTRACT

M.Sc. Thesis

ORDERING PARAGRAPHS BY IMPORTANCE FOR EXTRACTIVE SUMMARIZATION OF DOCUMENTS

Ahmet İlkay KISAYOL

İstanbul Commerce University

Graduate School of Applied and Natural Sciences
Department of Computer Enginear

Supervisor: Assist Prof. Dr. Metin TURAN
2018, 47 pages

Summarization is means of process of the abbreviation of a text. This abbreviation should be such that it contains important information about the texts. The purpose of this study is selecting paragraphs according to the importance of the information contained in paragraphs of documents such as articles, news, etc. written in English.

During the first phase of the study, important words that represents the document set were identified. At this stage, the words in all the documents are sorted according to the frequency of their stems in ascending order and paragraph vector are represented by a certain number of most frequently limited selected words.

In the next step, the paragraphs are separated into clusters according to the desired summary ratio. K-Means was used as the clustering algorithm. Two different methods were used to determine the starting points when the clusters were constructed. In the first method, the paragraphs with the highest frequency of passage of one of the first 10 words are selected as the cluster starts. In the second method, the number of keywords to be selected is determined according to the summary ratio determined by the user. Then the paragraphs most often passed by these keywords are set as starting points. As an extraction method in the summarization, the paragraph selection which is closest to Jaccard distance to the central point of the cluster is applied for all clusters. When the results were checked, it was observed that the second method gave a more successful result. Success rates according to the second method were 40% for the 20% summary rate, 50% for the 40% summary rate and 71% for the summary rate.

Keywords: feature extraction, multiple document summarize, paragraph base summarization, text grouping

TEŞEKKÜR

Doğal dil işleme konusuyla ilk tanışmama vesile olan Metin Turan, bu alandaki çalışmaları ve bilgi birikimi ile tez konumu bu alanda seçmeme ön ayak oldu. Araştırma ve geliştirme aşamasında karşılaştığım zorlukları aşmamda yardımcı olan değerli danışman hocam Dr. Öğr. Üyesi Metin TURAN'a sonsuz teşekkürlerimi sunarım. Tez çalışmamın testlerinde emeği geçen ve bu süreçte hep destek olan Pixelsoftoffice'de çalışan iş arkadaşlarıma teşekkürü bir borç bilirim.

Bu süreçte her zaman yanımda olan sevgili nişanlıma ve eğitim hayatım boyunca bana sürekli destek veren maddi ve manevi yardımlarını hiçbir zaman esirgemeyen değerli aileme teşekkür ederim.

Ahmet İlkay KISAYOL
İSTANBUL, 2018

ŞEKİLLER

ÇİZELGELER

SİMGELER VE KISALTMALAR

BE Bilgi erişimi
OMÖ Otomatik metin özetleme

1. GİRİŞ

İnternetin hızlı bir şekilde büyümesiyle, insanlar çok miktarda çevrimiçi bilgi ve belge içinde boğuluyorlar. Bu çok miktardaki belgelerin kullanılabilirliği, otomatik metin özetleme alanında kapsamlı araştırmalar gerektirmiştir. Redef ve arkadaşları özeti, “bir veya daha fazla metinden üretilen, orijinal metinde / metinlerde önemli bilgiler taşıyan ve orijinal metnin (metinlerin) yarısından daha uzun olmayan ve genellikle önemli bir metin" olarak tanımlamıştır.

Özetleme, tek veya aynı konuyu içeren bir veya birden fazla dokümandan çıkarılan ve en çok bilgiyi içeren metin parçalarını bulma ve sıralama işlemidir. Bu işlemin bilgisayar tarafından otomatik olarak yapılması "Otomatik Metin Özetleme (OMÖ)" olarak adlandırılır. Bilgisayarlar, insan bilgisi ve dil yeterliliğinden yoksun olduğundan dolayı otomatik metin özetlemesini çok daha zor ve önemli bir iş haline getirir. OMÖ işleminde özetlenecek doküman sayısına göre “tekli” veya “çoklu” doküman özetleme olarak ayırabiliriz. Tekli doküman özetlemede bir tane doküman mevcutken, çoklu doküman özetlemede birbirleri ile aynı konuyu içeren birden fazla dokümandan yararlanılmaktadır. Özetleme işlemi "yorum"a ya da "çıkarım"a dayalı olarak yapılabilir. Yoruma dayalı özetlemede, özetlenecek doküman veya dokümanlardaki ifadeler kısaltılarak yeniden yazılır. Çıkarım ile yapılan özetlemede özetlenecek doküman veya dokümanlardaki, cümleler veya paragraflar seçilerek yapılır.

Tez beş bölümden oluşmaktadır. İkinci bölümde özetleme ile ilgili yapılmış olan çalışmalardan bahsedilmiştir. Üçüncü bölümde metin özetleme sistemi incelenmiş ve K-Means kümeleme algoritmasında kullanılmak üzere başlangıç paragraf vektörlerinin belirlenmesi ile ilgili iki farklı yöntem tanıtılmıştır. Son bölümde ise sonuçlar değerlendirilmiştir.

1.1. Problem Tanımı

Günümüzde bilginin boyutu çok büyük hızla artmaktadır. Büyük boyutlu veriler, farklı alanlardaki büyük ölçekli veri tabanları içerisinde işlenebilir olarak

bulunan verileri içeren veri madenleri olarak düşünülebilir. Hızla yükselen bu bilgi boyutu ile birlikte aranan bir bilgiye ulaşmak da zor bir hal almıştır. Bilgi boyutunun artışı sonucu olarak çıkan sorunları incelemek için *bilgi erişimi* olarak ifade edilen araştırma konusu ortaya çıkmıştır. BE, yüksek boyutlu bilgi depolarından istenilen bilgiye ulaşmak için çalışmaların geliştirildiği araştırma konusudur (Baeza-Yates ve Ribeiro-Neto, 1999). BE'nin amacı kullanıcının istediği, aradığı bilgiyi en kısa sürede sunmaktır.

Metin özetlemenin BE sistemlerinde kullanılabileceği alanlara örnek olarak, farklı bir dile çevrilecek olan metinde tüm metini çevirmek yerine çıkarılabilecek olan özetin çevrilmesi gösterilebilir (Mani ve Maybury, 1999).

Doğal Dil İşleme (DDİ); ana işlevi bir doğal dili çözümleme, anlama, yorumlama ve üretme olan bilgisayar sistemlerinin tasarımını konu alan bir mühendislik alanıdır (Rich, 1991). BE'nin en çok ihtiyaç duyulduğu alanlardan biri de yazılı belgelerdir. Yazılı olarak saklanan belgeler üzerinde yapılan BE çalışmalarında Doğal Dil İşleme yöntemleri önemli yer tutmaktadır. BE sistemlerinde çalışılan alanlardan biri de metin özetlemedir. Metin özetleme, girdi olarak girilen bir belgenin çıktı olarak belgenin önemli yerlerini içeren, girdi olarak girilen belgeye göre çok daha kısa olan ve girdi olarak girilen belgeyi temsil edebilecek şekilde alınabilmesidir. Metin özetleme, uzun dokümanların anlam bütünlüğünü bozmadan daha kısa boyutlara indirgenerek daha kısa sürede kullanıcının aradığı bilginin bulunup bulunmadığını kontrol etmesine olanak sağlar. Metin özetleme yöntemleri, sürekli boyutu yükselen verilerin bulunduğu bir ortamda hem mevcut bilgilerin daha iyi anlaşılmasına hem de ilgili bilgilerin daha hızlı şekilde tespit edilmesine yardımcı olmak için gereklidir. Örnek olarak tez çalışması yapılırken literatür taraması aşamasında bulunan çalışmaların özet bölümleri incelenerek o çalışmanın tamamını okumadan aradığınız bilgiyi içerip içermediği tespiti yapılabilir. Özetleme, belgelerin okunma sürelerinin kısalması, araştırma yaparken seçim süreçlerinin kolaylaşmasına olanak sağlar.

1.2. Çalışma Konusu ve Amacı

Bu çalışmada çoklu dokümanlarda paragraf tabanlı çıkarıma dayalı özetleme sistemi üzerinde çalışılmıştır. Bu çalışmamızdaki amaç, sisteme girilen metinlerin yine kullanıcı tarafından yönetilebilir olan özetleme oranına göre metinlerin paragraf tabanlı çıkarımlarının sağlanarak kullanıcıya özet sunmaktır. Bu makalede ayrıca K-Means algoritması yardımıyla oluşturulan paragraf kümelerinin içerisinden paragraf seçimi yapmayı sağlayan iki farklı yöntem incelenmiş ve başarı oranları da karşılaştırılmıştır.

2. LİTERATÜR ÖZETİ

İnternet kullanımının artması ile birlikte erişilebilir bilgi kaynakları da çoğalmıştır. Günümüzde bir konu hakkında araştırma yaparken konu ile ilgili birçok farklı kaynağa erişmek mümkün olmuştur. Bu kaynaklar arasında işe yarar bilgiler olduğu gibi gereksiz bilgiler de bulunmaktadır. Bu kaynakların uzunlukları fazla olanlarını incelemek büyük bir zaman ve güç gerektirmektedir. Bu zaman ve güç israfının önüne geçmek için metinlerin tamamını okumak yerine metinin daha kısa ve anlamlı kısmını okuması yardımcı olacaktır. Bu yüzden metin özetleme günümüz yaşantısında büyük önem taşımaktadır.

OMÖ sistemlerinde yaygın olarak iki yaklaşım kullanılır. Özetin oluşturulma şekli olan yorumsal ve çıkarımsal özetleme olmak üzere ikiye ayrılır (Khan ve Salim, 2014).

Yorumsal özetleme, özetlenecek olan metinin teması belirlenir. Belirlenmiş olan tema ile ilgili metin içerisinde geçen terimler tespit edilir. Tespit edilmiş bu terimlere göre metinin temasını en iyi temsil edecek olan cümle ve kelimeler tespit edilmiş olan terimler çerçevesinde yeniden düzenlenerek özet metin oluşturulur (Dalal ve Malik, 2013).

Çıkarımsal özetleme, metinin temasını temsil edebilecek olan, metin içinde bulunan kelime, cümle ve paragrafların önemli olanlarının seçilmesiyle oluşturulmuş özetlemedir (Alguliev vd., 2012). Bu özetlemede seçilmiş olan bölümler önemlerine göre sıralanır. İstenilen özet oranına göre seçilip sıralanmış olan bölümlerden özet oluşturulur.

Otomatik metin özetleme konusunda ilk çalışma H.P. Luhn (Luhn, 1958) adlı bilim adamı tarafından 1959 yılında yapılmıştır. Bu çalışmada doküman özetleme işlemi 2 aşamada gerçekleştirilmiştir. Bu aşamalardan ilkinde özetlenecek olan doküman ön işlemeden geçirilir. Ön işlemede doküman içinde konu ile ilgili bir anlam ifade etmeyen etkisiz kelimeler (zamir, bağlaç vb.) temizlenir. Kalan kelimeleri ortak bir payda da buluşturmak için 6 farklı harften az olanlar ve aynı ön eke sahip kelimeler aynı sözcük olarak kabul edilir. İkinci

aşamada ön işlemeden geçirilip ortak bir yapıda kümelenmiş olan kelimelerden sayıca az olanlar temizlenir. Bu şekilde kelimelerden fazla sayıda geçen yani yüksek frekanslı kelimeler belirlenir. Bu kelimeler anahtar kelime olarak kullanılır. Burada belirlenmiş anahtar kelimeler kullanılarak cümleler içerdikleri anahtar kelimeler ile puanlandırılır. Puanlandırılmış olan cümleler sıralandırılarak özet için en yüksek puana sahip cümleler seçilir.1969 yılında H.P. Edmundson (Edmundson, 1969), 1995 yılında Brandow, Mitze ve Rau (Brandow vd., 1995) frekans tabanlı çalışmalara devam etmişlerdir. Edmundson yaptığı çalışmada, kelime sıklığına ek olarak, "ipucu sözcük öbekleri", "başlık terimleri" ve "cümle konumu" gibi üç yeni özellik daha kullanmıştır. Brandow, Mitze ve Rau, anahtar kelime seçimini cümle ağırlıklarının belirlenmesinde kullanmıştır. Hovy ve Lin (Hovy ve Lin, 1998) çalışmalarında, özetleme işini 3 aşamalı bir süreçle gerçekleştirirler. Bu aşamalar, konu belirleme kavram yorumlama ve son olarakta özetin oluşturulmasıdır. Pollock ve Zamora yaptıkları çalışma, kimya alt alanlarına ait ipucu kelime öbekleri yöntemi kullanımına dayanan, Kimya Özetçe Hizmeti'nde yapılmış bir özetleme programıdır (Fukumoto ve Suzuki, 2000). Jade Goldstein, Vibhu Mittal, Jaime Carbonell ve Mark Kantrowitzt (Goldstein vd., 2000) yaptıkları çalışma da tekli doküman üzerinde uygulanan cümle çıkarım yöntemini çoklu doküman özetlemede uygulamışlardır. Yapmış oldukları çalışmayı 1988-1992 yılları arasında Associated Press ve Wall Street Journal'da yayınlanmış olan ortalama 31 cümle içeren 200 haber metin üzerinde uygulamışlardır. Jaruskulchai (Jaruskulchai ve Kruengkrai, 2003) önerdiği yöntemde, Thai dilinde Luhn'un TF yöntemini uyarlayarak önemli kelime grubunu bulmuş ve paragraflar arası ilişki ağını oluşturmuştur. Yaptığı bu çalışmayı 3 veri kümesi üzerinde %20 ve %30 olmak üzere iki farklı özetleme oranı ile test etmiştir. %20 özet oranı için sırasıyla %50,%43 ve %40, %30 özet oranı için %55,%45 ve %48 başarı oranı elde etmiştir. Ebru Uzundere, Elda Dedja, Banu Diri ve M.Fatih Amasyalı (Uzundere vd., 2008) Türkçe haber metinleri için bir otomatik özetleme sistemi üzerinde çalışmışlardır. Çalışmalarında haber metinlerinin cümlelerini çeşitli özelliklere göre puanlayarak en yüksek puanlı cümlelerin seçimi ile metinlerin özetini çıkarmışlardır. Geliştirdikleri otomatik haber özetleme sisteminin performansı, kullanıcıların çıkardığı cümleler ile aynı olma olasılığı taban

alınarak ölçülmüş ve oran yaklaşık olarak %55 bulunmuştur. Meng Wang (Wang vd., 2009) ve arkadaşları çoklu dokümanlar da kelime ağırlıklarını kullanarak dokümanları kendi içinde alt konulara ayırdıktan sonra bu alt konuları temsil eden cümlelerin seçimi ile özet oluşturmuştur. Lloret ve Palomar (Lloret ve Palomar, 2010) özetleme de cümle seçimi konusu üzerinde, kod kalite prensibini (Code Quality Principle) kullanarak, daha fazla bilgi içeren cümlelerin seçimi ile çıkarımı yapmışlardır. Yaptıkları yaklaşım ile bu yöntemin belgenin önemli cümlelerini seçmek için uygun olabileceğini ve bir özetleme sistemi oluşturulurken bu özelliğin göz önünde bulundurulmasının iyi bir fikir olabileceğini belirtmişlerdir. Min ve arkadaşları (Min vd., 2001) çalışmalarında, Çince dilinde paragraf gruplarını oluşturmak için başlıkları, başlıkların olmadığı durumlarda ise benzerlik amacıyla önemli olarak belirledikleri kelimelerden oluşan vektörleri kullanmışlardır. Fumiyo Fukumoto ve Yoshimi Suzuki (Fukumoto ve Suzuki, 2000) çoklu dokümanlar da özetleme işini anahtar paragraf çıkarımı yöntemi ile yapmışlardır. %20 özet oranı ve 2 adet doküman ile %77,7'lik başarı oranını yakalamışlardır.

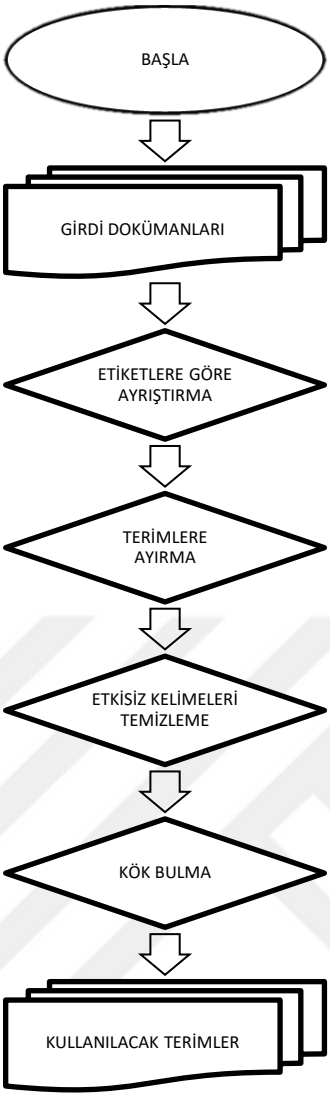
Günümüz uygulamarında, terim sıklığı, ipucu sözcük öbekleri, başlık yöntemi ve cümle konumu yöntemleri hala önemli yöntemler olarak kullanılmaktadır. Bu yöntemlerin uygulandığı sistemlerin başarı oranları kişilerin yaptığı özetler ile karşılaştırılarak performans değerleri ölçülür.

3. METİN ÖZET ÇIKARIMI

Çalışmanın bu bölümünde gerçeklenen metin özetleme sisteminin tasarım detayları anlatılmıştır. Sistem temel anlamda üç aşamadan oluşur. Birinci aşamada dokümanlar ön işlemeden geçirilir ve paragraf vektörleri oluşturulur. İkinci aşamada ilgili paragraflar organize edilerek alt-konu ilişkisi bulunan paragraf öbekleri oluşturulur. Üçüncü aşamada alt-konu öbeklerinden paragrafların seçimi ve özetleme işlemi gerçekleştirilir.

3.1. Doküman Ön İşleme

Sisteme girdi olarak verilen metinler üzerinde çalışmak ve analiz yapabilmek için bu metinlerin üzerinde işlemler yapılması gerekmektedir. Bu işlemler uygulanırken doğal dil işlemede önemli bir rolü olan ön işleme tekniklerinden yararlanılır. Çalışmamızda ön işleme tekniklerini veriyi temizleme ve uygun formata getirme işlemlerini gerçekleştirmek için kullanacağız. Uygulanan teknikler Şekil 3.1'de gösterilmiştir. Sisteme girilen dokümanlar ilk olarak işlem yapılacağı birimlere ayrılmalıdır. Bu işlem için metinlerin girdi olarak verilmeden önce paragraflarının HTML etiketleri ile etiketlenmesi gerekmektedir. Etiketlenmiş halde sisteme sokulan metinler çalışmamızda kullandığımız **"Beautiful Soup"** HTML ayrıştırıcı kütüphanesi ile paragraflarına ayrıştırılır. Elde edilen paragraflar, en küçük birimi olan ve terim olarak ifade edeceğimiz kelimelere ayrıştırlır. Etkisiz kelimeler (stop words) bir dilde çok sık kullanılan ve anlamı olmayan kelimelerdir. Bu kelimeler cümlelerden temizlendikten sonra anlam olarak bir kayba sebep olmazlar; ama varlıkları sistemimizde sonuçları negatif yönde etkileyip daha isabetsiz sonuçların dönmesine sebep olabilirler. Bu yüzden İngilizce diline ait etkisiz kelimeler paragraflar içinden temizlenir. Etkisiz elemanlar paragraflardan temizlendikten sonra kalan kelimelerin kökleri **"Porter Stemming"** algoritması yardımıyla bulunarak, ek almış olan kelimeler için ortak parçalar bulunmuş olur. Bu işlem terim sıklık hesabının doğru olabilmesi için önemlidir.



Şekil 3.1. Doküman ön işleme aşamaları

3.2. Terimlerin Ağırlıklandırılması

Metinlerde özetleme işlemi yapılmadan önce incelenerek o belgeyi temsil eden anahtar kelimeler belirlenmelidir. Bu işlem sayesinde özetleme işlemi için tüm metin girdi olarak verilmez, yalnızca o belgeye ait anahtar kelimeler gönderilir. Böylece belge içerisinde önemsiz olan, bir ayırıcılığı olmayan kelimeler elenecek böylece özetleme işleminin başarımı artacaktır.

3.2.1 Terim sıklığı

Bir kelimenin yer aldığı doküman içerisinde kaç kez tekrarlandığı bilgisi terim sıklığı olarak isimlendirilmektedir. Bir belge içinde diğer kelimelere oranla daha çok kullanılan bir kelime muhtemelen o belgenin konusu hakkında ipucu verebilecek bir kelimedir. Örneğin bir metin içerisinde, ders, okul, öğrenci kelimeleri çok sık tekrarlanıyorsa bu belgenin konusu eğitim olarak tahmin edilebilir.

Ön işleme aşamasında metinden çıkarılan, ayırt edilicilik özelliği olmayan sözcüklerin tespitinde, sözcüklerin İngilizce dil yapısına göre cümlede kullanılma olasılıkları göz önünde bulundurulmuştur. Ayırt edici özellik olarak sözcük sınıfı sadece isim olan kelimeler kullanılmıştır.

Kelimeler çekim eki aldıkları zaman, ifade ettikleri anlam aynı kalmakta fakat yazılışları değişmektedir. Örneğin **"educ"** kelime kökü metin içerisinde education, educations, educated, educational gibi çeşitli çekim ekleri almış halleri ile bulunabilir. Tüm bu değişik yazılışlı kelimeler aslında "**educ**"'dan bahsetmektedir. Kelimelerin terim sıklıkları hesaplanırken bu kelimeler ayrı birer kelime olarak sayılmamaktadır. Tüm bu kelimelerin toplam geçiş sayısı "**educ"** kelimesinin terim sıklığı olarak belirlenmiştir.

3.3. Paragrafların Vektörler İle Temsil Edilmesi

Dokümanların metin özetleme algoritmaları ile analizinin otomatik olarak yapılabilmesi için doküman içeriğinin bilgisayar üzerinde işlenebilecek bir yapıya aktarılması gerekmektedir. Dokümanlar kendilerini oluşturan harfler, kelimeler, cümleler ya da paragraflar halinde ifade edilebilirler. Dilde yer alan kavramlar, varlıklar, eylemler, durumlar vb. unsurlar kelimelerle ifade edilir. Bu nedenle bir belgeyi ifade edebilecek en küçük yapı taşı o belgeyi oluşturan kelimelerdir.

Doğal dil işlemede dokümanları bilgisayar ortamında kelime dizileri ile tasarlarız. Matematiksel olarak bunu "her doküman kelime vektörlerinden

oluşur" şeklinde ifade edebiliriz. Bu vektörler iki boyutlu bir dizi olarak modellenir. Bu vektörlerin içeriğini doküman geçen terimler belirler (Berry vd., 1999). Yaptığımız çalışmada vektörler paragraf bazında oluşturulmuştur. Yani bir dokümanı temsil eden vektör dizisi "anahtar kelime sayısı x paragraf sayısı" olarak modellenmiştir. Kelimeler hesaplanmış olan geçiş sıklıklarına göre sıralanır. Paragraf-anahtar kelime vektörleri için bu sıralanan anahtar kelimelerden belli sayıda seçim yapılır. Bu değer birinci yöntem için ilk 10, ikinci yöntemde ise seçilen özet oranına bağlı olarak oluşacak başlangıç kümesi sayısına eşdeğer ilk kelimeler olarak belirlenmiştir. Daha sonra seçilen bu anahtar kelimeler ile dokümanlardaki her bir paragrafa ait paragraf-anahtar kelime vektörleri Çizelge 3.1'deki gibi oluşturulur.

Çizelge 3.1. Örnek paragraf-anahtar kelime vektörleri

Paragraf	**Anahtar Kelimeler**									
	network	secur	İnform	busin	comput	system	atack	data	hacker	protect
p_1	1	0	1	0	0	0	2	0	1	0
p_2	0	0	0	0	0	1	1	0	1	0
p_3	0	0	1	0	1	0	0	0	1	0

3.4. Paragrafların Organize Edilmesi İçin Kullanılan Yöntemler

Bu adımda K-Means (Faber, 1994) kümeleme algoritması kullanılarak, anahtar kelime vektörleri haline getirdiğimiz paragrafların kümelendirilmesi yapılır. K-Means algoritmasında kullanacağımız küme sayısı kullanıcı tarafından girilen özetleme oranı parametresi ile bağlantılıdır (Formül 1).

$$k = \left\lceil \frac{r}{100} * p \right\rceil \quad (1)$$

- k: Küme sayısı
- r: Özetleme oranı
- p: Dokümanlardaki toplam paragraf sayısı

Çalışmamızda bu formülden yola çıkarak % 20 özet oranı istendiği durum için küme sayımız 20 olmuştur. Küme sayısının belirlenmesinde kullanılan bu

yaklaşımın arkasında yatan ana fikir, özetin tüm doküman kümesini kapsaması için farklı alt konuların seçilebilmesidir. Kümelerin başlangıç paragraf vektörleri (küme merkezleri) aşağıda açıklandığı gibi iki farklı yöntemle belirlenerek, sonuçlar karşılaştırılmıştır.

3.4.1. Yöntem 1

Paragraflar içerdiği anahtar kelimelerin toplamına göre sıralanır. Başlangıç kümesi sayısı kadar paragraf seçilir. Çizelge 3.1'deki değerlerden yola çıkarak örnek vermek gerekirse:

$$p_1 = 1 + 0 + 1 + 0 + 0 + 0 + 2 + 0 + 1 + 0 = 5$$
$$p_2 = 0 + 0 + 0 + 0 + 0 + 1 + 1 + 0 + 1 + 0 = 3$$
$$p_3 = 0 + 0 + 1 + 0 + 1 + 0 + 0 + 0 + 1 + 0 = 3$$

Her bir paragraf vektörü için toplam içerdiği anahtar kelime sayısı bulunur. Tüm vektörler için bu değer bulunduktan sonra bu değere göre sıralanır. Bir önceki adımda hesaplanan başlangıç kümesi sayısı kadar vektör, paragraf vektörleri başlangıç noktaları olarak seçilir.

3.4.2. Yöntem 2

Anahtar kelimeler tüm dokümanlarda geçme sayılarına göre sıralanır. Çizelge 3.2'de çalışmada kullanılan verilerden bir kesit alınmıştır. Bu kesit için kelimeleri geçiş sıklıklarına göre sıraladığımız da durum "network, inform, secur, hacker" şeklinde olacaktır. Bu sıralamaya göre kelimelerin en çok geçtiği paragraflar belirlenir. Yani "network" için p_{10}, "inform" için p_6, "secur" için p_{10}, "hacker" için p_3 seçilir. Örnekte görüldüğü gibi "network" ve "inform"ı temsil eden paragraflar aynı çıktı. Böyle bir durumda yeni paragrafı seçmek için ikinci en çok geçtiği paragrafa bakılır. Bu durumda "inform" için p_2 seçilir. Son durumda başlangıç noktalarımız p_{10}, p_6, p_3 ve p_2 olmuş olur.

Çizelge 3.2. Anahtar kelimelerin paragraflarda geçiş sayıları

Paragraf	Anahtar Kelime	Geçiş Sayısı
p_1	Network	2
p_1	Secur	1
p_2	Secur	1
p_2	İnform	1
p_3	Hacker	1
p_5	Network	2
p_6	İnform	3
p_{10}	Network	5
p_{10}	Secur	3

3.5. K Means Kümeleme Algoritması

K-means algoritması bir kümeleme algoritmasıdır. Kümeleme algoritmaları otomatik olarak verileri daha küçük kümelere ya da alt kümelere ayırmaya yarayan algoritmalardır. Algoritma istatistiksel olarak benzer nitelikteki kayıtları aynı gruba sokar. Bir elemanın yalnızca bir kümeye ait olmasına izin verilir (Evans vd., 2005). Küme merkezi kümeyi temsil eden değerdir.

K-means algoritmasının genel mantığı n adet veri nesnesinden oluşan bir veri kümesini, giriş parametresi olarak verilen k adet kümeye bölümlemektir. Amaç, gerçekleştirilen bölümleme işlemi sonunda elde edilen kümelerin, küme içi benzerliklerinin maksimum ve kümeler arası benzerliklerinin minimum olmasını sağlamaktır.

K-means yöntemini kullanmak için öncelikle bir uzaklık ölçüm yöntemi tanımlanmış olmalıdır. Öklid uzaklığı, ya da kosinüs benzerliği ölçüleri paragraf vektörleri ile komşuları arasındaki yakınlığı ölçmek için kullanılabilir.

3.6. K Means Kümeleme Algoritmasının Uygulanması

Tez kapsamında yapılan çalışmada paragrafların organize edilebilmesi için K-Means algoritması kullanılmıştır. K-Means yönteminde kullanabilmek için paragrafları temsil eden vektörleri daha önceki bölümde belirlemiştik. Belirlenen bu vektörlerin arasındaki benzerlikleri bulmak için bir uzaklık ölçüsüne ihtiyacımız vardı. Tez çalışmamızda yapılan modelde vektörler arası

uzaklığı hesaplamak ve dolayısıyla vektörler arasındaki benzerliği bulmak için öklid uzaklık formülünden yararlanılmıştır. Vektörler arası öklid uzaklık formülüne göre hesaplanan sonuçlara göre iki vektör arasındaki uzaklık ne kadar küçük ise bu iki vektör birbirine o kadar benzerdir sonucuna varılır.

3.6.1. Vektörler arası uzaklıkların belirlenmesi

Paragraf vektörlerinin birbirlerine uzaklıklarının hesaplanması için Öklid uzaklık formülü kullanılmıştır (Formül 2):

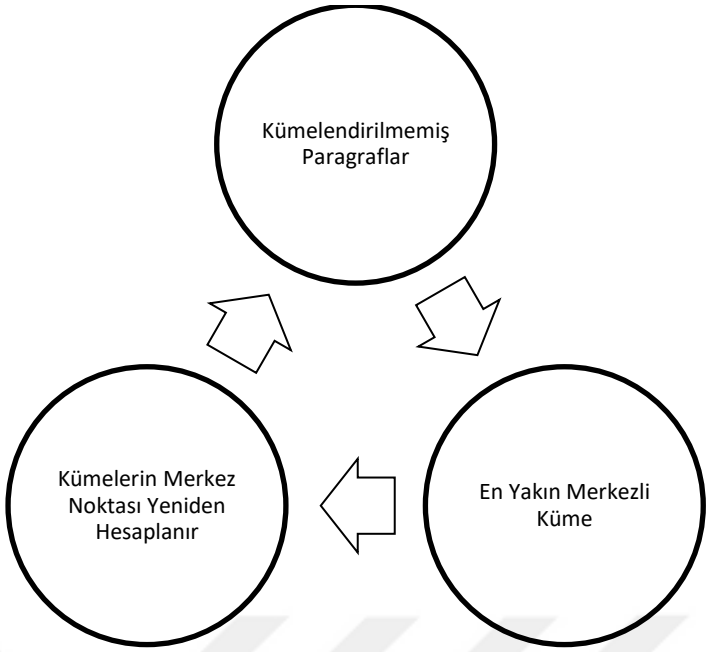
$$d(p,q) = \sqrt{\sum_{i=}^{n}(q_i - p_i)^2} \tag{2}$$

- p, q: Paragraf vektörleri
- p_i: p vektörüne ait kelime frekansları
- q_i: q vektörüne ait kelime frekansları

Örnek olarak Çizelge 3.1'deki p_1 ve p_2 vektörlerini ele alalım. Bu vektörlerden p_1'in k_1 kümesinin merkez noktası olduğunu kabul edelim. Bulmak istediğimiz şey p_2 vektörünün k_1 kümesine olan uzaklığıdır. Bunun için yukarıda belirttiğimiz Öklid formülünü aşağıdaki gibi uygulayarak p_2 vektörünün k_1 kümesine olan uzaklığını bulabiliriz (Formül 3):

$$d(p_1,p_2) = \sqrt{\begin{matrix}(0-1)^2+(0-0)^2+(0-1)^2+(0-0)^2+(1-0)^2+(1-2)^2+\\(0-0)^2+(1-1)^2+(0-0)^2\end{matrix}} \tag{3}$$

Sırasıyla tüm paragrafların birbirleriyle olan uzaklıkları hesaplanır. Çalışmanın **3.4.** bölümünde anlattığımız iki yönteme göre başlangıç vektörleri belirlenir. Şekil 3.2'deki döngüdeki gibi bir döngü ile tüm paragraflar kümelendirilir.



Şekil 3.2. K-Means kümeleme döngüsü

Bu döngüye göre başlangıç adımı olarak seçilen yönteme göre belirlenmiş başlangıç vektörleri birer küme olarak kabul edilir ve başlangıç adımı için aynı zamanda kümelerin merkezini de temsil eder. Başlangıç kümelerinden ilki için paragraflardan Öklid uzaklığı en kısa olan paragraf bu küme içine yerleştirilir. Paragraf kümeye yerleştirildikten sonra artık o kümenin de merkez noktası da değişmiştir. Değişen bu kümenin merkezi içerdiği paragraf vektörlerinin kelime frekanslarının toplamı küme içerisindeki paragraf sayısına bölünerek yeniden güncellenir. Kümelere yerleştirilmemiş olan paragraf vektörlerinin küme merkezlerine olan Öklid uzaklıkları tekrar hesaplanır. Bu işlem tüm vektörler bir kümeye yerleştirilene kadar tekrarlanır.

3.7. K Means Kümeleme Algoritması İle Özetin Oluşturulması

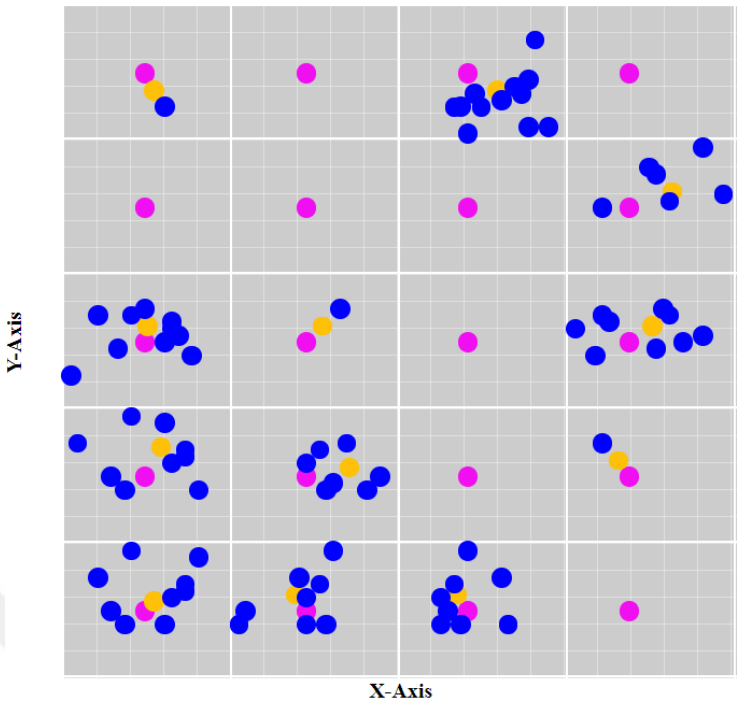
Özetlenecek olan metinler ön işlemlerden geçirildikten sonra, paragraf tabanında terim vektörleri bulunur ve bu vektörler birbirleri ile kıyaslanarak en yakın komşuları elde edilir. Bu aşamada en yakın komşuların belirlenmesi ile oluşturulmuş olan kümelerden o kümeyi en çok temsil ettiği düşünülen paragrafın seçimi yapılır.

Tüm paragraflarımız kümelerine yerleştirildikten sonra kümelerimiz paragraf seçimi için uygun hale gelmiştir. Başlangıç küme sayısını, istenilen özet oranına göre sonuçta olarak çıkması gereken paragraf sayısı kadar belirlediğimiz her kümeden, o kümenin merkezine Jaccard uzaklığına göre en yakın olan paragraf vektörü seçilerek istenilen özet oranı oluşturulmuş olur.

Aşağıda örnek olarak gösterilen Şekil 3.3'deki dağılım birinci yöntem %20 özet oranı için oluşturul temsili bir şekildir. Burada pembe noktalar başlangıçta belirlenen merkez noktalarını temsil eden paragraflardır. Görüldüğü gibi 102 paragraflık bir veri kümesinden %20 oranında özet istenildiğinde 20 adet başlangıç noktası belirlenmiştir. Mavi noktalar geriye kalan diğer paragrafları temsil eder. Sarı noktalar ise kümelenmenin sonunda belirlenmiş olan merkez noktadır. Paragrafların Jaccard uzaklığını bu merkez nokta ile karşılaştırarak paragraf seçimi yapılır. Paragrafın merkez noktasına olan uzaklığını bulmak için aşağıdaki formül kullanılır (Formül 4):

$$J_g(p,q) = \frac{\sum_i \min(p_i,q_i)}{\sum_i \max(p_i,q_i)} \quad (4)$$

- p: Hesaplanmış olan merkez noktasının vektörü
- q: Merkeze olan uzaklığını hesaplamak istediğimiz paragraf vektörü

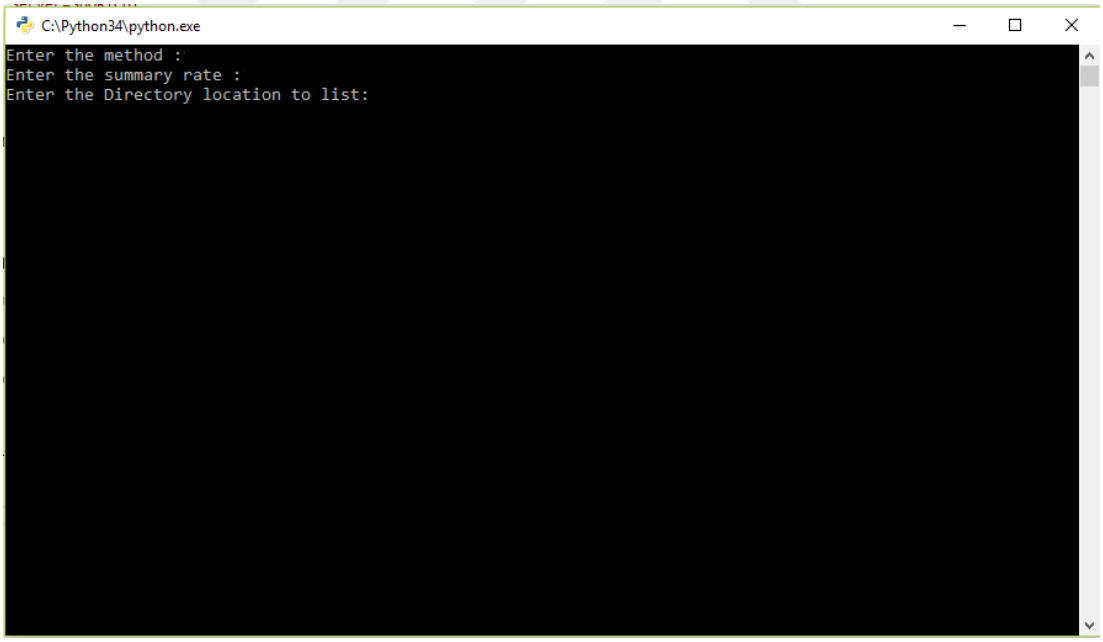


Şekil 3.3. Kümelendirilmiş paragraf vektörleri

Her kümenin içinde bulunan vektörlerin, Jaccard uzaklık formülü ile kümelerin merkezlere olan uzaklıkları hesaplanır. Her küme için merkeze en yakın olan vektör ilgili kümeyi en çok temsil eden vektör olarak kabul edilir.

4. YAZILIMIN AÇIKLANMASI

Tez kapsamında özetleme yapmak için geliştirilen yazılım bu bölümde tanıtılacaktır. Hazırlanmış olan yazılım Python 3.4 versiyonu ile geliştirilmiştir. Veritabanı işlemleri için MsSql Server kullanılmıştır. Python programlama dili için oluşturulmuş olan Doğal Dil İşleme konusunda bir çok kütüphane içeren "nltk" aracı uygulamaya dahil edilmiştir. Uygulamanın resimdeki gibi basit bir arayüzü vardır. Bu aşamada "Enter the method" da K-Means kümeleme algoritmasında başlangıç paragraflarını belirlemek için kullanılacak yöntem seçilir (1 değeri girilirse "1.Yöntem", 2 değeri girilirse "2.Yöntem"). "Enter the summary rate" ile istenilen özet oranı değeri girilir (1-100). "Enter the Directory location to list" ile özetlerinin çıkarılması istenilen dosyaların bilgisayardaki konumu yazılır (Örn: "C:\Users\[User]\[Özetlenecek Dosyaların Bulunduğu Klasör]").



Şekil 4.1. Program arayüzü

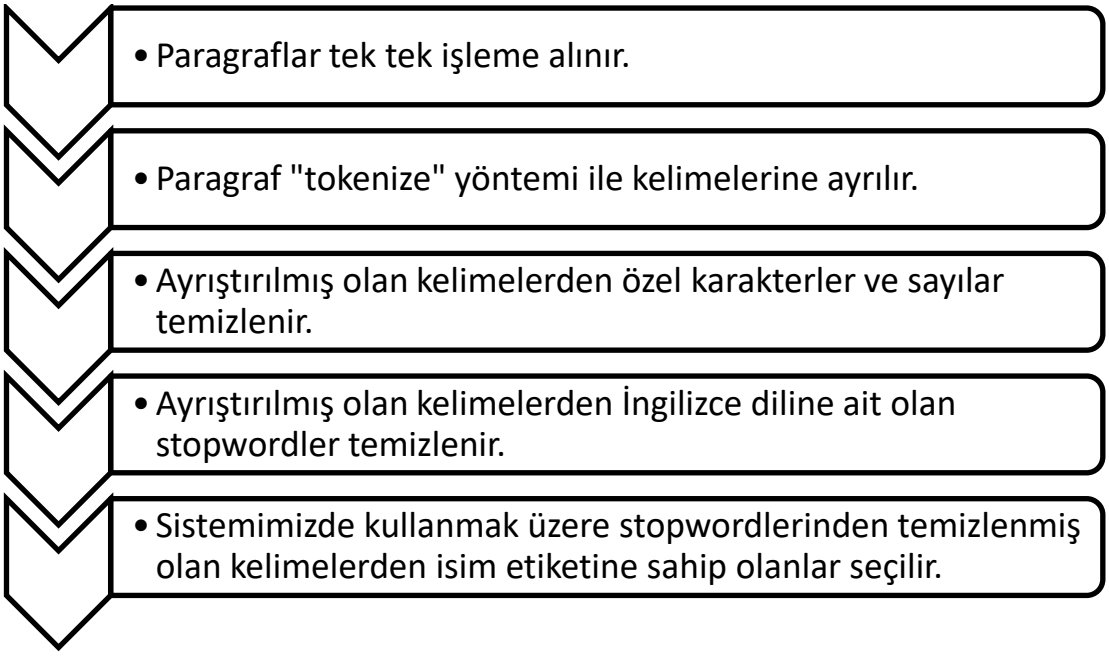
Yazılımda gerçekleştirilmiş adımlar aşağıdaki başlıklarda açıklanmıştır.

4.1 Paragrafların Tespiti

Sisteme girdi olarak verilen dokümanların paragraflarına ayrılması yazılımımızın ilk aşamasıdır. Sistemin işleyeceği yani özetinin çıkarılacağı belgelerin yazılımımızda işlenebilmesi için paragraflarına ayrılması gerekmektedir. Bu işlemde kullanılmak üzere HTML ve XML dosyaları gibi etiketlenmiş metinleri ayrıştırmak için oluşturulmuş olan BeautifulSoap kütüphanesi sistemimize eklenmiştir. Bu kütüphaneyi kullanabilmek için girdi olarak verilen belgeler de paragraf başı için " " ve paragraf sonu için " " etiketleri eklenmiş olmalıdır. Bu şekilde etiketlenmiş olan belgeler bu kütüphane yardımıyla kolayca paragraflarına ayrılır. Böylelikle her paragraf ayrı ayrı işleme alınabilecek hale gelmiştir.

4.2 Paragraflarda Bulunan Kelimelerin Tespiti ve İşlenmeye Uygun Hale Getirilmesi

Önceki bölümde belgelerin paragrafları ayrı ayrı tespit edilmiştir. Yazılımın bu bölümünde ayrıştırılmış olan paragrafların içerdiği kelimelerin tespiti yapılır ve ön işlemeden geçirilir. (Şekil 4.2).

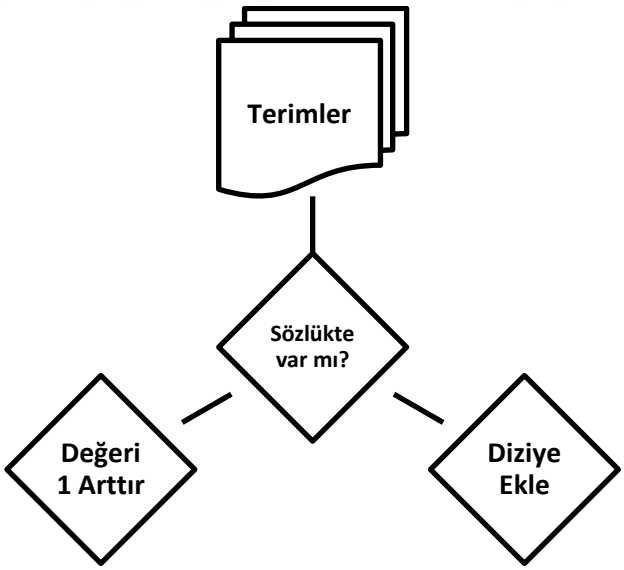


Şekil 4.2. Terimlerin tespit adımları

Bu adımda kullanılmak üzere uygulamaya dahil edilmiş olan “nltk” aracına ait olan “nltk.tokenize” kütüphanesi kullanılarak paragrafların içerdiği kelimeler ayrıştırılmış ve ayrı ayrı elde edilmiştir. Ayrıştırılmış olan kelimelerin içinden “!, #, $, %” gibi özel karakterler temizlenir. Özel karakterlerden temizlenmiş olan kelimeler yine nltk aracının bir kütüphanesi olan “nltk.corpus” yardımıyla İngilizce diline ait olan stopwordlerden ayrıştırılır. Stopwordlerden ayrıştırılmış olan kelimeler “Part of speech tagging” yöntemi ile isim, fiil, sıfat gibi sınıflardan hangisine dahil oluyor ise onun etiketi ile etiketlenir. Bu adımda yapılan işlem için yine nltk aracı kullanılmıştır. Etiketlenme işlemi tamamlandıktan sonra etiketi isim olan kelimeler çalışmanın geri kalanında kullanılacak olan terimleri oluşturmaktadır.

4.3 Paragraflarda Bulunan Terimlerin Geçiş Sayılarının Hesaplanması

Paragraflarda tespit edilmiş terimler basit bir algoritma ile paragraf bazında geçiş sayıları hesaplanmıştır.



Şekil 4.3. Terimlerin geçiş sayılarının hesaplanma süreci

Terimlerin geçiş sayılarını tutmak için anahtar kelimesi terim, değeri terimin geçiş sayısı olan bir **Sözlük** (**Dictionary)** tanımlanır. Şekil 4.3’deki gibi sırasıyla

paragraftaki tüm terimler bir döngü ile kontrol edilir. Eğer kontrol edilen terime ait tanımlanmış olan sözlük de bir anahtar mevcut ise bu anahtara ait değerdeki sayı bir arttırılır, eğer bu terim sözlük de bulunmuyorsa değeri bir olacak şekilde sözlüğe eklenir.

Çizelge 4.1. Terimlerin geçiş sayılarıyla birlikte kayıt edildiği tablo yapısı

Words	
PK	WordId
	DocumentId
	ParagraphId
	StemWord
	Count

Paragraf için tüm terimler kontrol edildikten sonra veri tabanında Çizelge 4.1'deki gibi tablo yapısı ile veritabanına kayıt edilir. Burada "DocumentId" sisteme girdi olarak verilen dokümanların sistem tarafından atanmış olan Id değerini tutar. "ParagraphId" ile dokümanın kaçıncı paragrafı olduğu bilgisi saklanır. "StemWord" alanında terimin değeri, "Count" alanında bu terime ait paragrafta kaç kez geçtiğinin bilgisi tutulur (Çizelge 4.2).

Çizelge 4.2. Words tablosu kayıt örnekleri

WordId	DocumentId	ParagraphId	StemWord	Count
11	1	1	network	2
12	1	1	atack	1
13	1	2	theft	1
14	1	2	privaci	1
15	1	2	victim	1
16	1	2	efect	1
17	1	2	concern	1
18	1	2	secur	1
19	1	2	network	1
20	1	2	ident	1

4.4 Terimlerin Ağırlıklandırılması

Sistemde bulunmuş olan terimlerin anahtar kelime seçimi yapılabilmesi için ağırlıklandırılması gerekmektedir. Bu işlem veritabanında **Saklı Yordam**'lar

(**stored procedure**) içerisinde yapılmaktadır. Bu aşamadan önce girdi olarak verilmiş tüm metinlerin terimleri ve terimlerin geçiş sayıları hesaplanıp veritabanına kayıtları eklenmiştir. Terim hesaplamasının yapılması adımında iki adet bir boyutlu vektör tanımlanmıştır. Bu vektörlerden biri doküman bazında dokümandaki terimlerin tekil olarak kullanılmasıyla oluşturulur. Örnek olarak terimleri Çizelge 4.2'deki gibi olan bir dokümanın vektörü (Çizelge 4.3);

Çizelge 4.3. Terim-Doküman Vektörü

network
atack
theft
privaci
victim
efect
concern
secur
ident

şeklinde oluşmaktadır. Burada dikkat edilmesi gereken nokta her bir terimin bir kez kullanılmasıdır. Bu adımda oluşturulan vektör doküman bazında oluşturulur. Uygulamaya kaç adet doküman girdi olarak verildiyse sistemde o kadar doküman vektöründen olmak zorundadır.

Benzer bir mantıkla fakat bu sefer dokümanların tümü birden göz önüne alınarak bir adet tüm dokümanlardaki terimlerin tekil bir şekilde bulunduğu ana vektör oluşturulur.

Terim ağırlıklandırılması yani terim geçiş sıklığı hesaplaması oluşturulmuş olan ana vektör üzerinden yapılır. Bu vektörde bulunan terimlerin sırasıyla dokümanlardaki geçiş sıklığı hesaplanır. Bu hesaplama için aşağıdaki formül 5 kullanılır;

$$\boldsymbol{t_j = \frac{n}{d}} \tag{5}$$

- $\boldsymbol{t_j}$: j. Terimin geçiş sıklığı değeri
- $\boldsymbol{n}$: Kaç adet doküman vektöründe terime rastlandı
- $\boldsymbol{d}$: Sisteme girilen doküman sayısı

Ana vektörde bulunan tüm terimler için terim ağırlıklandırılması yapılır. Uygulamanın ilk aşamasında seçilen yönteme göre bu terimlerden kaç adedinin anahtar kelime olarak seçileceği bilgisi belirlenir. Terimlerin ağırlık değerleri Çizelge 4.4'deki yapıyla veritabanında saklanır.

Çizelge 4.4. Terimlerin geçiş sıklık değerlerinin tutulduğu tablo yapısı

WordDispersions	
PK	WordDispersionId
	StemWord
	DispersionValue

Bu tabloda "StemWord" terimi, "DispersionValue" alanı ise terime karşılık hesaplanan ağırlığı temsil eder. Çizelge 4.5'de hesaplanan örnek ağırlıklandırma değerleri listelenmiştir.

Çizelge 4.5. WordDispersions tablosu kayıt örnekleri

WordDispersionId	StemWord	DispersionValue
299	Network	1
403	Secur	1
227	İnform	0.89474
49	Busin	0.84211
81	Comput	0.84211
443	System	0.73684
109	Data	0.68421
29	Atack	0.68421
200	Hacker	0.68421
364	Protect	0.57895

Geçiş sıklıklarına göre sıralanan terimlerden yönteme göre belirlenen sayı kadar terim anahtar kelime olarak atanır.

4.5 Paragrafların Kümelendirilmesi ve Paragraf Seçimi

Uygulamanın bu adımında geçiş sıklığı değerleri hesaplanan terimlerden seçilen anahtar kelimeler ile paragrafların vektörleri oluşturulmuştur. Sisteme girilen her dokümanın her bir paragrafı için vektörler belirlenir. Bu vektörler, seçilmiş

olan anahtar kelimelerin ilgili vektörde kaç kez geçtiği değeri ile oluşturulur. Oluşturulan değerler Çizelge 4.6'daki yapıyla saklanır.

Çizelge 4.6. Anahtar kelime-paragraf vektörlerinin tutulduğu tablo yapısı

ParagraphWordVectors	
PK	ParagraphWordVectorId
	DocumentId
	ParagraphId
	StemWord
	Count
	TFValue

Bu tabloda her bir paragraf için seçilen anahtar kelime sayısı kadar kayıt oluşturulmuştur. Çizelge 4.7'de örnek olarak seçilmiş 10 adet anahtar kelimeye göre oluşturulmuş olan 2 adet paragraf vektörü görülmektedir

Çizelge 4.7. ParagraphWordVectors tablosu kayıt örnekleri

ParagraphWordVectorId	DocumentId	ParagraphId	StemWord	Count	TFValue
1	1	1	atack	1	0.071
2	1	1	busin	0	0.000
3	1	1	comput	1	0.071
4	1	1	data	0	0.000
5	1	1	hacker	0	0.000
6	1	1	inform	0	0.000
7	1	1	network	2	0.142
8	1	1	protect	0	0.000
9	1	1	secur	1	0.071
10	1	1	system	0	0.000
11	1	2	atack	0	0.000
12	1	2	busin	0	0.000
13	1	2	comput	0	0.000
14	1	2	data	0	0.000
15	1	2	hacker	0	0.000
16	1	2	inform	1	0.083
17	1	2	network	1	0.083
18	1	2	protect	0	0.000
19	1	2	secur	1	0.083
20	1	2	system	0	0.000

Bu tabloda alanlardan anlaşılacağı üzere DocumentId, ParagraphId, StemWord hangi dokümanın kaçıncı paragrafına ait vektör değerini oluşturmaktadır. "Count" alanında kelimenin bulunduğu paragrafta kaç kez geçtiği değeri tutulmaktadır. "TFValue" alanı paragraf vektörlerindeki terimlerin, hesaplanan paragraf vektöründeki paragrafta kaç kez geçtiği değerinin normalize edilmiş halidir. Normalizasyon, çok uzun paragrafların daha kısa olan paragraflara göre değerlerinin daha anlamlı çıkması için yapılmıştır. Yapılan normalizasyon hesaplaması **formül 6** ile yapılır;

$$\boldsymbol{Value} = \frac{\boldsymbol{t_{ij}}}{\boldsymbol{a_i}} \qquad (6)$$

- $\boldsymbol{t_j}$: $\boldsymbol{i}$. Paragrafta $\boldsymbol{j}$. Anahtar kelimenin geçiş sayısı
- $\boldsymbol{a_i}$: $\boldsymbol{i}$. Paragrafta bulunan toplam terim sayısı

Paragraf vektörleri ve anahtar kelimelerin değerleri hesaplandıktan sonra paragrafların birbirleriyle olan Öklid uzaklıklarının hesaplanma adımına geçilir. Bu adımda sistemde bulunan tüm paragrafların birbirleriyle olan Öklid uzaklıkları hesaplanır. Hesaplama formülü olarak tez çalışmasının **3.6.1.** bölümünde yer alan Öklid uzaklık formülü kullanılır.

Çizelge 4.8. Paragraf vektörleri arasındaki uzaklıkların tutulduğu tablo yapısı

ParagraphsOklidLengths
Document1Id
Paragraph1Id
Document2Id
Paragraph2Id
OklidLength

Hesaplanan Öklid uzunlukları Çizelge 4.8'deki tablo yapısı ile değerleri saklanır. Burada Document1Id ve Paragraph1Id değerleri birlikte bir paragrafı, Document2Id ve Paragraph2Id değerleri birlikte farklı bir paragrafı temsil etmektedir. OklidLength değeri ise bu iki farklı paragrafın birbirine olan Öklid uzaklık değerlerini temsil eder. Çizelge 4.9'da örnek olarak kayıtlar gösterilmiştir. Bu kayılara göre 0 ile temsil edilen dokümanın 1. Paragrafı ile 0

ile temsil edilen dokümanın 2. Paragrafı arasındaki Öklid uzaklığı 0.144337567285 olarak bulunmuştur.

Çizelge 4.9. ParagraphOklidLengths tablosu kayıt örnekleri

Document1Id	Paragraph1Id	Document2Id	Paragraph2Id	OklidLength
0	1	0	2	0.144337567285
0	1	0	3	0.170034010191
0	1	0	4	0.182981263668
0	1	1	1	0.163663417668
0	1	1	2	0.256313575576
0	1	1	3	0.270014557683
0	1	1	4	0.165359456929

Paragrafların birbirleriyle olan Öklid uzunlukları hesaplandıktan sonra paragrafların K-Means algoritması ile kümelendirilme aşamasına geçilebilir hale gelmiştir.

Çalışmanın **3.6.1.** bölümünde anlatılan algoritmanın uygulanması ile birlikte paragraflar uygun olan kümelere yerleştirilme işlemi yapılır. Kümelendirme işlemi sonucu oluşan kayıtlar Çizelge 4.10'daki yapıda saklanır.

Çizelge 4.10. Paragraf kümelerinin tutulduğu tablo yapısı

ClusteredParagraphs
Document1Id
Paragraph1Id
Document2Id
Paragraph2Id

Tabloda bulunan Document2Id ve Paragraph2Id alanları çalışma başında seçilmiş olan başlangıç paragraflarını temsil eder. Document1Id ve Paragraph1Id ile ifade edilen paragraf ise Document2Id ve Paragraph2Id ile belirtilen kümenin içinde bulunan diğer paragrafları temsil eder. Aynı Document2Id ve Paragraph2Id değerlerine sahip olan paragraflar aynı kümeye aittir.

Çizelge 4.11. ClusteredParagraphs tablosu kayıt örnekleri

Document1Id	Paragraph1Id	Document2Id	Paragraph2Id
2	4	8	2
3	4	8	2
17	2	8	2
8	3	8	3
8	4	8	4
9	2	9	2
9	6	9	6
11	2	11	2
11	1	11	2
4	4	11	2
1	4	11	2
17	3	11	2

Yukarıdaki örnek kayıtlar üzerinden açıklamak gerekirse Document2Id-Paragraph2Id şeklinde ifade edersek 8-2, 8-3, 8-4, 9-2, 11-2'ye ait paragraflar başlangıç kümeleri olarak ifade edilir. Document1Id-Paragraph1Id ile ifade edilen 2-4, 3-4, 17-2 paragrafları 8-2 ile ifade edilen paragrafla aynı kümede yer alır. 8-3, 9-2, 9-6 başlangıç kümeleri için herhangi bir atama yapılmadığından dolayı bu kümelerin tek bir elemanı vardır ve bunlarda kendileridir.

Yazılımın en son adımında çalışmanın **3.7.** bölümünde anlatılan Jaccard uzaklık formülü uygulanarak oluşturulmuş olan her bir küme için merkez noktasına en yakın olan paragrafların seçimi yapılır. Her kümeden yapılan seçimlere göre istenilen özet oranına ait paragraf sayısı kadar paragraf öneme göre belirlenmiştir.

Yazılımın en son adımında çalışmanın **3.7.** bölümünde anlatılan Jaccard uzaklık formülü uygulanarak oluşturulmuş olan her bir küme için merkez noktasına en yakın olan paragrafların seçimi yapılır. Her kümeden yapılan seçimlere göre istenilen özet oranına ait paragraf sayısı kadar paragraf öneme göre belirlenmiştir.

5. SONUÇ VE ÖNERİLER

Paragraf tabanlı çıkarım metodunun başarı değerlendirilmesi Internet üzerinden toplanan "Network Güvenliği" konusunu anlatan 19 adet doküman üzerinde yapılmıştır. Bu 19 doküman 4 farklı kişiye verilmiş ve farklı oranlar da özetlerin çıkarılması istenmiştir.

Çizelge 5.1. Veri kümesi

Veri Kümesindeki Toplam Doküman Sayısı	19
Veri Kümesindeki Toplam Paragraf Sayısı	102

Yöntem ve özet oranına göre hata matrisi ve başarı oranları Çizelge 5.2, Çizelge 5.3 ve Çizelge 5.4'de verilmiştir. Çizelgelerde "1.Sınama, 2.Sınama, 3.Sınama, 4.Sınama" olarak sistemin başarı oranını test eden 4 farklı kullanıcının testi ifade edilmektedir.

Çizelge 5.2. %20 özet oranı için hata matrisleri

1.Yöntem %20 Özetleme												
	1.Sınama			**2.Sınama**			**3.Sınama**			**4.Sınama**		
	Yok	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**
Yok	71	11	82	66	16	82	66	16	82	73	9	82
Var	11	9	20	16	4	20	16	4	20	9	11	20
	82	20	102	82	20	102	82	20	102	82	20	102
Hata Oranı	0,215686			0,313725			0,313725			0,176471		
Doğruluk	0,784314			0,686275			0,686275			0,823529		

2.Yöntem %20 Özetleme												
	1.Sınama			**2.Sınama**			**3.Sınama**			**4.Sınama**		
	Yok	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**
Yok	73	9	82	66	16	82	69	13	82	68	14	82
Var	9	11	20	16	4	20	13	7	20	14	6	20
	82	20	102	82	20	102	82	20	102	82	20	102
Hata Oranı	0,176471			0,313725			0,254902			0,27451		
Doğruluk	0,823529			0,686275			0,745098			0,72549		

Çizelge 5.3. %40 özet oranı için hata matrisleri

1.Yöntem %40 Özetleme												
	1.Sınama			2.Sınama			3.Sınama			4.Sınama		
	Yok	Var	Toplam	Yok	Var	Toplam	Yok	Var	Toplam	Yok	Var	Toplam
Yok	44	18	62	37	25	62	38	24	62	47	15	62
Var	18	22	40	25	15	40	24	16	40	15	25	40
	62	40	102	62	40	102	62	40	102	62	40	102
Hata Oranı	0,352941			0,490196			0,470588			0,294118		
Doğruluk	0,647059			0,509804			0,52941 2			0,705882		

2.Yöntem %40 Özetleme												
	1.Sınama			**2.Sınama**			**3.Sınama**			**4.Sınama**		
	Yok	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**
Yok	45	17	62	39	23	62	43	19	62	46	16	62
Var	17	23	40	23	17	40	19	21	40	16	24	40
	62	40	102	62	40	102	62	40	102	62	40	102
Hata Oranı	0,333333			0,45098			0,372549			0,313725		
Doğruluk	0,666667			0,54902			0,627451			0,686275		

Çizelge 5.4. %60 özet oranı için hata matrisleri

1.Yöntem %60 Özetleme												
	1.Sınama			**2.Sınama**			**3.Sınama**			**4.Sınama**		
	Yok	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**
Yok	23	19	42	20	22	42	21	21	42	18	24	42
Var	19	41	60	22	38	60	21	39	60	24	36	60
	42	60	102	42	60	102	42	60	102	42	60	102
Hata Oranı	0,372549			0,431373			0,411765			0,470588		
Doğruluk	0,627451			0,568627			0,588235			0,529412		

2.Yöntem %60 Özetleme												
	1.Sınama			**2.Sınama**			**3.Sınama**			**4.Sınama**		
	Yok	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**	**Yok**	**Var**	**Toplam**
Yok	43	17	60	26	16	42	27	15	42	20	22	42
Var	25	43	68	16	44	60	15	45	60	22	38	60
	68	60	128	42	60	102	42	60	102	42	60	102
Hata Oranı	0,328125			0,313725			0,294118			0,431373		
Doğruluk	0,671875			0,686275			0,705882			0,568627		

Hata matrisinde TN değeri, kullanıcının çıkardığı özet sonucu seçilmemesi gereken paragraflardan kaçının doğru seçildiği sayısıdır. FP değeri, kullanıcının çıkardığı özet sonucu seçilmemesi gereken paragraflardan kaçının seçilmediği sayısıdır. FN değeri, kullanıcının çıkardığı özet sonucu seçilmesi gereken

paragraflardan kaçının seçilmediği sayısıdır. TP değeri, kullanıcının çıkardığı özet sonucu seçilmesi gereken paragraflardan kaçının doğru seçildiği sayısıdır.

Doğruluk değeri aşağıdaki formül 7 ile hesaplanır:

$$(TN + FP)/Toplam. \tag{7}$$

Hata oranı değeri ise formül 8 olarak tanımlıdır.

$$(1 - Doğruluk). \tag{8}$$

Çizelge 5.5. Sistemin başarı oranları

Özet Oranı	Başarı Oranı	
	1.Yöntem	2.Yöntem
%20	%35	%40
%40	%47	%50
%60	%61	%71

Başarı oranları dikkate alındığında, özetleme oranındaki artışa paralel olarak sezgisel olarak beklendiği üzere başarı oranında da artış olduğu gözlemlenmektedir.

Çizelge 5.6. Geçmiş yapılan çalışmalardaki başarı oranları

Başarı Oranı	Çalışma	Veri Kümesi	Özetleme Oranı
60,80%	Metin Turan, 2015	DUC 2006	250 Kelime
55%	Jaruskulchai, C. ve Kruengkrai, C., 2003	Thai dilinde haber metinleri (30 doküman)	%30

Çizelge 5.6'da paragraf tabanlı çıkarım tekniği ile oluşturulmuş bazı başarılı çalışmaların başarı oranları görülmektedir. Metin Turan (Turan, 2015) yaptığı tez çalışmasında DUC 2006 (Financial Times of London ve Los Angeles Times gazetelerinden her biri için seçilmiş 25 haberden oluşmaktadır.) doküman kümesini kullanarak özetleme oranını maksimum 250 kelime olarak belirlemiştir ve yakaladığı en yüksek başarı oranı %60,80'dir. Jaruskulchai, C. ve Kruengkrai (Jaruskulchai ve Kruengkrai, 2003) Thai dili üzerinde yaptıkları

çalışma sonucu ulaştıkları en yüksek başarı oranı %55'dir. Bu çalışmada geliştirdiğimiz sistemin en yüksek başarı oranı %60 özetleme için %71 olmuştur.

Geliştirilen bu sistemde İngilizce dokümanlar üzerinde bir konuya ait birden fazla doküman içinden otomatik özet çıkarılması amaçlanmıştır. Özet çıkarılırken, kümeleme algoritmaları kullanılarak aynı bilgiyi içeren paragrafların seçimi minimuma indirilmeye çalışılmıştır. Sistem, çıkardığı özet oranı bilgisi kullanıcı tarafından değiştirilebilecek şekilde geliştirilmiştir. Bu çalışma da 19 farklı "Network Güvenliği" konusundan bahseden doküman kümesi üzerinden 4 farklı kullanıcı çeşitli oranlar üzerinden özet çıkarmıştır. Bu çıkarılan özetler sistemin başarısını karşılaştırmak için kullanılmıştır. Bu karşılaştırma sonucu başarı oranlarının değerleri Çizelge 5.5'de verildiği gibidir. Burada k-means algoritmasında başlangıç merkezlerini oluşturmak için belirlediğimiz iki yöntemin sonuçlarını karşılaştırdığımız zaman da ikinci yöntemin bizim sistemimiz için daha uygun olduğu görülmüştür. Özetleme oranının sistemin başarısına etkisine bakıldığında özet oranı arttığı zaman sistemin başarı oranının sezgisel olarak beklendiği gibi arttığı görülmüştür.

KAYNAKLAR

Alguliev R. M., Aliguliyev R. M., Hajirahimova M. S. 2012. Generic document summarization based on maximum coverage and less redundancy. Expert Systems with Applications, 39, 12460-12473.

Baeza-Yates R., Ribeiro-Neto B., 1999. Modern Information Retrieval,Addison-Wesley, England.

Berry, M.W., Drmac, Z., Jessup, E.R., 1999. Matrices, vector spaces and information retrieval, SIAM Review, 41, 2, 335-362.

Brandow R., Mitze K., Rau L.F., 1995. Automatic condensation of electronic publications by sentence selection, Information Processing and Management, 31, 5, 675-685.

Dalal V., Malik L. G. 2013. A survey of extractive and abstractive text summarization techniques, Emerging Trends in Engineering And Technology, 109-110.

Edmundson H.P., 1969. New Methods in Automatic Abstracting, Journal of the ACM, 264-285.

Evans, S., Lloyd, J., Stoddard, G., Nekeber, J., Samone, M. 2005. Risk Factors For Adverse Drug Events. The Annals of Pharmacotherapy, 39, 1161-1168.

Faber V., 1994. Clustering and the Continuous k-Means Algorithm, Los Alamos Science Number 22.

Fukumoto F., Suzuki Y., 2000. Extracting key paragraph based on topic and event detection: towards multi-document summarization, NAACL-ANLP-AutoSum '00 Proceedings of the 2000 NAACL-ANLPWorkshop on Automatic summarization, 4, 31-39.

Goldstein J., Mittal V., Carbonell J., Kantrowitzt M., 2000. Multi-Document Summarization By Sentence Extraction, NAACL-ANLP-AutoSum '00 Proceedings of the 2000 NAACL-ANLP Workshop on Automatic Summarization, 40-48.

Hovy E., Lin C-Y., 1998. Automating Text Summarization in SUMMARIST. Proceeding TIPSTER '98 Proceedings of a workshop on held at Baltimore, Maryland, 197-214

Jaruskulchai, C., Kruengkrai, C., 2003. A Practical Text Summarizer by Paragraph Extraction for Thai, The Sixth International Workshop on Information Retrieval with Asian Language, 9-16.

Khan A., Salim N. 2014. A review on abstractive summarization methods, Journal of Theoritical and Applied Information Technology, 59, 64-72

Lloret, E., Palomar, M., 2010.Challenging Issues of Automatic Summarization: Relevance Detection and Quality-based Evaluation, Informatica, 34, 29-35.

Lunh H.P., 1958. The Automatic Creation of Literature Abstracts, IBM Journal, 159-165.

Mani, I., Maybury M.T., 1999. Advances in automatic text summarization, MIT Press, London.

Min, W., Zhensheng, L., Yuqing, G., 2001. Study on Semantic ParagraphPartition in Automatic Abstracting System, Systems, Man and Cybernetics, 892-897.

Rich E., 1991. Artificial Intelligence, McGraw Hill Inc., Second Edition, Newyork.

Turan M., 2015. Özgün Paragraf Tabanlı Çıkarım Tekniği Kullanarak Otomatik Çoklu Doküman Özetleme, Doktora Tezi, Bilgisayar Mühendisliği Programı, Yıldız Teknik Üniversitesi, İstanbul, Türkiye.

Uzundere E., Dedja E., Diri B., Amasyalı M.F., 2008. Türkçe Haber Metinleri İçin Otomatik Özetleme, Akıllı Sistemlerde Yenilikler ve Uygulamaları Sempozyumu.

Wang M., Wang X., Li C., 2009. Extracting Multi-document Summarization Based on Local Topics, 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery.

EKLER

EK A. Kodlar
EK B. ER Diagramı

EK A. Kodlar

```
ALTER PROCEDURE [dbo].[Insert_WordDispersion]
(
	@Method int,
	@Rate int
)
AS
Begin

	INSERT Into DocumentWordVector(
		DocumentId,
		StemWord
	)
	select distinct
		DocumentId,
		StemWord
	from Words

	INSERT Into MainDocumentWordVector(
		StemWord
	)
	SELECT distinct [StemWord]
	FROM [Summarize].[dbo].[DocumentWordVector]

	DECLARE @StemWord nvarchar(50)
	DECLARE @WordFrequency decimal(18,5)
	DECLARE @DocumentCount decimal(18,5)

	SELECT @DocumentCount=MAX(DocumentId) +1
	FROM DocumentWordVector


	INSERT Into WordDispersion(
		StemWord,
		DispersionValue
	)
	SELECT StemWord,
			(SELECT COUNT(*)  FROM DocumentWordVector WHERE
StemWord=MainDocumentWordVector.StemWord)/@DocumentCount
	FROM MainDocumentWordVector

	Declare @ParagprahCount int =(Select Count (*) from (select
DocumentId,ParagraphId from Words group by DocumentId,ParagraphId) as t)
	Declare @ClusterCount int = (@Rate*@ParagprahCount)/100

	DECLARE @sqlCommand varchar(MAX)

	SET @sqlCommand =
	dbo.FN_SelectBeginDispersionWord(@Method,@ClusterCount)
	EXEC (@sqlCommand)


END
```

Şekil A.1. Doküman vektörlerini bulan ve terimlerin geçiş sıklıklarını hesaplayan prosedür

```
ALTER PROCEDURE [dbo].[Insert_ParagraphsOklidLengths]
AS
Begin

	INSERT Into ParagraphsOklidLengths(
		Document1Id,
		Paragraph1Id,
		Document2Id,
		Paragraph2Id,
		OklidLength
	)
	Select
x.pwv1DocId,x.pwv1ParId,x.pwv2DocId,x.pwv2ParId,SQRT(SUM(x.Result)) from
		(SELECT pwv1.ParagraphWordVectorId
			,pwv1.DocumentId pwv1DocId
			,pwv1.ParagraphId pwv1ParId
			,pwv2.DocumentId pwv2DocId
			,pwv2.ParagraphId pwv2ParId
			,POWER((pwv1.TFValue-pwv2.TFValue),2) as Result
		  FROM ParagraphWordVector pwv1
		  left join ParagraphWordVector pwv2 on pwv1.StemWord
=pwv2.StemWord and  pwv2.ParagraphWordVectorId<>pwv1.ParagraphWordVectorId
		  --order by
pwv1.DocumentId,pwv1.ParagraphId,pwv2.DocumentId,pwv2.ParagraphId
		 ) as x
		 group by x.pwv1DocId,x.pwv1ParId,x.pwv2DocId,x.pwv2ParId

END
```

Şekil A.2. Doküman vektörlerini arasındaki uzaklığı hesaplayan prosedür

```
ALTER PROCEDURE [dbo].[Insert_ClusteredParagraphs]
(
	@Method int,
	@Rate int
)
AS
BEGIN

Declare @ParagprahCount int =(Select Count(*) from (select
DocumentId,ParagraphId from ParagraphWordVector group by
DocumentId,ParagraphId) as x)
Declare @ClusterCount int = (@Rate*@ParagprahCount)/100


DECLARE @sqlCommand varchar(MAX)


SET @sqlCommand =    dbo.FN_SelectBeginParagraph(@Method,@ClusterCount)
EXEC (@sqlCommand)


Declare @FirstParagraphWordVector table
(
	DocumentId int,
	ParagraphId int,
	StemWord nvarchar(100),
	Count decimal(18, 15),
	ClusterCount int
)
insert into @FirstParagraphWordVector
(
	DocumentId,
	ParagraphId,
	StemWord,
	Count,
	ClusterCount
)
select
	pwv.DocumentId,
	pwv.ParagraphId,
	StemWord,
	Count,
	2
From ParagraphWordVector pwv
inner join RandomSelectedParagraphs rsp on pwv.DocumentId=rsp.DocumentId and
pwv.ParagraphId=rsp.ParagraphId

--kümelenen vektörlerin bilgileri tutulur.İlk başta rastgele seçilen kümelerin
değerleri olur
Declare @ClusteredParagraphWordVector table
(
	ParagraphWordVectorId int
)
insert into @ClusteredParagraphWordVector
(
	ParagraphWordVectorId
)
select
	pwv.ParagraphWordVectorId
From ParagraphWordVector pwv
inner join RandomSelectedParagraphs rsp on pwv.DocumentId=rsp.DocumentId and
pwv.ParagraphId=rsp.ParagraphId
```

```
--En yakın sonuç çıkan vektör geçici olarak buraya yazılır.
Declare @TempVector table
(
	Document1Id int,
	Paragraph1Id int,
	Document2Id int,
	Paragraph2Id int
)


Declare @TempTable table
(
	Document1Id int,
	Paragraph1Id int,
	Document2Id int,
	Paragraph2Id int
)

insert into @TempTable
(
	Document1Id,
	Paragraph1Id,
	Document2Id,
	Paragraph2Id
)
select distinct pwv.DocumentId,pwv.ParagraphId,pwv.DocumentId,pwv.ParagraphId
from ParagraphWordVector pwv
inner join RandomSelectedParagraphs rsp on pwv.DocumentId=rsp.DocumentId and
pwv.ParagraphId=rsp.ParagraphId

DECLARE @Document1Id INT
DECLARE @Paragraph1Id INT


DECLARE CRS_Vector CURSOR FOR
select distinct pwv.DocumentId,pwv.ParagraphId
from ParagraphWordVector pwv
Where not EXISTS
(
	SELECT 1 FROM RandomSelectedParagraphs rsp
	WHERE pwv.DocumentId = rsp.DocumentId and
pwv.ParagraphId=rsp.ParagraphId
)

OPEN CRS_Vector

FETCH NEXT FROM CRS_Vector INTO @Document1Id,@Paragraph1Id

WHILE @@FETCH_STATUS =0
	BEGIN


		;WITH cte AS
			(
			select  pwv1DocId,pwv1ParId,pwv2DocId,pwv2ParId,result,
ROW_NUMBER() OVER ( ORDER BY result asc) rn
			from (
					SELECT
							pwv1.DocumentId pwv1DocId
							,pwv1.ParagraphId pwv1ParId
							,SQRT(SUM(POWER((pwv1.Count-
```

```
pwv2.Count),2))) as result
                                                ,pwv2.DocumentId pwv2DocId
                                                ,pwv2.ParagraphId pwv2ParId
                                    FROM ParagraphWordVector pwv1
                                    left join @FirstParagraphWordVector pwv2 on
pwv1.StemWord =pwv2.StemWord
                                    where not exists(
                                                                        select 1
from @ClusteredParagraphWordVector cpwv
                                                                        Where
cpwv.ParagraphWordVectorId=pwv1.ParagraphWordVectorId
                                    ) and pwv1.DocumentId=@Document1Id and
pwv1.ParagraphId=@Paragraph1Id
                                    Group  by
pwv1.DocumentId,pwv1.ParagraphId,pwv2.DocumentId,pwv2.ParagraphId
                                    ) as x
                        )

                        insert into @TempVector
                        (
                                Document1Id,
                                Paragraph1Id,
                                Document2Id,
                                Paragraph2Id
                        )
                        select pwv1DocId,pwv1ParId,pwv2DocId,pwv2ParId from cte
where rn=1


                        insert into @TempTable
                        (
                                Document1Id,
                                Paragraph1Id,
                                Document2Id,
                                Paragraph2Id
                        )
                        select Document1Id,
                                Paragraph1Id,
                                Document2Id,
                                Paragraph2Id
                        from @TempVector


                        insert into @ClusteredParagraphWordVector
                        (
                                ParagraphWordVectorId
                        )
                        select pwv.ParagraphWordVectorId
                        from ParagraphWordVector pwv
                        inner join @TempVector tv on pwv.DocumentId=tv.Document1Id
and pwv.ParagraphId = tv.Paragraph1Id


                        update pwv2 set
                                Count = (pwv2.Count+pwv.Count)/ClusterCount,
                                ClusterCount=ClusterCount+1
                        from @FirstParagraphWordVector pwv2
                        inner join @TempVector tv on
pwv2.DocumentId=tv.Document2Id and pwv2.ParagraphId = tv.Paragraph2Id
                        inner join ParagraphWordVector pwv on
pwv.DocumentId=tv.Document1Id and pwv.ParagraphId = tv.Paragraph1Id and
pwv.StemWord =pwv2.StemWord
```

```
                    Delete @TempVector

                FETCH NEXT FROM CRS_Vector INTO @Document1Id,@Paragraph1Id

        END

CLOSE CRS_Vector

DEALLOCATE CRS_Vector

        Insert into ClusteredParagraphs(
                Document1Id,
                Paragraph1Id,
                Document2Id,
                Paragraph2Id
        )select Document1Id,
                Paragraph1Id,
                Document2Id,
                Paragraph2Id
         from @TempTable order by Document2Id,Paragraph2Id

END
```

Şekil A.3. Vektörleri K-Means algoritmasına göre kümeleyen prosedür

```
ALTER PROCEDURE [dbo].[Get_Summary]

AS
BEGIN

Declare @Temp table
(
	DocumentId int,
	ParagraphId int,
	StemWord nvarchar(50),
	Count decimal(18, 15)
)



DECLARE @Document2Id INT
DECLARE @Paragraph2Id INT

DECLARE CRS_Vector CURSOR FOR
select distinct Document2Id,Paragraph2Id from ClusteredParagraphs

OPEN CRS_Vector

FETCH NEXT FROM CRS_Vector INTO @Document2Id,@Paragraph2Id

WHILE @@FETCH_STATUS =0
	BEGIN

	insert into @Temp(
		DocumentId,
		ParagraphId,
		StemWord,
		Count
	)
	Select @Document2Id,@Paragraph2Id,StemWord,Result from (SELECT
pwv1.StemWord
			,avg(pwv1.Count+pwv2.Count) as Result
		FROM ParagraphWordVector pwv1
		left join ParagraphWordVector pwv2 on pwv1.StemWord
=pwv2.StemWord
		where exists(
					select 1 from
ClusteredParagraphs cp
					where
cp.Document2Id=pwv2.DocumentId and cp.Paragraph2Id=pwv2.ParagraphId and
cp.Document1Id=pwv1.DocumentId and cp.Paragraph1Id=pwv1.ParagraphId
		)
		and pwv2.DocumentId=@Document2Id and
pwv2.ParagraphId=@Paragraph2Id
		group by pwv1.StemWord,pwv2.StemWord) as t



	FETCH NEXT FROM CRS_Vector INTO @Document2Id,@Paragraph2Id

	END

CLOSE CRS_Vector

DEALLOCATE CRS_Vector
```

```
	;WITH cte AS
	(
		Select *, ROW_NUMBER() OVER (PARTITION BY
tbl.Document2ID,tbl.Paragraph2ID ORDER BY result asc) AS rn
		From(
				SELECT cp.Document1Id,
					cp.Paragraph1Id,
					cp.Document2Id,
					cp.Paragraph2Id,
					sum(case when pwv1.Count> pwv2.Count then
pwv2.Count else pwv1.Count end)/sum(case when pwv1.Count> pwv2.Count then
pwv1.Count else pwv2.Count end) result
				FROM ClusteredParagraphs cp
				inner join ParagraphWordVector pwv1 on
cp.Document1Id=pwv1.DocumentId and cp.Paragraph1Id=pwv1.ParagraphId
				left join @Temp pwv2 on pwv1.StemWord
=pwv2.StemWord and cp.Document2Id=pwv2.DocumentId and
cp.Paragraph2Id=pwv2.ParagraphId
				group by cp.Document1Id,
							cp.Paragraph1Id,
							cp.Document2Id,
							cp.Paragraph2Id
		) as tbl
	)



	Insert into ResultTemp
	(
		Document1Id,
		Paragraph1Id,
		Document2Id,
		Paragraph2Id,
		Result,
		RowNumber
	)
		Select
			Document1Id,
			Paragraph1Id,
			Document2Id,
			Paragraph2Id,
			Result,
			rn
		FROM cte


		Declare @Count int = (Select Count(*) from ResultTemp where
RowNumber=1)

		DECLARE @sqlCommand varchar(MAX)='
		Select top '+Cast(@Count as nvarchar(100))+' * from (
		select t.* from ResultTemp t
		inner join (Select Document2Id,Paragraph2Id,rn from ( select
Document2Id,Paragraph2Id,MAX(RowNumber) rn from ResultTemp group by
Document2Id,Paragraph2Id) as x where x.rn>=3) x on t.Document2Id=x.Document2Id
and t.Paragraph2Id=x.Paragraph2Id
		where t.RowNumber=1
		union
		select t.* from ResultTemp t
```

```
            left join (Select Document2Id,Paragraph2Id,rn from ( select
Document2Id,Paragraph2Id,MAX(RowNumber) rn from ResultTemp group by
Document2Id,Paragraph2Id) as x where x.rn>=3) x on t.Document2Id=x.Document2Id
and t.Paragraph2Id=x.Paragraph2Id
            where x.Document2Id is null
            ) as bb
            order by Result'
            EXEC (@sqlCommand)
            Delete ResultTemp
END
```

Şekil A.4. Jaccard uzaklığına göre en yakın paragrafları seçip listeleyen prosedür

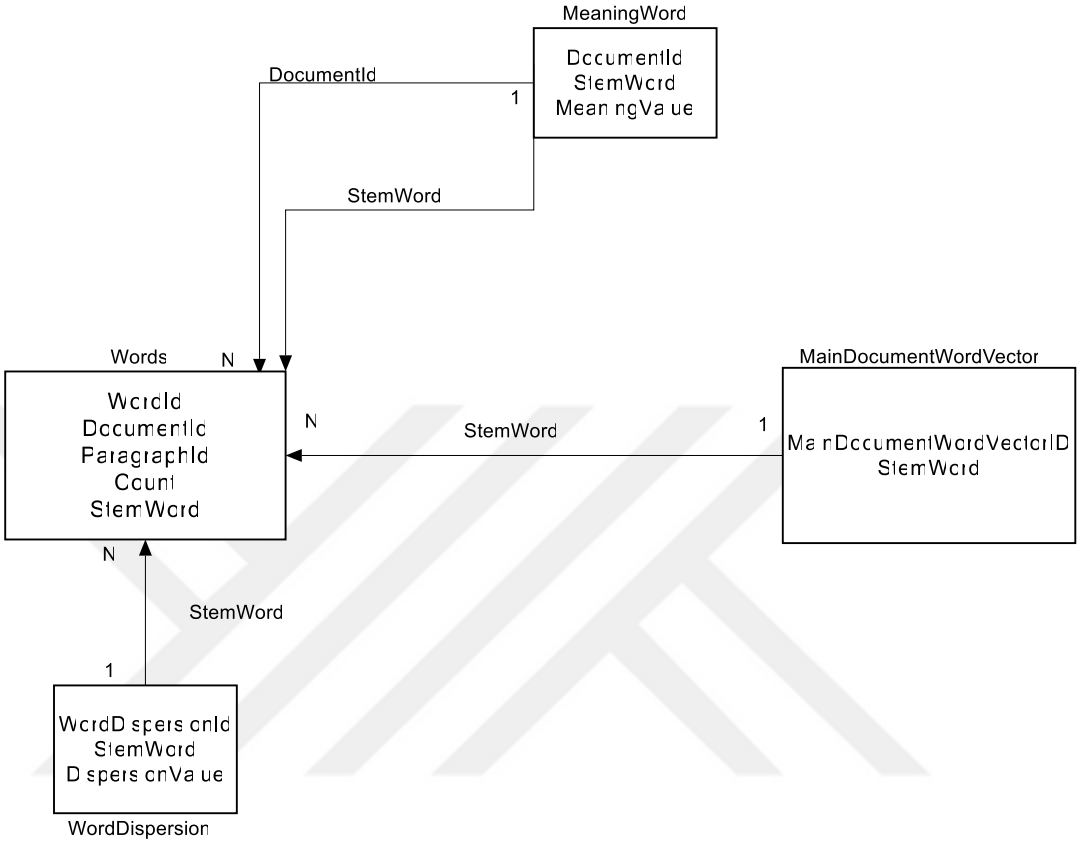
```
from bs4 import BeautifulSoup
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from string import punctuation
from nltk.corpus import wordnet as wn
from nltk import pos_tag
import nltk.data
import pypyodbc
import re
import shutil         #Contains functions for operating files
import os         #imports the os
import string

connection = pypyodbc.connect('Driver={SQL Server};'
                                                'Server=SHORTCUT;'
                                                 'Database=Summarize;'
                                                   'uid=sa;pwd=shortcut')

cursor = connection.cursor()

def ClearDatabase():
    SQLCommand = ("EXEC sp_MSForEachTable 'TRUNCATE TABLE ?'")
    cursor.execute(SQLCommand)
    connection.commit()

def GetContentFreq(content):
    #Read Words
    translator = str.maketrans('', '', string.punctuation)
    words = nltk.word_tokenize(content)
    words = [word.translate(translator) for word in words]
    #Remove one letters
    words = [word for word in words if len(word) > 1]
    #Remove numbers
    words = [word for word in words if not word.isnumeric()]
    #Convert lowercase
    words = [word.lower() for word in words]
    #Remove stop-words
    words = [word for word in words if word not in stopwords.words('english')]

    tempWords = []
    for word in words:
        word = re.sub(r'(\w)\1+', r'\1', word)
        tempWords.append((word))
    words=tempWords
    tempWords = []
    for word in words:
        if word!="" and len(word)>1:
            tempWords.append((word))
    words=tempWords
    return words

counts = dict()

docWordsDict = dict()

stop_words = set(stopwords.words('english'))

paraghraphId = 0
ClearDatabase()

method=input("Enter the method :")
```

```
rate =input("Enter the summary rate :")
path = input("Enter the Directory location to list:")
sortlist = sorted(os.listdir(path))       #Sorting and listing files
i = 0
while(i < len(sortlist)):
    dna = open(path+"\\"+sortlist[i],encoding='utf8',errors='ignore')
    soup = BeautifulSoup(dna)
    paragraphs = soup.find_all("p")
    paraghraphId = 1
    for element in paragraphs:
        stemmer = PorterStemmer()
        SQLCommand = ("INSERT INTO DocumentParagraphTexts
(DocumentId,DocumentName,Paragraph,Text)  VALUES (?,?,?,?)")
        Values = [i,sortlist[i],paraghraphId,element.string.lower()]
        cursor.execute(SQLCommand,Values)
        connection.commit()

        tokens = GetContentFreq(element.text)
        tagged = pos_tag(tokens)
        nouns = [word for word,pos in tagged \
               if (pos == 'NN' or pos == 'NNP' or pos == 'NNS' or pos ==
'NNPS')]

        for word in nouns:
                stems = stemmer.stem(word)
                if stems in counts.keys():
                    shortest,count = counts[stems]
                    counts[stems] = (shortest,count + 1)
                else:
                    counts[stems] = (word,1)
        for kok in counts:
            shortest,count = counts[kok]
            SQLCommand = ("INSERT INTO Words (DocumentId,Word,
Count,StemWord,ParagraphId)  VALUES (?,?,?,?,?)")
            Values = [i,shortest,count,kok,paraghraphId]
            cursor.execute(SQLCommand,Values)
            connection.commit()


        counts.clear()

        paraghraphId+=1
    i+=1


sql = """\
EXEC Insert_WordDispersion @Method=?,@Rate=?
"""
params = (method,rate,)
cursor.execute(sql, params)
cursor.execute("exec Insert_ParagraphsOklidLengths")
sql = """\
EXEC Insert_ClusteredParagraphs @Method=?,@Rate=?
"""
params = (method,rate,)
cursor.execute(sql, params)
connection.commit()

cursor.execute('exec Get_Summary')
row = cursor.fetchone()
print(row[0])
```

```
cursor.close()
connection.close()
```

Şekil A.5. Ön işleme ve prosedürlerin çalıştırıldığı python uygulaması

EK B. ER Diagramı



ÖZGEÇMİŞ

Adı Soyadı : Ahmet İlkay KISAYOL

Doğum Yeri ve Yılı : İstanbul, 19/02/1991

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : ilkaykisayol@gmail.com

Eğitim Durumu

Lise : ECA Elginkan Anadolu Listesi, 2009

Lisans : İstanbul Kültür Üniversitesi, Fen-Edebiyat Fakültesi, Matematik-Bilgisayar Bölümü, 2013

Yüksek Lisans : İstanbul Ticaret Üniversitesi, Fen Bilimleri Enstitüsü, Bilgisayar Mühendisliği

Mesleki Deneyim

XML Creative	2015-2017
Pixel Soft Office	2017-…(devam ediyor)

Yayınları

Ahmet İlkay KISAYOL, Metin TURAN, 2018. Paragraf Tabanlı Çıkarımsal Özetlemede Öbekleme Kullanan İki Yeni Yöntemin Kıyaslanması. Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 6, 1047 - 1057.