
**T.C.**

**ERZURUM TEKNİK ÜNİVERSİTESİ**

**MÜHENDİSLİK VE MİMARLIK FAKÜLTESİ**

**BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

# ZEMBEREK KÜTÜPHANESİ VE VEKTÖR UZAY MODELİ

**KÜBRA DEMİR**

**Nisan-2022**

**ERZURUM**

## RAPOR BİLDİRİMİ

Bu rapordaki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edildiğini ve bu çalışmada bana ait olmayan her türlü ifade ve bilginin kaynağına eksiksiz atıf yapıldığını bildiririm.

KÜBRA DEMİR

Tarih: 22/04/2022

# ÖZET

## ZEMBEREK KÜTÜPHANESİ ve VEKTÖR UZAY MODELİ

**KÜBRA DEMİR**

**Danışman: IŞIL KARABEY AKSAKALLI**

Çalışmanın ilk bölümünün konusu Zemberek kütüphanesinin araştırılmasıdır. Bu araştırmada Zemberek kütüphanesinin yapısından başlanarak, kütüphane içindeki problemlere daha sonrasında ise kullanım alanı olduğu kök bulma(stemming) işleminin derinlemesine incelemesi yapılmıştır. Kök bulma(stemming) başlığı altında ne olduğu, ne için kullanıldığı konularına değinilmiştir. Bunların yanında kök bulma işlemindeki hatalara, algoritmalara ve bu algoritmaların karşılaştırılmasına, son olarak Zemberek kütüphanesi yardımıyla yapılan bir örnek kod çalışmasına yer verilmiştir.

Çalışmanın ikinci kısmında ise Vektör Uzay Modelinin Bilgi Erişim açısından incelenmesine yer verilmiştir. İçerik olarak öncelikle vektör uzay modelinin ne olduğu araştırılmış ikinci adımda bilgi erişimdeki rolüne bakılmıştır. Bilgi erişim kısmının alt başlıklarında ise vektör uzay modelinin analizi yapılmıştır. Üçüncü adımda ise vektör uzay modelinin mesafe(distance) yöntemlerine bakılmıştır. Dördüncü adımda benzerlik ölçütleri incelenmiş ve bazıları arasında karşılaştırma yapılmıştır. Son olarak vektör uzay modelinin avantaj ve dezavantajlarına bakılarak başarım metrikleri hakkında bilgi edinilmiştir.

**Anahtar Kelimeler:** Zemberek, kök bulma(stemming), vektör uzay modeli, bilgi erişim

# İÇİNDEKİLER

# ŞEKİLLER VE KISALTMALAR

## ŞEKİL LİSTESİ



## KISALTMALAR

**DDİ :** Doğal Dil İşleme
**NLP :** Natural Language Processing
**POS :** Part of Speech
**IR :** Information Retrıeval
**TDK :** Türk Dil Kurumu
**Tf :** Term Frequency
**Df** : Document Frequency
**Cf :** Collection Frequency
**Idf :** Inverse Document Fr

# BİRİNCİ BÖLÜM

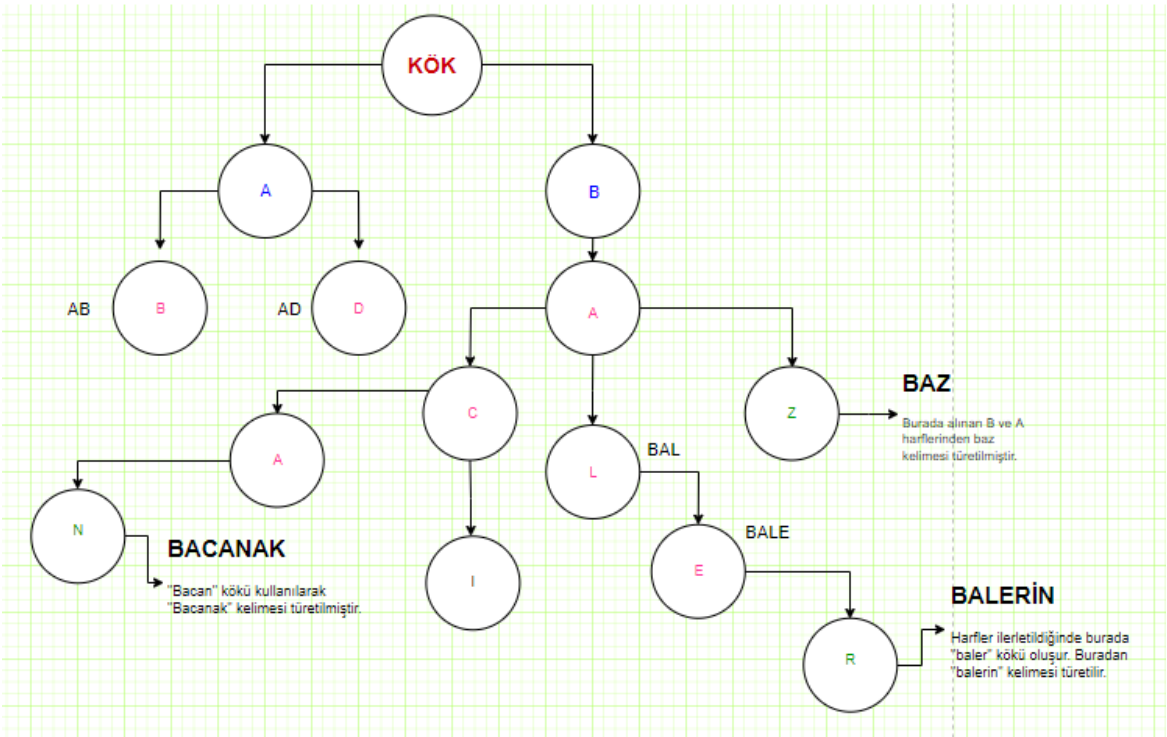
# ZEMBEREK KÜTÜPHANESİ NEDİR?

Bilgisayar dünyasında iki farklı dil vardır bunlardan biri makine dilleri yani programlama dilleri, diğeri ise doğal dillerdir. Doğal dillerden kasıt, insanların konuştuğu doğal dillerdir, bunlara; Türkçe, İngilizce gibi diller örnek olarak verilebilir. İnsanların konuştuğu dili makinelerin alıp işlemesine doğal dil işleme denilmiştir. Doğal dil işleme yani NLP doğal dillerin kurallı yapısını çözümleyerek anlaşılmasını ve gerekirse üretilmesini sağlar. Doğal dil işleme sayesinde yapılan çözümlemeler dokümanların anlaşılması, verilen komutları anlama gibi durumlar için kolaylıklar sağlar. Bu kolaylıkların Türkçe dili için çözümlemesi Zemberek kütüphanesidir.

Zemberek kütüphanesi Ahmet Akın tarafından geliştirilen açık kaynak kodlu Türkçe dili için kullanılan doğal dil işleme kütüphanesidir. Bu kütüphane kullanılarak stemming yani kelimenin kökünü bulma işlemi yapılabilir. Stemming haricinde kelimenin Türkçe olup olmaması, kelimenin yazım yanlışlarının düzeltilmesi, kelimenin sıfat, isim, fiil şeklinde ayrımı gibi daha birçok konuda kullanıcıya kolaylık sağlayacak çözümlemelerde yapılabilmektedir.

## 1. Zemberek Kütüphanesinin Yapısı

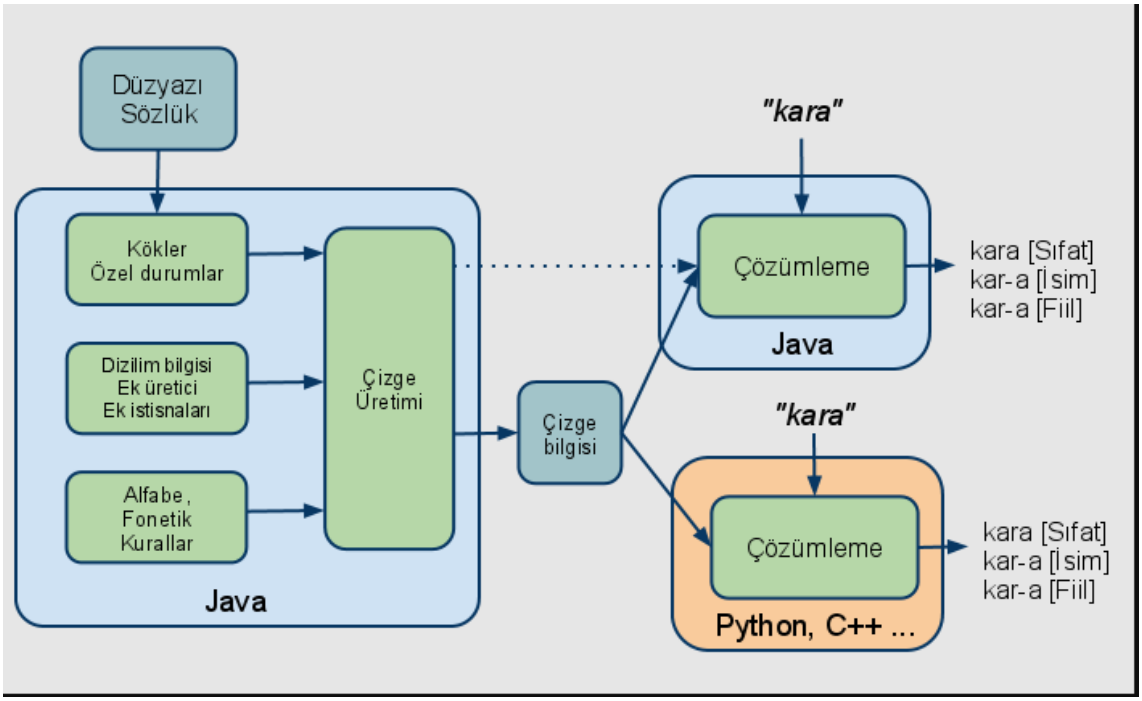
- Genel manada zemberek kütüphanesinin yapısı aşağıdaki gibidir; Burada bahsedildiği üzere stemming işlemine bir örnek olarak Zemberek kütüphanesi yapılandırılmıştır. Kütüphane içerisinde kullanılan bir kök ve ona eklenen harfler doğrultusunda köke uygun kelimeler türetilmiştir.




Şekil 1.1 Zemberek kütüphanesi stemming işlemi ile yapılandırılması

- Stemming işlemi dışında kelimenin isim, fiil olma durumunun işleyişi aşağıdaki örnek verilebilir;

Burada farklı programlama dilleri kullanılarak çözümleme yapılmış ve sonucunda "kara" kelimesinin Zemberek kütüphanesine uygun durumları ortaya çıkarılmıştır.




Şekil 1.2 Farklı programlama dillerinin zemberek kütüphanesi üzerinde şeması

## 2.Zemberek Kütüphanesi İçerisindeki Problemler

Zemberek kütüphanesinin kullanım alanı kelimenin kökünü bulmaya yöneliktir. Kök bulma içerisindeki iki farklı kelimenin aynı kökü vermesi, aynı kökü vermesi gereken iki farklı kelime ve yanlış tanımlama durumları Zemberek kütüphanesinin varlığıyla çözüme ulaşamaz. Yani bu problemlere çözüm olmamştır. Cümle analizi yapıldığında bu kök bulma işlemi stop wordsleri ele aldığında Zemberek kütüphanesi yetersiz kalabilir.

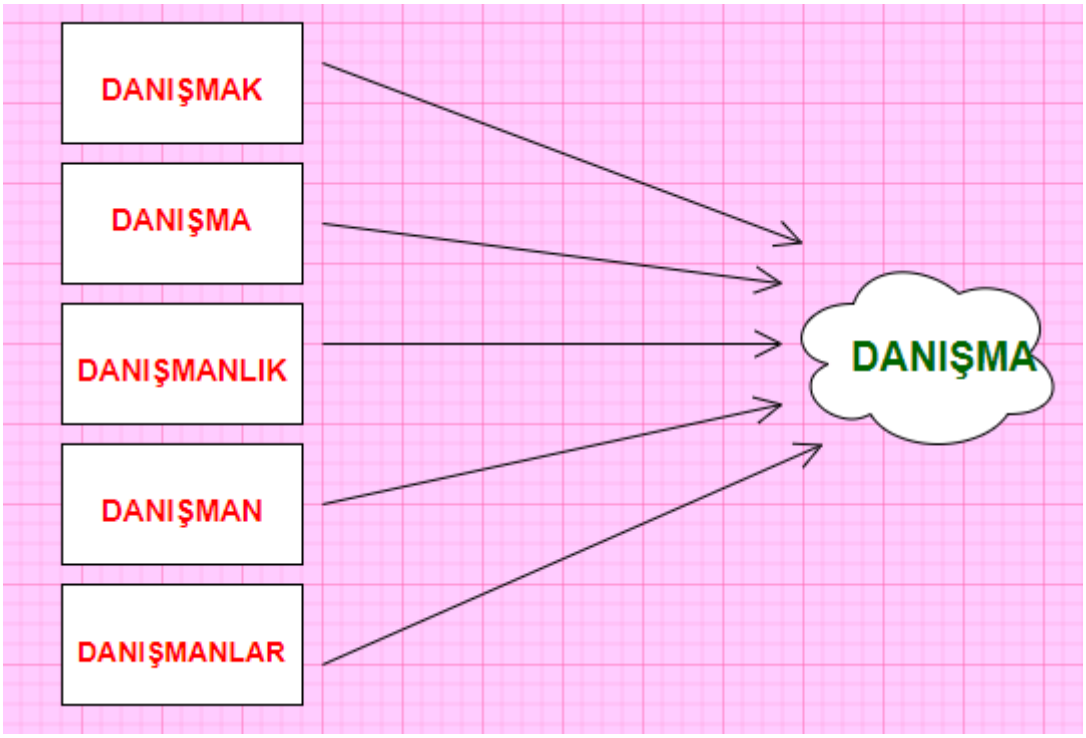
Türkçe içerisindeki bazı yazım kuralları ve TDK nın kabul etmediği bazı yazımlar için kullanıcıya Zemberek kütüphanesi yetersiz geldiği için kullanım durumuna göre yeni kütüphaneler geliştirilmiştir. Doğal dil işleme(NLP) içerisinde Zemberek kütüphanesinin dışında birçok kütüphane ile çözümler söz konusudur.

# 3.Kök Bulma(stemming)

## 3.1 Kök Bulma Nedir?

Kök bulma, bir kök kelimenin farklı versiyonlarını üretmektir. Kelimelerden ekleri çıkararak kelimelerin temel şeklini çıkarmak için kullanılan yöntemdir. Bu durumu bir ağacın dallarını kesip sadece gövdesini almak gibi düşünebiliriz. Örneğin elimizdeki kelimeler "ileri gitme", "ileri git" ve "ileri gider" gibiyse bunların stemmingi "ileri gitme" olacaktır.

Arama motorları, kelimeleri dizine eklemek için kök çıkarmayı kullanır. Bu nedenle, bir arama motoru bir kelimenin tüm formlarını saklamak yerine sadece köklerini saklayabilir. Bu şekilde, stemming indeksin boyutunu küçültür ve alma doğruluğunu artırır.




Şekil 3.1 Kök Bulma(stemming)

Stemming, kelimelerin uçlarını basitçe kesen kaba bir buluşsal süreç olarak görülebilir. Kaynak bulma, sözlük araması veya morfolojik analiz içermez. Kökün geçerli bir kelime olması veya morfolojik köküyle aynı olması bile gerekli değildir. Amaç, ilgili kelimeleri aynı köke indirgemektir.

Köklendirmenin ilk uygulamalarından biri bilgi erişim yani IR sistemleridir. Örneğin kullanıcı "patlama" kelimesini arattığında ilişkili olduğu "patlayıcılar" dökümanınında gelmesi gerekir. Stemming işlemi sayesinde IR sistemlerinin ilişkili dökümanı bulma işlemi gerçekleştirilmiş olur.

Sözcükler, genellikle ek olarak adlandırılan ön ekler ve son ekler içerebilir . Stemming genellikle soneklerle ilgilenir. Yani kelimenin sonuna getirilmiş eklemeleri bulara köke indirgeme işlemini yapar.

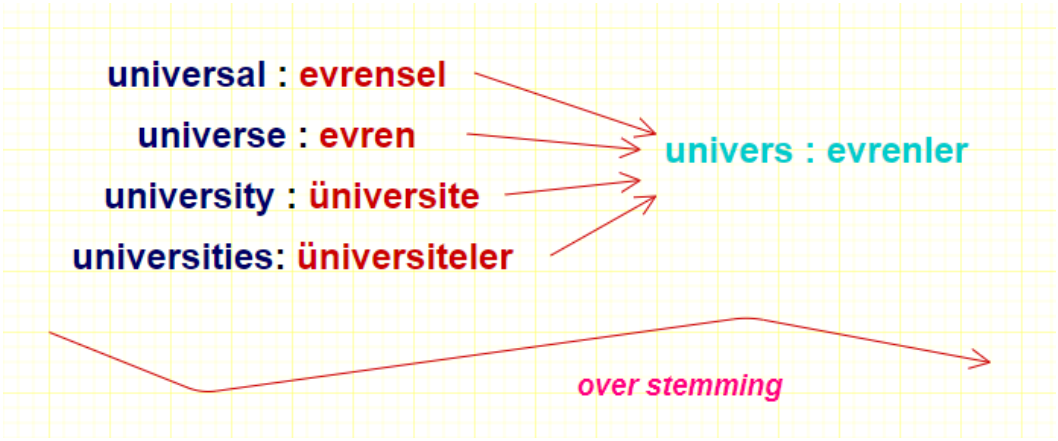
### 3.2 Kök Bulma(stemming) Hataları

Kök bulma işlemi sırasında bazı özel durumlarda kuralların başarısız olmasından kaynaklı hatalar meydana gelir. En kötü ihtimalle iki farklı kavramın aynı köke eşlenmesi görülebilir.

Örneğin, bir web portalında 'Withings', 'with' olarak kullanıldığında, kullanıcıya yanlış öğeler sunulur. Porter stemmer, farklı kavramlarla ilgili olsalar bile, 'meanness' çevrildiğinde "alçaklık" anlamına gelir, 'meaning' ise "anlam" şeklinde çevrilir. Kurallar uygulandığında ikisi de aynı kök olan 'mean' ile stemming işlemine uğrar. Burada iki farklı kavram aynı köke denk gelerek hata oluşturmuşlardır. Bunun aksine, 'goos' ve 'gees' kelimeleri aynı anlamı verir fakat stemming işleminde "goos" ve "gees" şeklinde köklendirilir. Bu durumda ters yönde aynı anlamı veren iki farklı kök ortaya çıkarak hata alınmış olur.

Genel olarak oluşan stemming hataları aşağıda verilmiştir;

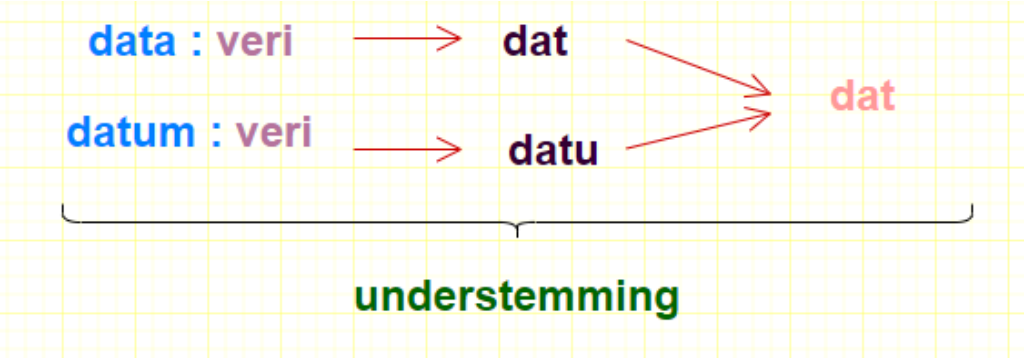
- Overstemming(aşırı köklenme) : Aşırı köklendirme, bir kelimenin gerekenden çok daha büyük bir bölümünün kesilmesi ve bunun sonucunda iki veya daha fazla kelimenin aynı kök kelimeye indirgenmesine veya iki veya daha fazla kelimeye indirgenmesi gerekirken yanlış köklenmesine neden olan süreçtir. Yani farklı köklere sahip iki kelime aynı kökten geldiğinde meydana gelir. Örneğin "university" ve "universe" kelimeleri için bakıldığında her iki kelime içinde stemming algoritmaları uygulandığında "univers" kelimesine indirgenebilir.




Şekil 3.2.1 Overstemming

- Understemming : Kökten ayırmada, aslında aynı kök kelimeye indirgenmeleri gerektiğinde, iki veya daha fazla kelime yanlışlıkla birden fazla kök kelimeye indirgenebilir. Örneğin, "data" ve "datum" kelimelerini ele alındığında aynı anlam olan "veri" kelimesine karşılık gelirler. Fakat bazı algoritmalar bu kelimeleri sırasıyla "dat" ve "datu"ya indirgeyebilir ki bu hata meydana getirir. Bunların her ikisinin de aynı kök "dat"a indirgenmesi gerekir . Ancak bu

tür modelleri optimize etmeye çalışmak, aynı zamanda aşırı köklendirmeye de yol açabilir.




Şekil 3.2.2 Understemming

- Misstemming(yanlış tanımlama) : Yanlış birleştirmelere yol açmadığı sürece iki farklı kelimenin aynı köke gelmesi sorun değildir fakat temelde yanlış bir kök alma biçimidir. Örneğin "relavity" "görecelik" anlamında ve "relative" "göreli" anlamındadır. Her ikisinin de köke indirgenmiş hali "relativ" şeklindedir. Yani doğru ama yanlış köklendirme yapılmıştır.

## 3.3 Kök Bulma(stemming) Algoritmaları

Kök bulma algoritmaları bir kelimenin değişik biçimleri üzerinden ortak bir biçime çevrilme sürecidir. Örneğin "yemek" , "yemekler", "yemekhane" kelimelerinin bu algoritmalardan birini kullanarak biçimlendirilmiş hali "yemek" olacaktır. Stemming için birçok farklı algoritma geliştirilmiştir.

Temelde stemming algoritmaları iki kategoriden birine girer;

1) **Kural tabanlı:** Köke ulaşmak için ekleri çıkarırken bir dizi kuralı uygulayan algoritmalardır.

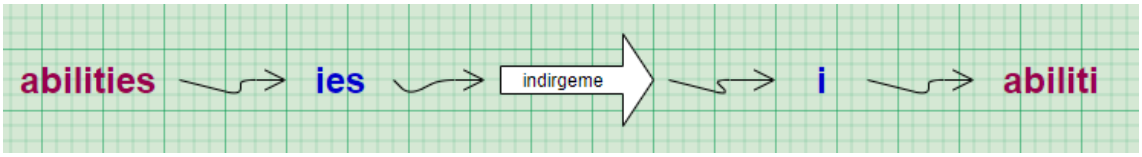
2) **İstatiksel:** Bir modeli eğiterek, son ek çıkarma kuralları örtük olarak türetilir.

Stemming için kullanılan algoritmalar aşağıdakiler gibidir;

### 3.3.1 Porter'ın Stemmer Algoritması

Uygulamalarının sadece İngilizce dili üzerinde olduğu ve İngilizce dilindeki kelimelere gelen eklerin daha küçük ve daha basit eklerin birleşiminden oluştuğu fikridir. Yani kelimelerin köküne indirgeme işlemidir. Porter algoritmasının kök alma(stemming) işlemi, hızı ve basitliği ile bilinir. Karmaşık bir yapıda değildir. Bu algoritmanın uygulandığı alanlar arasında bilgi erişimi ve veri madenciliği vardır. Aynı köke eşlendiği sürece çıkan kökün anlamlı bir kelime olması gerekmez.

Örneğin stemming işlemine yapacağımız kelime "abilities" olsun. Bu kelime için Porter algoritması uygulandığında "ies" soneki "i" ye indirgenerek arama yapılırken "abiliti" olaran aranmasını sağlar.




Şekil 3.3.1 Porter algoritması

Avantajı diğer sistemlere göre daha iyi çıktı verir ve daha az hata oranına sahiptir. Dezavantajı ise üretilen kelimeler her zaman gerçek aranan kelimeler değildir.

### 3.3.2 Lovins Stemmer Algoritması

Kelimenin en uzun eki çıkarıldıktan sonra kalan kelimeyi baz alınıp geçerli kelimelere dönüştürmek için kullanılmıştır. Bu işlem iki kısma ayrılır. Birinci kısımda kelime önceden belirlenmiş bir sonek listesiyle karşılaştırılır ve bu soneklerden birini içerdiği anlaşıldığında kelimenin eki kaldırılarak sadece kökü kalması sağlanır. İkinci kısımda ise daha farklı köklerin eşleşmesini sağlamak için birinci aşamadan çıkan yazım istisnaları kullanılarak bu işlem yapılır.

Örneğin "dried" kelimesi için birinci aşama "dri" alınır ve ikinci aşamada farklı köklerin eşleşmesi yapılarak "dry" kelimesine ulaşılır.


Şekil 3.3.2 Lovins Algoritması

Avantajı hızlıdır ve düzensiz çoğulların işlenmesini sağlar. Dezavantajı ise zaman alır ve çoğu zaman kökten kelime oluşturmada başarısız olur.

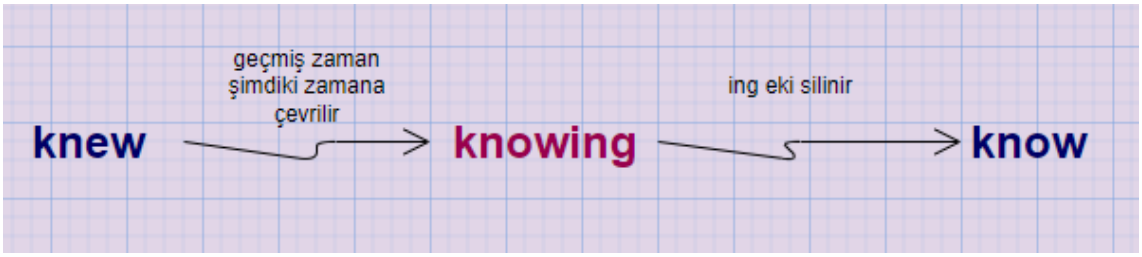
### 3.3.3 Dawson Stemmer Algoritması

Lovin algoritması üzerinden türetilmiş son eklerin uzunluklarına ve son harflerine göre indekslenerek ters sırada saklandığı stemming algoritmasıdır. İkisi arasındaki en büyük fark Dawson çok daha geniş bir kök alma işlemi gerçekleştirir. Çalışma prensibi Lovins algoritması ile aynıdır. Tek geçişli bir stemmer ve Lovin's'den daha hızlı olduğu kanıtlanmıştır. Avantajı uygulaması hızlıdır ve daha fazla kapsama alanı vardır. Dezavantajı ise karmaşık bir uygulaması vardır.

### 3.3.4 Krovetz Stemmer Algoritması

İki yöntemi vardır. Birincisi bir kelimenin çoğul halini tekil hale dönüştürerek elde edilen sonuçtur. İkincisi ise kelimenin geçmiş zamanını şimdiki zamana çevirip ve sondaki şimdiki zamandan gelen ing eki kaldırarak ulaşılan sonuçtur. Genellikle başka algoritmalar kullanılarak stemming işlemi yapar.

Örneğin “children” kelimesi çoğul halden tekile geçtiğinde “child” olur. Başka bir örnek olarak geçmiş zamanda alınan “knew” kelimesi şimdiki zamanda “knowing” olur ve “ing” eki atıldığında arama “know” kelimesi sonucunu elde eder.




Şekil 3.3.4 Krovetz Algoritması

Avantajı hafif bir algoritmadır ve diğer stemmer yöntemleri için yol gösterici olabilir. Dezavantajı ise büyük belgeler olması durumunda verimsiz hale gelmesidir.

### 3.3.5 Xerox Stemmer Algoritması

Büyük verileri üretmek için geliştirilmiştir. Geçerli sözcükler üretebilir. Aşırı köklenme olasılığı yüksektir, sözlüğe bağımlılığı onu dile bağımlı kılar. Bu nedenle sınırlaması, diline özgüdür. Yani kullanıldığı dil çerçevesinde işlem yapabilir.

- ‘children’ -> ‘child’

- ‘understood’ -> ‘understand’

- ‘whom’ -> ‘who’

- ‘best’ -> ‘good’

Yukarıdaki gibi kelime üretimi vardır. Avantajı büyük veriler üzerinde işlem yapmasıdır fakat bu durum algoritmanın aşırı köklenme durumuna geçerek dezavantajı haline gelmesini sağlamaktadır.

### 3.3.6 N-Gram Stemmer Algoritması

Belirlenen bir n değeri doğrultusunda kelimenin bu n değerine bölünerek tanımlama yapılmasıdır. Yani bir n-gram algoritması benzer sözcüklerin yüksek oranda ortak n-grama sahip olacağı bir sözcükten çıkarılan n ardışık karakter kümesidir.

Örneğin n=2 alındığı “introduction” kelimesi n-gramlarına bölündüğünde “*i , in, nt, tr, ro, od, du, uc, ct, ti, io, on” şekline olacaktır.

Buna ihtiyaç duyulmasının nedeni kelimeler içerisinde arama yapılarak daha etkin sonuçlar elde etmektir.

Avantajı dizilerin karşılaştırılmalarına bağlıdır ve dile bağlıdır. Dezavantajı ise yapılan bölümleme yani indeksleme işlemi sonucunda memory içinde fazla yer kaplamasıdır. Verileri çalıştırmak ve depolamak için yüksek memory gücüne ihtiyaç vardır.

### 3.3.7 Snowball Stemmer Algoritması

Porter algoritması prensibinden yola çıkılarak geliştirilmiş ve Porter algoritmasındaki gibisadece İngilizce için değil daha geniş dil seçeneklerini de içerisinde barındıran bir stemming algoritmasıdır. Bu algoritma kullanılarak belirli bir kelime veya kelimeler için hangi kökün döndürüleceği hızlı bir şekilde kontrol edilebilir. Kısacası Porter algoritmasının bir üst sürümü denilebilir. Porter da olduğu gibi örneğin "ingly" gibi bir ekle biten "sportingly" kelimesi köke indirgenerek "sport" halini alır. Buna göre de arama ve sonuçlar elde edilir. Snowball algoritması Porter algoritmasına göre kök bulma işleminde daha sert kurallara sahiptir.

İki algoritma arasındaki en temel fark snowball algoritmasının daha doğru kelimelere yönlendirme yapmasıdır.

### 3.3.8 Lancaster Stemmer Algoritması

Porter ve Snowball algoritmalarına göre daha çok köke indirgeme durumudur. Kök bulma işlemi sırasında daha ileri seviyede kurallara uyum söz konusudur. İki algoritmaya göre dinamik bir yapısı vardır. İleri seviye kök bulma işleminden dolayı daha küçük kelimelerke uğraşması Lancaster algortimaysını daha karmaşık hale getirir.

Lancaster stemmer, kuralları harici olarak kaydeder ve temel olarak yinelemeli bir algoritma kullanır. Kurallara uyumluluğu sebebiyle yeni kuralların eklenmesinde esnek bir yapıdadır.

Örneğin kök bulma işleminde kullanılan kelime "considerations" olsun. Porter ve Snowball içerisinde bu kelimenin kök bulma işlemi "considerat" veya "consider" gibi daha az köke indiregenerek yapılır. Lancaster stemmer da ise daha ileri gidilerek "consid" kelimesi üzerinden kök bulma işlemi yapılır. Buradan da anlaşılacağı gibi Lancaster daha küçük kelimeler üzerinden işlem yapar. Bu yüzden karmaşıklaşır ve Snowball algoritmasına göre daha verimsizdir.

### 3.3.9 YASS Stemmer Algoritması

Kök bulma işlemini bir kelime problemi olarak ele alır. Yani verimli bir çalışma çıkarmak için hesaplama kaynağına ihtiyaç duyar. Uzun olan ortak ön eke sahip iki kelimeyi benzer olarak kabul eder. Bu kelimelerden anlamlı kümeler oluşturmak için eşik değeri seçer. Bu algoritmanın amacı dilden bağımsız olarak işlenmesidir .

### 3.3.10 HMM Stemmer Algoritması

Kelime harfleri gizli durumlar olarak kabul edilir ve kök durumundan sonek durumuna geçiş, bölünme noktası olarak alınır. Yani gizli olarak aldığı her olası çıktı için bir olasılık seçer ve en olası olanı alır.

### 3.3.11 Corpus Based Stemmer Algoritması

Korpus dilbiliminde, istatistiksel analiz ve hipotez testi yapmak, oluşumları kontrol etmek veya belirli bir dil bölgesindeki dil kurallarını doğrulamak için kullanılır. Corpus based stemmer ise herhangi bir insan müdahalesi veya dile özgü bilgi olmadan

korpustan morfolojik olarak ilişkili kelimeleri keşfetmek için bilişsel-ilhamlı hesaplama gerçekleştirir.

Bu algoritma, birkaç dilde kural tabanlı olanlar da dahil olmak üzere bir dizi güçlü kök ayırıcıdan önemli ölçüde daha iyi performans gösterir.

### 3.3.12 Context Sensitive Stemmer Algoritması

Porter kök bulmanın değiştirilmiş bir versiyonudur. Burada bağlama duyarlıdan kasıt uygulama alanı ve kullanıcı bağlam bilgisine dayanarak, dinamik değişim yada adaptasyon

yeteneğine sahip olma durumudur. Sonuç olarak bağlam kullanma temelinde kullanıcı işlemlerine bağlı olarak uygun bilgiyi ve hizmetleri sağlayan sistemi, bağlama-dayalı sistem olarak adlandırır.

Porter'ın kelime içindeki durumlarına bakışı denebilir. Örnek olarak Porter üzerinden alınan bir kelime kökünün büyük, küçük harf durumuna bakılarak değiştirilmiş hali düşünülebilir.

### 3.3.13 Paice/Husk Stemmer Algoritması

Dilde iki ana stemmer ailesi bulunmaktadır. Bunlardan algoritmik stemmerlar bilinmeyen sözcüklerin kökünün alınmasını sağlar. Dezavantajı daha yüksek hata oranına sahip olmasıdır fakat bu durum bir arada bulunması gereken kelimeleri birleştirilmesini sağlar.

Paice/Husk algoritması stemmerın algoritmik ailesine girer. Kök bulma için birçok kurala dayanır fakat bu kuralları kodunun dışında tutar. Böylece yeni bir dili yeni bir dili, kodunu yeniden yazmadan, sadece birkaç ayarlama ile başka bir kurallar kümesini kullanarak işlemek mümkündür. Bu avantaj sayesinde yeni bir dilin işlenmesi kolaylaşır.

- Paice/Husk algoritmasının işleyişi aşağıdaki adımlarla sağlanır;
    - Öncelikle yeni dilin tüm aksanlı harfler dahil yeni bir kural kalıbı oluşturulur.
    - Alınan yeni dilin tüm ünlü harfleri listelenir.
    - Dil içinde sık olarak kullanılan kelimeler alınır ve frekansları elde edilir.
    - Frekansa göre sıralanmış listenin ilk kelimesi alınır ve test dosyasındaki bilgiler ile ilişkilendirilir.
    - Son olarak Paice/Husk algoritması çalıştırılır.

    Bulunan kök beklentiyle uyuşmuyorsa tekrar Paice/Husk algoritması içindeki kurallara ekleme gerekiyorsa eklenmeli, değiştirilmesi gerekiyorsa değiştirilmelidir. Paice/Husk algoritması test dosyasından her biri için "good" kök döndürene kadar test dosyasındaki bazı gövdeleri değiştirmek gerekir. Burada hangi kelimelerin aynı köklere sahip olacağını ve bu kuralları kullanacak olan stemmerın gücünü belirlemek için seçimler yapmamız gerektiğini görebiliriz.

## 4.Zemberek Kütüphanesi Kullanılarak Stemming(kök bulma) İşlemi

kubra_demir_190303090.py*

```python
#type-annotations -> bir nevi kodun daha iyi okunmasını sağlamak için
from typing import List
# Zemberek kütüphanesi Java temelli olduğu ve zemberek kütüphanesini kullanmak için
from jpype import JClass, isJVMStarted, getDefaultJVMPath, startJVM, java

ZEMBEREK_PATH = r'C:\Users\kubradmr\zemberek-full.jar' #zemberek kütüphanesinin bulunduğu konum

#JVM'i açıp kapatmamız gerekiyor eğer halihazırda açıksa tekrar açmaya çalışınca hata veriyor
#Bu hatayı önlemek önce açık olup olmadığını kontrol edip ona göre açmamız lazım
if not isJVMStarted():
    startJVM(getDefaultJVMPath(), '-ea', '-Djava.class.path=%s' % (ZEMBEREK_PATH))

#Burası zemberek kütüphanesini JVM aracılığıyla tanımladığımız/kullandığımız kısım
TurkishMorphology = JClass('zemberek.morphology.TurkishMorphology')
morphology = TurkishMorphology.createWithDefaults()

#Köklerini bulacağımız kelimeleri buraya yazıyoruz
kelimeler = 'ademimerkeziyetçilik çekoslovakyalılaştıramadıklarımız muvaffakiyetsiz köklerine'

#Python'da List Java'da ArrayList üzerinden kök analizi yaptığımız kısım
analysis: java.util.ArrayList = (morphology.analyzeAndDisambiguate(kelimeler).bestAnalysis())

#Doğal dil işlemedeki POS taglar için
pos: List[str] = []

#Ve Listeyi gezip köklerini buluyoruz
for i, analysis in enumerate(analysis, start=1):
    f'\nAnalysis {i}: {analysis}',
    f'\nPrimary POS {i}: {analysis.getPos()}'
    f'\nPrimary POS (Short Form) {i}: {analysis.getPos().shortForm}'

    #Kök analizi yaptığımız kelimelerin köklerini listeye ekliyoruz
    pos.append(f'{str(analysis.getLemmas()[0])}')

#Ekran çıktısı
print(f'\n Kelime Kökleri: {" ".join(pos)}')
```

Şekil 4.1 Zemberek Stemming İşlemi

```
 Kelime Kökleri: ademimerkeziyet çekoslovakyalı muvaffakiyet kök

I|15:54:35.644|Root lexicon created in 419 ms.
| DictionarySerializer#getDictionaryItems
I|15:54:35.645|Dictionary generated in 533 ms
| RootLexicon#defaultBinaryLexicon
I|15:54:35.965|Initialized in 895 ms.
| TurkishMorphology#createWithDefaults
I|15:56:06.826|Initialized in 105 ms.
| TurkishMorphology#createWithDefaults
I|18:09:15.674|Initialized in 87 ms.
| TurkishMorphology#createWithDefaults
I|18:10:03.811|Initialized in 181 ms.
| TurkishMorphology#createWithDefaults
I|18:10:52.508|Initialized in 77 ms.
| TurkishMorphology#createWithDefaults
I|18:11:11.476|Initialized in 81 ms.
| TurkishMorphology#createWithDefaults
I|18:11:39.403|Initialized in 75 ms.
| TurkishMorphology#createWithDefaults
I|18:11:55.917|Initialized in 85 ms.
| TurkishMorphology#createWithDefaults
I|18:12:15.227|Initialized in 91 ms.
| TurkishMorphology#createWithDefaults
I|18:12:26.866|Initialized in 72 ms.
| TurkishMorphology#createWithDefaults
I|18:12:31.996|Initialized in 81 ms.
| TurkishMorphology#createWithDefaults
I|18:13:04.654|Initialized in 97 ms.
```

Şekil 4.2 Zemberek Stemming Çıktısı

# İKİNCİ BÖLÜM

## VEKTÖR UZAY MODELİ NEDİR?

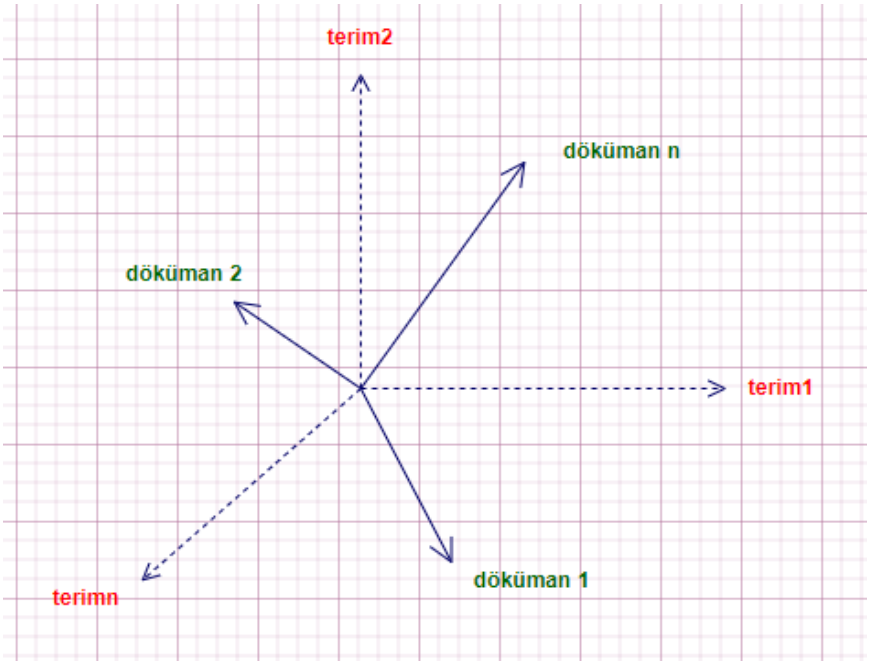
Bilgisayar bilimlerinde özellikle istatistiksel doğal dil işleme(NLP), metin madenciliği (text mining), bilgi erişim(IR) gibi konularda vektör uzay modeli(vector space model) kullanılmaktadır.

Vektörün sonsuz sayıda boyutu olabilir. [1,2,3,2,1,3] şeklinde 6 boyutlu bir vektörden bahsedebiliriz. Bu vektörün gösterimi veya çizilmesi gerçekten gerekmemektedir çünkü amaç birbiri ile ilişkisi olmayan boyutlarda farklı değerlere sahip bir vektörü temsil etmektir. Buradaki önemli nokta bu boyutların arasında ilişki bulunmamasıdır. Diğer bir deyişle birinci boyutun kaç olduğunun 2. boyuta veya ikinci boyutun kaç olduğunun 3. boyuta bir etkisi yoktur.

Vektör uzay modeli yöntemine göre herhangi bir metin veya metin parçasını vektörel olarak göstermek ve bir uzay içerisinde modellemek mümkündür.
Metinler için kavramları, x ve y koordinatlarının dokümanları ifade etmesi durumudur. Kısaca vektör uzay modeli bir indeks teriminin vektörü olarak metin dokümanlarını sunan cebirsel bir modeldir.

### 1. Vektör Uzay Modelinin Bilgi Erişimdeki Rolü

Bilgi erişim metin sorgulama sistemi, belgelerden çıkarılan belirteçleri indekslemek için vektör uzay modelini kullanır. Belirteçler, her belge metni aracılığıyla ayrıştırılarak noktalama işaretleri, boşluklar gibi detayların temizlenmesinin ardından vektör uzay modeli kavramlarına dayalı olarak indekslenir. Aranan kelime, dökümanlar içindeki sıklığı ve ilişkili belge dosyaları hakkında bilgi içeren bir sözlükte saklanır.



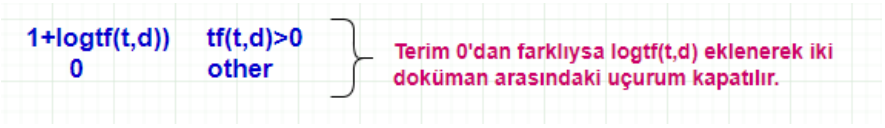

Şekil 1.1 Bilgi Erişimde Vektör Uzay Modeli

Bilgi erişimde bu modelin temeldeki işlevi aranan kelimenin sıklık durumudur. Döküman metninden çıkarılmış veya deneyimli dizinleme yapan kişiler tarafından elle atanan terimler ve sorgudan çıkarılmış terimler, döküman içeriğini belirlemede kullanılabilir. Her iki durumda da dökümanlar terim vektörleri olarak gösterilebilir. Belge vektörleri bir araya getirilerek bütün belgeleri içeren matris oluşturulur.

## 1.1 Vektör Uzay Modelinin Analizi

### 1.1.1 Term Frequency(tf) :

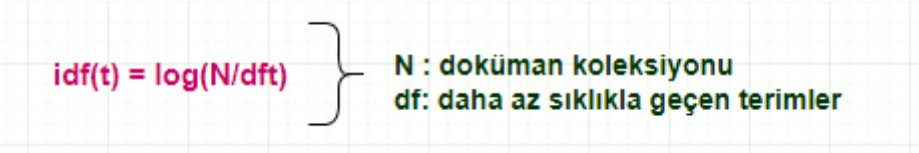
Terimin sıklığını ölçer. Terim sıklığı, bir terimin bir belgede ne sıklıkta geçtiğinin ölçümüdür. En kolay hesaplama, bir kelimenin kaç kez göründüğünü saymaktır. Ancak, belgenin uzunluğuna veya belgede en sık kullanılan kelimenin sıklığına göre bu değeri değiştirmenin yolları vardır. Gösterimde tf(t,d) durumu d dokümanında t terimin sıklığını verir. Term frequency ölçüsü tek başına yeterli değildir. Örneğin A dokümanında terimin 10 kere geçmesi ve B dokümanında 1 kere geçmesi arasında karşılaştırma yapıldığında A dokümanında daha fazla geçiyor diye daha ilişkilidir denmesi yanlış bir ifadedir. Hesaplama içerisinde eğer terimin sıklığı 0'dan farklıysa 1+logtf(t,d) eklemesi yapılarak iki doküman arasındaki uçurum kapatılır. Term frequency için alınan kelimeler ham haldedir. Ham kelime sıklığını kullanmanın sorunu, alaka düzeyinin kullanımla orantılı olarak artmamasıdır. Bunun için term frequency nin log-frequency versiyonu kullanılmaktadır.




Şekil 1.1.1 Term Frequency

### 1.1.2 Document Frequency(df) :

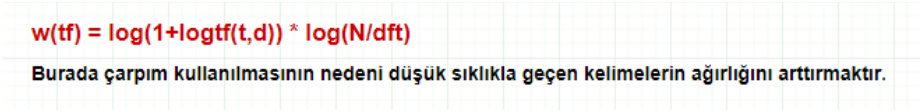
Az sıklıkla geçen kelimeler çok sıklıkla geçen kelimelere göre daha çok bilgi verir. Document frequency de az sıklıklı kelimelere daha yüksek ağırlık verilir. Çeşitli kullanım yöntemleri vardır. Örneğin stop wordsleri yani önemsiz kelimeleri analizden çıkartmaktır. Document frequency nin kelimelerin sıklığına, önemine göre arttırıp, azaltmasıyla verilen ağırlıklara inverse document frequency(idf) denir. Burada ele alınan document frequency ne kadar yüksekse inverse document frequency(idf) o kadar düşük çıkmasıdır. Bu durum bilgi erişimde konuyla ilgili dokümanların ağırlıklarının konu dışı dokümanların ağırlıklarına göre daha yüksek olmasını sağlayan bilgi erişim algoritmalarında kullanılır. Sonuç olarak df ve idf ters orantılıdır.



Şekil 1.1.2 Document Frequency

#### 1.1.2.1 Collection frequency(cf) ve document frequency(idf):

Dokümanda geçen tüm terimlerin sayısıdır. Örneğin "try" ve "insurance" kelimelerinin sırasıyla cf değerleri 10422 ve 10440 olsun. Df değerleri ise sırasıyla 8760 ve 3997 olsun. Burada "try" doküman içinde 10422 ve "insurance" 10440 kere geçmiştir. Fakat collection frequency ayırt edici bir özellik olmadığından burada arama terimi olarak df kullanılır. Document frequency e bakıldığında ise arama terimi olarak "insurance" alınmalıdır. Bulunduğu döküman sıklığı daha azdır.




Şekil 1.1.2.1 Collection Frequency(cf) ve Document Frequency(idf)

#### 1.1.2.2 Term frequency(tf) ve inverse document frequency(idf) :

Bir kelimenin bir doküman için ne kadar önemli olduğunu yansıtmayı amaçlayan istatistiksel yapıdır.
Genellikle bilgi erişim, metin madenciliği ve kullanıcı modelleme aramalarında bir ağırlıklandırma modeli olarak kullanılırlar. tf–idf değeri orantılı olarak artar. Bir sözcüğün belgede görünme sayısına ve cümledeki sözcüğü içeren belge sayısına göre dengelenir. Bu, bazı sözcüklerin genel olarak daha sık göründüğü gerçeğini düzeltmeye yardımcı olur. tf-idf değerleri genellikle arama motorları tarafından verilen kullanıcı sorgusunun bir dokümanla ilişkisini puanlama ve sıralamada kullanılır. tf-idf dokümanları özetleme ve sınıflandırma gibi konularda stop wordsleri filtreleme konusunda başarıyla kullanılmaktadır.

Vektör içinde kelimelere ait sayısal bir değer olur. Örneğin o dokümanda var olup olmaması (0, 1) değerleri ifade edilebilir. Bu değerlere ağırlık(w) denir. Bir metnin vektörel olarak gösterilmesinde birçok farklı yöntem vardır. Bunlardan bazıları metin etiketleme(POS), n-gram sayımı, LSA gibi yöntemlerdir. Ağırlık değerlerini ifade etmek için en çok bilinen yöntem ise yukarıda da belirtildiği gibi tf.idf(term frequency–inverse document frequency) ağırlıklandırma yöntemidir.

Örneğin;

**doc1 = "kırmızı renk ayakkabı"**

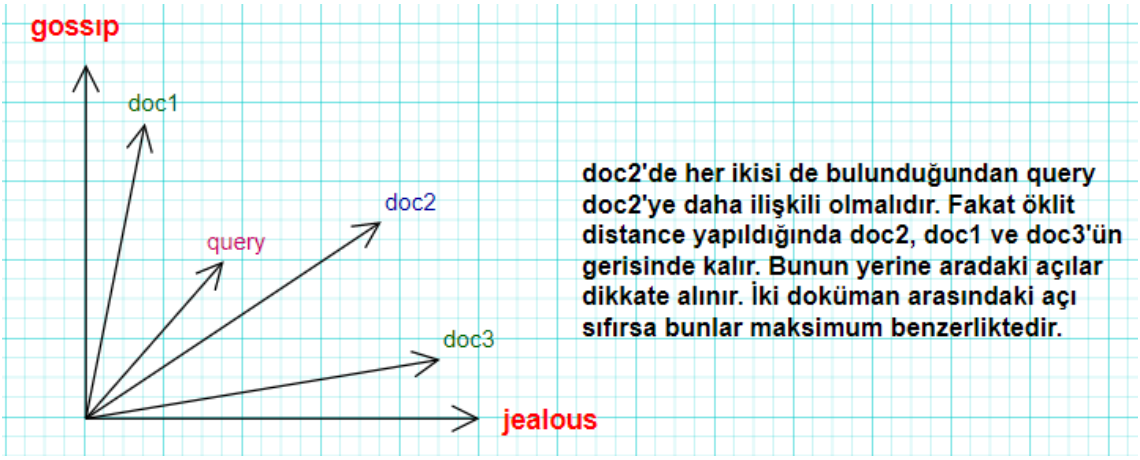
**doc2 = "kırmızı renk elbise"**

**doc3 = "sarı çizgili ayakkabı"**

Yukarıda verildiği gibi dokümanlar içinde geçen kelimelerin vektör içindeki sayısal değerleri tf-idf ile arandığında öncelikle tf değerleri her bir doküman içinde "kırmızı, renk, ayakkabı, elbise, sarı, çizgili" kelimelerinin sıklığına göre bulunur. Daha sonra idf değerleri için tüm doküman sayısının toplam kelime sayısına oranının iki tabanında logaritması alınarak bulunur. Bu iki adımdan sonra bir query oluşturulup bu query üzerinden tf.idf değerleri bulunur. Çıkan vektörel formulün sonuçlarına göre örneğin doc1 dokümanı query e en yakın kabul edilirken doc2 ve doc3 benzer doküman olarak alınır.

## 2.Vektör Uzay Modeli Öklit Uzaklığı

Dokümanlar bir vektör uzayında alındığı zaman buradaki amaç en yakın dokümanı bulmaktır. Böyle bir amaç için vektör uzay modelinde mesafe(distance) değeri bulunmalıdır. Mesafe yani iki nokta arasındaki uzaklık için kullanılan yöntemlerden biri öklit uzaklığıdır. Öklit uzaklığı, bir nevi benzerlik metriğidir. Bu fikir temelde iyi değildir. Örneğin "gossip" kelimesi doc1'de "jealous" kelimesi doc3' de daha fazla geçmiş olsun. Her iki kelimede doc2'de ortak olarak bulunsun. Bu durum için aşağıdaki gibi bir sonuç elde edilir.




Şekil 2.1 Öklit Uzaklığı

## 3.Vektör Uzay Modelinin Benzerlik Ölçütleri

- Vektör uzay modelinin similarity(benzerlik) konusunda dört ana tekniği vardır. Bunlar;
    1. Inner Product(iç çarpım)
    2. Cosine Similarity(kosinüs benzerliği)
    3. Dice Similarity(zar benzerliği)
    4. Jaccard Similarity(jaccard benzerliği)

3.1 İnner Product(iç çarpım) : İç çarpımda ele alınan fikir her belgeyi bilgi erişim sisteminde pozitif vektörler olarak sorgulanması ve bu belgelerin sorgu ile arasındaki benzerliğin bulunmasıdır. Query nin vektörünü yön bakımından benzer olan vektörün querysinin benzeri olarak kabul edilecektir. Yani bir belgenin içerisinde çarpımlar yapılarak benzerlik bulma ölçütüdür.

3.2 Cosine Similarity(kosinüs benzerliği): Veri analizinde iki sayı dizisi arasındaki benzerliğin ölçüsüdür. Diziler iç çarpım uzayında vektörler olarak bilinir ve kosinüs benzerliği aralarındaki açının kosinüsünü yani vektörlerin nokta çarpımının uzunluklarının çarpımı ile bölünmesi olarak tanımlanır. Buradan, kosinüs benzerliğinin vektörlerin büyüklüklerine değil, sadece açılarına bağlı olduğu sonucu çıkar. Kosinüs benzerliği özellikle sonucun düzgün bir şekilde sınırlandığı pozitif uzayda kullanılır. Pozitif uzay iki orantılı vektörün kosinüs benzerliğinin 1 olması durumudur. Bilgi erişimde her kelimeye her kelimeye farklı bir koordinat atanır ve bir belge, belgedeki her kelimenin oluşum sayısının vektörü ile temsil edilir. Kosinüs benzerliği, daha sonra, konularına göre ve belgelerin uzunluğundan bağımsız olarak, iki belgenin ne kadar benzer olabileceğinin yararlı bir ölçüsünü verir. Vektör uzay modeli öklit uzaklığı örneği cosinüs benzerliği için kullanılabilir. Buradaki dokümanlar arasında uzaklık yerine açının kosinüs değerine bakılarak daha verimli bir benzerlik bulunur.

3.3 Dice Similarity(zar benzerliği): Dice katsayısı da denilen dice similarity doküman benzerliğinde kullanılan bir ölçüttür. Burada, benzer kelime sayısı toplam kelime sayısından çıkarma işlemi yerine 2 ile çarpılmaktadır. Tüm ölçütlerde [0,1] arasında bir değer üretilmektedir. Yapılan ilk işlem iki doküman arasında özellikelrin çıkarılıp D(A,B) = 2|AandB|/(|A| + |B|) formülüne göre benzerlik hesabının yapılmasıdır. Örneğin doc1'de "bilgi" doc2'de ise "bilim" kelimeleri bulunsun. Bu iki doküman arasında özellikler ortaya çıkarılır. Çıkarılan özellikler örneğin her harf için veya bigram kullanılarak bulunur. Burada bigram kullanıldığı varsayılırsa sonuçlar doc1 için {bi,il,lg,gi} ve doc2 içinse {bi,il,li,im} şeklinde olacaktır. Dice similarity ye göre bu iki kümenin kesişim sayısı 2 ve kümelerin eleman sayısı 4 bulunacaktır. Dice similarity formülünde gerekli yerlere yazıldığında çıkan sonucun yüksek olması benzerliğin fazla olduğunu gösterir.

3.4 Jaccard Similarity(jaccard benzerliği): Jaccard benzerliği, hangi üyelerin paylaşıldığını ve farklı olduğunu görmek için iki veri seti arasındaki benzerliği ölçer. Jaccard benzerliği, her iki kümedeki gözlem sayısının herhangi bir kümedeki gözlem sayısına bölünmesiyle hesaplanır. Başka bir deyişle, Jaccard benzerliği, kesişimin boyutunun iki kümenin birleşiminin boyutuna bölünmesiyle hesaplanabilir. Kısaca iki asimetrik değişken arasındaki benzerliktir. Örneğin elimizde 0 ve 1 gibi iki değişken olsun. Jaccard katsayısı bu iki değişkende nerede 0 yok onu bilmemizi sağlar ve 0'ın olmaması 1 değişkeninin olduğu anlamına gelir. Her durum simetrik ikili nitelikler için eşit derecede değerliyken, bu iki durum asimetrik ikili değişkenlerde eşit derecede önemli değildir. Bunlara ek olarak Dice katsayısı üçgen eşitsizliğini karşılamadığından , Jaccard indeksinin semimetrik versiyonu olarak kabul edilir.

## 4.Vektör Uzay Modeli Avantaj ve Dezavantajları

Vektör uzaylarının kullanılmasının avantajları;

- Doğrusal cebir (linear algebra) kullanılarak işlenebilen veri yapılarının elde edilmesi
- İkilik tabandaki sayılar yerine ağırlıkların hesaba katılabilir olması
- Vektörler arasında tanımlı olan bütün fonksiyonların metinler arasında da tanımlanabilir olması (örneğiN kosinüs benzerliği gibi)
- Metinler üzerinde sıralama (ranking) fonksiyonlarının çalıştırılabilir olması
- Metnin tamamı yerine bir parçası üzerinde çalışabilir olması

Vektör uzaylarının dezavantajları;

Vektör uzaylarının kullanıldığında çok yüksek miktarda özellik içeren veriyle uğraşılması gerekir. Örneğin twitterda paylaşılan metinlerin verilerini çektiğimizde, çekilen metinlerin bir dildeki karşılığında birçok farklı kelimenin kullanıldığını ve bu kelimeleri tutan özellik vektörünün yüksek gb değerlerinde olduğu görülür. Bu bilgi verisini bilgisayarın RAM'ine yüklesek bile işlemek için yer kalmadığı ve özel yöntemler geliştirmenin gerektiği görülür. Terim frekansının yanında anlambilimsel yaklaşımlar kullanılarak bu boyutların azaltılması mümkündür.

Vektör uzay modelinin en büyük dez avantajlarından birisi, metin boyutu uzadıkça kullanışsız hale gelmesidir. Çünkü uzayan metinler birbirine benzemeye başlar ve metinleri ayırt etmede önemli rol oynayan kelime farklılıkları azalır. Ayrıca kelimeler arasındaki benzerlikler göz ardı edilmektedir. Örneğin ağaç kelimesi ile ağaçlar kelimesi farklı iki kelimedir ve farklı metinlerde geçmesi halinde iki metin birbirinden farklı olarak sınıflandırılacaktır. Oysaki metinler arasında anlamsal bağlantılar olabilir. Ayrıca metin içerisindeki kelimelerin sırası da vektör uzay modelinde kaybedilmektedir. Örneğin bir kelimenin nerede geçtiği önemsizdir.

.

# KAYNAKLAR


https://melikebektas95.medium.com/zemberek-kütüphanesi-ile-türkçe-metinlerde-kelime-köklerinin-bulunması-6ddd3a875d5f

https://www.emo.org.tr/ekler/c7a625d5077d3ba_ek.pdf?dergi=483

https://tr.wikipedia.org/wiki/Doğal_dil_işleme

https://www.veribilimiokulu.com/dogal-dil-isleme-nedir-ve-uygulama-alanlari-nelerdir/

http://zembereknlp.blogspot.com/

http://1.bp.blogspot.com/-Dg322Pt9uFs/TaGVKBuVegI/AAAAAAAARpQ/yR0d630Wh1U/s1600/Zemberek3-Blok.png

https://burcu-can.github.io/BurakOzturk-MscThesis.pdf

https://en.wikipedia.org/wiki/Stemming

https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_stemming_lemmatization.htm

https://devopedia.org/stemming

https://xapian.org/docs/stemming.html

https://vijinimallawaarachchi.com/2017/05/09/porter-stemming-algorithm/

https://en.wikipedia.org/wiki/Julie_Beth_Lovins

https://www.researchgate.net/publication/311931154_A_Comparative_Study_of_Stemming_Algorithms_for_use_with_the_Uzbek_Language

https://www.geeksforgeeks.org/snowball-stemmer-nlp/#:~:text=Snowball

https://krithikanagi.medium.com/an-overview-of-stemming-algorithms-501ad413653

https://www.tutorialspoint.com/natural_language_toolkit/natural_language_toolkit_stemming_lemmatization.htm

http://alx2002.free.fr/utilitarism/stemmer/stemmer_en.html

https://www.semanticscholar.org/paper/An-Efficient-Corpus-Based-Stemmer-Singh-Gupta/92798684d83f14f86d3408a1c7ed3c474d8d026d#extracted

https://www.emo.org.tr/ekler/8995418dd0b5d24_ek.pdf

https://www.geeksforgeeks.org/introduction-to-stemming/

https://www.diagrameditor.com/webapp/?splash=0&ui=atlas&tr=0&gh=0&gl=0&gapi=0&od=0&db=0&lang=en

https://towardsdatascience.com/stemming-lemmatization-what-ba782b7c0bd8

https://github.com/words/lancaster-stemmer

https://egealpay1.medium.com/hidden-markov-models-saklı-markov-modeli-b381380d0aca

https://www.researchgate.net/publication/221300044_A_novel_corpus-based_stemming_algorithm_using_co-occurrence_statistics

https://en.wikipedia.org/wiki/Text_corpus

http://www.openaccess.hacettepe.edu.tr:8080/xmlui/bitstream/handle/11655/2607/cee531a9-5d87-4eec-bd4d-559346fb0a91.pdf?sequence=1&isAllowed=n

http://alx2002.free.fr/utilitarism/stemmer/stemmer_en.html

https://quick-adviser.com/what-is-overstemming-and-understemming/

https://towardsdatascience.com/stemming-of-words-in-natural-language-processing-what-is-it-41a33e8996e2#:~:text=Over

https://bilgisayarkavramlari.com/2012/12/10/vektor-uzay-modeli-vector-space-model/

https://blog.marketmuse.com/glossary/term-frequency-definition/#:~:text=Term

https://miramirkhani.ir/project/vector-space-information-retrieval-model/

https://kavita-ganesan.com/what-is-document-frequency/#.Ylq0sdpBzIU

https://erdincuzun.com/bilgi_erisimi/vektor-uzayi-modeli-vector-space-model/

https://bilgisayarkavramlari.com/2012/12/10/vektor-uzay-modeli-vector-space-model/

https://en.wikipedia.org/wiki/Tf–idf

https://medium.com/novaresearchlab/doğal-dil-işleme-serisi-3-vektör-uzayı-modelleri-de84d4a4ec88

https://en.wikipedia.org/wiki/Cosine_similarity

https://towardsdatascience.com/vector-space-models-48b42a15d86d

https://bilgisayarkavramlari.com/2013/07/01/sorenson-dice-katsayisi-dice-sorensen-coefficient/

https://www.learndatasci.com/glossary/jaccard-similarity/#:~:text=The

https://stringfixer.com/tr/Sorenson_index

https://drive.google.com/drive/u/0/folders/1FN80VbqesnqU21us4c4Pvgv2VqUsSf2z

https://ugurozker.medium.com/zemberek-nlp-7add032881e9

https://github.com/ozturkberkay/Zemberek-Python-Examples

https://github.com/Loodos/zemberek-python

https://github.com/ahmetaa/zemberek-nlp