# DUPLICATE RECORD DETECTION: A SURVEY (2007)

**Proje Notu:** Veritabanlarının üzerinde bulunan benzer kayıtların çıkarılması için kullanılan geleneksel algoritmaların anlatıldığı bir araştırma yayınıdır.

Bu çalışmada kayıt çakışması tespiti üzerine bir literatür araştırması yapılmıştır. Kayıt çakışması tespiti, aynı veya farklı veritabanlarındaki, tamamen aynı olmasa bile, benzer veriyi temsil eden kayıtları tespit etmektir. Benzer veri girişlerini tespit etmek için string benzerliklerini bulma yöntemlerine dayanan farklı algoritmalar ile verilerin benzerlik metrikleri ele alınmıştır. Algoritmaların verimliliği ve ölçeklendirilebilirliğini genişletmek için farklı teknikler karşılaştırılmıştır. Yapılsal farklılıklar ve sözcüksel farklılıklara üzerinden veri benzerliğini bulmaya çalışan algoritma çeşitleri bulunmaktadır.

Bu çalışmada, sözcüksel farklılıklar üzerine yoğunlaşılmıştır. Sözcüksel farklılıklar aranırken girdi kümesi yapılandırılmış ve uygun şekillerde bölümlere ayrılmış bir veri kümesinden oluşmaktadır. Verilerin benzerliklerini tespit etmek için çeşitli algoritmalar kullanılmış ve sonuçları karşılaştırılmıştır. Sözcüksel farklılıklara dayanan algoritmalar, karakter tabanlı benzerlik, simge tabanlı benzerlik, fonetik benzerlik ve sayısal benzerlik metrikleri olmak üzere dört ana parçaya ayrılmaktadır. Fakat bu metrikler tek başına uygulandığında çok yüksek bir hata oranı vermektedir. Tüm veri kümeleri için uygun tek bir benzerlik metriği bulunmamaktadır. Bu nedenle birden fazla benzerlik metriğinin bir arada kullanımını öneren yaklaşımlar bulunmaktadır.

Sözcüksel farklılıklara dayanan algoritmalarda veri bütün bileşenleri ile alınmak yerine özellik özellik ele alınır ama aslında veri bütün özellikleri ile beraber bir bütündür. Bizim için önemli olan bütün bir kaydın tekrar etmemesidir. Bu nedenle, veriler karşılaştırılırken birçok özellik göz önüne alınmalı ve birden fazla metriğin beraber kullanımını öneren yaklaşımlar kullanılmalıdır. Denetimli makine öğrenme algoritmalarının kullanımı ile veri benzerliklerini elde etme ve alan bilgisi ve genel mesafe metriklerini kullanarak veri benzerliklerini elde etme olmak üzere temelde iki farklı yaklaşım bulunmaktadır.

# PERFORMANCE ANALYSIS OF DUPLICATE RECORD DETECTION TECHNIQUES (2019)

**Proje Notu:** Veritabanları üzerinde benzer kayıtların filtreleme yöntemleri ile çıkarılmasının ETL süreçlerine uygun olarak gerçekleştirilmesinin gösterildiği akademik çalışmadır.

Bu çalışmada, ilişkisel veritabanları için mükerrer kayıt tespit yöntemlerinin kapsamlı bir performansı analizi uygulanmıştır. Ayıkla, Dönüştür ve Yükle (ETL) aşamaları uygulanarak, ekleme işlemi gerçekleştirilirken veritabanında zaten var olan kayıtları bulmak ve ortadan kaldırmak için geleneksel SQL tabanlı ve modern bloom filter tekniklerine odaklanmaktadır. Veri tabanlarında kullanılan ISAM, B-Tree, Bitmap, and Hash indexing tekniklerine rağmen mükerrer kayıt tespiti hala bir problem teşkil etmekte ve zaman almaktadır. Mükerrer kayıt kaydının tespit edilmesinin, dağıtık veri tabanlarında oldukça önemli bir problem olduğu bilinmektedir. Burada dağıtık veritabanları birleştirilirken ETL teknikleri kullanılarak mükerrer kayıtlar tespit edilmeli ve silinmelidir. Bloom filter, ilişkisel veritabanlarında mükerrer kayıt tespitinde geleneksel yöntemlere oranla potansiyel olarak daha verimli bir şekilde uygulanabilmektedir. Bu nedenle, bu çalışmada veritabanlarında mükerrer kayıt tespitinde bloom filter’ın uygulamaları üzerine çalışılmakta ve bloom filter, modern bloom filter ve SQL karşılaştırılmaktadır.

Bloom filter yöntemi hash fonksiyon tabanlı bir yöntemdir. False negatif ihtimali 0’dır, fakat false pozitif ihtimali 0’dan yüksektir. Parametre seçimi ile false pozitif ihtimalini minimize etmek mümkündür. False pozitif, filtrenin zaten varolan bir verinin yeni bir girdi olmamasına rağmen yeni girdi olarak kaydedilme ihtimali olarak tanımlanmaktadır. Ekleme ve arama bloom filter’da oldukça hızlı gerçekleştirilmektedir. Silme ek işlemlere ihtiyaç duyduğu için genellikle gerçekleştirilmesine izin verilmez. Bloom filter seçilirken artıları ve eksilerini karşılaştırırsak alan açısından artıları ile false pozitif açısından eksileri karşılatırılmaktadır. Bizim sistemimiz için alan(space) daha önemli ise ufak bir false pozitif oranı ile bloom filter bizim için en uygun seçim olmaktadır. Fakat ufak bir false pozitif bile bizim veri tabanımız için tolore edilemeyen bir hata oluşturuyorsa, bu durumda bloom filter bu veri tabanında kullanılamaz. Bloom filter’ın, counting bloom filter, d-Left counting bloom filter, compressed bloom filter, bloomier filter, space-code bloom filter, dynamic bloom olmak üzere birçok farklı versiyonu bulunmaktadır.

Bloom filtrelerinin uygulamaları, yazım denetimi, P2P ağlarında işbirliği, kaynak ve paket yönlendirme, önbellek optimizasyonu, URL kısaltma, video önerisi, dize eşleme, spam filtreleme, DoS ve DDoS algılama alanlarında kullanılabilir ancak kullanım alanları bunlarla sınırlı değildir.

Bu araştırmada, farklı sayıda tuple (yani tablolarda bin ila bir milyon tuple) kullanarak veritabanına, dağıtılmış veritabanına ve veri ambarına toplu ekleme işlemi gerçekleştirirken mükerrer kayıtların algılanması için SQL, çiçek filtresi ve paralel çiçek filtresi uygulanmıştır.

## Önerilen Metodoloji

Mükerrer kayıtların bulunması büyük veritabanları için oldukça zahmetli bir iş olmaktadır. Çünkü eklenen her verinin veritabanında bulunan bütün veriler ile karşılaştırılması gerekmektedir. Yalnızca birincil anahtarların karşılaştırılması ile verilerin karşılaştırılmış olarak varsayamayız çünkü veriler farklı veritabanlarından veya kaynaklardan çekilmektedir. Bu çalışmada, SQL, bloom filter, paralllel bloom filter olmak üzere üç farklı yaklaşım kullanılmakta ve farklı kayıt sayısına göre mükerrer kaydın tespitindeki performansları karşılaştırılmaktadır.

### SQL tabanlı yaklaşım

SQL tabanlı yaklaşım ile mükerrer kayıt tespit etme adımları:

1. **Adım:** Çıktı dosyasındaki veri uygulama tarafından yüklenir.
2. **Adım:** Dosyadan bir sonraki kayıt çekilir, çekilen kayıt veritabanında varsa 3. Adıma geçilir, yoksa işlem sonlandırılır.
3. **Adım:** Kaydın anahtar değeri hariç bütün sütunları birleştirilir.
4. **Adım:** Birleştirilen kayıt bütün veritabanı (birleştirilmiş veritabanı) ile karşılaştırılır.
5. **Adım:** Eğer kayıt veritabanında yoksa gerçek tabloya eklenir, varsa mükerrer kayıt tablosuna eklenir ve 2. Adıma geri dönülür.

### Bloom filter tabanlı yaklaşım

Bloom Filter tabanlı yaklaşım ile mükerrer kayıt tespit etme adımları:

1. **Adım:** Çıktı dosyasındaki veri uygulama tarafından yüklenir.
2. **Adım:** Dosyadan bir sonraki kayıt çekilir, çekilen kayıt veritabanında varsa 3. Adıma geçilir, yoksa işlem sonlandırılır.
3. **Adım:** Kaydın anahtar değeri hariç bütün sütunları birleştirilir.
4. **Adım:** Birleştirilen kayıt bütün veritabanı (birleştirilmiş veritabanı) ile karşılaştırılır. Bloom filtresinin, kaynak tabloya eklenen her kayıt için güncellenmesi gerekir. Bu nedenle, bloom filtresi her zaman veritabanındaki tablonun geçerli durumunu yansıtır.
5. **Adım:** Eğer kayıt veritabanında yoksa gerçek tabloya eklenir, varsa mükerrer kayıt tablosuna eklenir ve 2. Adıma geri dönülür.

**Not:** 1-3,5 adımları SQL tabanlı yaklaşımla tamamen aynıdır

### Parallel bloom filter tabanlı yaklaşım

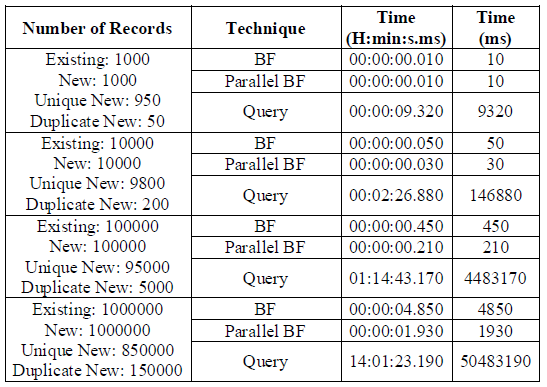
Parallel Bloom Filter tabanlı yaklaşım ile mükerrer kayıt tespit etme adımları:

1. **Adım:** Çıktı dosyasındaki veri uygulama tarafından yüklenir.
2. **Adım:** Dosyadan bir sonraki kayıt çekilir, çekilen kayıt veritabanında varsa 3. Adıma geçilir, yoksa işlem sonlandırılır.
3. **Adım:** Kaydın anahtar değeri hariç bütün sütunları birleştirilir.
4. **Adım:** Bu adım bloom filtresinin 4. adımı gibidir, ancak tek fark bloom filtresindeki kayıtların (yani birleştirilmiş sütunlar) mevcut tüm kayıtlarla (yani birleştirilmiş sütunları içeren her bir kayıt) paralel olarak eşleştirilmesidir. Bu, ana makinenin birden çok çekirdeğinin kullanılmasına yardımcı olur ve tüm kayıtlarla eşleşmesi için gereken süreyi azaltır.
5. **Adım:** Eğer kayıt veritabanında yoksa gerçek tabloya eklenir, varsa mükerrer kayıt tablosuna eklenir ve 2. Adıma geri dönülür.

**Not:** 1-3,5 adımları SQL tabanlı yaklaşımla tamamen aynıdır

## Karşılaştırma ve Sonuç

Bu çalışmada sunulan üç yaklaşım farklı veri boyutlarına göre Tablo 1’de karşılaştırılmıştır.



Tablo 1 SQL, BF ve Parallel BF Yaklaşımlarının Karşılaştırılması

Tablo 1’e bakıldığında, mükerrer kayıt tespiti ve ekleme işlemleri için SQL ve bloom filter tabanlı yaklaşımların arasındaki bağıl zaman farkı, kayıt sayısındaki artışla birlikte hızla artmakttığı görülmektedir. Bloom filter ve parallel bloom filter arasındaki zaman farkı, SQL ile arasındaki fark kadar hızlı bir artış göstermese de kayıt sayısı ile orantılı olarak artmaktadır. Parallel bloom filter algoritmasında, paralel hesaplama mimarisi kullanılarak bloom filter uygulandığı için normal algoritmaya oranla oldukça efektif ve hızlı bir versiyon elde edilmektedir. Bu nedenle, veritabanındaki satır sayısının fazla olduğu ve mimarinin uygun olduğu durumlarda veri mükerrer kayıt tespiti için en uygun çözümün parallel bloom filter olduğu görülmektedir.

# NEAR DUPLICATE DETECTION IN RELATIONAL DATABASE (2019)

**Proje Notu:** Benzer kayıtların çıkarılması adımında uygulanan ön işlem adımları ile levenshtein ve cosinüs benzerlik algoritmalarının kullanımının gösterildiği akademik çalışmadır.

Mükerrer kayıt tespitini zorlaştıran nokta, mükerrer kaydın veritabanındaki kayıt ile tamamen aynı olmadığı durum olmaktadır. Bazı alanları aynı olan iki kaydın birinin mükerrer kayıt olup olmadığının tahmin edilmesi zor bir problemdir.

Mükerrer kayıtların tespit edilmesi problemi ilk olarak Newbombe tarafından farklı zamanlarda alınan, birebir aynı ya da benzer medikal kayıtlar için tanımlanmıştır. Bu yöntemde kayıtlar karşılaştırılırken, soy isimler ilk olarak soundex kod olarak bilinen, ismin ilk harfi ve üç basamaklı bir sayıyla sesçil kodlara dönüştürülmüş, karşılaştırma bu kodlara göre yapılmıştır. Hernandez ve Stalfo potansiyel adayları limitlemek için uzaklık hesaplaması gerektiren sıralanmış komşuluk yöntemini geliştirilmişken, McCallum kayıtlatın benzerlikleri ölçüldükten sonra, bu benzerliklere göre uygulanan bir öbekleme algoritması önermiştir. Cohen ve Richman kayıt tespiti için, benzerliklerin beraber kullanıldığı bir çatı önermişlerdir.

Bu çalışmada, ilişkisel veritabanında farklı kaynaklardan gelen kayıtların birleştirilmesi aşamasında mükerrer kayıtların tespiti sırasında alan uzmanlık bilgisine ve benzerlik fonksiyonlarına göre işaretlemeler yapıldıktan sonra mükerrer kayıt makine öğrenmesi yöntemleri kullanılmıştır.

## Veri Ön İşleme

Bu çalışmada,

1. Kategorik verilerin harfleri büyütülmüş ve satur sonu boşluğu gibi özel karakterler temizlenmiştir.
2. Tarih bilgisinden saat bilgisi kısmı çıkarılarak standartlaştırılmıştır.
3. Ülke kodları eklenmiştır.
4. Belirlenen kurallara uymayan hücre değerleri sıfırlanmıştır.
5. Şehir ve ülke hücresindeki uyumsuz değerler benzerlik oranına bakılarak yeniden düzenlenmiştir.

5. Adımda gerçekleştirilen işlem için şehir veya ülkelerin doğru yazımlarının olduğu ilişkisel veritabanı tablosu, verideki şehir veya ülke bilgileri gruplandıktan sonra kartezyen çarpım ile aynı satırda yan yana getirilmiştir. Ardından Levenshtein ve cosine benzerlikleri kullanılarak, bozuk olan şehir veya ülke verileri en yüksek eşleme sağlanan şehir veya ülke bilgisi ile güncellenmiştir. Her iki benzerlik için de alt sınır 0.80 alınmış, bunun altında kalan şehir veya ülke bilgileri silinmiştir.

Bu çalışmada kullanılan veritabanındaki kayıt sayısının fazla olmasından dolayı satırların birbirleri ile kartezyen çarpımın alınması oldukça fazla zaman ve hafıza maliyetine sebep olmaktadır. Bu nedenle, bütün satılar için kartezyen çapımı yapmak yerine veriler benzerlik belirten kolonlar ile birleştirilmiştir. Birleştirme için cep telefonu, isim ve verideki önemli öznitelikler ayrı ayrı anahtar olarak alınmakta ve veri kolon bazında birleştirildikten sonra sonuçlar satır bazında alt alta yazılmaktadır. İsim bazında yapılan birleştirmede, harflerin yanlış yazılması olasılığına karşı ismin sesli harfleri silinmiş ve sessiz harfler alfabetik olarak sıralanmıştır.

## Veri Benzerliği

Bu çalışmada, Levenshtein ve Kosinüs benzerliği kullanılmıştır. Cep telefonu numarası, kimlik numarısı gibi öznitelikler için statik bir karşılaştırma ile eşit olup olmama durumuna göre 0 veya 1 verilmiş, kategorik olan isim özniteliğinde benzerlik oranına göre [0,1] aralığında bir değer verilmiştir. Bu iki benzerlik algoritmasının beraber kullanılmasının sebebi Levenshtein algoritmasının kelime bazında, Kosinüs algoritmasının ise vektör bazında karşılaştırma yapmasıdır.

**Not:** Öznitelik puanları, özniteliğin bir kişiyi tek başına ifade etme oranına bakılarak belirlenmenmiştir. {tam isim, cep telefonu numarası, şehir, doğum tarihi, cinsiyet} öznitelikleri için {8, 5, 3, 2, 1} puanları belirlenmiştir. İsim özniteliğinin en büyük olmasının sebebi isim haricindeki özniteliklerin diğer müşteri bilgilerine göre doldurulma ihtimalidir. İsim öznitelikleri puanlanırken her iki benzerlik oranından da en az 0.80 puan alması beklenmektedir.

### Levenshtein Benzerliği

İki metnin birbirlerine olan benzerliklerini ölçmek amacıyla kullanılmaktadır. Metinleri kartezyen çarpımı yaparak karşılaştırmakta ve metinler arasındaki farkı sayısal bir değer olarak vermektedir. Bu fark ne kadar az ise iki metin arasındaki benzerlik oranı o kadar fazla olmaktadır. Eğer metinler aynı ise sonuç “0” olacak farklı ise aradaki farka göre “0” dan büyük bir değer dönecektir. Basitçe, iki dizi, iki kelime, iki cümle gibi varlıklar arasındaki değiştirme ve ekleme işlemlerini tutar. Kartezyen çarpımının daha iyi anlaşılması için CAVİT ve ÇEVİK kelimeleri arasındaki uzaklık hesaplanması Tablo 2’de verilmektedir.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Ç** | **E** | **V** | **İ** | **K** |
| **C** | 1 | 1 | 2 | 3 | 4 |
| **A** | 1 | 2 | 2 | 3 | 4 |
| **V** | 2 | 2 | 2 | 3 | 4 |
| **İ** | 3 | 3 | 3 | 2 | 3 |
| **T** | 4 | 4 | 4 | 3 | **3** |

Tablo 2 CAVİT ve ÇEVİK kelimeleri arasındaki uzaklık hesaplanması

Hesaplama yapılırken iki farklı durum bulunmaktadır:

1. Karşılaştırılan harflerin aynı olması durumu: sol, sol-üst ve üst bölmede bulunan değerlerden en küçük olan değer boş kutucuğa yazılır.
2. Karşılaştırılan harflerin farklı olması durumu: sol, sol-üst ve üst bölmede bulunan değerlerden en küçük olan değer bir ile toplanarak boş kutucuğa yazılır.

Sonuç olarak sağ alt köşede elde edilen değer CAVİT kelimesi ile ÇEVİK kelimesi arasındaki farkı bize verir. Bu değer CAVİT kelimesinden ÇEVİK kelimesine geçebilmek için; C’nin Ç’ye, A’nın E’ye ve T’nin K’ye dönüştürülmesi olmak üzere 3 işlem yapmamız gerektiğini bize söylemektedir.

### Kosinüs Benzerliği

Basitçe iki farklı doküman (text) arasındaki benzerliği, trigonometrideki kosinüs (cosine) fonksiyonu üzerinden formülize etmek amaçlanmaktadır. Metinlerin birer vektör (yöney, vector) olarak düşünüldüğü bu yaklaşımda, iki vektörün birbirine göre olan ilişkisi bir açı ile ifade edilmektedir.

Örneğin;

1. Aynı yönü gösteren ve biribiri ile aynı olan iki vektör için kosinüs değeri 1 olacaktır (cos(0) = 1) .
2. Birbiri ile ilişkisiz olan vektörler için ise kosinüs değeri 0 olacaktır (cos(90) = 0).
3. Son olarak tamamen birbirini zıddı olan vektörler için, bu vektörler için aralarında ilişki yoktur denemez, kosinüs değeri -1 olacaktır ( cos(180) = -1 )
4. Bunların arasında kalan benzerlik değerleri, +1> benzerlik > -1 arasında yer alacaktır.

Dolayısıyla iki vektör arasındaki benzerlik bu yöntemle sayısallaştırılabilir. İki vektör arasındaki kosinüs bağlantısı için iki vektörün çarpımının (dot product) iki vektörün boylarının (euclidian space) çarpımına oranı alınmıştır.

## Makine Öğrenmesi ile Mükerrer Kayıtların Tespit Edilmesi

Veri üzerinde sırasıyla Naive Bayes, Destekçi Vektör Makinesi (SVM), Rassal Ormanlar ve K En Yakın Komşu algoritmaları kullanılmıştır. Navie Bayes gibi basit bir algoritma diğer karmaşık algoritmalara oranla daha başarız sonuçlar elde etmiştir. Algoritmalardaki parametrelere performansa göre denenerek karar verilmiştir.

1. Rassal ormanlar algoritması için ağaç sayısı belirlenirken [0, 600] aralığına bakılarak 300 seçilmiştir.
2. Destekçi Vektör Makinesi için C, gamma ve kernel parametreleri sırasıyla; 2154, 0.1, rbf olarak seçilmiştir.
3. K en Yakın Komşu algoritmasında n\_neighbors parametresi [0, 50] aralığına bakılarak 5 olarak belirlenmiştir.

## Sonuçlar

Benzerlik oranlarına göre sonuçlara bakıldığında hafıza ve işlem gücü gibi kısıtlardan dolayı her kolona kartezyen çarpımı uygulanmamasından dolayı yakalanamayan mükerrer kayıtlar bulunmaktadır.

Alan uzmanlık bilgisi ve benzerlik algoritmaları sonucunda puanlanan eğitim kümesi üzerinden uygulanan makine öğrenmesi algoritmaları veri tabanındaki mükerrer kayıtları %99 oranında belirlemiştir.

# DETECTION OF NEAR DUBLICATES IN TABLES BASED ON THE LOCALITY-SENSITIVE HASHING METHOD AND THE NEAREST NEIGHBOR METHOD (2016)

**Proje Notu:** Birbirine benzer kayıtların gruplanması işlemlerinde kullanılabilecek algoritmaların ve önerilen hibrit modelin gösterildiği akademik çalışmadır.

Literatürde yakın veri tespitinde düz metin karşılaştırılması yapılırken, kelime bazlı karşılaştırma yapan Levenshtein algoritması kullanılmaktadır. Metin, alt metinlere ayrılıp benzerlik karşılaştırılması yapılırken Wu-Manber değişimi ile Bitmap algoritması kullanılmaktadır. Mükerrer kayıt analizinde ayrıca BK-tree algoritması da kullanılmaktadır. Bu algoritma sözlüğü kullanarak satırlar arası benzerliği tanımlamaktadır. Bu çalışma, bir veri içerisinde mükerrer kayıt arama problemi için matematiksel bir model sunmaktadır. Ayrıca bu çalışmada, mükerrer kayıt bulma problemi için sunulan farklı yöntemler, buldukları kelimelerin miktarına ve algoritmanın çalışma zamanına göre karşılaştırılmıştır. Nümerik data karşılaştırılmasında KNN algoritması ile yapılmıştır.

Bu çalışmada, Locality-sensitive hashing ile KNN algoritması kullanılarak bir model kurulmuş ve nümerik veriler üzerinde mükerrer kayıtlar aranmıştır. Çalışma için kullanılan veri tabanında; nümerik, metin, formül (MathType, KFormula, MathCastmula, TeX, MathML vb.), tarih ve saat, fotoğraf ve birleştirilmiş tip olmak üzere altı farklı türde veri bulunmaktadır.

1. Giriş tablosu B grafik nesnelerinden ayırın: görüntüler ve formüller. Bu nesneler ayrı ayrı incelenir. Formülleri karşılaştırmak için, örnekler veya şablonlar arasındaki karşılaştırmaya dayanan bir yöntem uygulanabilir. Matematiksel formüllerde kopyaların yakınında bulunma yöntemi aşağıdaki aşamaları içerir:

1) incelenen metinlerdeki formüllerin tanımlanması;

2) belirlenen formüllere göre numune oluşturma;

3) bulunan formüllerin örneklerinin aralarında karşılaştırılması;

4) aynı formüllere sahip formüllerin bağlamını, bu formüllerde yaygın olarak kullanılan sembollerin varlığını dikkate alarak kontrol etmek;

5) aynı numune ve içeriğe sahip formüllerin kopyaların yakınında olduğu kabul edilir.

2. Tablo B'de tarih türünde veriler içeren hücreler varsa, tüm tarihlerin birleşik bir formata, örneğin “day.month.year” a getirilmesi ve hücre verilerinin şu şekilde araştırılması önerilmektedir. metin türünün içeriğine sahip hücre.

3. Sayısal tipteki hücrelerden gelen tüm içerik tek tip bir tipe getirilmelidir: ondalık ayırıcı virgül «,» şeklinde belirtilmelidir.

4. «Hayır. Tablodan »girilirse, bu satır mevcutsa satırların sayısını temsil eder.

5. Bir dizi metin verisi S ve bir dizi sayısal değer oluşturun N Bölünürken, birleştirilmiş içeriğe sahip hücrelere dikkat edin: belirli bir hücrenin sayısal ve metin verileri içerdiği durumlarda, verilen hücrenin metni ve sayısal veriler sekansların ayrı ayrı elemanlarına bölünür.

6. B giriş tablosunun S dizisi için, kanonize edilmiş W formundaki bir kelime dizisini oluşturun. Ardından, - + 1 2 mh 1 E, E,, E dizilerini tanımlayın ve bölgeye duyarlı karma yöntemiyle, endeks unsurları (Ek).

7. Ardından, giriş tablosunun her dizin öğesi dizisinden Hamming mesafelerini, formül (7) ile tabanda bulunan bu tablo dizilerinin dizin öğelerine kadar hesaplayın ve atanan eşik değeri λH for için koşulu (8) kontrol edin.   0,1. Koşul yerine getirilirse, neredeyse bir kopyasının tanımlandığı anlamına gelir.

8. B giriş tablosunun N sayısal dizisi N için bir dizi alt dizi Build - + 1 2 v g 1 K, K,, K oluşturun.

9. subse - + 1 2 vg 1 K, K,, K ve - + yyy 1 2 vg 1 K, K,, K vektörlerini formlarda gösterin ve mesafeleri formül (9) - (11'e göre hesaplayın) ). Temel olarak, bu formüllerden birini seçebilirsiniz.

11. Formülleri (12), (13) uygulayın ve durumunu (14) kontrol edin. Atanan eşik değeri ρ λ Î0,1 için bu koşul yerine getirilirse, neredeyse bir kopyasının tanımlandığı söylenebilir. Λρ ve λH değeri önceden deney yoluyla belirlenir.

12. Bu yöntemle neredeyse kopyaları belirlerken, özellikle hesaplanan mesafe eşik değerden önemli ölçüde daha azsa, (8) ve (14) koşullarının ayrı ayrı karşılandığı tablolara özel dikkat gösterilmelidir. Bu koşullar belirtilen λρ ve λH değerleri için eşzamanlı olarak yerine getirilirse, böyle bir tablo açık bir şekilde neredeyse iki kopya içerir ve ödünç alındıysa uzman tarafından ayrı ayrı incelenmelidir.

Bu yöntem, antiplagiarizm sistemlerinde ve tablo halinde sunulan bilgilerdeki benzerlikleri tanımlamak için akıllı analiz çalıştırmak üzere tasarlanmış diğer sistemlerde kullanılabilir. Yöntemin bir uygulaması, özel özellikleri nedeniyle mümkündür: aynı anda farklı veri türleri içeren tabloların karşılaştırılması (metin, sayısal) ve yakın kopyaların tespiti için hibrit yöntemin uygulanması nispeten kolaydır, çünkü bilinen en yakın komşu ve bölgeye duyarlı karma yöntemlerine dayanır.

Bu yöntem, antiplagiarizm sistemlerinde ve tablo halinde sunulan bilgilerdeki benzerlikleri tanımlamak için akıllı analiz çalıştırmak üzere tasarlanmış diğer sistemlerde kullanılabilir. Özel özelliklerinden dolayı yöntemin bir uygulaması mümkündür: bir kerede farklı veri türlerini içeren tabloların karşılaştırılması (metin, sayısal) ve yakın kopyaların tespiti için hibrit yöntemin nispeten olması uygulaması kolay, çünkü bilinen en yakın komşu yöntemlere ve bölgeye duyarlı karma yöntemlere dayanmaktadır.

# SEMI-SUPERVISED CLUSTERIN FOR DE-DUPLICATION (2018)

**Proje Notu:** Mükerer kayıt tespiti için yarı eğitimli tabanlı bir yöntem geliştirilmiştir. Projemizde kullanılması düşünülmektedir.

# DISCRETE-TIME HOPFIELD NEURAL NETWORK BASED TEXT CLUSTERING ALGORITHM

**Proje Notu:** Mükerrer kayıt tespitinde kullanmayı planladığımız çizge tabanlı bir kümeleme algoritmasıdır.