FEN BİLİMLERİ ENSTİTÜSÜ

DOĞAL DİL İŞLEME İLE İNGİLİZCE OTOMATİK SÖZLÜK OLUŞTURMA

Ahmet TOPRAK

Danışman
Dr. Öğr. Üyesi Metin TURAN

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
İSTANBUL - 2019

KABUL VE ONAY SAYFASI

Ahmet TOPRAK tarafından hazırlanan **"Doğal Dil İşleme ile İngilizce Otomatik Sözlük Oluşturma"** adlı tez çalışması 24/06/2019 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. Selim AKYOKUŞ** Medipol Üniversitesi
Jüri Üyesi	**Dr. Öğr. Üyesi Alper ÖZPINAR** İstanbul Ticaret Üniversitesi

Onay Tarihi : 09.07.2019

Prof. Dr. Necip ŞİMŞEK
Enstitü Müdürü

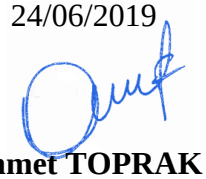
AKADEMİK VE ETİK KURALLARA UYGUNLUK BEYANI

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.

24/06/2019



Ahmet TOPRAK

İÇİNDEKİLER

ÖZET

Yüksek Lisans Tezi

DOĞAL DİL İŞLEME İLE İNGİLİZCE OTOMATİK SÖZLÜK OLUŞTURMA

Ahmet TOPRAK

İstanbul Ticaret Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Dr. Öğr. Üyesi Metin TURAN
2019, 77 sayfa

Dil sözlüğü alanındaki çalışmalar, otomatik sözlük oluşturma konusuna yoğunlaşmış durumdadır. Bu makalede başlangıç olarak verilen bir İngilizce doküman referans alınarak, makale konusuna ait sözlüğün otomatik oluşturulması sağlanmıştır. Çalışmada öncelikli olarak, referans dokümanı temsil eden anlamlı kelimeler tespit edilmiştir. Bu amaçla hem Helmholtz Prensibi hem de TF-IDF metrikleri uygulanmıştır. İlk sözlük kelimeleri bu tohum dediğimiz referans dokümanına ait anlamlı kelimelerden oluşmaktadır. Daha sonra bir döngü ile, en son işlenen dokümana ait anlamlı kelimeler kullanılarak Azure Web Cognitive Web Search sisteminde Web araması yapılmaktadır. Arama sonucu gelen ilk dokümanın, referans dokümanına da uygulandığı üzere Helmholtz Prensibi ve TF-IDF metrikleri ile anlamlı kelimeleri bulunmaktadır. Döngü esnasında bulunan anlamlı kelimeler bu sefer sözlüğe doğrudan eklenmemekte, sapmaları önlemek üzere WordNet sözlüğü kullanılarak her anlamlı kelimenin oluşmuş sözlük ile benzerliği hesaplanmaktadır. Benzerlik değerleri, belirli bir eşik değerinden yüksek olan anlamlı kelimeler sözlüğe eklenmekte ve bu kelimeler kullanılarak Web'te arama döngüsü tekrarlanmakta, nihai olarak sözlük için istenilen kelime sayısına ulaşıldığında ise sonlanmaktadır. Sözlüğün başarımını ölçmek üzere, Hash Similarity benzerlik hesaplaması yöntemi kullanılmıştır. Farklı konularda verilen referans dokümanlarla yapılan sınamalarda, Helmholtz Prensibi uygulanarak yapılan çalışmalarda ortalama % 52,50, TF-IDF metrikleri uygulanarak yapılan çalışmalarda ise % 75,2 oranında benzerliğe sahip sözlükler oluşturulabilmektedir.

Anahtar Kelimeler: Büyük veri, hash similarity, helmholtz prensibi, otomatik sözlük oluşturma, tf-idf metrikleri, wordnet.

ABSTRACT

M.Sc. Thesis

CREATING ENGLISH AUTOMATIC DICTIONARY WITH NATURAL LANGUAGE PROCESSING

Ahmet TOPRAK

İstanbul Commerce University
Graduate School of Applied and Natural Sciences
Department of Computer Engineering

Supervisor: Assist. Prof. Dr. Metin TURAN

Studies in the area of language lexicography are focused on automatic dictionary creation. In this article, an English document is given as an initial reference. In the study, meaningful words representing the reference document were identified. For this purpose, both the Helmholtz Principle and TF-IDF metrics were applied. The first dictionary words consist of the meaningful words of the reference document we call this seed. Then, with a loop, Web search is performed in the Azure Web Cognitive Web Search system using meaningful words from the most recently processed document. The first document from the search result has meaningful words with the Helmholtz Principle and TF-IDF metrics as applied to the reference document. The meaningful words found during the cycle are not added directly to the dictionary this time, and using the WordNet dictionary to avoid deviations, the similarity of each meaningful word with the dictionary formed is calculated. The meaningful words with similarity values higher than a certain threshold value are added to the dictionary and the search cycle is repeated using these words, and finally, when the desired number of words for the dictionary is reached, it ends. Hash similarity similarity calculation method was used to measure the performance of the dictionary. In the tests carried out with reference documents given in different subjects, in the studies conducted by applying Helmholtz Principle 52,50 %, while TF-IDF metrics are applied, dictionaries with a similarity of 75,2 % can be created in the studies.

Keywords: Automatic dictionary creation, big data, hash similarity, helmholtz principle, tf-idf metrics, wordnet.

TEŞEKKÜR

Bu çalışmanın yürütülmesi sırasında desteğini esirgemeyip, bilgi birikimi ve tecrübesi ile çalışmam boyunca ihtiyaç duyduğum her zaman yardım eden değerli danışman hocam Dr. Öğr. Üyesi Metin TURAN'a teşekkürlerimi sunarım.

Tez çalışmam boyunca yanımda olup, çalışmamın her aşamasında desteğini hiçbir zaman eksik etmeyen sevgili nişanlıma ve eğitim hayatım boyunca her zaman desteklerini sunan ve yanımda olan sevgili aileme teşekkür ederim.

Ahmet TOPRAK
İSTANBUL, 2019

ŞEKİLLER

ÇİZELGELER

SİMGELER VE KISALTMALAR

YAS	Yanlış Alarm Sayısı
IR	Bilgi Getirimi
TF	Terim Frekansı
IDF	Ters Metin Frekansı
NLP	Doğal Dil İşleme
NLTK	Natural Language Toolkit

1. GİRİŞ

Teknolojinin bu denli hızlı olarak geliştiği, bilginin bu kadar kolay elde edilebildiği günümüzde doğru bilgiye erişim hayati önem taşımaktadır. Bilginin yoğun olması nedeniyle büyük veri üzerinden anlamlı bilgi çıkarımı oldukça önemlidir. Bu noktada istenen bilgiye özgü sözlükler bize yardımcı olacaktır.
Bu sözlükler Web arama motoru optimizasyonu, otomatik özetleme sistemleri, tema belirleme ve metin sınıflandırma araştırmaları için kullanılabilir. Bu amaçtan yola çıkarak dokümanları işleyerek bu dokümanlara ait anlamlı kelimeler belirlenmiş, bizim için önemli olmayan kelimeleri eleyerek istenilen amaca yönelik sözlüğün oluşturulması sağlanmaya çalışılmıştır. Böylece bilgiye daha kolay bir şekilde ulaşmamız sağlanmaktadır.

Sözlük oluşturma işlemi elle (Kepuska ve Rojanasthien, 2011), yarı-otomatik (Koeva vd., 2016) ve otomatik (Ellen, 1993) olmak üzere 3 farklı şekilde yapılabilir. Elle oluşturulan dokümanlar ve sözlükler, statik olmakta ve sözlüğün büyümesi için dışarıdan sürekli bir müdahale gerekmektedir. Bu durumda, sürekli bakım maliyeti ortaya çıkacaktır. Bu bakım maliyetinden kurtulmak ve daha yapısal bir uygulama elde edebilmek adına bu süreçleri otomatik hale getirmek önem arz etmektedir.

Ellen R. çalışmasında (Ellen, 1993) elle oluşturulan sözlüklerin eksikliklerine değinmiştir. “Bilgiye dayalı doğal dil işleme sistemleri, belirli görevlerle iyi bir başarı elde etmelerine rağmen, çoğu zaman eleştirilmektedirler. Çünkü bunlar, büyük ölçüde elle bilgi mühendisliği gerektiren alana özgü bir sözlüğe bağımlıdırlar. Bu bilgi mühendisliği darboğazı, bilgi tabanlı NLP sistemlerini gerçek dünya uygulamalarında hayata geçirmek için pratik değildir. Böylece kolayca ölçeklenemez veya yeni alanlara aktarılamazlar.”

1.1. Problem Tanımı

Teknolojinin gelişmesi, hem olumlu hem de olumsuz sonuçları beraberinde getirmiştir. Bilgi erişiminin kolaylaşması, zamandan tasarruf edilmesi gibi

olumlu etkilerinin yanında, elde edilen bu bilgiler içerisinden ihtiyacımız olan anlamlı bilgiyi çıkarma güçlüğü gibi olumsuz etkileri de bulunmaktadır. Bu sorunun çözümü için problem tanımına istinaden farklı şekillerde anahtar kelime belirleme yöntemleriyle çalışmalar yapılmaktadır. Bu doküman anahtar kelime belirleme çalışmalarında yer alan yöntemlere Helmholtz Prensibi (Khoury vd., 2016) ve TF-IDF metrikleri (Qaiser ve Ali, 2018) örnek verilebilir. Doküman(lar)a ait anlamlı kelimeler tespit edildikten sonra, bu anlamlı kelimeler için dil sözlüğü oluşturulması sağlanacaktır.

1.2. Çalışma Konusu ve Amacı

Bu çalışmada otomatik sözlük oluşturmak için bir iş akışı önerilmiş ve deneysel olarak sınanmıştır. Temel yaklaşım, kullanıcının amaçladığı sözlüğe referans olarak verdiği doküman(lar)dan önemli kelimelerin bulunmasıdır. Bu önemli kelimelerin bulunmasında farklı teknikler uygulanmıştır. Daha sonra bu kelimeler kullanılarak, Web ortamında ilişkili yeni dokümanlar bulunur ve sözlük için diğer kelimeler elde edilir. Diğer kelimeler, sözlüğe WordNet benzerlik değeri en yüksek olanlardan seçilir. Böylece anlamsız sözcüklerin sözlüğe eklenmesi mümkün olduğunca önlenmiş olur. Bu işlem sözlük kelime sayısına ulaşılana kadar tekrarlanır. Böylece sözlük dışarıdan herhangi bir müdahale olmadan otomatik ve sürekli olarak büyümeye devam edecektir. Amacımız, özelleşmiş problemler üzerinde yüksek başarı oranları elde edecek genelleştirilebilir otomatik sözlük oluşturma modeli geliştirmektir.

2. LİTERATÜR ÖZETİ

Sözlük oluşturma ile ilgili geçmişten günümüze çalışmalar yapılmış olup, bu çalışmalar farklı yöntem ve farklı analizler içererek günümüze kadar gelmiştir. İlk olarak 1990 (Ellen, 1993) ve 2000 (Kepuska ve Rojanasthien, 2011) yılları arasında bu çalışmalara başlanmış ve zaman içerisinde değişen ihtiyaçlar doğrultusunda farklı teknikler geliştirilmiştir. Sonuçların bu tekniklere bağlı olarak iyi yönde değiştiği gözlenmektedir. Bu çalışmaların çoğunun amacı yığın veri içerisinden istenilen bilgiyi elde etmektir.

Helmholtz Prensibi kullanılarak önemli kelime tespit çalışmaları yapılmıştır. Agnes Desolneux ve arkadaşları (Desolneux vd., 2001) çalışmalarında, Helmholtz Prensibi'ne bağlı temel algı ilkesine göre, dijital görüntüdeki geometrik yapıları önceden bilinen bir bilgi olmadan hesaplamak için kenar algılama yöntemi adında bir teoriyi anlatmışlardır. Bu teori, bir görüntüdeki kenarları ve sınırları (kapalı kenarları) parametresiz bir yöntemle tanımlamayı ve hesaplamayı sağlamaktadır. Bu çalışmanın görüntü analizi çalışmalarında ara bir katman olarak kullanılabileceği savunulmaktadır.

Raphael Khoury ve takımının yaptıkları (Khoury vd., 2016) çalışmalarında, Helmholtz Prensibi'ni uygulamışlardır. Yapılan çalışmada, yazılım bakım maliyetlerini kolaylaştırmak, önemli bilgileri büyük bir iz üzerinden etkin olarak tanımlayabilmek için bir yaklaşım geliştirmişlerdir. Bu yaklaşım, hata ayıklama, performans analizi, özellik geliştirme gibi büyük verilerin olduğu alanlarda, bu verilerin içinden anlamlı izlerin (bilgi) elde edilmesinde Gestalt teorisi ve Helmholtz Prensibi'ne uygulanmasını ele almaktadır.

Metin Turan ve Sena Ögtelik yaptıkları (Turan ve Ögtelik, 2018) çalışmalarında, belirli tema ve bu temalara ait alt kavramları içeren bir dokümanın, hangi sınıfa ait olduğunun tahmin edilmesi üzerine bir çalışma yapmışlardır. Dokümanlarda tema ve alt kavramların tespiti için kullanılabilecek anlamlı sözcüklerin belirlenmesi amacıyla Helmholtz Prensibi temelli Gestalt teorisi kullanılmıştır. 70 adet sınama dokümanı ile farklı sayıda (5, 10, 20) anlamlı kelime seçilerek

deneyler yapılmış, başarı oranının konularda yaklaşık olarak % 95, alt kavramlarda ise % 80 olduğu belirtilmiştir.

Helmholtz Prensibi ile birlikte, TF-IDF metrikleri de anahtar kelime tespit çalışmalarında önemli rol oynamaktadır. Bu çalışmalardan biri de, Christian Caldera ve arkadaşlarının (Caldera vd., 2014) çalışmasıdır. Bu çalışmada, PRIMA adında bir araç geliştirilmiştir. Bu aracın amacı, konferanslarda en fazla iş yüküne neden olan Uluslararası Program Komite'sinin (IPM) üyelerine gönderilen başvuruları atamaktır. Çünkü her hakemin uzmanlık alanına göre, makale ataması yapılması gerekmektedir. Bu amaçtan yola çıkarak, sisteme yüklenen makaleler otomatik olarak incelenmiş ve daha sonra hakemlerin uzmanlık alanına göre atama işlemi gerçekleştirilmiştir. Uygun hakeme atama işlemini gerçekleştirmek için TF-IDF metrikleri kullanılmıştır. Yapılan deneylerde, 100 makale için yapılan hakem atama işlemi 3 dakika sürmüştür.

Qaiser ve Ali (Qaiser ve Ali, 2018) çalışmalarında, dokümandaki anahtar kelimelerin tespiti için TF-IDF metriklerini kullanmış ve bazı noktalarda beklenmedik sonuçlar aldıkları için bu yöntemin eksikliklerine değinmişlerdir. Çalışmalarında, dokümanlarda geçen (“go” “goes”), (“play” “playing”), (“mark” “marking”), (“year” “years”) kelimeleri aynı kelime köküne sahip olmasına rağmen, bu kelimelerin farklı kelimeler olarak algılanması nedeniyle sonuçları etkilediği belirtilmiştir. Aynı zamanda kelimelerin birlikte oluşumu ve dokümanda anlam bazında değil, sözlük bazında kontrol sağlandığı için eleştirilmiştir. TF-IDF metriklerinden daha iyi sonuçlar elde etmek için, Naive Bayes gibi diğer tekniklerle birleştirilebileceği belirtilmiştir.

Bir önceki bölümde de belirtildiği şekilde, sözlük oluşturma işlemi otomatik, yarı otomatik ve elle olmak üzere 3 farklı şekilde işletilmektedirler. Bu türlerin her biri için zaman içinde çalışmalar yapılmıştır. Bu çalışmalardan biri de, Riloff Ellen’in otomatik sözlük çalışmasıdır. Riloff Ellen (Ellen, 1993) çalışmasında, metinden bilgi çıkarmak için otomatik olarak alana özgü kavramlar sözlüğü olan “AutoSlog” adlı bir sistem geliştirmiştir. AutoSlog'a bir metin verildiğinde, AutoSlog istenen bilgileri bu metinden çıkararak bir dizi sözlük girişi oluşturur.

AutoSlog'a verilen metinler istenen bilgileri temsil ediyorsa, AutoSlog tarafından oluşturulan sözlük, önemli ölçüde başarılı sonuçlar verecektir. AutoSlog sözlüğü ile 5 kişi-saatte, terör olaylarını içeren bir sözlük oluşturulmuştur. AutoSlog sözlüğü daha sonra iki yetenekli lisansüstü öğrenci tarafından yapılan ve yaklaşık 1500 kişi-saat çaba gerektiren el yapımı bir sözlükle karşılaştırılmıştır. İki sözlük, her biri 100 metin içeren iki test seti kullanılarak değerlendirilmiştir. Sonuç olarak, AutoSlog sözlüğünün, el yapımı sözlüğün performansının % 98'ini sağladığı görülmüştür.

Silverman ve arkadaşlarının 1999 yılında yaptıkları (Silverman vd., 1999) otomatik sözlük oluşturma çalışmasında, Apple Computer'da konuşma sentezi araştırma ve geliştirmesini desteklemek için oluşturulan Victoria sözlüğünün tasarımı ve yapısı açıklanmaktadır. Victoria sözlüğü 5 ana bölümden oluşmaktadır. Bunlar polifon, prosodik bağlam, tekrar eden konuşma, fonksiyon kelime dizileri ve sürekli konuşmadır. Bu sözlük, her biri konuşma sentezinin belirli bir yönünü kapsayacak şekilde tasarlanmıştır. Victoria sözlüğü, genel olarak ABD İngilizcesi ile oluşturulmuştur. Victoria sözlüğünün amacı, konuşma üzerinden anlamsal metinlerin toplanmasıdır. Sözlük, Apple'ın gelecek nesil text-to-speech sistemi MacinTalk 4 için süre ve adım modellerinin istatistiksel tahmininde kullanılmaktadır.

Vijay ve arkadaşları (Vijay vd., 2018) çalışmalarında, son 8 yılda çevrimiçi olarak yayınlanan tweetleri kullanarak Hintçe-İngilizce kod karışık sözlük oluşturdular. Bu sözlüğü oluşturmak için, öncelikle Twitter'ın gelişmiş arama seçeneğini kullanan Twitter Python API1 kullanılarak tweetler Twitter'dan alındı. Alınan tweetler, zaman damgası, URL, metin, kullanıcı, retweetler, cevaplar, tam ad, kimlik ve beğeniler gibi tüm bilgiler json formatına dönüştürüldü. Tüm gürültülü tweetleri kaldırmak için kapsamlı bir yarı otomatik işlem gerçekleştirildi. Tüm bu adımlarından sonra 2866 kelimelik bir sözlük oluşturuldu ve sözlük kelimeleri mutluluk, hüzün, öfke, sürpriz, nefret ve çoklu duygu olarak sınıflandırıldı. Daha sonra çevrimiçi yayınlanan tweetler için duygu analizi çalışması yapılmış ve % 58,2 oranında doğruluk tespiti elde edilmiştir.

Bir diğer otomatik sözlük oluşturma çalışması da, S. Vorapatratorn ve arkadaşlarının yaptıkları (Vorapatratorn vd., 2012) çalışmadır. Bu çalışmada, özel fonetik dağılım kullanan otomatik sözlük oluşturma yöntemi açıklanmaktadır. Genellikle, sistem bir web tarayıcısı üzerinden, İnternet'ten sürekli metin indirerek verilerini seçer. Açgözlü algoritma(covetous algorithm), uygun kelimeleri çıkarmak için metinlere uygulanır, bu işlem uygun metin sözlüğü kuruluncaya kadar devam eder. Çalışmada elde edilen sonuçlar, internetten çekilen veri sayısının hedef fonetik dağılımını gerçekleştirebildiğini ve % 99,13 oranında telefon kapsama alanı oluşturduğunu göstermektedir. Bu metin sözlüğünün, daha sonra konuşma sözlüğünü verimli bir şekilde üretmek için kullanılabileceği belirtilmiştir.

Kepuska Veton Z. ve Rojanasthien (Kepuska ve Rojanasthien, 2011) çalışmalarında, elle sözlük oluşturma işlemi yapmışlardır. Bu çalışma ile film, TV dizileri ve DVD'lerden konuşma sözlüğü oluşturmak için bir veri toplama sistemi oluşturmuşlardır. Bu DVD'lerden yapılan sözlük üretimi, geleneksel bir konuşma sözlük elde etme yöntemine kıyasla, daha düşük maliyetli bir çözüm sunmaktadır. Ek olarak, verilerin toplanması ve bir sözlüğe işlenmesinin daha kısa sürdüğü belirtilmiştir.

Metin Turan ve Coşkun Sönmez (Turan ve Sönmez, 2015) tema ve alt kavram tespiti çalışmalarında, aynı alt kavram ile ilgili kelimeleri bulmak için elle bir sözlük oluşturmuşlardır. Bu sözlükte, 2 konu, 37 alt konu ve başlangıçta toplam 2313 kelime yer almaktadır. Önerilen yöntem, eğitim ve spordan rastgele seçilen 40 dokümanı içeren bir veri setinde değerlendirilmiştir. Sonuç olarak, tema algılama başarı oranı ortalama % 83, alt kavram tespiti ise ortalama % 68 olarak bulunmuştur.

Bir diğer elle sözlük oluşturma çalışması da, Behdad Behmadi Moghaddas ve arkadaşlarının (Moghaddas vd., 2013) yaptıkları çalışmadır. Bu çalışmada, Farsça otomatik metin özetleme işlemi yapabilmenin önündeki engellerin kaldırılmasına yönelik bir öneri sunulmaktadır. Bu engelin nedeni, Farsça metin

özetleyicilerin değerlendirilmesinde standart bir sözlük olmamasıdır. Bu sorununun çözümü için Pasokh adında bir sözlük oluşturulmuştur. Pasokh sözlüğü, çeşitli konularda çok sayıda Farsça haber belgesinden oluşan bir veri kümesidir. Bu veri setini üretmek için 2000 kişi-saat harcanmıştır. Daha sonra Pasokh sözlüğüne tekli ve çoklu dokümanlar verilerek, özetleme başarısı gözlemlenmiştir. Pasokh sözlüğü tekli doküman özetlerinde % 83,56, çoklu doküman özetlerinde ise % 60,24 başarı oranı elde etmiştir.

Marc Bertin ve Iana Atanassova (Bertin ve Atanassova, 2018) çalışmalarında, InTeReC adında elle bir sözlük oluşturmuşlardır. Bu sözlüğün amacı, akademik camiaya büyük bir veri kümesi sağlayarak, alıntı bağlamı analizlerini ve bu alıntı yapılan makalelerin tür bakımından dağılımlarını kolaylaştırmaktır. Bu amaçtan yola çıkarak, 85.660 adet makale PLOS adresinden XML formatında alındı. Daha sonra veri ön işleme adımları uygulanarak veriler temizlendi ve InTeReC sözlüğüne aktarıldı. Sonuç olarak, 314.023 kelimelik InTeReC sözlüğü oluşturuldu.

Türkçe sözlük oluşturma ile ilgili çalışmalar da yapılmaya başlanmıştır. Bunlardan biri de Türkçe WordNet çalışmasıdır. Bu çalışmada (Aktaş vd., 2016), özellikle Türkçe WordNet olmakla birlikte, detaylı WordNet literatür araştırması yapılmıştır. Uzun vadede ise bugüne kadar hazırlanmış ilişkili sözlüklerle birlikte en geniş Türkçe bilişim sözlüğü hazırlanmıştır. Böylelikle büyük bilişim projelerinin zemini olan ilişkisel bilişim terimleri sözlüğünün diğer sözlüklere göre ilişki ve kelime sayısı çokluğu olarak en büyük sözlük olduğu belirtilmiştir.

3. OTOMATİK DİL SÖZLÜĞÜ OLUŞTURMA

Bu çalışma, sisteme başlangıç olarak verilen dokümanlardan, o dokümanın konusuyla ilgili sözlüğün elde edilmesi üzerinedir. Sınama amaçlı seçilmiş dokümanlardan elde edilen anlamlı kelimeler sistemde ön işleme ve ağırlıklandırma aşamalarından geçtikten sonra, kelimeler sözlüğe eklenmiş ve bu kelimeler üzerinden Web araması yapılmıştır. Aşağıda bu adımlar detaylı olarak açıklanacaktır.

3.1. Ön İşleme

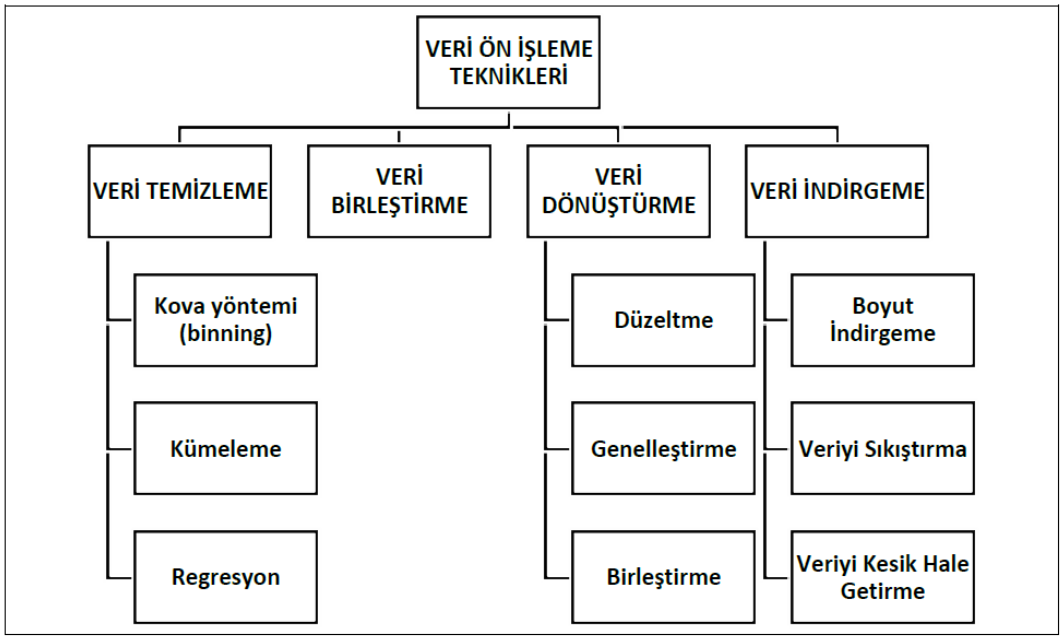
Bilişim dünyasının gelişimi insan yaşamına birçok açıdan büyük kolaylıklar getirmiştir. Bu kolaylıkların yanında başka açıdan birtakım zorluklar da getirdiği bilinmektedir. Kullanılan teknolojik araçların çokluğu, mobil uygulamaların yaygınlaşması, internetin kolay erişilebilirliği gibi nedenlerden ötürü çok büyük miktarda veri bolluğu oluşmuştur. Bu büyük verileri işlemek artık klasik yöntemler ile yapılması güçleşmiş hatta imkansız hale gelmiştir. Tam bu noktada veri madenciliği tekniği ortaya çıkmıştır. Veri madenciliği, büyük veri tabanlarındaki gizli bilgi ve yapıyı açığa çıkarmak için, çok sayıda veri analizi aracını kullanan bir süreçtir. Veri madenciliğinin üç farklı bakış açısı vardır; veri tabanı bakış açısı, makine öğrenim bakış açısı ve istatistiksel bakış açısıdır. Yazılan kitaplar ve geliştirilen bilgisayar programları da bu farklı bakış açılarına uygun olarak yapılmaktadır (Zhou, 2002).

Klasik istatistiksel uygulamalar ile veri madenciliği arasındaki en temel farklılık, veri kümesinin büyüklüğüdür. Bir istatistikçi için ‘büyük’ veri kümesi birkaç yüz veya bin veri içerir. Veri madenciliği ile uğraşan birileri için ise milyon veya milyarlık veri beklenmeyen bir sayı değildir. Bu tip büyük veri tabanları gerçek hayatta sıkça ortaya çıkmaktadır (Oğuzlar, 2003).

Veri madenciliği çalışmalarında başarılı sonuçlar elde etmenin en önemli noktası kullanılacak verilerin kaliteli olması gerekir. Kaliteli veriler elde etmek için kullanılacak verilerin ön işleme süreçlerinden geçmesi gerekir. Böylece veri madenciliği çalışmalarında başarılı sonuçlar elde edilebilir. Çok sayıda uygulamada veri ön işlemenin türlerinin bir tanesinden daha fazlasına ihtiyaç duyulmaktadır. Bu sebeple veri ön işlemede tür belirlenmesi önemlidir (Famili vd., 1997). Bu yaklaşımdan yola çıkarak çok sayıda veri ön işleme tekniği geliştirilmiştir. Bunlara; veri temizleme, veri birleştirme, veri dönüştürme ve ayrıklaştırma, veri indirgeme gibi teknikler örnek olarak verilebilir.

3.2. Veri Ön İşleme Teknikleri

Ön işleme yöntemlerinin detaylı hali Şekil 3.1.' de gösterilmiştir.



Şekil 3.1. Veri ön işleme yöntemlerinin genel şeması

Veri temizleme: Toplanan verilerin ilk ön işleme aşamasıdır. Bu aşamada, eksik verilerin tamamlanması, gürültünün düzeltilmesi ve verilerdeki tutarsızlıkların giderilmesi işlemleri yapılmaktadır. Herhangi bir değişkene ilişkin eksik

değerlerin doldurulması için farklı yollar vardır. Bunlardan bazıları aşağıda kısaca açıklanmaktadır.

1. Eksik değer içeren kayıt veya kayıtlar atılabilir.

2. Değişkenin ortalaması eksik değerlerin yerine kullanılabilir.

3. Aynı sınıfa ait tüm örneklemler için değişkenin ortalaması kullanılabilir.

4. Var olan verilere dayalı olarak en uygun değer kullanılabilir. Burada sözü edilen en uygun değerin belirlenmesi için regresyon veya karar ağacı gibi teknikler kullanılabilir. Örneğin yaşı x, eğitim düzeyi y olan bir kişi için ücret durumu, mevcut verilerden yukarıdaki tekniklerden birinin kullanılmasıyla tahmin edilebilir.

Veri temizleme tekniğinin kullanıldığı bir diğer problem ise gürültülü verilerdir. Gürültülü verilerin oluşmasının nedeni, bir değerin hatalı olarak ölçülmesi, hatalı veri toplama aracı, ya da verinin hatalı niteliklerinden kaynaklı olabilir. Gürültü verilerinin giderilmesi içinde çeşitli yöntemler mevcuttur. Aşağıda veri düzeltme tekniklerinden bazıları açıklanmaya çalışılmıştır:

- **Kova yöntemi (binning):** Kova yöntemi, küçükten büyüğe veya büyükten küçüğe sıralanmış verileri düzeltmek için kullanılır. Binning yönteminde öncelikle sıralanmış veriler eşit büyüklükteki bin'lere ayrılır. Daha sonra bin'ler, bin ortalamaları, bin medyanları veya bin sınırları yardımıyla düzeltilir.

- **Kümeleme (Clustering):** Veriler içerisindeki aykırı değerler kümeler ile belirlenebilir. Benzer değerler aynı grup veya aynı küme içinde yer alırken, aykırı değerler kümelerin dışında yer alacak şekilde sınıflandırılır.

- **Regresyon (Regression):** Veriler, regresyon ile verilere bir fonksiyon uydurularak düzeltilebilir. Uydurulan fonksiyona uymayan noktalar aykırı değerlerdir.

Veri Birleştirme: Veri madenciliğinde genellikle farklı veri tabanlarındaki veriler veri ambarı denilen alanda birleştirilirler. Farklı veri tabanlarındaki verilerin tek bir veri tabanında birleştirilmesiyle şema birleştirme hataları (schema integration errors) oluşur. Bu hatalar çoğunlukla tablo kolonlarının farklı isimlerde verilmesi kaynaklı olarak ortaya çıkmaktadır. Şema birleştirme hatalarından kaçınmak için meta veriler kullanılır. Meta veri, veriye ilişkin veridir. Raporlama işlemlerinin sık olarak yapıldığı kurumlarda veri ambarı alanı oluşturulur ve kayıtlar meta veri olarak tutulur.

Veri Dönüştürme: Veri dönüştürme işleminin amacı, verileri veri madenciliği için uygun formlara dönüştürmektir. Veri dönüştürme; düzeltme, birleştirme, genelleştirme ve normalleştirme işlemlerden bazılarını içerebilir.

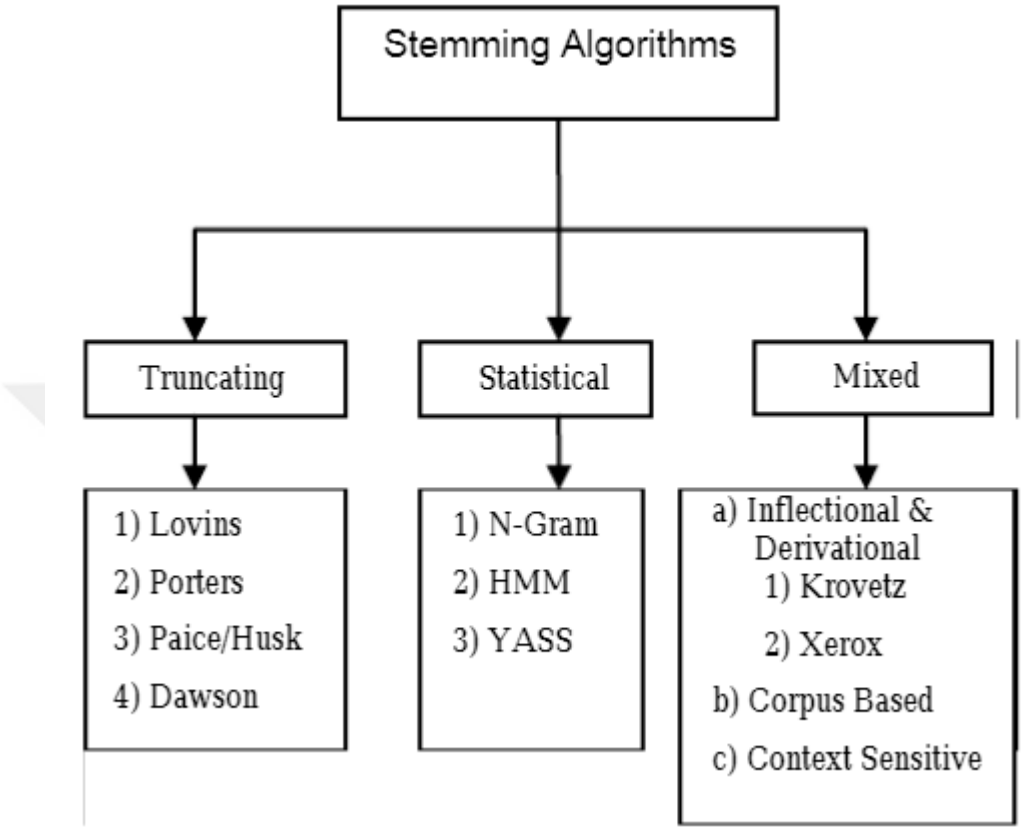
Veri İndirgeme: Veri indirgeme, işlem yapılacak veri boyutunun çok büyük olması durumunda, veriyi daha küçük boyutlu hale getirme işlemidir. Bu işlem sonucunda elde edilen küçük boyutlu veriler ile daha etkin sonuçlar elde edilebilir.

Ön işleme tekniklerine, verilerle çalışılan her alanda ihtiyaç duyulmaktadır. Yukarıda bahsedilen teknikler veri madenciliği çalışmalarında, çalışma bazında farklılık gösterse de, çoğu zaman bu sırada ilerler. Doğal dil işleme çalışmalarında ise, ön işleme teknikleri sisteme verilen dokümanlara uygulanarak, doküman içindeki kelimelerin standart bir forma dönüştürülmesi sağlanır. D. Tanasa (Tanasa ve Trousse, 2004) ve V. Chitraa, (Chitraa ve Davamani, 2010) çalışmalarında ön işleme tekniklerine yer vermişlerdir.

3.2.1. Çalışmada uygulanan ön işleme teknikleri

Çalışmamızda çeşitli kaynaklardan toparlanan dokümanlar üzerinde işlemler yapılmıştır. Doküman türleri ve formatları birbirinden farklı olarak verilebilmektedir. Bu da beraberinde verilerin ön işlemeden geçirilerek veri temizleme yöntemi ile dokümanların uygun formata getirilmesini gerektirmektedir. İlk aşama olarak sisteme girilen dokümanların işleme alınabilecekleri birimlere ayrıştırılması gerekir. Çalışmamızda işlemler paragraf tabanlı olarak ele alınmaktadır. Bu nedenle dokümanların HTML taglarına ayrılarak <p> tagına sahip alanlarının tespit edilmesi gerekmektedir. Dokümanları HTML taglarına ayırmak için "BeautifulSoup" HTML ayrıştırıcı kütüphanesi kullanılır. BeautifulSoup, HTML veya XML dosyalarını işlemek için oluşturulmuş güçlü ve hızlı bir kütüphanedir. Bu kütüphane sayesinde <p> tagına sahip alanlar tespit edilir. Paragraflarına ayrılmış metin parçacıkları, paragraf bazında kelimelerine ayrılır. Dokümanlar metin içinde anlamı olmayan birçok kelime içermektedir ve bu kelimeler (stop words) cümleden çıkarıldıklarında da anlamsal bir kayba yol açmazlar. Ancak bu kelimeler ön işleme adımlarına tabi tutulmazsa sonuçlara olumsuz yönde etki edeceklerdir. Çünkü paragraf bazında işlem yapılacağı ve bu paragraftaki anlamlı kelimeler frekans bazında tespit edileceğinden sonuçları etkileyecektir. Çalışmada İngilizce sözlük oluşturma işlemi yapılacağından İngilizce diline ait sonlama kelimelerinin dokümanlardan atılması gerekmektedir. Aşağıda İngilizce diline ait sık kullanılan sonlama kelimeleri yer almaktadır. "a", "about", "above", "after", "again", "against", "all", "am", "an", "and", "any", "are", "aren't" vb... Sonlama kelimeleri ayırt edici özelliğe sahip olmadıklarından, ön işleme sırasında temizlenirler. Sonraki adımda, dokümandaki boşluk, rakam gibi anlamsız kelimelerde dokümandan çıkarılır. Daha sonra dokümandaki tüm kelimeler küçük harf formatında yazılarak harf uyumluluğu sağlanır. Bu işlemin ardından kelimelerin yapım ve çekim ekleri tek bir forma dönüştürülür. Bu işlemin yapılmasının nedeni anlamlı kelimelerin bulunması aşamasında frekans bazlı işlem yapılmasıdır. Ekleri tek bir forma dönüştürme işlemi için kök bulma (stemming) algoritmaları kullanılır. İngilizce dili için geliştirilmiş algoritmalardan en yaygın olarak bilinen ve kullanımı kolay olan (Moral vd., 2014) PorterStemmer kök bulma algoritması bu projede uygulanmıştır. Aşağıda

(Şekil 3.2.)'de NLP'de kullanılan kök bulma algoritmaları verilmiştir (Khoury vd., 2016).



Şekil 3.2. Kök bulma algoritmaları

3.3. Kelimelerin Ağırlıklandırılması

Dokümanların ön işleme adımları sonucunda PorterStemmer algoritması ile kökleri bulunduktan sonra, bu kelimeler arasından anlamlı kelimelerin tespit edilmesi gerekmektedir. Farklı anahtar kelime belirleme yöntemleri bulunmaktadır. Helmholtz tabanlı Gestalt İnsan algı teoremi ve TF-IDF metrikleri en bilinenleridir. Çalışmamızda her 2 yöntem içinde uygulamalar yapılmış sonuçlar karşılaştırmalı olarak değerlendirilmiştir.

Ağırlıklandırma işleminde; hem başlangıç olarak sisteme verilen doküman(lar), hem de Web araması sonucu elde edilen doküman(lar) için anlam değerlerinin

(meaning value) bulunması sağlanmıştır. Kelimelerin anlam değerlerinin bulunması ile birlikte bu kelimeler içinden anahtar kelime seçimi yapılmasına elverişli hale getirilmiştir.

3.3.1. Helmholtz Prensibi tabanlı Gestalt insan algı teoremi

Gestalt İlkeleri, 20. yüzyılın ortalarında Almanya'da ortaya çıkan, adını Almanca şekil ya da form anlamına gelen "gestalt" sözcüğünden alan, bilişsel süreçler içerisinde algı ve algısal örgütlenme konularına yoğunlaşan psikoloji teorisinin temelini oluşturan prensiplerdir. Bu teori şöyle özetlenebilir: "Bütün, kendisini oluşturan parçaların toplamından bağımsızdır" (Sherpa, 2019).

Gestalt kuramının ana bileşenleri olan Gestalt ilkeleri, aşağıdaki gibidir.

Şekil-Zemin İlişkisi: Dış çevrede algılanan obje şekil, şeklin üzerine işlendiği ve onu kapsayan alan da zemindir (Yeşilyaprak, 2005).

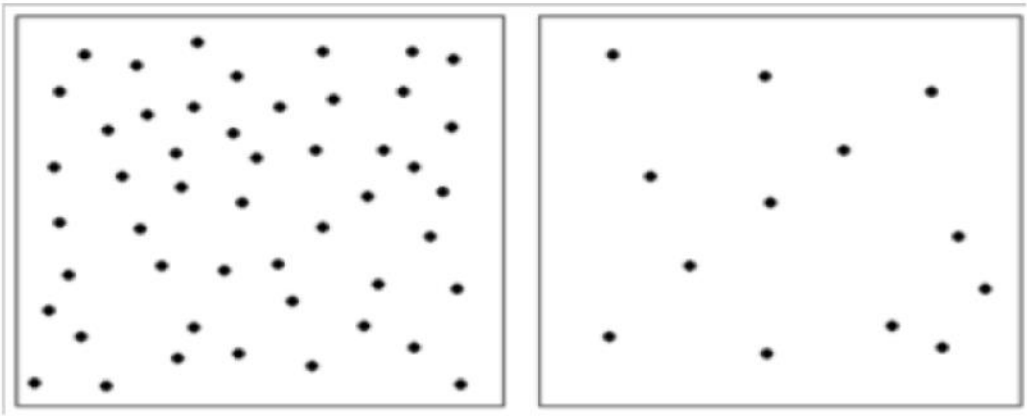
Yakınlık İlkesi: İnsanlar birbirlerine yakın olan nesneleri algılarken hepsini bir grup ve bütün olarak anlamlandırırlar. Birbirlerine yakın olan nesneler mekân, zaman ve anlam olarak beraber gruplandırılır(Senemoğlu, 2012).

Tamamlama İlkesi: Şekil olarak bitirilmemiş, devam ettirilmemiş objelerin şeklinde bir eksiklik olduğu halde tammış şeklinde algılanmasıdır. İnsan beyni bu
yarım kalan şekilleri kendiliğinden tamamlama eğilimindedir.

Benzerlik İlkesi: Bazı duyusal özellikler yönünden (şekil, renk, doku v.b) benzer olan cisimler bir küme olarak algılanır. Hem görsel hem işitsel olarak da birbirine benzerlik gösteren uyarıcılar yine aynı şekilde bir bütünmüş gibi algılanır (Senemoğlu, 2012)

Süreklilik İlkesi: Aynı yönde belli bir süreklilikle devam eden objeler sanki birbirlerinin devamıymış gibi bir bütünlük içinde algılanır (Zeren, 2007).

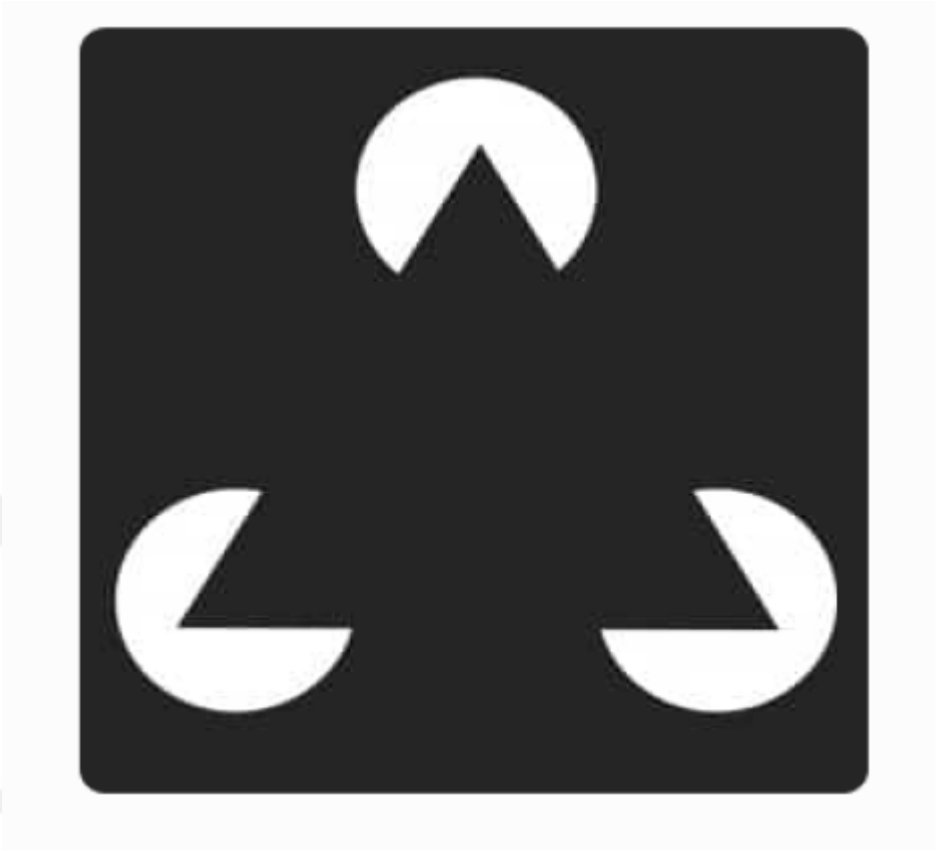
Anlam değeri görüntü işlemede de kullanılan Helmholtz Prensibi (Balinsky vd., 2011) tabanlı Gestalt insan algı teorisine (Desolneux vd., 2007) dayanmaktadır. İstatistiksel fizik ile doğrulanabilen bu yöntem beklenti değeri hesaplamaları kullanılarak elde edilmiştir (Balinsky vd., 2011). Bu teori rastgele oluşturulmuş bir görüntü içerisinde rahatlıkla algılanabilen bir geometrik yapının şans eseri olmayacağının, bunun bir anlamı olduğunu söylemektedir. Bu durumu anlatmak için en açık örnek rastgele noktalardan oluşturulmuş (Şekil 3.3.)'deki iki görüntüdür.



Şekil 3.3. İnsan algısında Helmholtz Prensibi(Balinsky vd., 2011)

Soldaki şekilde rastgele noktalara bakıldığında insan algısı ile herhangi bir geometrik yapı algılanamıyor. Bu beklentisel bir durumdur ve görüntünün rastgele oluşturulduğu rahatlıkla söylenebilirken, sağdaki şekilde sanki görünmeyen bir çizgi üzerine konulmuş 5 nokta görülmektedir. Rastgele noktalarla oluşturulmuş bir görüntü içerisinde böyle geometrik bir yapının olma olasılığının beklenti değeri çok düşük olmasına karşın bu olay gerçekleşmiş ve insan bunu rahatlıkla algılayabilmektedir. Öyle ise bu yapının anlamlı bir yapı olması gerekir. Çünkü beklenti değeri düşük bir olayın gerçekleşmesi şans eseri olamayacak kadar düşük bir olma olasılığına sahiptir. Gerçekleşme olasılığı çok düşük olan bir olayın gerçekleşmiş olması bu olayın anlamlı ya da önemli olduğunu göstermektedir (Balinsky vd., 2011). Şekil 3.3.'de

verilen insan algısında Helmholtz Prensibi'nin bir benzeri de Şekil 3.4.'de verilmiştir.



Şekil 3.4. Helmholtz Prensibi

Şekil 3.4. incelendiğinde İlk bakışta siyah zemin üzerinde oluşturulmuş bir üçgen görülmektedir. Bunun sebebi beynimizin görsel bilgiyi alıp bize anlamlı gelen, tanıdık, düzenli simetrik veya algılayabileceğimiz şekliyle bize görüntüyü sunuyor olmasıdır. Bu şekilde asıl olan siyah zemin üzerinde 3 adet beyaz "pac man" olmasıdır (Turan ve Ögtelik, 2018).

3.3.2. Helmholtz Prensibi'nin uygulanması

Helmholtz Prensibi, Gestalt insan algı teorisini kullanır. Bu teori metin madenciliğinde her bir kelime için; dokümanın paragrafında, kelimenin m kez geçmesinin olası olup olmadığının belirlenmesinde kullanılmıştır. Bu teoriye dayanan anlam değeri bazı formüllerle hesaplanır. Bu formüller şu şekildedir:

$$\mathrm{YAS}(k, P, D) = \binom{K}{m} \frac{1}{N^{m-1}} \quad (3.1)$$

$$\mathrm{Anlam}(k, P, D) = -\frac{1}{m} \log \mathrm{YAS}(k, P, D) \quad (3.2)$$

$$N = \frac{|D|}{|P|} \quad (3.3)$$

$$\log \mathrm{YAS}(k, P, D) = \log\left(\binom{K}{m} \frac{1}{N^{m-1}}\right) \quad (3.4)$$

$$\binom{K}{m} = \frac{K!}{m!(K-m)!} \quad (3.5)$$

Helmholtz Prensibi'ne göre kullanılan bu formüllerde;

D: Dokümanın paragrafları
P: Paragrafta yer alan cümleler, anlamı taşımaktadır.

Çalışmada dokümanları paragraflarına ayırarak, paragraf tabanlı bir çalışma yaptığımız için bu formülü kullanırken D'yi dokuman, P'yi de dokümanda yer alan paragraf olarak ele aldık. Yaptığımız çalışma için formülde yer alan diğer terimlerin de açıklaması aşağıdaki gibidir:

k: İşlem yapılan kelime
P: Veri kümesinde yer alan paragraf sayısı
D: Veri kümesinde yer alan dokumanlar
m: Hesapladığımız kelimenin toplam kaç tane paragrafta geçtiği
K: Hesapladığımız kelimenin doküman da toplam kaç kez geçtiğinin sayısı

N: Tüm veri kümesindeki toplam kelime sayısının, bir dokümandaki toplam kelime sayısına bölümü olarak ele alınmıştır(Turan ve Ögtelik, 2018).

YAS, kelimenin m kere P'de geçmesinin beklenti değeridir. Beklenti değeri 1'den küçük ise w kelimesinin m kez P'de geçmesi beklenmeyen bir olaydır. Fakat w kelimesi m kez P'de zaten geçmiş ise w kelimesi P için anlamlı önemli bir kelime olarak düşünülebilir. Eğer anlam değeri sıfırdan büyük ise YAS değeri 1'den küçüktür (Dadachev vd., 2012).

Anlam değerinin hesaplanmasında kullanılan YAS değeri, anlam değeri ile ters orantılıdır. Bunun anlamı YAS değeri ne kadar küçükse o özelliğin o sınıf için anlam değeri o kadar büyüktür. Anlam değerinin büyük olması ise özelliklerin daha etkin ve önemli bir özellik olduğunu ifade etmektedir.

3.3.3. TF-IDF metrikleri

Anlamlı kelimelerin tespit edilmesi için Helmholtz Prensibi dışında TF-IDF metrikleri de kullanılabilir. TF-IDF metrikleri, doğal dil işleme ve veri madenciliği konularının ortak çalışma alanı olan metinler üzerinde istatistiksel incelemeler konusunda kullanılmaktadır. TF-IDF kavramı IR gibi konuların altında bir sıralama(ranking) algoritması olarak sıkça geçmektedir (Bilgisayar Kavramları, 2012).

Term Frequency: Bir doküman içerisinde geçen terim ağırlıklarını hesaplamak için kullanılan yöntemdir.

Aşağıda Şekil 3.5.'de TF ile ağırlık hesaplamasında kullanılan yöntemler açıklanmaktadır.

Weighting scheme	TF weight
binary	{0,1}
Raw frequency	$f_{t,d}$
Log normalization	$\log 1 + f_{t,d}$
Double normalization 0.5	$0.5 + 0.5 \frac{f_{t,d}}{\max f_{t,d}}$
Double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max f_{t,d}}$

Şekil 3.5. Term Frequency ağırlık hesaplama yöntemleri (Medium Corporation, 2015).

Aşağıda Şekil 3.5.'de verilen formüller detaylı olarak açıklanmaktadır.

Binary: Doküman içerisinde terimin olup olmadığını temsil etmektedir.

Raw Frequency: Terimin dokümanda geçme sayısı / dokümandaki kelime sayısı oranına eşittir.

Double Normalization: 0,5–1 arasında bir değer oluşturur. **Raw Freq / Maksimun geçen Terim Raw Freq** değerine bölerek doküman ne kadar uzun olursa olsun terimin diğer terimlere olan oranını bularak frekansı normalize etmektedir.

Inverse Document Frequency: Birden fazla dokümanda kelimenin geçme sayısını bularak bu kelimenin terim olup olmadığını, bağlaç vb. (Stop Words)

olduğu anlamaya çalışır. Bunun için terimin geçtiği doküman sayısı / doküman sayısının logaritmasının mutlak değeri alınmaktadır.

TF-IDF metriklerini örnek üzerinden açıklamak gerekirse;

TF (Term Frequency): "D1" adında 5000 kelimeye sahip bir dokümanımız olsun. Bu doküman içerisinde "Computer" kelimesi 10 kez geçtiğini varsayarsak, "Computer" kelimesinin TF değeri aşağıdaki gibi hesaplanır.

$$\text{TF} = \frac{10}{5000} = 0.002$$

IDF (Inverse Document Frequency): 10 adet dokümanımız olsun. Bu dokümanların 5 tanesinde "Computer" kelimesinin geçtiğini varsayarsak, "Computer" kelimesinin IDF değeri aşağıdaki gibi hesaplanır.

$$\text{IDF} = \log_e \frac{10}{5} = 0.3010$$

IDF hesabı sırasında bazı noktalara dikkat etmek gerekir. Öncelikle logaritmanın tabanının bir önemi yoktur. Amaç üssel fonksiyonun tersi yönde bir hesap yapmaktır. Doğal logaritma kökü e, 2 veya 10 gibi sayılar en çok kullanılan değerlerdir. Genelde TF-IDF değerinin kıyas için kullanıldığını ve diğer terimlerin TF-IDF değerleri ile kıyaslandığını düşünecek olursak hepsinde aynı tabanın kullanılıyor olması sonucu değiştirmeyecektir.

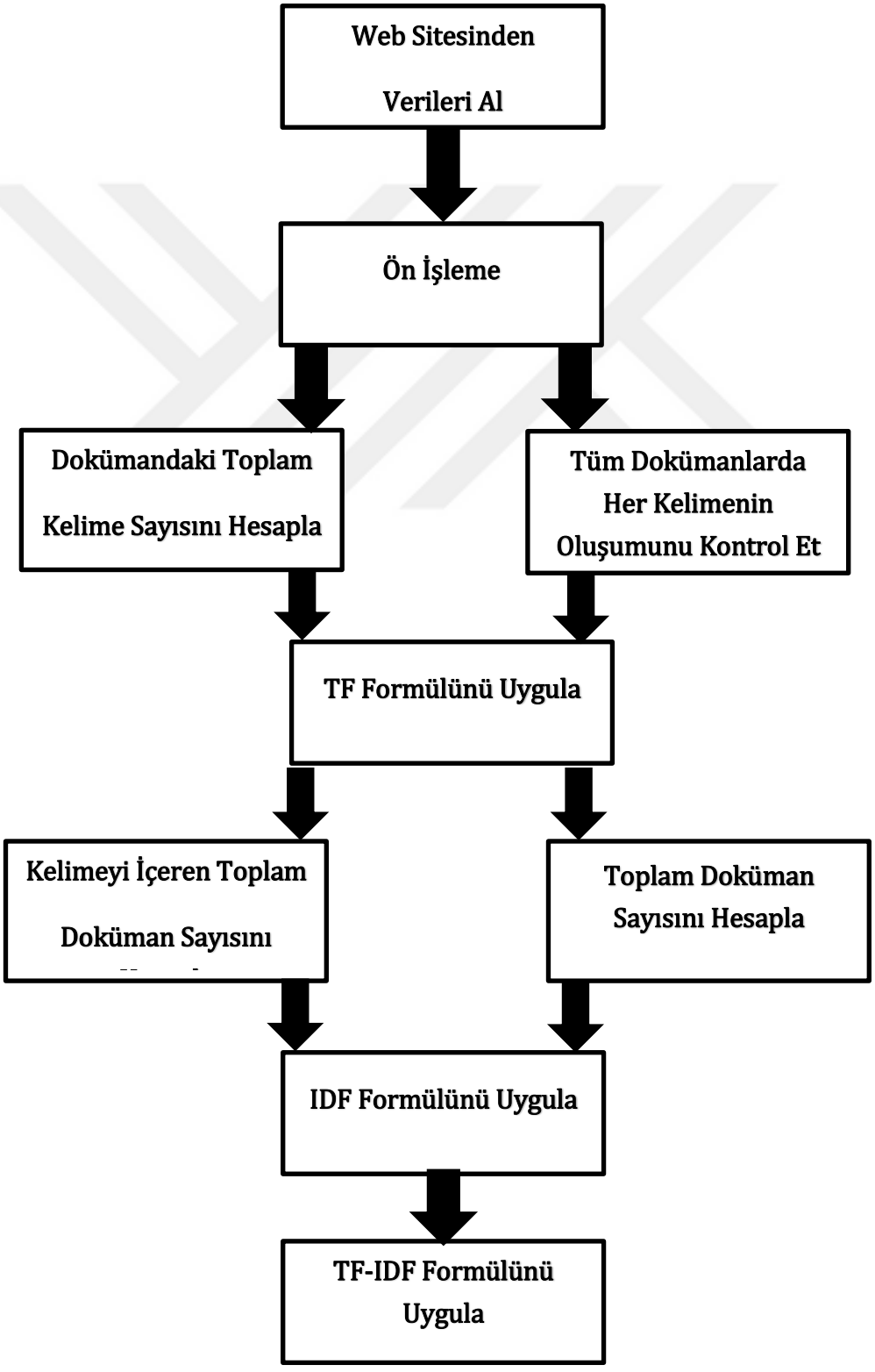
Diğer dikkat edilecek bir husus ise, IDF hesabı sırasında geçen "terimi içeren doküman sayısı" değeridir. Bu değer hesaplama sırasında paydada yer almaktadır ve bu değerin 0 (sıfır) olma ihtimali vardır. Bu durumda sonuç sıfıra bölüm belirsizliğine götürebileceğinden genelde bu değere 1 eklemek sıkça yapılan bir programlama yaklaşımıdır (Bilgisayar Kavramları, 2012).

TF-IDF (Term Frequency—Inverse Document Frequency): TF ve IDF değerleri hesaplandıktan sonra TF-IDF hesaplaması yapılır. TF-IDF değeri, TF ve

IDF değerlerinin çarpımına eşittir. Yukarıda elde ettiğimiz TF ve IDF değerlerini baz alarak TF-IDF değerini hesaplarsak, sonuç aşağıdaki gibi olacaktır.

$$\mathrm{TF} - \mathrm{IDF} = 0.002 * 0.3010 = 0.000602$$

Aşağıda Şekil 3.6.'da TF-IDF metrikleri sürecinin Web üzerinden alınan dokümanların işlenmesi sürecinden itibaren akışını anlatmaktadır.



Şekil 3.6. TF-IDF süreci

3.3.4. TF-IDF metriklerinin uygulanması

Çalışmamızda, başlangıç ve Web araması sonucu elde edilen dokümanların sisteme beslenmesi sonrası ön işleme adımlarına tabi tutulup PorterStemmer algoritması ile kökleri bulunduktan sonra, bu dokümanlara ait anlamlı kelimelerin tespit edilmesi sağlanmıştır. Anlamlı kelimelerin tespiti için TF-IDF metrikleri kullanılmıştır. TF-IDF değeri belli bir değerin üzerinde olan kelimeler sözlüğe eklenmiş ve daha sonra sözlüğe eklenen bu kelimeler üzerinden Web araması yapılmıştır.

3.3.5. Benzerlik oranı tespiti

Sözlük oluşturma çalışmamızda, TF-IDF ya da Helmholtz Prensibi ile elde edilen anlamlı kelimeler doğrudan sözlüğe eklenmemiş bazı kontroller yapılmıştır. Bu kontrollerden biri de benzerlik oranıdır. TF-IDF ya da Helmholtz Prensibi ile anlamlı kelime seçimi yapıldıktan sonra, kelimeleri sözlüğe eklemeden önce her anlamlı kelimenin oluşmuş sözlük ile benzerliği hesaplanmaktadır. WordNet ile kelimelerin benzerlik hesaplaması yapıldıktan sonra benzerlik oranı belirlenen eşik değerin üzerinde olan kelimeler sözlüğe eklenmiştir. Çalışmada, benzerlik oranı hesaplanması için WordNet benzerlik değeri kullanılmıştır.

WordNet, Pricenton Üniversitesi Bilişsel Bilimler Laboratuarı'nda hazırlanmış İngilizce sözlük veri tabanıdır. İsimler, fiiller, sıfatlar ve zarflar, her biri ayrı bir kavram ifade eden bilişsel eşanlamlılar (synsets) kümeleri halinde gruplandırılır. Synsets kavramsal, anlamsal ve sözcüksel ilişkiler aracılığıyla birbirine bağlanır (Miller vd., 1990). WordNet ücretsizdir ve herkese açık olarak indirilebilir. WordNet'in yapısı dolayısıyla, hesaplamalı dilbilim ve doğal dil işleme için kullanışlı bir araçtır (Medium Corporation, 2018).

3.3.6. Web aramasından dokümanların elde edilmesi

TF-IDF ya da Helmholtz Prensibi uygulanarak elde edilen anahtar kelimelerin, WordNet sistemi ile benzerlik oranları hesaplandıktan sonra, parametre olarak belirlenen eşik benzerlik oranından yüksek orana sahip olan kelimeler sözlüğe eklenmiştir. Sözlüğe eklenen kelimelerden ise, sadece parametre olarak belirlenen değerin üstünde olanlar Web aramasında kullanılmıştır. Örneğin, sözlüğe 10 adet kelime eklenirse ve uygulamada sözlüğe eklenen kelimelerin %50'si Web aramasında kullanılsın şeklinde parametre tanımlanırsa bu kelimelerden sadece 5 tanesi Web aramasında kullanılacaktır. Projede Web üzerinde arama yöntemi olarak Azure Cognitive Web Search yöntemi uygulanmıştır. Web aramasında kullanılacak kelimeler boşluk karakteri ile birleştirilerek Azure Web Search servisine parametre olarak verilmiştir. Örnek vermek gerekirse, Web aramasında kullanılacak kelimeler sırasıyla "game", "football", "sport" kelimeleri ise, bu kelimeler Web'te "game football sport" şeklinde aranmaktadır. Bu işlem sözlük kelime sayısı belirlenen parametre değerine ulaşana kadar devam eder. Böylece sistemde sürekli bir döngü sağlanmıştır.

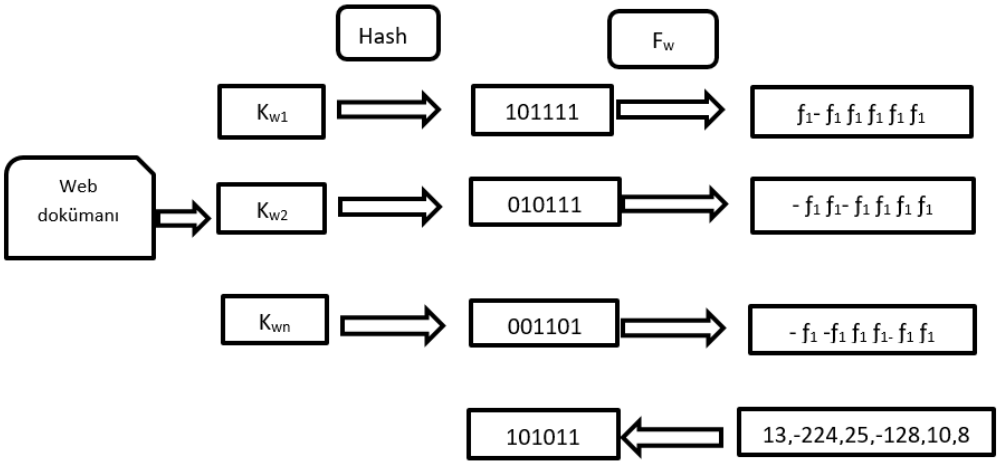
3.3.7. Genel benzerlik oranı tespiti

Bu adımda, uygulama sonucunda elde edilen sözlüğün başlangıç olarak verilen kelimeler ile benzerliği tespit edilmeye çalışılmıştır. Başarı oranı tespit yöntemi olarak SimHash (Benzerlik Özeti) algoritması kullanılmıştır. SimHash algoritması, özellikle metin işlemenin yoğun olduğu, arama motoru gibi uygulamalarda dosyaların veya web sitelerinin birbirine olan benzerliğini bulmak için kullanılan bir algoritmadır (Bilgisayar Kavramları,2012). SimHash algoritması, iki dosyayı birer vektör olarak görür ve bu vektörler (yöney, vector) arasındaki cosinüs (cosine) bağlantısını bulmaya çalışır. SimHash algoritması,

benzerlik tespiti amacıyla birçok farklı çalışmada kullanılmıştır. Jiang (Jiang ve Sun, 2011) ve Pi (Pi vd., 2019) çalışmalarında, doküman benzerliğini elde etmek amacıyla SimHash algoritması kullanılmıştır. Aşağıda Şekil 3.7.'de SimHash algoritmasının sözde kodu, Şekil 3.8.'de ise SimHash algoritmasının çalışma prosedürü yer almaktadır.

```
(1) function Simhash (document D)
(2) {
(3)    Init vector Sim[0,...,(f − 1)] = 0;
(4)    for each Keyword Kw in D
(5)    {
(6)       Kw is hahsed into f bit Hash value X;
(7)          for (i = 0; i < f; i ++)
(8)          {
(9)             if (X == 1)
(10)            {
(11)               Sim[i] = Sim[i] + weight(Kw);
(12)            } else {
(13)               Sim[i] = Sim[i] − weight(Kw);
(14)            } //end if
(15)         } //end for
(16)   }//end for
(17)   for (i = 0; i < f; i ++)
(18)   {
(19)      if (Sim[i] > 0) Sim[i] 1;
(20)      else Sim[i] = 0;
(21)   } //end for
(22) }
```

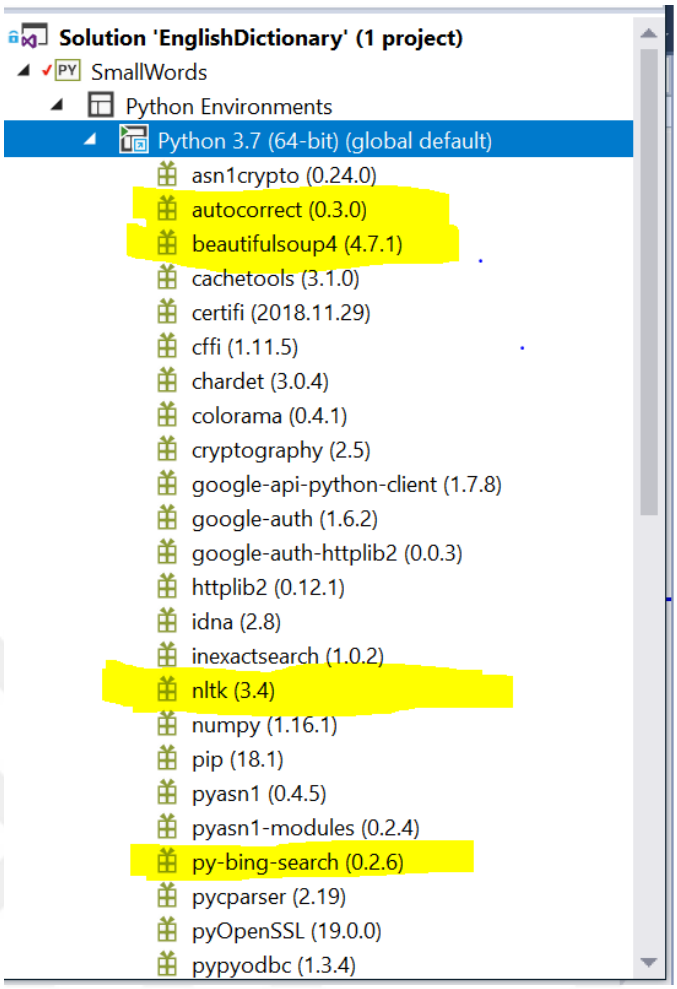
Şekil 3.7. SimHash algoritmasının sözde kodu (Ho ve Sung, 2014)



Şekil 3.8. SimHash algoritmasının çalışma prosedürü (Ho ve Sung, 2014)

4. YAZILIMIN GERÇEKLENMESİ

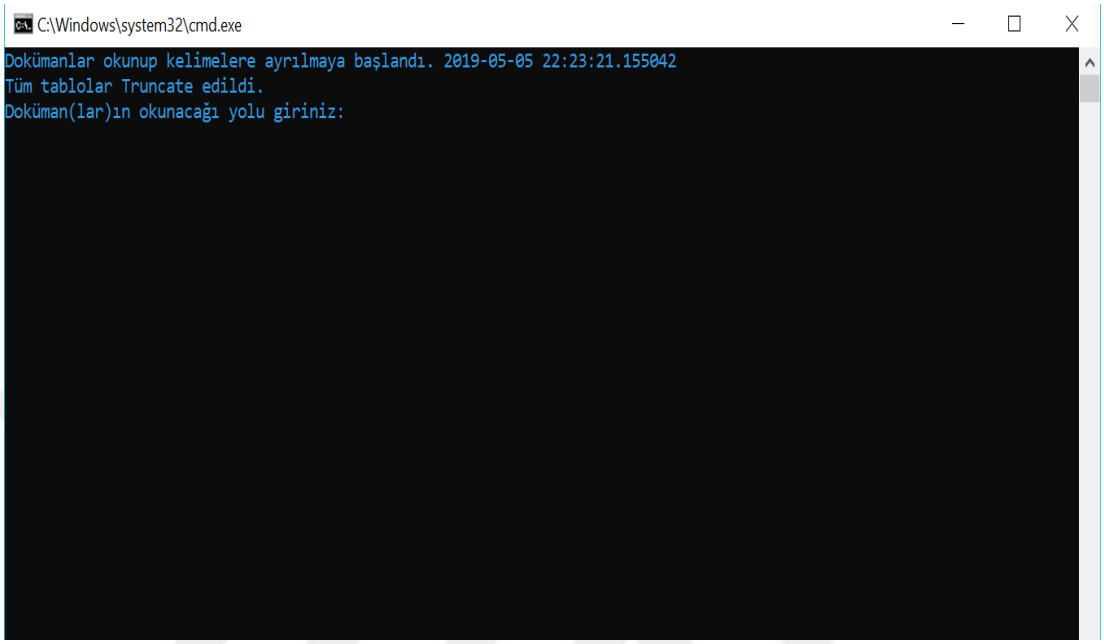
Otomatik sözlük oluşturma çalışması ile ilgili tüm yazılımsal süreç aşağıda detayları bahsedilecek otomasyon ile sağlanmıştır. Uygulamanın yazılımsal süreci Python ile kodlanmıştır. Proje, Python 3.7 versiyonu ile geliştirilmiştir. Projede Python dilinin tercih edilmesinin nedeni, hem esnek kodlama standartlarına sahip olması, hem de doğal dil işleme problemlerine yönelik geniş çözümler sunmasıdır. Veritabanı işlemleri için MsSql Server kullanılmıştır. Aşağıdaki Şekil 4.1.'de belirtilen tüm paketler uygulamaya dahil edilmiştir. Sarı ile işaretlenen paketler uygulamada sıkça kullanılan önemli işlem paketleridir.



Şekil 4.1. Yazılımın gerçeklenmesinde kullanılan paketler

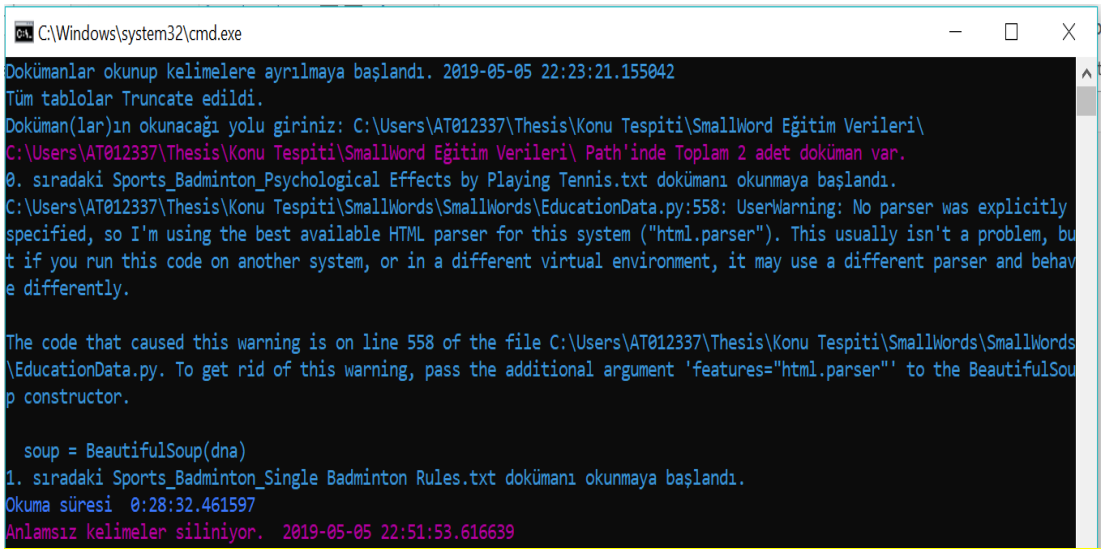
Proje içerisinde, 3 farklı uygulama bulunmaktadır. Bunlardan biri, tüm süreci yöneten projenin sonlanıp sonlanmayacağına karar veren uygulamadır. İkincisi, kullanıcıdan aldığı doküman(lar)a göre dokümanları ön işleme adımlarına tabi tutan, kelimelerin köklerini bulan, ardından bu kelimelerin ağırlıklarını tespit edip ağırlıkları en yüksek olan kelimeleri belirleyen uygulamadır. Üçüncüsü ise, ikinci adım sonunda anlamlı kelimeler sözlüğe eklendikten sonra uygulamanın yaşam döngüsüne devam edebilmesi için Web üzerinden sözlüğe eklenen kelimeler ile arama yapıp yeni dokümanlar oluşturan uygulamadır. Kullanıcıdan oluşturmak istediği sözlük konusuyla ilişkili başlangıç dokümanlarını isteyen arayüz Şekil 4.2.'deki gibidir. Uygulama arayüzünde tüm süreci takip edebilmek adına her adım konsol ekranına yazdırılmıştır. Kullanıcıdan aldığı dosya

yolunda bulunan tüm dokümanlar işleme alınacaktır. (Örn: “C:\Users\[User]\[Klasör Adı]”).



Şekil 4.2. Proje arayüzü

Kullanıcıdan aldığı dosya yolunda bulunan tüm dokümanların tespit edilmesi ve okunması işlemi aşağıda Şekil 4.3.’de gösterilmiştir. Eğer belirtilen dosya yolunda herhangi bir doküman bulunmadığı taktirde (Şekil 4.2.)’de belirtilen ekran tekrar kullanıcıya sunulacaktır.



Şekil 4.3. Proje arayüzünde dokümanların işlenmesi

Önceki bölümlerde de belirtildiği üzere, bu çalışmada kelimelerin ağırlıklandırılması ile ilgili hem Helmholtz Prensibi hem de TF-IDF metrikleri kullanılmıştır. Bölüm 4.3.'deki "Terimlerin Ağırlıklandırılması" başlığına kadar yer alan tüm süreçler hem Helmholtz Prensibi hem de TF-IDF metriklerini içeren uygulamalar için ortaktır. "Terimlerin Ağırlıklandırılması" başlığından itibaren Helmholtz Prensibi ve TF-IDF metrikleri için yapılan çalışmalar, farklı başlıklar halinde açıklanacaktır.

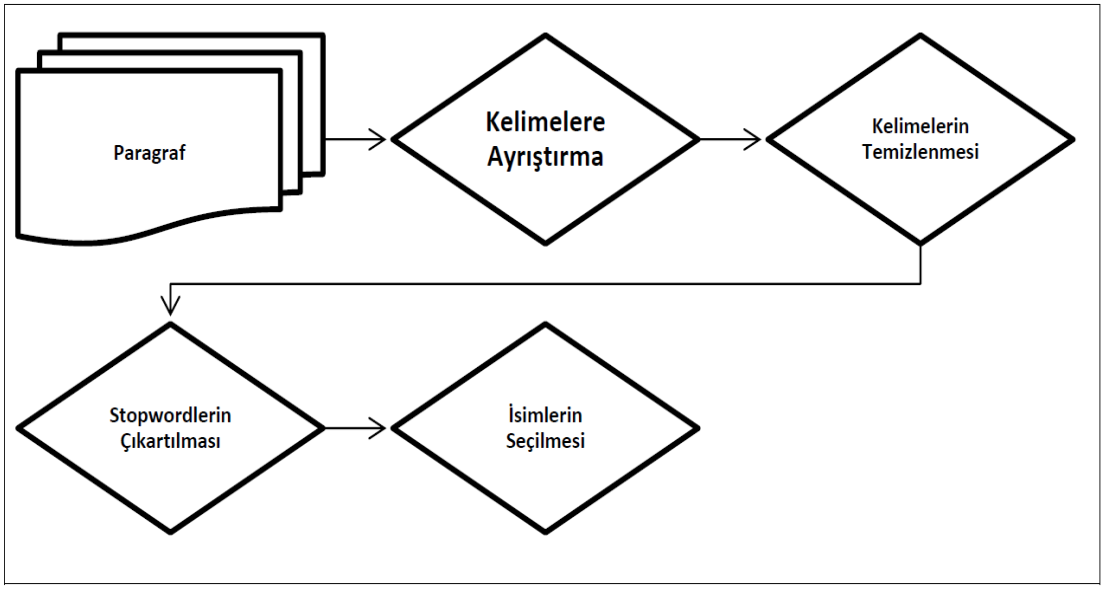
4.1. Paragrafların Tespiti

Çalışmada, otomatik sözlük oluşturma işlemi sadece kullanıcıdan alınan başlangıç doküman(lar) ile değil, aynı zamanda başlangıç olarak verilen kelime(ler) ile de oluşturulabilmektedir. Başlangıç doküman(lar) ile sözlük oluşturma çalışması yapıldığında, bu dokümanların ön işleme adımına tabi tutulması gerekmektedir. Bu bölüm yazılımda gerçekleştirilen ilk adımdır. Kullanıcıdan alınan tohum dokümanlar, sisteme paragraf başı " " ve paragraf sonu " " etiketleri ile etiketlenmiş şekilde girdi olarak verilir. BeautifulSoup, HTML veya XML dosyalarını işlemek için oluşturulmuş bir kütüphanedir. Bu kütüphane ile uygulamamıza girdi olarak aldığımız belgeleri HTML kodlarına

yani etiketlenmiş olan "<p>" etiketlerine ayrıştırıp her bir paragrafı ayrı ayrı elde etmemizi sağlamıştır.

4.2. Paragraf Bazında Kelimelerin Tespiti ve İşlenmeye Uygun Hale Getirilmesi

BeautifulSoup kütüphanesi ile dokümanlara ait paragraflar tespit edildikten sonra, paragraflarda yer alan kelimelerin de tespit edilmesi gerekmektedir. Paragrafa ait kelimeler bulunduktan sonra, son olarak bu kelimelerin işlenmeye uygun formata dönüştürülmesi sağlanır. Aşağıda (Şekil 4.4.)'de kelimelerin seçim aşaması adımları yer almaktadır.



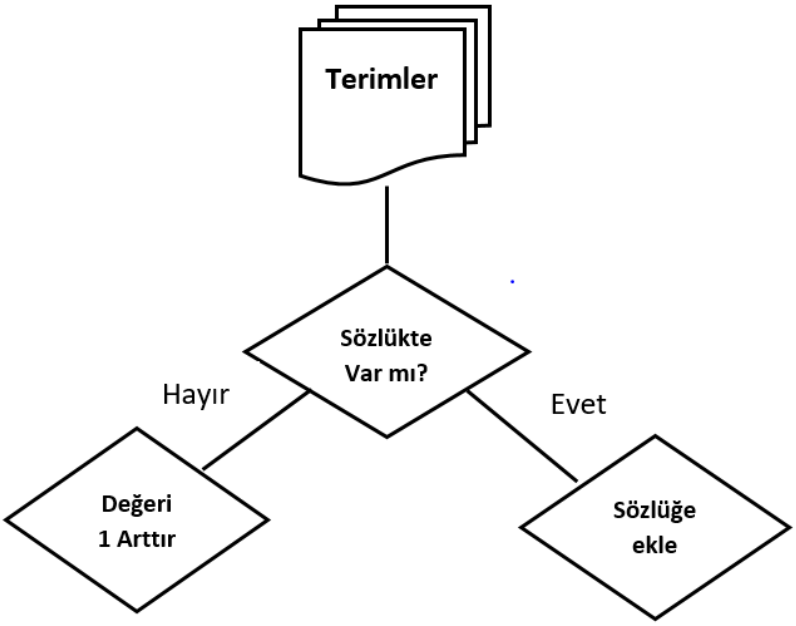
Şekil 4.4. Kelimelerin seçim aşaması adımları

Bir dokümana ait paragraflar döngü içerisine sokulur ve son paragraf işleninceye kadar döngü devam ettirilir. Döngü içerisinde NLTK kütüphanesine ait tokenize metodu ile paragrafın içerdiği kelimeler ayrıştırılır. Daha sonra kelimeler içerisindeki "!, #, $, %" gibi özel karakterler temizlenir. Özel karakterlerden temizlenmiş olan kelimeler NLTK kütüphanesine ait "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. Sadece isim olanları seçmemizin nedeni, Helmholtz Prensibi ve TF-IDF metriklerinde frekans bazlı işlem yaptığımız için, kelimelerin en yalın hali üzerinde işlem yapmamız daha doğru sonuç verecektir.

4.2.1. Paragraf bazında her bir kelimenin geçiş adedinin hesaplanması

Paragraf bazında tespit edilen isim etiketindeki terimlerin geçiş sayılarını tespit etmek amacıyla (Şekil 4.5)'deki basit algoritma kullanılmıştır.



Şekil 4.5. Terim geçiş sayısı hesaplama algoritması

(Şekil 4.5.)'deki "Sözlükte Var mı?" kısmında kontrol edilen değer o an üzerinde işlem yapılan kelimedir. Sözlük olarak kastettiğimiz yapı bir **Dictionary<key,value>** sözlüğüdür. Bu sözlük için anahtar kelime (key) terim, anahtar değeri (value) terimin geçiş sayısı olarak verilmiştir. Döngü içerisinde kontrol edilen kelimenin "student" olduğunu varsayalım. "student" kelimesi

sözlükte var mı diye kontrol edilir. Kelime sözlükte var ise sözlük girdisi Dictionary<student,2> şeklinde olacaktır. Eğer student kelimesi daha önce sözlüğe eklenmemiş olsaydı, sözlük girdisi Dictionary<student,1> şeklinde eklenecekti.

Çizelge 4.1. Sistemdeki terimlerin tutulduğu tablo yapısı

Words (Table)
WordID
DocumentId
Word
Count
StemWord
Paragraph

"student" terimini baz alarak devam edelim. Bu kelimenin 2. Dokümanın 1. paragrafında 1 kez geçtiğini tespit ettikten sonra bu kelimeyi (Çizelge 4.1.)'de verilen Words tablosuna kaydetmemiz gerekiyor. Words tablosundaki "DocumentId" alanı 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. Aşağıda (Çizelge 4.2.)'de "student" kelimesinin Words tablosuna kaydedildikten sonraki hali yer almaktadır.

Çizelge 4.2. Sistemdeki terimlerin tutulduğu tablo kayıt örneği

WordID	**DocumentId**	**Count**	**StemWord**	**ParagraphId**
110	2	1	student	1

4.3. Terimlerin Ağırlıklandırılması

4. bölümün başında da belirtildiği üzere kelimelerin ağırlıklandırılması kısmı hem Helmholtz Prensibi hem de TF-IDF metrikleri ile gerçeklenmiştir. Bu kısımda her 2 ağırlıklandırma süreci detaylı olarak açıklanacaktır.

4.3.1. Helmholtz Prensibi ile kelimelerin ağırlıklandırılması

Paragraf bazında kelimelerin sayıları tespit edilip Words tablosuna kaydedildikten sonra, bu kelimeler içerisinden dokümanı en fazla temsil eden anahtar (anlamlı) kelimelerin tespit edilmesi gerekmektedir. Daha sonra bu anlamlı kelimelerin veritabanı tablosuna kaydedilmesi sağlanır. Helmholtz prensibi uygulanarak elde edilen kelimeler yapısı aşağıda (Çizelge 4.3.)'de verilen tabloya kaydedilir.

Çizelge 4.3. Helmholtz Prensibi uygulanan kelimelerin anlam değerlerinin tutulduğu tablo yapısı

MeaningWord (Table)
DocumentId
StemWord
MeaningValue

(Çizelge 4.3.)'deki yapıya göre "DocumentId" alanı sisteme alınan doküman için üretilmiş Id değeridir. "StemWord" alanı anlam değeri hesaplanan kelimeyi temsil eder. "MeaningValue" alanı ise, "StemWord" alanında belirtilen kelime için hesaplanan anlam değeridir. "MeaningValue" değerini hesaplamak için MsSql içerisinde bulunan "CalculateMeaningValue" isimli saklı yordam (stored procedure) çalıştırılır (Şekil 4.6.).

```
CREATE PROCEDURE [dbo].[CalculateMeaningValue]

AS
BEGIN
  INSERT INTO MeaningWord (DocumentId,
  StemWord,
  MeaningValue)
    SELECT
      DocumentId,
      Stemword,
      CAST(LOG(A * B) AS decimal) / -M AS MV
    FROM (SELECT
      DocumentId,
      StemWord,
      K,
      M,
      CAST((SELECT
        1 / (SELECT
          POWER((SELECT
            CAST((SELECT
              SUM(Count)
            FROM Words)
            AS float) / (SELECT
              SUM(x.TotalWord)
            FROM (SELECT
              SUM(COUNT) AS TotalWord
            FROM Words
            WHERE Paragraph IN (SELECT DISTINCT
              Paragraph
            FROM Words
            WHERE x.StemWord = StemWord
            AND x.DocumentId = DocumentId)
            GROUP BY Paragraph) AS x)), M - 1)))
      AS decimal(18, 2)) A,
      (SELECT
        dbo.Factorial(K) / dbo.Factorial(M) * dbo.Factorial(K - M))
      B
    FROM (SELECT DISTINCT
      DocumentId,
      StemWord,
      (SELECT
        COUNT(Paragraph)
      FROM Words
      WHERE StemWord = w.StemWord
      AND DocumentId = w.DocumentId)
      M,
      (SELECT
        SUM(Count)
      FROM Words
      WHERE StemWord = w.StemWord
      AND DocumentId = w.DocumentId)
      K
    FROM Words w) AS x) AS y
    WHERE A * B > 0
    AND CAST(LOG(A * B) AS decimal) / -M > 0
    ORDER BY DocumentId, MV DESC

END
```

Şekil 4.6. Helmholtz Prensibi ile kelimelerin anlam değerlerini hesaplama sorgusu

Aşağıda (Çizelge 4.4.)'de, (Şekil 4.6.)'da verilen formülün uygulanması sonucu elde edilen anlamlı kelimelerin tutulduğu Database tablosu yer almaktadır.

Çizelge 4.4. Helmholtz Prensibi uygulanan kelimelerin terim ağırlıklarının tutulduğu tablo

DocumentId	StemWord	MeaningValue
1	time	1.6667
1	team	1.6667
1	recept	1.5000
1	goal	1.5000
1	end	1.5000
1	field	1.3333
1	stadium	1.2500
1	year	1.0000
1	week	1.0000
1	steeler	1.0000
1	second	1.0000
1	receiv	1.0000
1	quarter	1.0000
1	play	1.0000
1	packer	1.0000
1	matt	1.0000
1	line	1.0000
1	giant	1.0000
1	career	1.0000

4.3.2 TF-IDF metrikleri ile kelimelerin ağırlıklandırılması

Paragraf bazında kelimelerin sayıları tespit edilip Words tablosuna kaydedildikten sonra, bu kelimeler içerisinden dokümanı en fazla temsil eden

anahtar (anlamlı) kelimelerin tespit edilmesi gerekmektedir. Daha sonra bu anlamlı kelimeleri veritabanı tablosuna kaydedilmesi sağlanır. TF-IDF metrikleri uygulanarak elde edilen kelimeler yapısı aşağıda (Çizelge 4.5.)'de verilen tabloya kaydedilir.

Çizelge 4.5. TF-IDF metrikleri uygulanan kelimelerin anlam değerlerinin tutulduğu tablo yapısı

TFIDFWords (Table)
WordID
DocumentId
Word
TF
IDF
TF_IDF

(Çizelge 4.5.)'deki yapıya göre, WordID alanı tablo birincil anahtar değeridir. "DocumentId" alanı sisteme alınan doküman için üretilmiş Id değeridir. "Word" alanı anlam değeri hesaplanan kelimeyi temsil eder. "TF", "IDF" ve "TF-IDF" alanları ise, "Word" alanında belirtilen kelime için hesaplanan kelime için hesaplanan anlam değerleridir. "TF", "IDF" ve "TF-IDF" değerini hesaplamak aşağıdaki formüller kullanılmaktadır.

$$\mathrm{TF} = \frac{\text{Kelimenin geçiş sayısı toplamı}}{\text{Tüm kelimelerin geçiş sayısı toplamı}} \quad (4.1)$$

$$\mathrm{IDF} = \log\left(\frac{\text{Toplam doküman sayısı}}{\text{Kelimeyi içeren toplam doküman sayısı}}\right) \quad (4.2)$$

$$\mathrm{TF} - \mathrm{IDF} = \mathrm{TF} * \mathrm{IDF} \quad (4.3)$$

Word tablosunda bulunan tüm kelimelerin TF, IDF ve TF-IDF değerlerinin hesaplanması için sisteme yüklenmiş tüm dokümanlara ait birbirinden farklı kaç adet kelime olduğu hesaplanır. Daha sonra bu kelimelerin her biri için bir döngü aracılığıyla toplam geçiş sayıları bulunur. Örnek vermek gerekirse, "football" kelimesi 1. dokümanda 3 kez, 2. dokümanda 5 kez geçmiş olsun. Sonuç olarak "football" kelimesi için geçiş sayısı 3+5=8 olacak şekilde belirlenir. Daha sonra Word tablosuna kaydedilmiş tüm kelimelerin geçiş adetleri toplanır. TF hesaplaması kelime bazında yapılır. "football" kelimesi ile devam edersek, "football" kelimesi için TF değeri, "football" kelimesinin dokümanlarda toplam geçme sayısının, tüm kelimelerin toplam geçiş sayısı toplamının oranına eşittir. "football" kelimesinin toplam 8 kez geçtiğini ve tüm kelimelerin toplam geçiş sayısını 800 olduğunu varsayarsak, TF değeri 0,01 olarak bulunacaktır.

TF değerini hesapladıktan sonra, IDF değerinin hesaplanması gerekiyor. Öncelikle döngü içerisinde her bir kelime için, ilgili kelimeyi içeren doküman sayısı bulunur. Daha sonra sisteme yüklenmiş toplam doküman sayısı hesaplanır. IDF değeri, toplam doküman sayısının, kelimeyi içeren toplam doküman sayısına bölümünün logaritmasına eşittir. "football" kelimesi örneğinden devam edersek, "football" kelimesi toplam 3 adet dokümanda geçmiş olsun. Toplam doküman sayısının 6 olduğunu varsayarsak, IDF değeri $\log_{10}(2) = 0{,}30$ olacaktır. IDF değerini hesaplarken kelimeyi içeren toplam doküman sayısının 0 olması durumunda (sıfıra bölünme hatası) bu değeri 1 alarak "Normalization" işlemi yapılabilir.

TF ve IDF değerleri hesapladıktan sonra, son olarak TF-IDF değerinin hesaplanması gerekir. TF-IDF değeri, TF ve IDF değerlerinin çarpımına eşittir. Yukarıda TF değerini 0,01, IDF değerini 0,30 olarak hesaplamıştık. Bu sonuçlara göre TF-IDF değeri, 0,003 olacaktır.

Aşağıda (Çizelge 4.6.)'da TF-IDF metriklerinin uygulanması sonucu elde edilen anlamlı kelimelerin tutulduğu Database tablosu yer almaktadır.

Çizelge 4.6. TF-IDF metrikleri uygulanan kelimelerin terim ağırlıklarının tutulduğu tablo

WordID	DocumentId	Word	TF	IDF	TF_IDF
185	1	season	0.0422	1	0.0422
228	1	yard	0.0578	0.6931	0.0400
138	1	pass	0.0533	0.6931	0.0370
144	1	player	0.0222	1	0.0222
84	1	game	0.0200	0.6931	0.0139
213	1	touchdown	0.0200	0.6931	0.0139
80	1	footbal	0.0133	1	0.0133
69	1	drug	0.0178	0.6931	0.0123
210	1	time	0.0111	1	0.0111
97	1	intercept	0.0156	0.6931	0.0108
207	1	test	0.0156	0.6931	0.0108
17	1	bay	0.0133	0.6931	0.0092
204	1	team	0.0089	1	0.0089
1	1	aaron	0.0067	1	0.0067
191	1	sport	0.0067	1	0.0067
221	1	week	0.0067	1	0.0067
115	1	lion	0.0089	0.6931	0.0062
154	1	quarterback	0.0089	0.6931	0.0062
162	1	record	0.0089	0.6931	0.0062
187	1	seed	0.0089	0.6931	0.0062
193	1	stadium	0.0089	0.6931	0.0062
217	1	victori	0.0089	0.6931	0.0062
229	1	year	0.0089	0.6931	0.0062
230	1	york	0.0089	0.6931	0.0062
27	1	brown	0.0067	0.6931	0.0046

5. SONUÇ VE ÖNERİLER

Veri kümesi oluşturmak amacıyla, 2-gram olasılıklarına dayalı benzerlik modeli kullanan Python uygulaması yazılmıştır. Helmholtz Prensibi ve TF-IDF metrikleri için ayrı ayrı elde edilen sözlükler için benzerlik oranları hesaplanmış, sonuçlar karşılaştırmalı olarak değerlendirilmiştir.

5.1. Helmholtz Prensibi Uygulanarak Elde Edilen Sonuçlar

Sisteme başlangıç olarak 1,2,3,5 adet şeklinde farklı sayıda spor dokümanı verilmiştir. Bu dokümanlar "txt" türünde sisteme sokulmuştur. Yine aynı şekilde sözlük kelime sayısı azami tüm sözlüklerde 100 adet olacak şekilde sınırlandırılmıştır. Böylece, aynı kelime sayısına sahip sözlüklerin, farklı başlangıç doküman sayılarıyla beslenmesi sonucu elde ettikleri sözlük benzerlik oranı değerleri gözlenmeye çalışılmıştır. Aşağıda sırasıyla bu parametre değerleri için alınan sonuçlar paylaşılmıştır.

Parametre değerleri kısaca şöyle tanımlanabilir;

Web araması için azami sonuç sayısı: Anlamlı kelimeler içinden seçilen kelimeler, Web aramasında kullanılmaktadır. Bu kelimeler ile Web'te arama yapıldığında, oldukça fazla sayıda doküman geldiği için, gelen sonuçları filtrelememiz gerekmektedir. Bu parametre değeri, Web araması sonucu gelen sonuçlardan, istenen adet kadarını almamızı sağlamaktadır.

Sözlük azami kelime sayısı: Oluşturulmak istenen sözlüğün kaç kelimeden oluşması isteniyorsa, bu değer o şekilde verilir.

Sözlüğe eklenecek kelime için kullanılacak benzerlik oranı: Çalışmada, Helmholtz Prensibi ile anlamlı kelimeler bulunmaktadır. Bulunan bu anlamlı kelimeler sözlüğe doğrudan eklenmemekte, sapmaları önlemek üzere WordNet sözlüğü kullanılarak her anlamlı kelimenin oluşmuş sözlük ile benzerliği

hesaplanmaktadır. Benzerlik değerleri, belirli bir eşik değerinden yüksek olan anlamlı kelimeler sözlüğe eklenmektedir. Eşik değer, bu parametre ile sağlanmaktadır.

Sözlüğe ekleme işlemi için kullanılacak anahtar kelime %'si: Web araması sonucu elde edilen dokümanların Helmholtz Prensibi ile anlamlı kelimeleri bulunmaktadır. Anlamlı kelimelerden sadece bu parametrede belirlenen % değeri kadarı sözlüğe ekleme işlemine tabi tutulmaktadır.

Sistem başlangıç olarak 1 adet spor dokümanı ile beslendiğinde, sözlüğün oluşması için kullanılan parametreler ve sözlük benzerlik oranı değeri Çizelge 5.1.'deki gibidir.

Çizelge 5.1. Helmholtz Prensibi uygulanan 100 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Web Araması İçin Azami Sonuç Sayısı	100
Sözlük Azami Kelime Sayısı	100
Sözlüğe Eklenecek Kelime İçin Kullanılacak Benzerlik Oranı	0,1
Sözlüğe Ekleme İşlemi İçin Kullanılacak Anahtar Kelime %'si	50
Başlangıç Doküman Sayısı	1
Sözlük Benzerlik Oranı % (SimHash)	**50,25**
Sözlük Benzerlik Oranı % (WordNet)	**22,77**

Sistem 1 adet başlangıç spor dokümanı verilerek beslendiğinde, 100 kelimelik sözlüğün oluşması sonucu WordNet benzerlik değeriyle % 22,77, SimHash algoritması ile % 50,25 oranında spor (badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmektedir. Bu sözlük, diğer oluşan sözlüklere göre daha az benzerlik oranına sahiptir. Bu sonucun oluşması, birbirine bağlı iki farklı durumdan kaynaklanmış olabilir. Sisteme girdi olarak verilen başlangıç

doküman seti spor konusunda olmasına rağmen, badminton türüne ait kelimeleri az oranda içeriyor olabilir. Buna bağlı olarak Helmholtz Prensibi sonucu elde edilen anlamlı kelimelerinde badminton türüyle ilgisi az olacaktır. Bu durum, Web'ten farklı konularda doküman bulunmasına sebep olacak ve döngü böyle çalışacağı için sistem spor (badminton) türüne ait olmayan kelimeleri sözlüğe ekleyecektir.

Sistem başlangıç olarak 2 adet spor dokümanı ile beslendiğinde, sözlüğün oluşması için kullanılan parametreler ve sözlük benzerlik oranı değeri Çizelge 5.2.'deki gibidir.

Çizelge 5.2. Helmholtz Prensibi uygulanan 100 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Web Araması İçin Azami Sonuç Sayısı	100
Sözlük Azami Kelime Sayısı	100
Sözlüğe Eklenecek Kelime İçin Kullanılacak Benzerlik Oranı	0,1
Sözlüğe Ekleme İşlemi İçin Kullanılacak Anahtar Kelime %'si	50
Başlangıç Doküman Sayısı	2
Sözlük Benzerlik Oranı % (SimHash)	**51,69**
Sözlük Benzerlik Oranı % (WordNet)	**25,87**

Sistem 2 adet başlangıç spor dokümanı verilerek beslendiğinde, 100 kelimelik sözlüğün oluşması sonucu WordNet benzerlik değeriyle % 25,87, SimHash algoritması ile % 51,69 oranında spor (badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmektedir. Çalışmada, sözlüğe eklenecek kelimeler için benzerlik eşik değeri % 0,1 belirlenmiştir. Elde edilen oranın, belirlenen eşik değerinden yüksek olduğu görülmektedir. Aynı şekilde oluşan sözlük, 1 başlangıç dokümanın sisteme beslenmesiyle elde edilen sözlüğe göre daha fazla spor(badminton) verilerini içermektedir. Bu oranı arttırmak için, sisteme girdi

olarak verilen başlangıç doküman seti spor (badminton) konusuyla ilgili olmalıdır. Web araması sonucu gelen dokümanlarda belli kurallar dahilinde sisteme sokulmalıdır.

Sistem başlangıç olarak 3 adet spor dokümanı ile beslendiğinde, sözlüğün oluşması için kullanılan parametreler ve sözlük benzerlik oranı değeri Çizelge 5.3.'deki gibidir.

Çizelge 5.3. Helmholtz Prensibi uygulanan 100 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Web Araması İçin Azami Sonuç Sayısı	100
Sözlük Azami Kelime Sayısı	100
Sözlüğe Eklenecek Kelime İçin Kullanılacak Benzerlik Oranı	0,1
Sözlüğe Ekleme İşlemi İçin Kullanılacak Anahtar Kelime %'si	50
Başlangıç Doküman Sayısı	3
Sözlük Benzerlik Oranı % (SimHash)	**52,50**
Sözlük Benzerlik Oranı % (WordNet)	**39,03**

Sistem 3 adet başlangıç spor dokümanı verilerek beslendiğinde, 100 kelimelik sözlüğün oluşması sonucu WordNet benzerlik değeriyle % 39,03, SimHash algoritması ile % 52,50 oranında spor (badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmektedir. Çalışmada, sözlüğe eklenecek kelimeler için benzerlik eşik değeri % 0,1 belirlenmiştir. Elde edilen oranın, belirlenen eşik değerinden yüksek olduğu görülmektedir. Bu oran aynı zamanda oluşturulan diğer sözlüklere göre en yüksek değerdir. Bu oranın elde edilmesindeki en önemli faktör, hem başlangıç olarak sisteme beslenen doküman seti, hem de Web araması sonucu gelen doküman seti spor (badminton) verilerini iyi oranda içermektedir.

Sistem başlangıç olarak 5 adet spor dokümanı ile beslendiğinde, sözlüğün oluşması için kullanılan parametreler ve sözlük benzerlik oranı değeri Çizelge 5.4.'deki gibidir.

Çizelge 5.4. Helmholtz Prensibi uygulanan 100 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Web Araması İçin Azami Sonuç Sayısı	100
Sözlük Azami Kelime Sayısı	100
Sözlüğe Eklenecek Kelime İçin Kullanılacak Benzerlik Oranı	0,1
Sözlüğe Ekleme İşlemi İçin Kullanılacak Anahtar Kelime %'si	50
Başlangıç Doküman Sayısı	5
Sözlük Benzerlik Oranı % (SimHash)	**52,50**
Sözlük Benzerlik Oranı % (WordNet)	**33,20**

Sistem 5 adet başlangıç spor dokümanı verilerek beslendiğinde, 100 kelimelik sözlüğün oluşması sonucu WordNet benzerlik değeriyle % 33,2, SimHash algoritması ile % 52,50 oranında spor (badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmektedir. Bu sözlüğün, sözlük benzerlik oranı, 1 ve 2 adet başlangıç dokümanıyla oluşturulan sözlüklerden yüksek, 3 adet başlangıç dokümanıyla oluşturulan sözlükten ise düşüktür. Başlangıç doküman sayısı 3 doküman üzerine çıktığı zaman, anlamlı kelime sayısı artmakta ve bu anlamlı kelimelerin % 50'si (**Sözlüğe Ekleme İşlemi İçin Kullanılacak Anahtar Kelime %'si** parametre değerine bağlı olarak, bu oran değişebilmektedir.) sözlüğe eklendiği için sözlük ilişkisiz kelimeleri de içermiş olacaktır. Bu durum, Web araması sonucu elde edilen dokümanların farklı türlerde olmasına neden olmaktadır.

5.2. TF-IDF Metrikleri Uygulanarak Elde Edilen Sonuçlar

TF-IDF metrikleri ile anahtar kelime tespiti yapıldıktan sonra, bu anahtar kelimelerin sözlüğe eklenmesi ve Web aramasında kullanılması sırasında anlam değeri belli bir değerin altında olan kelimeler ile işlem yapıldığında sapmalara neden olduğunu gözlemledik. Yaptığımız deneysel sonuçlara göre anahtar kelimeler içerisinden anlam değeri 0,3'ten büyük olan kelimeler ile işlem yapıldığında daha iyi sonuçlar elde edildiği görülmüştür. Bu nedenle TF-IDF metrikleri ile yaptığımız deneylerde anlam değeri 0,3'ten büyük olan kelimeler ile işlem yapılmıştır. Anlam değeri 0,3'ten küçük kelimeleri sözlüğe ekleyince elde edilen sözlükler de ayrıca paylaşılacak ve sonuçlar karşılaştırmalı olarak değerlendirilecektir.

Tüm sözlükler için, sisteme başlangıç olarak 1 adet spor dokümanı verilmiştir. Bu doküman "txt" türünde sisteme sokulmuştur. Yine aynı şekilde sözlük kelime sayısı azami 25, 50 ve 100 adet olacak şekilde sınırlandırılmıştır. Böylece, farklı kelime sayısına sahip sözlüklerin, aynı başlangıç doküman sayılarıyla beslenmesi sonucu elde ettikleri sözlük benzerlik oranı değerleri gözlenmeye çalışılmıştır. Aşağıda sırasıyla bu parametre değerleri için alınan sonuçlar paylaşılmıştır.

5.2.1. TF-IDF anlam değeri 0,3 ve üzerinde olan kelimelerin sözlüğe eklenmesi

Yukarıda bölüm 5.2'de de bahsedildiği üzere TF-IDF anlam değeri 0,3 ve üzerinde olan kelimeleri sözlüğe ekleyerek işlem yapıldığında başarı oranı yüksek sözlükler elde edilmektedir. Aşağıda sırasıyla elde edilen sözlükler ve başarı oranları paylaşılacaktır.

Aşağıda Çizelge 5.5.'de, sisteme başlangıç olarak verilen 1 adet spor dokümanı ile elde edilen 25 kelimelik sözlüğün genel benzerlik oranı yer almaktadır.

Çizelge 5.5. TF-IDF anlam değeri 0,3 uygulanan 25 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Kelimeyi Sözlüğe Eklemek İçin Kullanılacak TF-IDF Anlam Değeri	0,3
Sözlük Azami Kelime Sayısı	25
Başlangıç Doküman Sayısı	1
Sözlük Benzerlik Oranı % (SimHash)	**75,2**
Sözlük Benzerlik Oranı % (WordNet)	**40,02**

Sisteme başlangıç olarak 1 adet spor dokümanı verilip ve sözlüğe ekleme kuralı olarak TF-IDF metrikleri hesaplaması sonucu anlam değeri 0,3 olan kelimelerin eklenmesi ile 25 kelimelik sözlük elde edilmiştir. Elde edilen bu sözlüğün genel başarı oranı, Hash Similarity yöntemiyle hesaplandığında % 75,2, WordNet benzerlik değeriyle hesaplandığında ise % 40,02 oranında spor(badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmiştir. Bu başarı oranı 50 ve 100 kelimelik diğer 2 sözlüğün başarı oranından daha yüksektir. Bu başarı oranının elde edilmesindeki en önemli faktör, başlangıç olarak sisteme beslenen dokümanın, oluşturulmak istenen sözlüğün konusuyla ilgili olmasıdır. Aynı zamanda TF-IDF değeri 0,3 üzerinde olan kelimelerinin sözlüğe eklenmesi sapmaları oldukça önlemiştir. Aşağıda Çizelge 5.6.'da elde edilen sözlüğe ait kelimeler yer almaktadır.

Çizelge 5.6. TF-IDF metrikleri uygulanarak elde edilen 25 kelimelik sözlük

feder	medal	Metr	racket	injuri
olymp	sport	Peopl	shop	player
game	committe	Entri	sear	court

badminton	bwf	Event	set	shoe
exercis	world	Debut	academi	select

Aşağıda Çizelge 5.7.'de, sisteme başlangıç olarak verilen 1 adet spor dokümanı ile elde edilen 50 kelimelik sözlüğün genel benzerlik oranı yer almaktadır.

Çizelge 5.7. TF-IDF anlam değeri 0,3 uygulanan 50 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Kelimeyi Sözlüğe Eklemek İçin Kullanılacak TF-IDF Anlam Değeri	0,3
Sözlük Azami Kelime Sayısı	50
Başlangıç Doküman Sayısı	1
Sözlük Benzerlik Oranı % (SimHash)	**73,8**
Sözlük Benzerlik Oranı % (WordNet)	**36,67**

Sisteme başlangıç olarak 1 adet spor dokümanı verilip ve sözlüğe ekleme kuralı olarak TF-IDF metrikleri hesaplaması sonucu anlam değeri 0,3 olan kelimelerin eklenmesi ile 50 kelimelik sözlük elde edilmiştir. Elde edilen bu sözlüğün genel başarı oranı, Hash Similarity yöntemiyle hesaplandığında % 73,8, WordNet benzerlik değeriyle hesaplandığında ise % 36,67 oranında spor(badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmiştir. Bu başarı oranı, 25 kelimelik sözlüğün başarı oranından daha düşüktür. Bu sözlüğün sözlük benzerlik oranının, 25 kelimelik sözlükten daha düşük olması, sözlük kelime sayısının artışına bağlı olarak, sözlüğe spor(badminton) konusuyla daha az ilişkili kelimelerin eklenmesi ve Web arama işleminin bu kelimeler üzerinden yapılması sonucu sözlükte sapmaların oluşmasından kaynaklanmaktadır. Aşağıda Çizelge 5.8.'de elde edilen sözlüğe ait kelimeler yer almaktadır.

Çizelge 5.8. TF-IDF metrikleri uygulanarak elde edilen 50 kelimelik sözlük

wrist	shoe	Academi	arcadia	cutter
injuri	select	Metr	school	cooki
player	polici	Countri	enjoy	entri
committe	box	Leagu	techniqu	event
bwf	member	Excel	servic	debut
world	league	Equip	ship	medal
feder	dimens	Compani	postag	sport
olymp	court	Brand	shop	dark
game	tenni	Racket	sear	blais
badminton	ligament	Yonex	set	wayn

Aşağıda Çizelge 5.9.'da, sisteme başlangıç olarak verilen 1 adet spor dokümanı ile elde edilen 100 kelimelik sözlüğün genel benzerlik oranı yer almaktadır.

Çizelge 5.9. TF-IDF anlam değeri 0,3 uygulanan 100 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Kelimeyi Sözlüğe Eklemek İçin Kullanılacak TF-IDF Anlam Değeri	0,3
Sözlük Azami Kelime Sayısı	100
Başlangıç Doküman Sayısı	1
Sözlük Benzerlik Oranı % (SimHash)	**68,7**
Sözlük Benzerlik Oranı % (WordNet)	**33,96**

Sisteme başlangıç olarak 1 adet spor dokümanı verilip ve sözlüğe ekleme kuralı olarak TF-IDF metrikleri hesaplaması sonucu anlam değeri 0,3 olan kelimelerin eklenmesi ile 100 kelimelik sözlük elde edilmiştir. Elde edilen bu sözlüğün genel başarı oranı, Hash Similarity yöntemiyle hesaplandığında % 68,7, WordNet benzerlik değeriyle hesaplandığında ise % 33,96 oranında spor(badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmiştir. Bu başarı oranı 25 ve 50 kelimelik 2 sözlüğün başarı oranından daha düşüktür. Bu sonuçla birlikte şöyle bir çıkarım yapabiliriz. Sözlük azami kelime sayısı parametre değeri arttırıldığında, sözlükte belli bir kelime sayısından sonra sapmalar meydana gelmektedir. Sapmaya neden olan faktör ise, hatalı bir kelimenin sözlüğe eklenmesi sonrası Web aramasının bu kelime üzerinden yapılıyor olmasıdır. Sözlükte sapmaların önüne geçebilmek için, sözlük belli periyotlarda kontrol edeilmeli ve başlangıç olarak verilen doküman baz alınarak tekrardan bu doküman konusuyla ilgili kelimelerin sözlüğe eklenmesi sağlanmalıdır. Aşağıda Çizelge 5.10.'da elde edilen sözlüğe ait kelimeler yer almaktadır.

Çizelge 5.10. TF-IDF metrikleri uygulanarak elde edilen 100 kelimelik sözlük

question	racket	Box	languag	inform
use	yonex	Member	sign	compani
tournament	video	Group	fast-paced	brand
season	backhand	Wib	skill	merg
har	dan	championship	program	bird
ingen	lin	Canada	experi	birdi
och	match	Alberta	job	shuttlecock
vatn	fact	Address	card	basic
knee	ntrue	Shoe	member	coach
pain	answer	Select	player	point
ligament	debut	Sear	associ	weight
wrist	medal	Set	singapor	fat
injuri	sport	Academi	partner	build
player	committe	Metr	format	club
court	bwf	Year	doubl	guest
read	world	İmag	countri	hotel
cutter	feder	Draw	leagu	thesen

cooki	olymp	Team	excel	techniqu
entri	game	Procedur	equip	oppon
event	badminton	Polici	shop	servic

5.2.2. TF-IDF anlam değeri 0,3 altında olan kelimelerin sözlüğe eklenmesi

Yukarıda da bahsedildiği üzere TF-IDF metrikleri sonucu anlam değeri belirlenen kelimelerden, anlam değeri 0,3 ve üzeri kelimeler sözlüğe eklendiğinde yüksek başarı oranları elde edilirken, anlam değeri 0,3 altında olan kelimelerin eklenmesi sonucu daha düşük başarı oranları elde edilmektedir. Bu bölümde 25 ve 50 kelimelik 2 sözlük oluşturulmuş ve sözlüğe ekleme kuralı olarak TF-IDF anlam değeri 0,2 olarak belirlenmiştir.

Aşağıda Çizelge 5.11.'de, sisteme başlangıç olarak verilen 1 adet spor dokümanı ile elde edilen 25 kelimelik sözlüğün genel benzerlik oranı yer almaktadır.

Çizelge 5.11. TF-IDF anlam değeri 0,2 uygulanan 25 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Kelimeyi Sözlüğe Eklemek İçin Kullanılacak TF-IDF Anlam Değeri	0,2
Sözlük Azami Kelime Sayısı	25
Başlangıç Doküman Sayısı	1
Sözlük Benzerlik Oranı % (SimHash)	**62,5**
Sözlük Benzerlik Oranı % (WordNet)	**31,32**

Sisteme başlangıç olarak 1 adet spor dokümanı verilip ve sözlüğe ekleme kuralı olarak, TF-IDF metrikleri hesaplaması sonucu anlam değeri 0,2 olan kelimelerin eklenmesi ile 25 kelimelik sözlük elde edilmiştir. Elde edilen bu sözlüğün genel başarı oranı, Hash Similarity yöntemiyle hesaplandığında % 62,5, WordNet benzerlik değeriyle hesaplandığında ise % 31,32 oranında spor(badminton)

konusu ile ilgili kelimeler içerdiği sonucu elde edilmiştir. Bu başarı oranı TF-IDF metrikleri hesaplaması sonucu anlam değeri 0,3 olan kelimelerin sözlüğe eklenmesi sonucu elde edilen tüm sözlüklerin başarı oranından daha düşüktür. TF-IDF anlam değeri düşük olan kelimelerin sözlüğe eklenmesi sözlükte sapmalara neden olduğu görülmektedir. Aşağıda Çizelge 5.12.'de elde edilen sözlüğe ait kelimeler yer almaktadır.

Çizelge 5.12. TF-IDF metrikleri uygulanarak elde edilen 25 kelimelik sözlük

badminton	court	Feder	olymp	sport
bwf	day	Game	player	state
citi	di	Languag	point	street
coach	dimens	Metr	quarterfin	tenni
committe	exercis	Nn	sign	world

Aşağıda Çizelge 5.13.'de, sisteme başlangıç olarak verilen 1 adet spor dokümanı ile elde edilen 50 kelimelik sözlüğün genel benzerlik oranı yer almaktadır.

Çizelge 5.13. TF-IDF anlam değeri 0,2 uygulanan 50 kelimelik sözlük parametreleri ve genel benzerlik oranı

Parametre Adı	Parametre Değeri
Kelimeyi Sözlüğe Eklemek İçin Kullanılacak TF-IDF Anlam Değeri	0,2
Sözlük Azami Kelime Sayısı	50
Başlangıç Doküman Sayısı	1
Sözlük Benzerlik Oranı % (SimHash)	**59,2**
Sözlük Benzerlik Oranı % (WordNet)	**31,60**

Sisteme başlangıç olarak 1 adet spor dokümanı verilip ve sözlüğe ekleme kuralı olarak TF-IDF metrikleri hesaplaması sonucu anlam değeri 0,2 olan kelimelerin eklenmesi ile 50 kelimelik sözlük elde edilmiştir. Elde edilen bu sözlüğün genel

başarı oranı, Hash Similarity yöntemiyle hesaplandığında % 59,2, WordNet benzerlik değeriyle hesaplandığında ise % 31,60 oranında spor(badminton) konusu ile ilgili kelimeler içerdiği sonucu elde edilmiştir. Bu başarı oranı 25 kelimelik sözlüğün başarı oranından düşüktür. Bu durumun nedenleri, hem TF-IDF anlam değeri 0,3'ten küçük kelimeler ile işlem yapılması, hem de sözlük kelime sayısının daha fazla olması kaynaklı, sözlükte sapmaların oluşmasıdır. Sapmaları önlemek için, sözlük belli periyotlarda kontrol edilmeli ve amaca yönelik dokümanların Web aramasından elde edilmesi sağlanmalıdır. Aşağıda Çizelge 5.14.'de elde edilen sözlüğe ait kelimeler yer almaktadır.

Çizelge 5.14. TF-IDF metrikleri uygulanarak elde edilen 50 kelimelik sözlük

assist	champion	Play	event	world
racket	descript	Cup	feder	bwf
shoe	book	Year	olymp	court
select	read	Klcc	medal	feedback
polici	cutter	associ	sport	set
box	cooki	singapor	point	academi
member	tournament	group	servic	metr
nn	competit	draw	side	badminton
list	editor	team	player	game
championship	articl	entri	committe	tenni

Elde edilen sonuçlar incelendiğinde, Helmholtz Prensibi ile anlamlı kelime tespiti

yapılıp oluşturulan sözlüklerde % 39,03 sözlük benzerlik oranı elde edilirken, TF-IDF metrikleri ile anlamlı kelime tespiti sonucu oluşturulan sözlükler de ise, %75,2 sözlük benzerlik oranı sağlanmıştır. TF-IDF metrikleri kullanılarak oluşturulan sözlüğün genel başarısının, Helmholtz Prensibi kullanılarak elde edilen sözlükten daha yüksek olmasının farklı nedenleri bulunmaktadır. Bunlar, Helmholtz Prensibi ile oluşturulan sözlükler TF-IDF metrikleri ile oluşturulan sözlüklere göre daha fazla sayıda başlangıç dokümanları ile işlem yapmıştır. Bu

durum oluşturulmak istenen sözlüğe ait anlamlı kelimelerin net olarak belirlenmesini engellemektedir. Bu durumun sonucu olarak da, sözlükte oluşturulmak istenen sözlüğün konusunun dışında kelimelerin oluşmasına neden olmaktadır. Bunun dışında, Helmholtz Prensibi ile oluşturulan sözlüklerin sözlük kelime sayısı, TF-IDF metrikleri ile oluşturulan sözlüklerin kelime sayısından daha fazla olduğu görülmektedir. Böyle sözlük kelime sayısı belli bir değerin üzerinde olan sözlük oluşumlarında, sözlük belli periyotlarda kontrol edilmeli ve sapmaların önlenmesi gerekmektedir. Helmholtz Prensibi ile oluşturulan sözlüklerin düşük başarı oranı elde etmesinin bir diğer nedeni ise, Web araması sonucu birden fazla doküman elde edilmesi ve daha sonra bu dokümanlar arasından anlamlı kelime tespiti yapılmasıdır. Bu durumun sonucu olarak da, ilişkisiz kelimeler sözlüğe eklenecektir. TF-IDF metrikleri ile oluşturulan sözlüklerin başarı oranlarını incelediğimizde, genel olarak yüksek başarı elde edilmesine rağmen, sözlük kelime sayısı az olan sözlüklerde bu oran daha yüksektir. Bu durumun nedeni, sözlük kelime sayısı az olduğu için sözlükte daha az sapma oluşmuştur. Bir diğer nokta, TF-IDF metrikleri ile oluşturulan sözlükler Web araması sonucu sürekli tek bir doküman üzerinden işlem yapmıştır. Örneğin sözlüğe eklenen kelimeler “football” ve “stadium” ise bu kelimeler ile “football stadium” şeklinde Web araması yapılıp elde edilen dokümanlardan sadece 1 tanesi ön işleme adımlarına sokulup anlamlı kelimeleri tespit edilmektedir. Buna bağlı olarak sadece amaca yönelik doküman üzerinde işlem yapılmakta ve bu durum başarı oranı yüksek sözlüklerin elde edilmesini sağlamaktadır.

Benzer çalışmalar ile kıyaslandığında, yapılan çalışmanın ortalama sonuçlar, sözlük oluşturma ve büyüme oranı bakımından oldukça başarılı olduğu görülmektedir. Literatür taraması kısmında bahsedilen benzer (Vijay vd., 2018) çalışması ile kıyaslandığında, her 2 çalışmada otomatik sözlük oluşturma işleminde başarılı sonuç elde etmiştir. Ancak (Vijay vd., 2018) çalışmasından farklı olarak, bu çalışmaya Web araması kısmı da dâhil edilmiştir. Bu çalışma, anlık ve güncel veriler üzerinden çalışması bakımından (Vijay vd., 2018) çalışmasından değerlidir. Bununla birlikte bu çalışmada WordNet benzerlik

hesaplaması kullanılmış, sözlüğe eklenecek kelimelerin gerçekten oluşturulmak istenen konu ile ilgili olup olmadığı belirlenmiştir.

Literatüre bundan sonraki süreçlerde katkı sağlayacağı düşünülen aşağıdaki çalışmalar ele alınabilir.

- Mevcut çalışmanın yapısına yapay sinir ağı eklenerek, yapay sinir ağının öğrenme ile sözlük oluşturma başarısı gözlemlenebilir.
- Mevcut sistem sürekli bir büyüme içerisinde çalışacağından sistemin en iyi sonuç verdiği iterasyon sayısı tespit edilip, belirlenen en uygun aralıklar içerisinde sistemin çalışması sağlanabilir.
- Mevcut çalışma kapsamında başlangıç olarak spor dokümanları kullanılmış ve sözlüğün spor konusu ile ilişkili kelimeler barındırmasına göre benzerlik hesaplaması yapılmıştır. Farklı konularda başarım oranlarını tespit etmek amacıyla, farklı çalışma türlerinde sözlükler oluşturulabilir. Sonrasında ortaya çıkan sonuçlar karşılaştırmalı olarak incelenebilir.
- Sözlüğe eklenecek kelimelerin mevcut sözlükteki kelimeler ile arasındaki benzerlik oranı hesaplamasında, WordNet benzerlik yöntemi kullanılmıştır. WordNet benzerlik sonucuna göre kelimenin sözlüğe eklenip eklenmeyeceğine karar verilmiştir. Bundan sonraki çalışmalarda farklı benzerlik hesaplaması teknikleri kullanılarak oluşan sözlüklerin yapısı incelenebilir. Örneğin Word2vec benzerlik yöntemi ve WordNet benzerlik yöntemi için ayrı ayrı çalışmalar yapılabilir. Ardından ortaya çıkan iki sözlüğün yapısal farklılıkları ortaya koyulabilir.
- Sözlüğe eklenecek kelimeyi belirledikten sonra, bu kelime ile birlikte, bu kelimenin tüm eş anlamlı kelimeleri de sözlüğe eklenebilir. Daha sonra oluşan sözlüğün benzerlik oranı SimHash algoritması hesaplanıp, sonuçlar karşılaştırılabilir.

KAYNAKLAR

Aktaş, Y., İnce, E.Y., Çakır, A., Kutlu, A., 2016. Wordnet ve Bilgisayar Ağ Terimleri Sözlüğünün Oluşturulması. Akademik Bilişim 2016, Adnan Menderes Üniversitesi, 30 Ocak- 5 Şubat, Aydın, 1-5.

Balinsky, H., Balinsky, A., Simske, S., 2011. Document sentences as a small world. 2011 IEEE International Conference on Systems Man and Cybernetics (SMC), 9-12 October, Anchorage, ABD, 2583-2588.

Bertin, M., Atanassova, I., 2018. InTeReC: In-text Reference Corpus for Applying Natural Language Processing to Bibliometrics. 7th International Workshop on Bibliometric-enhanced Information Retrieval (BIR 2018) to be held as part of the 40th European Conference on Information Retrieval (ECIR), 26 March, Grenoble, France, 54-62.

Bilgisayar Kavramları, 2012. TF-IDF. Erişim Tarihi: 16.07.2019. http://bilgisayarkavramlari.sadievrenseker.com/2012/10/22/tf-idf/

Caldera, C., Berndt, R., Eggeling, E., Schröttner, M., Fellner, D.W., 2014. PRIMA-Towards an Automatic Review / Paper Matching Score Calculation. The Sixth International Conference on Creative Content Technologies (CONTENT 2014), 25-29 May, Venice, Italy, 70-75.

Chitraa, V., Dr. Davamani, A. S., 2010. A Survey on Preprocessing Methods for Web Usage Data. International Journal of Computer Science and Information Security, 7(3), 78-83.

Dadachev, B., Balinsky, A., Balinsky, H., Simske, S., 2012. On the helmholtz principle for data mining. IEEE In Emerging Security Technologies (EST), 2012 Third International Conference, 5-7 September, Lisbon, 99-102.

Desolneux, A., Moisan, L., Morel, J. M., 2001. Edge Detection by Helmholtz Principle. Journal of Mathematical Imaging and Vision, 14, 271-284.

Desolneux, A., Moisan, L., Morel, J. M., 2007. From Gestalt Theory to İmage Analysis: A Probabilistic Approach. Springer Science & Business Media, 276, New York.

Ellen, R., 1993. Automatically Constructing a Dictionary for Information Extraction Tasks. Proceedings of the Eleventh National Conference on Artificial Intelligence, AAAI Press / MIT Press, USA, 811–816.

Famili, A., Shen, W., Weber, R., Simoudis, E., 1997. Data Preprocessing and Intelligent Data Analysis. Intell Data Anal, 1(4), 3-23.

Jiang, Q., Sun, M., 2011. Semi-Supervised SimHash for Efficient Document Similarity Search. The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 19-24 June, Portland, Oregon, USA, 93-101.

Kepuska, V. Z., 2011. Rojanasthien, P., Speech Corpus Generation from DVDs of Movies and TV Series. Journal of International Technology and Information Management, 20(1), 49-82.

Khoury, R., Shi, L., Hamou-Lhadj, A., 2016. Key Elements Extraction and Traces Comprehension Using Gestalt Theory and the Helmholtz Principle. 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), 2-7 October, Raleigh, NC, USA, 478-482.

Koeva, S., Stoyanova, I., Todorova, M., Leseva, S., 2016. Semi-automatic Compilation of the Dictionary of Bulgarian Multiword Expressions. Proceedings of GLOBALEX 2016: Lexicographic Resources for Human Language Technology, Workshop at LREC2016, 24 May, Portorož, Slovenia. 86-95.

Ho, T., Sung, Kim., 2014. Fingerprint-Based Near-Duplicate Document Detection with Applications to SNS Spam Detection. International Journal of Distributed Sensor Networks, 10, 1-8.

Medium Corporation, 2015. TF-IDF (Term Frequency—Inverse Document Frequency). Erişim Tarihi: 16.07.2019. https://medium.com/algorithms-data-structures/tf-idf-term-frequency-inverse-document-frequency-53feb22a17c6

Medium Corporation, 2018. Wordnet nedir. Erişim Tarihi: 16.07.2019. https://medium.com/5bayt/wordnet-nedir-6910c6f98837

Miller, G.A., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.J., 1990. Introduction to WordNet: An On-line Lexical Database. International Journal of Lexicography, 3(4), 235–244.

Moghaddas, B.B., Kahani, M., Toosi, S.A., Pourmasoumi, A., Estiri, A., 2013. Pasokh: A standard corpus for the evaluation of Persian text summarizers. ICCKE 2013, 31 October -1 November,Mashhad, Iran, 471-475.

Moral, C., Antonio, A., Imbert, R., Ramírez, J., 2014. A Survey Of Stemming Algorithms İn İnformation Retrieval. Information Research: An International Electronic Journal, 19(1). 1-22.

Oğuzlar, A., 2003. Veri Ön İşleme, Erciyes Üniversitesi İktisadi ve İdari Bilimler Fakültesi Dergisi, 21, 67-76.

Pi, B., Fu, S., Wang, W., Han, S., Inc, R., China, P., 2019. SimHash-based Effective and Efficient Detecting of Near-Duplicate Short Messages. Proceedings of the Second Symposium International Computer Science and Computational Technology(ISCSCT '09), 26-28 December, Huangshan, P. R. China, 20-25.

Qaiser, S., Ali, R., 2018. Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. International Journal of Computer Applications, 181(1), 25-29.

Senemoğlu, N., 2012. Gelişim, öğrenim ve öğretim. Pegem, 35, Ankara.

Sherpa, 2019. Gestalt İlkeleri Nedir. Erişim Tarihi: 16.07.2019. https://sherpa.blog/sozluk/gestalt-ilkeleri-nedir

Silverman, K.E., Anderson, V., Bellegarda, J.R, Lenzo, K.A., Naik, D., 1999. Design and collection of a corpus of polyphones and prosodic contexts for speech synthesis research and development. 6th European Conference on Speech Communication and Technology (EUROSPEECH'99), 5-9 September, Budapest, Hungary, 1-4.

Tanasa, D., Trousse, B., 2004. Advanced data preprocessing for intersites web usage mining, IEEE Intelligent Systems, 19(2), 59–65.

Turan, M., Ögtelik, S., 2018. İngilizce Dokümanlarda Tema ve Alt Kavramlar Tespit Modeli. DergiPark Düzce Üniversitesi Bilim ve Teknoloji Dergisi, 6(4), 754-764.

Turan, M., Sönmez, C., 2015. Automatize Document Topic and Subtopic Detection with Support of a Corpus. Procedia - Social and Behavioral Sciences, 177, 169 – 177.

Vijay, D., Bohra, A., Singh, V., Akhtar, S.S., Shrivastava, M., 2018. Corpus Creation and Emotion Prediction for Hindi-English Code-Mixed Social Media Text. Proceedings of NAACL-HLT 2018: Student Research Workshop, 2-4 June, New Orleans, Louisiana, 128-135.

Vorapatratorn, S., Suchato, A., Punyabukkana, P., 2012. Automatic online text selection for constructing text corpus with custom phonetic distribution. Ninth International Conference on Computer Science and Software Engineering (JCSSE), 30 May- 1 June, Bangkok, 6-11.

Yeşilyaprak, B., 2005. Gestalt ve insancıl yaklaşımında öğrenme. Pegem, 434 Ankara.

Zeren, G., 2007. Gestalt kuramı. Anı Yayıncılık, 527, İstanbul.

Zhou, Z., 2002. Three Perspectives of Data Mining. Artificial Intelligence, 143(1), 139-146.

EKLER

EK A. Kodlar

EK B. ER Diagramı

EK A. Kodlar

```
import EducationData
import WebSearch
import os
import time
import pypyodbc
import colorama
from colorama import Fore, Back, Style


colorama.init()

connection = pypyodbc.connect('Driver={SQL Server};'
                                                'Server=NB-AT012337;'
                                                 'Database=SmallWordsEducation;'
                                                   'uid=PhytonThesisUser;pwd=1')



cursor = connection.cursor()

thresholdEnglishDictionaryCountForStopProject=50



def CountDirectoryItems():
        path="C:\\Users\AT012337\\Thesis\Konu Tespiti\\SmallWord Eğitim
Verileri\\"
        list = os.listdir(path)
        number_files = len(list)
        if number_files==0:
             print("Not found.Please fill path with .txt files")
        return number_files



def GetWordCountOfEnglishDictionary():
    #EducationData.LastCheckEnglishDictionary()
    cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] (nolock)")
    result_set = cursor.fetchall()
    number_of_rows_EnglishDictionary = result_set[0][0]
    return number_of_rows_EnglishDictionary


def RunProjectCountOfEnglishDictionary():

             while(GetWordCountOfEnglishDictionary()<thresholdEnglishDictionary
             CountForStopProject):
                     EducationData.RunEducationDataProject()
                     WebSearch.WebSearch()
             EducationData.findTheSuccessOfTheProject()

RunProjectCountOfEnglishDictionary()
```

```
from bs4 import BeautifulSoup
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn
from nltk.tokenize import word_tokenize
from nltk import pos_tag
import nltk.data
import pypyodbc
import re
import os
import nltk
import string
from string import punctuation
import WordnetSimilarity
import SequenceMatcherSimilarity
from nltk.corpus import wordnet
from nltk.corpus import words
import WebSearch
import datetime
from autocorrect import spell
import math
from nltk import wordpunct_tokenize
import colorama
from colorama import Fore, Back, Style


colorama.init()



connection = pypyodbc.connect('Driver={SQL Server};'
                              'Server=NB-AT012337;'
                              'Database=SmallWordsEducation;'
                              'uid=PhytonThesisUser;pwd=1')

cursor = connection.cursor()

tallyTableCount=10000
RemovingfolderPath= "C:\\Users\AT012337\\Thesis\\Konu Tespiti\\SmallWord Eğitim
Verileri\\"
min_MeaningValue=0.3
max_MeaningValue=0.7
max_TopCount=5
meaningWords_TopCount=10
thresholdRateForSimilarity=0.3
sourceTopic="education"
useWordnetSimilarity=1
percentageOfSelectingMeaningWord=10
backupPath="C:\\Users\AT012337\\Thesis\\Konu Tespiti\\SmallWord Eğitim Verileri
Backup\\"
topCountForBackupPath=3
backupWebSearchWordlist = ["football", "sport", "spor", "ronaldinho", "ball",
"faul","spor","foot","captain","el classicco"]
count_Paragraph_Threshold=100
sum_Count_Threshold=100
thresholdCountOfEnglishDictionaryInsert=20
thresholdCountOfSelectingWord=300
parameterMeaningValueCountIfListEmpty=2
thresholdTF_IDFValue=0.03
```

```
topCountTFIDF=15
thresholdEnglishDictionaryTempTableInsert=2
thresholdWordCountPerDocument=20

def TruncateSelectedDatabase():
    SQLCommand = ("EXEC TruncateSelectedDatabase")
    cursor.execute(SQLCommand)
    connection.commit()



def InsertTallyTable():
    SQLCommand = ("EXEC InsertTally ?")
    Values = [tallyTableCount]
    cursor.execute(SQLCommand,Values)
    connection.commit()
    print(Fore.MAGENTA + 'Tally tablosu dolduruldu.')



def TruncateAllDatabase():
    SQLCommand = ("EXEC TruncateAllDatabase")
    cursor.execute(SQLCommand)
    connection.commit()
    print(Fore.CYAN + 'Tüm tablolar Truncate edildi.')




def findTheSuccessOfTheProject():
    totalSimilarityRate=GetTotalSimilarityRate()
         print(Fore.LIGHTBLUE_EX +"Projenin başarı oranı % :
     ",totalSimilarityRate*100)







def CheckIsFirstDocument():
    cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[SearchedDocuments] (nolock)")
    result_set = cursor.fetchall()
    number_of_rows_SearchedDocuments = result_set[0][0]
    if number_of_rows_SearchedDocuments==0:
        return True
    else:
        return False
```

```
def CheckSimilarityAndSaveDB2():
    #cursor.execute("SELECT Word FROM [SmallWordsEducation].[dbo].[TFIDFWords]
(nolock)")
    #tfIDFWordList = cursor.fetchall()
    #spelledTFIDFWordList=SpellWord(tfIDFWordList)
    #LastCheckMeaningWords(spelledTFIDFWordList,True)
    maxLevelMeaningWord=GetMaxLevelMeaningWord()
    originalWord=GetOriginalWord(maxLevelMeaningWord)
    isFirstDocument=CheckIsFirstDocument()

    #if isFirstDocument==False:
    if len(maxLevelMeaningWord)>0:
        TruncateEnglishDictionaryTempTable()
        cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] (nolock) where
Word=?",[maxLevelMeaningWord])
        result_set = cursor.fetchall()
        number_of_rows_MeaningWord = result_set[0][0]
        if number_of_rows_MeaningWord ==0:
            InsertEnglishDictionary(maxLevelMeaningWord)
            #synonymsWordList=FindSynonymsWordsGivenParameterWord(originalWord)
            #if len(synonymsWordList)>0:
            #    i=0
            #    for synonymsWord in synonymsWordList:
            #        if i<2:
            #            InsertEnglishDictionary(synonymsWord)
            #            print(synonymsWord +" kelimesi " + maxLevelMeaningWord
+" kelimesi ile benzer olduğu için sözlüğe eklendi.")
            #            i+=1
        else:
            print(Fore.RED + maxLevelMeaningWord +" kelimesi sözlükte olduğu
için tekrar eklenmedi.")




    else:
        cursor.execute("EXEC GetEnglishDictionaryTemp")
        englishDictionaryTempWord= cursor.fetchall()
        if len(englishDictionaryTempWord)>0:

synonymsWordList=FindSynonymsWordsGivenParameterWord(englishDictionaryTempWord[
0][0])
            if len(synonymsWordList)>0:
                InsertEnglishDictionaryTemp(synonymsWordList[0])
                InsertOriginalWords(synonymsWordList[0],synonymsWordList[0])
                print(Fore.RED + "TFIDFWords tablosundaki hiç bir kelimenin TF
değeri "+ str(thresholdTF_IDFValue)+" değerinden yüksek olmadığı için "+
englishDictionaryTempWord[0][0]+ " kelimesinin synonyms kelimesi temp sözlüğe
eklendi ve tekrar web araması yapılacak. Temp sözlüğe eklenen synonyms kelime :
"+ synonymsWordList[0])
             return
```

```
else:
        print("Fore.LIGHTBLUE_EX + İlk doküman olduğu için Sözlüğe "+
str(topCountTFIDF)+" adet kayıt eklenecektir.")
        meaningWordList=GetParameterTopCountMeaningWord()
        for meaningWord in meaningWordList:
            cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] (nolock) where
Word=?",[meaningWord[0]])
            result_set = cursor.fetchall()
            number_of_rows_MeaningWord = result_set[0][0]
            if number_of_rows_MeaningWord ==0:
                SQLCommand = ("EXEC InsertEnglishDictionary ?")
                Values = [meaningWord[0]]
                cursor.execute(SQLCommand,Values)
                connection.commit
                print(meaningWord[0] +" kelimesi sözlüğe eklendi.")

#synonymsWordList=FindSynonymsWordsGivenParameterWord(meaningWord[0])
                #if len(synonymsWordList)>0:
                #    i=0
                #    for synonymsWord in synonymsWordList:
                #        if i<2:
                #            InsertEnglishDictionary(synonymsWord)
                #            print(synonymsWord +" kelimesi " + meaningWord[0]
+" kelimesi ile benzer olduğu için sözlüğe eklendi.")
                #            i+=1
            else:
                        print(meaningWord[0] +" kelimesi sözlükte olduğu için
     tekrar eklenmedi.")


InsertEnglishDictionaryTemp(maxLevelMeaningWord)
InsertOriginalWords(maxLevelMeaningWord,maxLevelMeaningWord)



def InsertOriginalWords(stemWord,originalWord):
    SQLCommand = ("EXEC InsertOriginalWords ?,?")
    Values = [stemWord,originalWord]
    cursor.execute(SQLCommand,Values)
    connection.commit()


def InsertEnglishDictionary(meaningWord):
    SQLCommand = ("EXEC InsertEnglishDictionary ?")
    Values = [meaningWord]
    cursor.execute(SQLCommand,Values)
    connection.commit
    print(Fore.GREEN + meaningWord +" kelimesi sözlüğe eklendi.")
```

```
def InsertEnglishDictionaryTemp(meaningWord):
    SQLCommand = ("EXEC InsertEnglishDictionaryTemp ?")
    Values = [meaningWord]
    cursor.execute(SQLCommand,Values)
    connection.commit
    print(Fore.GREEN + meaningWord +" kelimesi temp sözlüğe eklendi.")



def GetMaxLevelMeaningWord():
    cursor.execute("EXEC GetMaxLevelMeaningWord ?",[thresholdTF_IDFValue])
    meaningWord = cursor.fetchall()
    if len(meaningWord)==0:
        print(Fore.GREEN + "TFIDFWords tablosundaki tüm kayıtlar tabloya
eklendi.")
        return meaningWord
    else:
        return meaningWord[0][0]



def GetOriginalWord(word):
    if len(word)>0:
        cursor.execute("EXEC GetOriginalWord ?",[word])
        originalWord = cursor.fetchall()
        if len(originalWord)==0:
            return word
        else:
            return originalWord[0][0]



def GetParameterTopCountMeaningWord():
    cursor.execute("EXEC GetParameterTopCountMeaningWord ?",[topCountTFIDF])
    meaningWordList = cursor.fetchall()
    return meaningWordList



def GetWordList():
    Values = [thresholdCountOfSelectingWord]
    cursor.execute("EXEC GetWords_TopCount ?",Values)
    wordList = cursor.fetchall()
    return wordList
```

```
def CheckSimilarityAndSaveDB():
     cursor.execute("EXEC GetEnglishDictionary")
     englishDictionaryWordList = cursor.fetchall()
     cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[MeaningWord] (nolock)")
     result_set = cursor.fetchall()
     number_of_rows_MeaningWords = result_set[0][0]
     topCount=CalculateParameterMeaningValueCount(number_of_rows_MeaningWords)
     params = [topCount]
     cursor.execute("EXEC GetMeaningWords_TopCount ?",params)
     meaningWordList = cursor.fetchall()
     spelledMeaningWordListTemp=SpellWord(meaningWordList)

spelledMeaningWordList=LastCheckSpelledMeaningWordList(spelledMeaningWordListTe
mp)
     count=0
     TruncateMeaningWordsTable()
     if len(englishDictionaryWordList)>0:
            for meaningWord in spelledMeaningWordList:
                totalSimilarityRate=0
                for englishDictionaryWord in englishDictionaryWordList:

similarityRate=SequenceMatcherSimilarity.SimilarityRate(meaningWord,englishDict
ionaryWord[0])
                    totalSimilarityRate+=similarityRate

averageSmilarityRate=GetAverageSimilarityRate(totalSimilarityRate,len(englishDi
ctionaryWordList))
                if averageSmilarityRate>=thresholdRateForSimilarity:
                    try:
                        SQLCommand = ("EXEC InsertEnglishDictionary ?")
                        Values = [meaningWord]
                        cursor.execute(SQLCommand,Values)
                        connection.commit()
                        count+=1
                        print(Fore.GREEN + meaningWord +" kelimesi threshold
değerinden yüksek olduğu için sözlüğe eklendi.")


                        FindWordSynonymsAndSaveDB(meaningWord)
                        print(Fore.CYAN + meaningWord +" kelimesi için Synonyms
kelimeler bulundu.")

                        SQLCommand = ("EXEC InsertEnglishDictionaryTemp ?")
                        Values = [meaningWord]
                        cursor.execute(SQLCommand,Values)
                        connection.commit()
                        print(Fore.GREEN + meaningWord +" kelimesi Temp sözlüğe
eklendi.")

                    except:
                        pass
```

```
else:
          print(Fore.LIGHTWHITE_EX + "Sözlük boş olduğu için Anlamlı
kelimelerin kaynak topic ile benzerlikleri karşılaştırılacak.")
          for meaningWord in spelledMeaningWordList:
              try:

similarityRate=SequenceMatcherSimilarity.SimilarityRate(meaningWord,sourceTopic
)
                  if similarityRate>thresholdRateForSimilarity:
                      SQLCommand = ("EXEC InsertEnglishDictionary ?")
                      Values = [meaningWord]
                      cursor.execute(SQLCommand,Values)
                      connection.commit()
                      count+=1
                      print(Fore.GREEN + meaningWord +" kelimesi threshold
değerinden yüksek olduğu için sözlüğe eklendi.")

                      FindWordSynonymsAndSaveDB(meaningWord)
                      print(Fore.CYAN + meaningWord +" kelimesi için Synonyms
kelimer bulundu.")

                      SQLCommand = ("EXEC InsertEnglishDictionaryTemp ?")
                      Values = [meaningWord]
                      cursor.execute(SQLCommand,Values)
                      connection.commit()
                      print(Fore.GREEN + meaningWord +" kelimesi Temp sözlüğe
eklendi.")
              except:
                  pass



def LastCheckSpelledMeaningWordList(wordList):
    start=datetime.datetime.now()
    if len(wordList)==0:
        print(Fore.RED + "Anlamlı kelime listesi boş.")
    for word in wordList:
        if not wordnet.synsets(word):
            wordList.remove(word)
            print(Fore.RED + word+" kelimesi anlamlı bir kelime olmadığı için
SpelledMeaningWordList listesinden silindi.")
    end=datetime.datetime.now()
    print(Fore.BLUE + "LastCheckSpelledMeaningWordList metodu çalışma süresi:
",(end-start))
    return wordList
```

```
def LastCheckMeaningWords(spelledWordList,tfIDFWord):
    if tfIDFWord==False:
        english_vocab = set(word.lower() for word in nltk.corpus.words.words())
        for meaningWord in spelledWordList:
            result=meaningWord in english_vocab
            if result==False:
                SQLCommand = ("DELETE FROM [SmallWordsEducation].[dbo].[Words]
WHERE Word=?")
                Values = [meaningWord]
                cursor.execute(SQLCommand,Values)
                connection.commit()
                print(Fore.RED + meaningWord+" kelimesi anlamlı bir kelime
olmadığı için Words tablosundan silindi.")
    else:
        for meaningWord in spelledWordList:
            result=meaningWord in words.words()
            if result==False:
                SQLCommand = ("DELETE FROM
[SmallWordsEducation].[dbo].[TFIDFWords] WHERE Word=?")
                Values = [meaningWord]
                cursor.execute(SQLCommand,Values)
                connection.commit()
                print(Fore.RED + meaningWord+" kelimesi anlamlı bir kelime
olmadığı için TFIDFWords tablosundan silindi.")


def LastCheckEnglishDictionary():
    SQLCommand = ("select OriginalWord from
SmallWordsEducation.dbo.EnglishDictionary e(nolock) inner join
SmallWordsEducation.dbo.OriginalWords o(nolock) on e.Word=o.StemWord")
    cursor.execute(SQLCommand)
    englishDictionaryWordList = cursor.fetchall()
    if len(englishDictionaryWordList)>0:
        english_vocab = set(word.lower() for word in nltk.corpus.words.words())
        for englishDictionaryWord in englishDictionaryWordList:
            result=englishDictionaryWord in english_vocab
            if result==False:
                SQLCommand = ("DELETE FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] WHERE Word=?")
                Values = [englishDictionaryWord[0]]
                cursor.execute(SQLCommand,Values)
                connection.commit()
                print(Fore.RED + englishDictionaryWord[0]+" kelimesi anlamlı
bir kelime olmadığı için EnglishDictionary tablosundan silindi.")



def CalculateParameterMeaningValueCount(meaningWordsCount):
    result=int((percentageOfSelectingMeaningWord/100)*meaningWordsCount)
    if result==0:
        return parameterMeaningValueCountIfListEmpty
    return result
```

```
def FillPathIfFolderPathIsEmpty():
     SQLCommand = ("SELECT TOP(?) Word FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] (nolock) order by Id desc")
     Values = [topCountForBackupPath]
     cursor.execute(SQLCommand,Values)
     englishDictionaryWordList = cursor.fetchall()
     if len(englishDictionaryWordList)==0:
         for backupWebSearchWord in backupWebSearchWordlist:
             WebSearch.searchAndSaveToFile(backupWebSearchWord)
     else :
         for englishDictionaryWord in englishDictionaryWordList:
             WebSearch.searchAndSaveToFile(englishDictionaryWord[0])


def SpellWord(wordList):
    start=datetime.datetime.now()
    spelledWordList=[]
    if len(wordList)==0:
        print(Fore.RED + "Anlamlı kelime listesi boş.")
    else:
        for word in wordList:
            spelledWordList.append(spell(word[0]))
    end=datetime.datetime.now()
    print(Fore.LIGHTBLUE_EX + "SpellWord metodu çalışma süresi: ",(end-start))
    return spelledWordList


def FindSynonymsWordsGivenParameterWord(word):
    SynonymsWordList=[]
    for syn in wordnet.synsets(word):
                      for l in syn.lemmas():
                         if l.name() !=word:
                             SynonymsWordList.append(l.name())
    uniqueList=unique(SynonymsWordList)
    for synonymsWord in SynonymsWordList:
        if not wordnet.synsets(synonymsWord):
            uniqueList.remove(synonymsWord)
    return uniqueList



def unique(list):
       unique_list = []
       for x in list:
              if x not in unique_list:
                     unique_list.append(x)
       return unique_list



def TruncateMeaningWordsTable():
    SQLCommand = ("Truncate Table [SmallWordsEducation].[dbo].[MeaningWord]")
    cursor.execute(SQLCommand)
    connection.commit()
```

```
def FindWordSynonymsAndSaveDB(word):
    global sourceTopic
    sourceTopicList=FindSynonymsWordsGivenParameterWord(sourceTopic)
    synonymsWordList=FindSynonymsWordsGivenParameterWord(word)
    for synonymsWord in synonymsWordList:
        totalSimilarityRate=0
        for sourceTopic in sourceTopicList:

similarityRate=SequenceMatcherSimilarity.SimilarityRate(synonymsWord,sourceTopi
c)
            totalSimilarityRate+=similarityRate

averageSmilarityRate=GetAverageSimilarityRate(totalSimilarityRate,len(sourceTop
icList))
        if averageSmilarityRate>=thresholdRateForSimilarity:
            SQLCommand = ("EXEC InsertEnglishDictionary ?")
            Values = [synonymsWord]
            cursor.execute(SQLCommand,Values)
            connection.commit()
            print(Fore.GREEN + synonymsWord +" kelimesi " + word +" kelimesi
ile benzer olduğu için sözlüğe eklendi.")
        else:
            print(Fore.RED + synonymsWord +" kelimesi " + word +" kelimesi ile
benzer olmadığı için sözlüğe eklenmedi.")


def DeleteWordsForMeaningWordsCalculation():
    try:
        SQLCommand=("EXEC DeleteWordsForMeaningWordsCalculation ?,?")
        Values = [count_Paragraph_Threshold,sum_Count_Threshold]
        cursor.execute(SQLCommand,Values)
        connection.commit()
        print(Fore.LIGHTBLUE_EX + "DeleteWordsForMeaningWordsCalculation SP'si
başarılı olarak çalıştı.")
    except Exception as e:
        print(Fore.RED + "DeleteWordsForMeaningWordsCalculation SP'sinde hata
oluştu. Hata " + str(e))

def GetTotalSimilarityRate():
     cursor.execute("EXEC GetEnglishDictionary")
     englishDictionaryWordList = cursor.fetchall()
     cursor.execute("SELECT distinct Word FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] (nolock)")
     distinctEnglishDictionaryWordList = cursor.fetchall()
     if len(englishDictionaryWordList) > 0:
         totalSimilarityRate=0
         for englishDictionaryWord in englishDictionaryWordList:

similarityRate=SequenceMatcherSimilarity.SimilarityRate(sourceTopic,englishDict
ionaryWord[0])
             totalSimilarityRate+=similarityRate

averageSmilarityRate=GetAverageSimilarityRate(totalSimilarityRate,len(englishDi
ctionaryWordList))
         return averageSmilarityRate
     else:
         return 0
```

```
def GetContentFreq(content):
    translator = str.maketrans('', '', string.punctuation)
    words = nltk.word_tokenize(content)
    words = [word.translate(translator) for word in words]
    words = [word for word in words if len(word) > 1]
    words = [word for word in words if not word.isnumeric()]
    words = [word.lower() for word in words]
    words = [word for word in words if word not in stopwords.words('english')]
    tempWords = []
    for word in words:
        if word != "" and len(word) > 1:
            tempWords.append((word))
    words = tempWords
    return words



def TruncateEnglishDictionaryTempTable():
    SQLCommand = ("Truncate Table
[SmallWordsEducation].[dbo].[EnglishDictionaryTemp]")
    cursor.execute(SQLCommand)
    connection.commit()


def GetAverageSimilarityRate(totalSimilarityRate,englishDictionaryWord):
    if englishDictionaryWord==0:
        return 0
    else:
        result=totalSimilarityRate/englishDictionaryWord
    return result


def SpellChecker():
    spell = SpellChecker()
    misspelled = spell.unknown(['something', 'is', 'hapenning', 'here'])
    for word in misspelled:
        print(spell.correction(word))



def FindMeaningWordsAndSaveDB():
    try:
        SQLCommand = ("EXEC InsertMeaningValue")
        cursor.execute(SQLCommand)
        connection.commit()

        cursor.execute("SELECT StemWord FROM
[SmallWordsEducation].[dbo].[MeaningWord] (nolock)")
        meaningWordList = cursor.fetchall()
        print(Fore.LIGHTBLUE_EX + "MeaningWord Listesi : ",meaningWordList)
    except Exception as e:
        print(Fore.RED + "Anlamlı kelimeleri bulurken hata oluştu.Hata :
"+str(e))
        print(Fore.LIGHTBLUE_EX + "Bu nedenle
DeleteWordsForMeaningWordsCalculation SP'si çalışmaya başladı.")
        DeleteWordsForMeaningWordsCalculation()
        FindMeaningWordsAndSaveDB()
        print(Fore.CYAN + "Anlamlı kelimeler tekrar bulunuyor.")
```

```
def SeparateWordAndSaveDB():
    counts = dict()
    paraghraphId = 0
    TruncateAllDatabase()
    path=RemovingfolderPath
    sortlist = sorted(os.listdir(path))
    if len(sortlist)==0:
        print(Fore.RED + path+ " Path'inde doküman kalmadı. Yeni kelimeler ile
path dolduruluyor.")
        FillPathIfFolderPathIsEmpty()
        sortlist = sorted(os.listdir(path))
    i = 0
    documentCount=str(len(sortlist))
    print(Fore.LIGHTMAGENTA_EX + path +" Path'inde Toplam "+ documentCount+ "
adet doküman var.")
    while(i < len(sortlist)):
        print(Fore.CYAN + str(i)+". sıradaki " + sortlist[i]+ " dokümanı
okunmaya başlandı.")
        dna = open(path + "\\" + sortlist[i],encoding='utf8',errors='ignore')
        try:
            soup = BeautifulSoup(dna)
        except Exception as e:
            print(Fore.RED + sortlist[i]+" dokümanında hata oluştu. Hata " +
str(e))
        paragraphs = soup.find_all("p")
        paraghraphId = 1
        stemmer = PorterStemmer()
        for element in paragraphs:
            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:
                originalWord=word
                stems = stemmer.stem(word)
                try:
                    SQLCommand = ("EXEC InsertOriginalWords ?,?")
                    Values = [stems,originalWord]
                    cursor.execute(SQLCommand,Values)
                    connection.commit()
                except Exception as e:
                    print(Fore.RED + "OriginalWords tablosuna insert ederken
hata oluştu. Hata :" +str(e))

                if stems in counts.keys():
                    shortest,count = counts[stems]
                    counts[stems] = (shortest,count + 1)
                else:
                    counts[stems] = (stems,1)
            for kok in counts:
                shortest,count = counts[kok]
                try:
                    SQLCommand = ("INSERT INTO Words (DocumentId,Word,
Count,StemWord,Paragraph)  VALUES (?,?,?,?,?)")
                    Values = [i,shortest,count,kok,paraghraphId]
                    cursor.execute(SQLCommand,Values)
                    connection.commit()
                except Exception as e:
```

```
print(Fore.RED + "Word tablosuna insert ederken oluştu. Hata : "+ str(e))


            counts.clear()
            tokens.clear()
            nouns.clear()
            paraghraphId+=1

            number_of_rows_Words = GetWordsCount(i,False)

        if len(paragraphs)>0 and number_of_rows_Words>0:
            SQLCommand = ("INSERT INTO Documents
(Id,DocumentName,Topic,SubTopic)  VALUES (?,?,?,?)")
            Values = [i,sortlist[i],sortlist[i],sortlist[i]]
            cursor.execute(SQLCommand,Values)
            connection.commit()
        i+=1



def FindMeaningWordsAndSaveDB():
    try:
        SQLCommand = ("EXEC InsertMeaningValue")
        cursor.execute(SQLCommand)
        connection.commit()

        cursor.execute("SELECT StemWord FROM
[SmallWordsEducation].[dbo].[MeaningWord] (nolock)")
        meaningWordList = cursor.fetchall()
        print(Fore.LIGHTBLUE_EX + "MeaningWord Listesi : ",meaningWordList)
    except Exception as e:
        print(Fore.RED + "Anlamlı kelimeleri bulurken hata oluştu.Hata :
"+str(e))
        print(Fore.LIGHTBLUE_EX + "Bu nedenle
DeleteWordsForMeaningWordsCalculation SP'si çalışmaya başladı.")
        DeleteWordsForMeaningWordsCalculation()
        FindMeaningWordsAndSaveDB()
        print(Fore.CYAN + "Anlamlı kelimeler tekrar bulunuyor.")



def NewFindMeaningWordsAndSaveDB():

    SQLCommand = ("EXEC GetTotalDocumentCount")
    cursor.execute(SQLCommand)
    result = cursor.fetchall()
    totalDocumentCount=result[0][0]
    documentId=0




def CloseDBConnection():
    cursor.close()
    connection.close()
```

```
def GetWordsCount(documentId,type):
    if type==False:
        cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[Words] (nolock) where DocumentId=?",[documentId])
        result_set = cursor.fetchall()
        number_of_rows_Words = result_set[0][0]
        return number_of_rows_Words
    else:
        cursor.execute("SELECT distinct count(*) FROM
[SmallWordsEducation].[dbo].[Words] (nolock)")
        result_set = cursor.fetchall()
        number_of_rows_Words = result_set[0][0]
        return number_of_rows_Words



def NewFindMeaningWordsAndSaveDB2():

    SQLCommand = ("EXEC GetTotalDocumentCount")
    cursor.execute(SQLCommand)
    result = cursor.fetchall()
    totalDocumentCount=result[0][0]
    documentId=0

    SQLCommand = ("EXEC GetDistinctWordList")
    cursor.execute(SQLCommand)
    wordList = cursor.fetchall()

    SQLCommand = ("EXEC GetTotalWordsFrequency")
    cursor.execute(SQLCommand)
    result = cursor.fetchall()
    totalWordsFrequency = result[0][0]

    for word in wordList:
        SQLCommand = ("EXEC GetTermFrequency ?")
        Values = [word[0]]
        cursor.execute(SQLCommand,Values)
        term = cursor.fetchall()
        termFrequency = term[0][0]

        TF=CalculateTF(termFrequency,totalWordsFrequency)

        SQLCommand = ("EXEC GetDocumentCountOfIncludeTerm ?")
        Values = [word[0]]
        cursor.execute(SQLCommand,Values)
        document = cursor.fetchall()
        documentCountOfIncludeTerm = len(document)


        IDF=CalculateIDF(documentCountOfIncludeTerm,totalDocumentCount)

        TF_IDF=Calculate_TF_IDF(TF,IDF)

        SQLCommand = ("EXEC InsertTFIDFWords ?,?,?,?,?")
        Values = [documentId,word[0],TF,IDF,TF_IDF]
        cursor.execute(SQLCommand,Values)
        connection.commit()
```

```
def CalculateTF(termFrequency,totalWordsFrequency):
    return termFrequency/totalWordsFrequency

def CalculateIDF(documentCountOfIncludeTerm,totalDocumentCount):
    normalizationValue=0
    if documentCountOfIncludeTerm==0:
        documentCountOfIncludeTerm+=1
    if math.log(totalDocumentCount/documentCountOfIncludeTerm)==0:
        normalizationValue=1
    return
normalizationValue+math.log(totalDocumentCount/documentCountOfIncludeTerm)

def Calculate_TF_IDF(TF,IDF):
    return TF*IDF



def DeleteMeaningLessWords():
    cursor.execute("SELECT distinct Word FROM
[SmallWordsEducation].[dbo].[Words] (nolock)")
    wordList = cursor.fetchall()
    spelledWordList=SpellWord(wordList)
    LastCheckMeaningWords(spelledWordList,False)



def RunEducationDataProject():
    a=datetime.datetime.now()
    print(Fore.CYAN + "Dokümanlar okunup kelimelere ayrılmaya başlandı.",a)
    SeparateWordAndSaveDB()
    b=datetime.datetime.now()
    print(Fore.LIGHTBLUE_EX + "Okuma süresi ", b-a)
    number_of_rows_Words = GetWordsCount(0,True)
    if number_of_rows_Words<thresholdWordCountPerDocument:
        print(Fore.RED + "Okunan dokümandaki kelime sayısı "+
str(thresholdWordCountPerDocument)+" kelimeden az olduğu için Web araması
tekrar yapılacak.")
        return
    print(Fore.LIGHTMAGENTA_EX + "Anlamsız kelimeler siliniyor. " , b)
    DeleteMeaningLessWords()
    c=datetime.datetime.now()
    print(Fore.LIGHTBLUE_EX + "Anlamsız kelimeleri silme süresi " , c-b)
    print(Fore.LIGHTYELLOW_EX + "Anlamlı kelimeler bulunuyor. " , c)
    NewFindMeaningWordsAndSaveDB2()
    d=datetime.datetime.now()
    print(Fore.LIGHTGREEN_EX + "Anlamlı kelimeleri bulma süresi " , d-c)
    print(Fore.LIGHTBLUE_EX + "Benzerliğe göre sözlüğe eklenecek. " , d)
    CheckSimilarityAndSaveDB2()
    e=datetime.datetime.now()
    print(Fore.CYAN + "Benzerliğe göre sözlüğe ekleme süresi " , e-d)
```

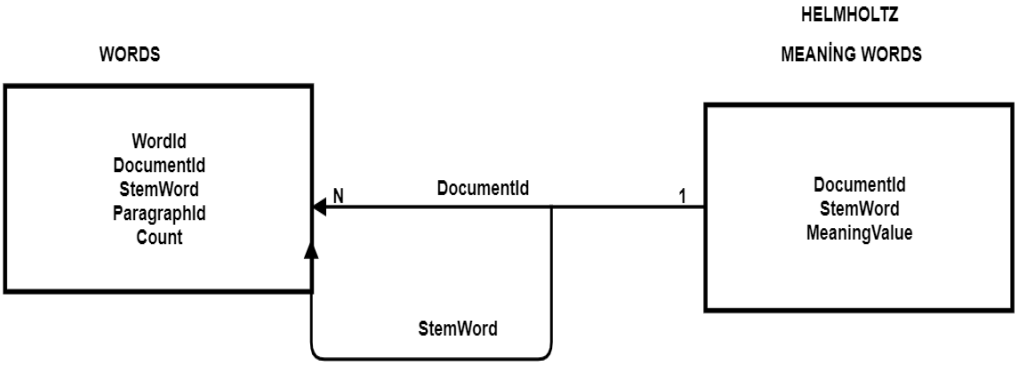
```
from bs4 import BeautifulSoup
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.corpus import wordnet as wn
from nltk.tokenize import word_tokenize
from nltk import pos_tag
import nltk.data
import pypyodbc
import re
import os
import nltk
import string
import requests
import re
import urllib
from string import punctuation
import datetime
import EducationData
import colorama
from colorama import Fore, Back, Style


colorama.init()


subscription_key='806fdf38921049c29a2e0d808a1b202c'
search_url = "https://api.cognitive.microsoft.com/bing/v7.0/search"
headers = {"Ocp-Apim-Subscription-Key" : subscription_key,"mkt":"en-US"}
webSearchSaveAndRemoveDirectory = "C:\\Users\AT012337\\Thesis\Konu
Tespiti\\SmallWord Eğitim Verileri\\"
percentageOfSelectingMeaningWord=10
thresholdCountOfDocumentsOnPath=150
thresholdEnglishDictionaryCountForStopProject=50
prefix_Search_term="education scholarship"
number_of_results_to_return_in_the_response=3
parameter_specifies_the_number_of_results_to_skip=0



connection = pypyodbc.connect('Driver={SQL Server};'
                                                'Server=NB-AT012337;'
                                                 'Database=SmallWordsEducation;'
                                                   'uid=PhytonThesisUser;pwd=1')
cursor = connection.cursor()




def GetCountOfDocumentsOnPath():
     sortlist = sorted(os.listdir(webSearchSaveAndRemoveDirectory))
     return len(sortlist)
```

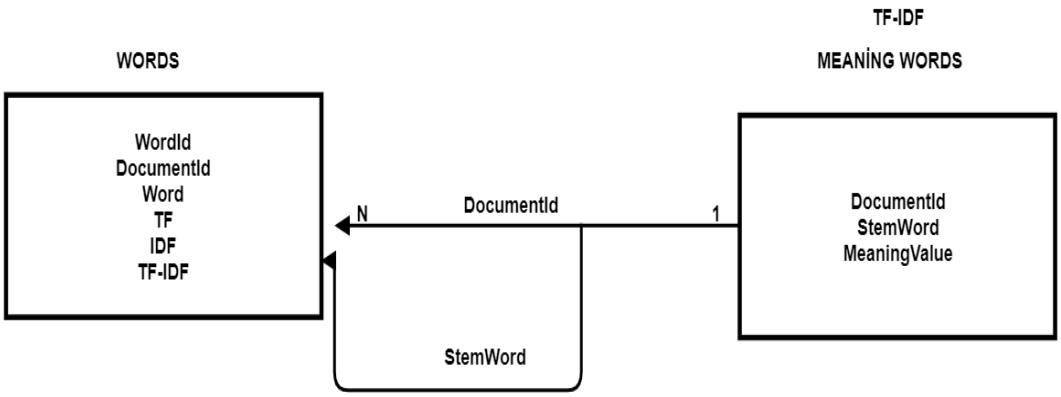
```
def GetWordCountOfEnglishDictionary():
    #LastCheckEnglishDictionary()
    cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] (nolock)")
    result_set = cursor.fetchall()
    number_of_rows_EnglishDictionary = result_set[0][0]
    return number_of_rows_EnglishDictionary



def LastCheckEnglishDictionary():
    SQLCommand = ("select OriginalWord from
SmallWordsEducation.dbo.EnglishDictionary e(nolock) inner join
SmallWordsEducation.dbo.OriginalWords o(nolock) on e.Word=o.StemWord")
    cursor.execute(SQLCommand)
    englishDictionaryWordList = cursor.fetchall()
    if len(englishDictionaryWordList)>0:
        english_vocab = set(word.lower() for word in nltk.corpus.words.words())
        for englishDictionaryWord in englishDictionaryWordList:
            result=englishDictionaryWord in english_vocab
            if result==False:
                SQLCommand = ("DELETE FROM
[SmallWordsEducation].[dbo].[EnglishDictionary] WHERE Word=?")
                Values = [englishDictionaryWord[0]]
                cursor.execute(SQLCommand,Values)
                connection.commit()
                print(Fore.RED + englishDictionaryWord[0]+" kelimesi anlamlı
bir kelime olmadığı için EnglishDictionary tablosundan silindi.")



def RemoveAllItemsFromFolder():
    for root, dirs, files in os.walk(webSearchSaveAndRemoveDirectory):
      for f in files:
        os.unlink(os.path.join(root, f))
      for d in dirs:
        shutil.rmtree(os.path.join(root, d))
    print(Fore.YELLOW +"Path temizlendi.")



def getMeaningWordForWebSearching2():
    cursor.execute("select OriginalWord from
SmallWordsEducation.dbo.EnglishDictionaryTemp e(nolock) inner join
SmallWordsEducation.dbo.OriginalWords o(nolock) on e.Word=o.StemWord")
    joined_Result=cursor.fetchall()
    if len(joined_Result)>0:
        return joined_Result
    else:
        cursor.execute("select Word from
SmallWordsEducation.dbo.EnglishDictionaryTemp e(nolock)")
        result=cursor.fetchall()
        return result

def CalculateParameterMeaningValueCount(meaningWordsCount):
    result= int((percentageOfSelectingMeaningWord/100)*meaningWordsCount)
    return result
```

```
def searchAndSaveToFile(search_term):
    if search_term!=prefix_Search_term:
        search_term += " " + prefix_Search_term

    search_term += " language:en"
    print(Fore.LIGHTCYAN_EX +"Arama yapılacak kelime : "+ search_term)
    params  = {"q": search_term, "textDecorations":True, "textFormat":"HTML",
               #"count":number_of_results_to_return_in_the_response,
               #"offset":parameter_specifies_the_number_of_results_to_skip,
               "mkt":"en-US"}
    try:
      response = requests.get(search_url, headers=headers,
params=params,timeout=7)
      response.raise_for_status()
      for page in response.json()['webPages']['value']:
        page_url = page['url']
        try:
            page_response = requests.get(page_url,headers=headers,
params=params,timeout=10)
            soup = BeautifulSoup(page_response.content)
            paragraphs = soup.find_all("p")
            customFileName=re.sub(r'\W+', '', search_term.split(" ", 1)[0] +
"_"+page_url) + '.txt'
            cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[SearchedDocuments] (nolock) where
DocumentName=?",[customFileName])
            result_set = cursor.fetchall()
            number_of_rows_SearchedDocuments = result_set[0][0]
            if len(paragraphs)>0 and number_of_rows_SearchedDocuments==0:
                file_name = re.sub(r'\W+', '', search_term.split(" ", 1)[0] +
"_"+page_url) + '.txt'
                file_name = webSearchSaveAndRemoveDirectory+file_name
                f = open(file_name, "w")
                f.write(str(page_response.content))
                f.close()

                SQLCommand = ("EXEC InsertSearchedDocuments ?")
                Values = [customFileName]
                cursor.execute(SQLCommand,Values)
                connection.commit()
                print(Fore.LIGHTYELLOW_EX +search_term +" kelimesi için Web
araması tamamlandı.")
                return
        except:
            pass
    except:
       pass
```

```
def getMeaningWordForWebSearching():
     cursor.execute("SELECT count(*) FROM
[SmallWordsEducation].[dbo].[EnglishDictionaryTemp] (nolock)")
     result_set = cursor.fetchall()
     number_of_rows_MeaningWords = result_set[0][0]
     topCount=CalculateParameterMeaningValueCount(number_of_rows_MeaningWords)
     if topCount<=0:
            topCount=1
     cursor.execute("select top(?) OriginalWord from
SmallWordsEducation.dbo.EnglishDictionaryTemp e(nolock) inner join
SmallWordsEducation.dbo.OriginalWords o(nolock) on e.Word=o.StemWord Order By
e.Id desc", [topCount])
     joined_Result=cursor.fetchall()
     if len(joined_Result)>0:
             return joined_Result
     else:
         cursor.execute("select top(?) Word from
SmallWordsEducation.dbo.EnglishDictionaryTemp e(nolock)", [topCount])
         result=cursor.fetchall()
         return result




def WebSearch():
    a=datetime.datetime.now()
    print(Fore.LIGHTBLUE_EX +"Web aramasına başlanıyor.", a)
    RemoveAllItemsFromFolder()
    result_set=getMeaningWordForWebSearching2()
    number_of_rows_MeaningWords = len(result_set)
    if number_of_rows_MeaningWords > 0:
        searchTerm=""
        for row in result_set:
            searchTerm+=row[0]+" "
        if GetCountOfDocumentsOnPath()<thresholdCountOfDocumentsOnPath and
GetWordCountOfEnglishDictionary()<thresholdEnglishDictionaryCountForStopProject
:
            searchAndSaveToFile(searchTerm)
    else:
        searchAndSaveToFile(prefix_Search_term)
    b=datetime.datetime.now()
    print(Fore.MAGENTA +"Web araması süresi " ,b-a)
```

EK B. ER Diagramı





ÖZGEÇMİŞ

Adı Soyadı : Ahmet TOPRAK

Doğum Yeri ve Yılı : ŞANLIURFA, 30/06/1994

Medeni Hali : Bekar

Yabancı Dili : İngilizce

E-posta : ahmetoprak190363@gmail.com

Eğitim Durumu

Lise : Şeyh Şamil Lisesi, 2012

Lisans : Trakya Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

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

Mesleki Deneyim

Türkiye Finans Katılım Bankası 2016-...(devam ediyor)

Yayınları

Özdoğan, A.G., Toprak, A., 2018. The Effect Of Gamification in Information Technologies Projects. 2018 3rd International Conference on Computer Science and Engineering (UBMK), 20-23 September, Sarajevo, 36-40.

Özdoğan, A.G., Toprak, A., 2019. Kurumsal Şirketlerde DevOps Süreçlerinin Zorlukları. 21. Akademik Bilişim Konferansı(AB 2019), 13-15 Şubat, Ordu, 49-55.

Toprak, A., Turan, M., 2019. Konuya Özel Web Kaynaklı İngilizce Otomatik Sözlük Oluşturma, Technologies and Applied Sciences, 2(1), 13-21.