Büyük Veri Problemlerine Çözüm Olarak Veri Akış Madenciliği
Data Stream Mining to Address Big Data Problems

Erdi Ölmezoğulları, İsmail Arı
Bilgisayar Mühendisliği Bölümü, Özyeğin Üniversitesi
İstanbul, Türkiye
erdi.olmezogullari@ozu.edu.tr, ismail.ari@ozyegin.edu.tr

Ömer Faruk Çelebi, Salih Ergüt
AveaLabs
İstanbul, Türkiye
{omerfaruk.celebi,salih.ergut}@avea.com.tr

***Özetçe*— Günümüzde bilişim dünyası faydalı bilgiye ulaşma yolunda "büyük veri" problemleri (verinin kütlesi, hızı, çeşitliliği, tutarsızlığı) ile baş etmeye çalışmaktadır. Bu makalede, büyük veri akışları üzerinde İlişkisel Kural Madenciliği'nin (İKM) daha önce literatürde yapılmamış bir şekilde "çevrimiçi" olarak gerçeklenme detayları ile başarım bulguları paylaşılacaktır. Akış madenciliği için Apriori ile FP-Growth algoritmaları Esper isimli olay akış motoruna eklenmiştir. Elde edilen sistem üzerinde bu iki algoritma kayan penceler ve LastFM sosyal müzik sitesi verileri kullanılarak karşılaştırılmıştır. Başarımı yüksek olan FP-Growth seçilerek gerçek-zamanlı ve kural-tabanlı bir tavsiye motoru oluşturulması sağlanmıştır. En önemli bulgularımız çevrimiçi kural çıkarımı sayesinde: (1) çevrimdışı kural çıkarımından çok daha fazla kuralın, (2) çok daha hızlı ve etkin olarak, ve (3) çok daha önceden hesaplanabileceği gösterilmiştir. Ayrıca müzik zevklerine uygun "George Harrison⇒The Beatles" gibi pekçok ilginç ve gerçekçi kural bulunmuştur. Sonuçlarımızın ileride diğer büyük veri analitik sistemlerinin tasarım ve gerçeklemesine ışık tutacağını ummaktayız.**

Anahtar Kelimeler — Veri akış madenciliği, ilişkisel kural madenciliği, karmaşık olay işleme, Apriori, FP-Growth.

***Abstract*—Today, the IT world is trying to cope with "big data" problems (data volume, velocity, variety, veracity) on the path to obtaining useful information. In this paper, we present implementation details and performance results of realizing "online" Association Rule Mining (ARM) over big data streams for the first time in the literature. Specifically, we added Apriori and FP-Growth algorithms for stream mining inside an event processing engine, called Esper. Using the system, these two algorithms were compared over LastFM social music site data and by using tumbling windows. The better-performing FP-Growth was selected and used in creation of a real-time rule-based recommendation engine. Our most important findings show that online association rule mining can generate (1) more rules, (2) much faster and more efficiently, and (3) much sooner than offline rule mining. In addition, we have found many interesting and realistic musical preference rules such as "George Harrison⇒Beatles". We hope that our findings can shed light on the design and implementation of other big data analytics systems in the future.**

Keywords — Data stream mining, association rule mining, complex event processing, Apriori, FP-Growth.

I. GİRİŞ

Günümüzde milyarlarca kullanıcısı olan mobil iletişim ağları, Internet hizmetleri ve duyargalar gibi pekçok yeni teknoloji, bu teknolojilerin uygulama alanları, ve bunların hepsini izleyen sistemler sayesinde sürekli ve yüksek-ölçekli veriler üretilmektedir. Bu veriler kurumsal altyapılarda defalarca kopyalanarak kütlesi artmakta ve "büyük veri" problemlerini ortaya çıkarmaktadır. Bu problemlerle baş edip analitik çalışmalar yapabilmek ve faydalı, sonuca yönelik bilgilere, desenlere, kurallara ulaşabilmek için bazı kurumlar firma içerisinde paylaşımlı olarak kullanılan bulut altyapısı ve platformları (IaaS, PaaS) kurmuşlar, bazıları da genel bulut sistemlerini kullanmaya başlamışlardır. Fakat her iki durumda da yüksek-ölçekli verilerin depolandıktan sonra analitik çalışmalara tabi tutulması ek kaynak maliyetlerine ve aksiyonel bilgilere ulaşmada gecikmelere sebep olmaktadır. Bu sebeple, kurumsal arakatman yazılımlarda, çevrimiçi veri filtreleme ve diğer analizlere imkan sağlayan veri akış işleme veya *Karmaşık Olay İşleme* (KOİ) motorlarına [5] görev verildiği görülmektedir. Ayrıca bu motorların yakın zamanda veri madencilik gereçleriyle güçlendirilmeye çalışıldığı da gözlenmektedir [13]. Ancak bu gereçlerin çoğunda veri madenciliğinin desen çıkarımı (*extraction*, *testing*) aşaması çevrimdışı olarak tarihsel verilerle yapılmakta ve ancak desen tespiti (*detection, scoring*) çevrimiçi yapılabilmektedir.

"Büyük veri" olarak adlandırılan bilişim probleminin dört boyutu vardır: büyük kütleler, yüksek hızlar, çok çeşitlilik, ve verideki tutarsızlıklar [16]. Bugün içlerinde telekom operatörleri, bankalar, e-ticaret siteleri, güvenlik güçleri ve belediyelerin de bulunduğu pekçok kuruluş hergün, operasyonel sistemlerinden akış halinde Terabyte (TB)'larca veri toplamakta ve bu verileri içinde Petabyte (PB)'lar bulunan veritabanlarına eklemeye çalışmaktadırlar. Ayrıca bu veriler farklı kaynaklardan geldikleri için yapısal çeşitlilik de göstermektedirler (örnek: txt-csv, XML, JSON, ses-video). Son on yılda ortaya çıkan Apache Hadoop çerçevesi [2] (HDFS, MapReduce, Hbase, Hive) ve benzer dağıtık NoSQL veritabanları (Cassandra, MongoDB) büyük veri problemlerinin yüksek boyut ve veri çeşitliliği ile baş etmek için tasarlanmışlardır. Ancak Hadoop ve diğerleri gerçek-zamanlı veri işleme veya çok sayıda döngü içeren veri madenciliği için tasarlanmamışlardır. Son zamanlarda Hadoop'un bu eksiklikleri giderici pekçok çalışma yapılmakta olsa da [17], veri akışlarında KOİ motorlarının kullanımı halen etkin olarak devam etmektedir.

II. ÖNCEKİ ÇALIŞMALAR

Veri akışlarının içindeki desenleri kaybetmeden kütlesini azaltma veya veriyi özetlemeyle ilgili yapılan önceki çalışmalar arasında örnekleme, yük dökme, çizgileme (sketching), özet (synopsis) çıkarımı ve tümleştirme (aggregation) bulunmaktadır [6]. Akışlar üzerinde tümleştirme yapmak için basit sayma veya toplama fonksiyonları yanında ortalama ±

Bu çalışma TUBITAK 3501 109E194, Avea Labs, EU FP7 Marie Curie BI4MASSES ve IBM SUR programları tarafından desteklenmiştir.

978-1-4673-5563-6/13/$31.00 ©2013 IEEE

varyans ($\mu \pm \sigma$) bulma veya akışları korelasyon ile ilişkilendirme gibi tekniklere başvurulmaktadır. StatStream [10] akışlar arasında ölçeklenebilir korelasyon bulma işlemini DFT kullanarak literatürde ilk defa yapan çalışmalar arasında gelir. Bizim ön çalışmalarımızda ise korelasyonun veri azaltımı sağlama potansiyeline değinilmektedir [3]. Bu makalenin amacı akış halindeki veriden ilişikisel kural çıkarımı tekniğiyle hızlı bir şekilde özetleme ve aksiyonel bilgi çıkarımı yapılabileceğini ortaya koymaktır.

Geleneksel Veritabanı (DBMS) mimarisi verileri önce depolayıp sonra incelemeye göre tasarlandığından, veri akışları üzerinde gerçek-zamanlı analizlerde başarım sorunları ortaya çıkarmaktadır. KOİ mimarisi yüksek hızlı akışların farklı sürekli-sorgularla bellek içinde hızla işlenebilmesini sağlamakta ve yeni ortaya çıkan büyük ölçekli sinyal işleme uygulamalarının analiz ihtiyaçlarına daha iyi cevap vermektedir. KOİ motorları kullancılara veriler arasındaki ilintileri ve olaylar arasındaki desenleri bulabilmek için gerekli sorgu deyimlerini bir dil olarak sunarlar. SQL benzeri bu sorgu dillerine genel adıyla Olay İşleme Dili (OİD) denir [11]. Bu çalışmada, OİD diline bütünleşik ilişkisel kural madenciliği operatörleri gerçeklenmiş, KOİ motorları üzerinde "sürekli-sorgular" olarak koşulmuş, ve bu sayede "çevrimiçi" akış madenciliği yapılması sağlanmıştır. Aşağıda önce literatürdeki temel kural çıkarım teknikleri daha sonra da bizim geliştirdiğimiz çevrimiçi kural çıkarımı anlatılacaktır.

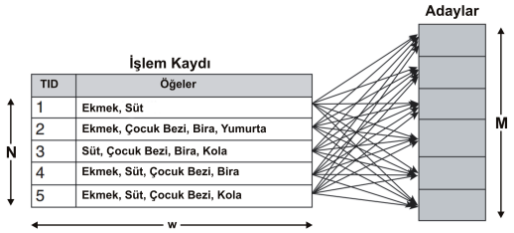
III. ÇEVRİMDIŞI İLİŞKİSEL KURAL MADENCİLİĞİ

Veri madenciğilinin popüler konularından biri olan ilişkisel kural madenciliği (İKM) için kullanılan en popüler iki algoritma Apriori [14] ve FP-Growth [4] algoritmalarıdır. İKM matematiksel olarak şu şekilde tanımlanmaktadır; $\boldsymbol{I} = i_1, i_2 \ldots i_m$: *öğe kümesi*, $\boldsymbol{D}$: *veritabanı*, $\boldsymbol{X \Rightarrow Y}$ da $X \subset I$, $Y \subset I$ ve $X \cap Y = \varnothing$ özelliklerine sahip *kural* ifadesiyle gösterilir. Buna göre öğelerin toplam *destek* (support) ve *güvence* (confidence) değerleri Denklem (1) ve Denklem (2) ile hesaplanır.

$$Destek(X \Rightarrow Y) = \frac{s(X \cup Y)}{|D|} \qquad (1)$$

$$Güvence(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)} \qquad (2)$$

Apriori algoritmasında, öncelikli olarak veritabanında kayıtlı tüm *öğeler* (items), yani test öğeleri, Şekil 1'de gösterildiği gibi *işlem kaydı* (transaction) formatına getirilerek her bir öğenin toplam *destek* değeri Denklem (1) kullanılarak hesaplanır. Ardından belirlenen bir *minimum destek* (*minDestek*) eşik değeri altında kalan öğeler budama işlemiyle kayıttan çıkarılır. Geriye kalan tüm öğelerin önce birli, sonra ikili ve üçlü olmak üzere tüm kombinasyonları, yani *öğe kümeleri* (itemsets), oluşturularak devam edilir. Oluşan öğe kümeleri için Denklem (2)'de gösterilen güvence değerleri hesaplanır ve belli bir *minimum güvence* (*minGüvence*) eşiği üzerinde olanlar *kural* (*rule*) olarak alınır. Apriori algoritması ilk İKM algoritmalarından olmasına rağmen kombinatoryal bir yaklaşım sergilemektedir ve oluşturduğu aday öğe kümelerinin hesaplanması zaman-mekan karmaşıklığı açısından oldukça maliyetlidir. Şekil 1'de gösterildiği gibi, veritabanındaki toplam işlem kaydı sayısı N, benzersiz öğe sayısı k, toplam aday öğe küme sayısı $M=2^k-1$, maksimum işlem kaydı uzunluğu w ile ifade edilirse, Apriori'nin zaman karmaşıklığı $O(N.M.w)$ ile ifade edilebilir [12]. Yani hesap maliyeti öğe sayısından (k) üstel olarak etkilenmektedir.



Şekil 1: İşlem kaydından ilgili adayların oluşturulması [12].

Apriori algoritmasının başarım sorunlarını gidermek amacıyla geliştirilen en popüler algoritma FP-Growth'tur [4]. FP-Growth algoritması da parametre olarak *minDestek* ve *minGüvence* eşik değerlerini kullanır. Apriori'den farklı olarak öğeler ilk aşamada veritabanındaki sıklıklarına göre sıralanır ve bir *sıklık listesi* (*FrequencyList, FList*) oluşturulur. Bu sıklık listesinde öğeler belirlenen *minDestek* eşiğine göre budanır. Budama işleminden sonra liste üzerinden bir "örüntü ağacı" (*FP-Tree*) veri yapısı oluşturulur. Kurallar FP-tree üzerinden yine *minGüvence* eşik değeri kullanılarak çıkartılır. Daha önceki çalışmamızda [15] Apriori ile FP-Growth algoritmaları karşılaştırılmış olup Apriori'nin çevrimiçi İKM'ye uygun olmadığı gösterilmiştir. Bu sebeple bu makalede sadece FP-Growth algoritmasının çevrimiçi İKM sonuçlarına yer verilecektir.

IV. ÇEVRİMİÇİ İKM VE YÖNTEM

İKM çıkarım algoritmalarının *"büyük veri"* olarak adlandırılan verilere uygulanması, algoritmaların hesaplama karmaşıklığı yüzünden çevrimdışı olarak yapıldığında kimi zaman zor, kimi zaman imkansız bir hal almaktadır. Ayrıca desenlerin, kuralların hızla değiştiği günümüzde geç elde edilen sonuçlar, daha bulundukları anda geçerliliğini kaybedebilmektedir. Bu problemleri en aza indirgemek için Şekil 2'de gösterilen İKM sorgusu Esper KOİ motoru [5] üzerinde gerçeklenmiş ve akan veriler üzerinde çevrimiçi akış madenciliği yapılması sağlanmıştır. Gereç olarak makine öğrenmesi konusundaki en popüler açık kaynak yazılım olan *Weka'nın* Java kütüphaneleri [8] kullanılmıştır. FP-Growth operatöründeki *parametre* alanı minDestek ve minGüvence; *tablo.özellik1* işlem kaydı şeması; *tablo.özellik2*: öğe sayısı değerlerini içerir. Tüm pencere boyutlarında *minSupport=0.1, minGüvence=0.9* değerleri kulanılmıştır. Bu değerler kural bulunamadığı zaman adaptif olarak değiştirilebilmektedir, ancak kontrollu deneylerimiz sırasında sabit tutulmuşlardır. Çıkarılan kurallar daha sonra sisteme otomatik eklenerek, gerçek-zamanlı desen tespiti (scoring) yapılması sağlanabilir.

```
SELECT FPgrowth(parametre,tablo.özellik1, tablo.özellik2)
FROM event.win:length_batch(10000) AS tablo
```

Şekil 2 FP-Growth İKM sorgusu.

Deneylerimiz sosyal müzik paylaşım sitesi *LastFM* [7] veri kümesi üzerinde gerçekleştirilmiştir. Orjinal ham veri kümesi [2005-2011] tarih aralığını ve *<zaman, kullanıcıNo, artistNo, artistAdı, parçaNo, parçaAdı>* bilgilerini içerir. 3GB büyüklüğündedir ve yaklaşık 1K'lık benzersiz kullanıcı ve 75K'lık benzersiz artist sayısına sahiptir. Orjinal veri, ön işlemlerden geçirilerek, *parçaNo* ve *parçaAdı* alanları çıkarılmış ve veri KOİ motorunda çalışacak hale getirilmiştir. Bu çalışmada başarım sonuçlarının daha iyi anlaşılabilmesi için sadece 2008 yılına ait 5.7 Milyon satır kullanılmıştır.

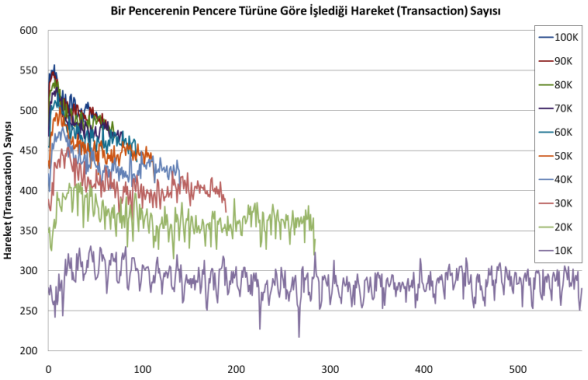
İKM için yapılan çalışmalar literatürde çoğunlukla perakendicilik alanından, yani işlem kaydı sınırlarının bir "fiş" ile çizilmiş olduğu bir sektörden gelir. Peki belirli bir oturum sınırı belirtilmeyen sonsuz akışlar üzerinde öğeler nasıl gruplanmalıdır? Şekil 3'deki 10K-100K'lık satır sayısına sahip zıplayan pencerelerin (*tumbling window*) birbirleriyle ilişkisi görülmektedir. İşlem sayıları, örnek *N=10,000* için *10k* şeklinde gösterilmektedir. Yani analizler 100K'lık 0 no'lu tek bütün bir pencere veya 10 adet 10K'lık pencereler (0,1,2,..,9) üzerinde yapılabilir. Her pencere işlendikten sonra bulduğu kuralları yayınlamaktadır. Şekil 3'de ayrıca hangi kuralın hangi pencereye denk geldiği ve yayınlanmadan önce ne kadar zaman gecikebileceği görülmektedir. Önceki çalışmamızda hem zıplayan hem de kayan pencereler karşılaştırılmıştır [15].

		100K	90K	80K	70K	60K	50K	40K	30K	20K	10K	Veri	Aralık	Kural
N	k	0	0	0	0	0	0	0	0	0	0	10K	[00K-10K)	
	k										1	10K	[10K-20K)	K0
	k									1	2	10K	[20K-30K)	
	k								1		3	10K	[30K-40K)	K1
	k							1		2	4	10K	[40K-50K)	
	k						1		2		5	10K	[50K-60K)	K2
	k					1		2		3	6	10K	[60K-70K)	
	k				1				3		7	10K	[70K-80K)	K3
	k			1				3		4	8	10K	[80K-90K)	
	k		1						4		9	10K	[90K-100K]	

Şekil 3 Zıplayan pencere kapasiteleri, aralıkları ve yakaladıkları kurallar.

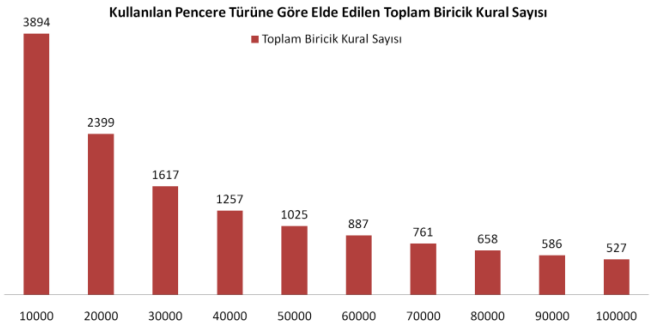
V. BULGULAR VE YORUM

Deneyler sırasında sorduğumuz sorular özetle şöyledir: (1) Çevrimiçi kural çıkarımı sayesinde, çevrimdışı kural çıkarımında görülmeyen "geçici kurallar" bulunabilir mi? (2) Çevrimiçi kural çıkarımı daha hızlı ve etkin olarak yapılabilir mi? (3) Genel kurallar çevrimdışı metodlara göre çok daha önce hesaplanıp yayınlanabilir mi? Verideki toplam satır sayısı sabittir. Şekil 4'te gösterildiği üzere kullaıcıNo'ya göre gruplanan işlem kaydı sayısı (*N,w*) büyük pencerelerde daha çok (100K→550), küçük pencerelerde daha azdır (10K→300).



Şekil 4 Zıplayan pencerelerin işlediği işlem sayısı (N, w).

Pencere boyutu arttıkça, pencerelerde işlenen öğelerin toplam sayıları (M) ve toplam işlem kaydı sayısı (N, w) artmaktadır. Bu sebeple her bir öğenin destek ve güvence sayıları, eşik değerlerinin altında kalmaktadır. Eşikler altında kalan öğe listeleri ve kurallar elenmektedir. Bu iddamızı desteklen sonuçlar Şekil 5'de gözlemlenmektedir. Zamana-bağlı bazı kurallar kendi küçük pencerelerinde daha fazla destek bulmaktadır. Eğer aynı kuralların daha büyük pencerelerde de bulunması istenirse pencere boyutuyla ters orantılı olarak eşik değerlerinin düşürülmesi gerekecektir.



Şekil 5 Zıplayan pencerelerle elde edilen toplam benzersiz kural sayısı.

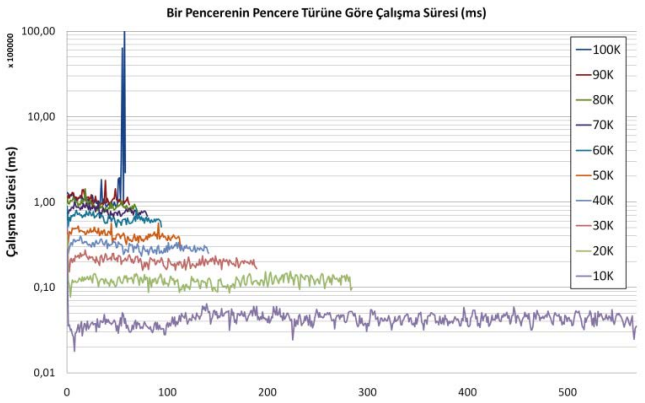
Şekil 5'te görüldüğü gibi küçük pencerelerde çok fazla kural yayınlanması karşı, farklı pencerelerde yayınlanan kural kümelerinin birbirleriyle ilişkileri matematiksel olarak küme teorileriyle (∪,∩,-,⊆) analiz edilmiş ve kümelerin benzerlik-farklılıkları bulunmuştur. Benzerlik ölçütleri olarak Jaccard=∩/∪ kullanılmıştır. Yapılan bu analiz ve sonuçlara göre tüm pencere boyutlarında kendini gösteren sadece iki tane popüler kural bulunmaktadır. Bunlar, **George Harrison=>The Beatles** (Kural-1) ve **Hilary Duff=>Britney Spears** (Kural-2)'dir. George Harrison, The Beatles grubunun gitaristi olup daha sonradan benzer nitelikli şarkılar söylemeye devam etmiştir ve bu kural göstermektedir ki, şarkıları benzer kişilere hitab etmektedir. Yine Hilary Duff ve Britney Spears'in şarkılarıyla benzer genç bir kitleye hitab ettiği söylenebilir. Bu iki kuralın hangi pencere türünde, hangi pencere nosunda geçtiğini ve toplamda kaç kere geçtiği Şekil 6'da gösterilmektedir. Bulunan diğer ilginç durum ise her zaman geçerli olmayan, yani zamana-dayalı geçerlilikleri olan **Iron & Wine ve Coldplay=>Radiohead (Kural-3) ve Blackfield=>Placebo (Kural-4)** gibi kuralların varlığıdır. Böylece, elde edilen kuralların hangi zaman aralığında popüler olduğu ya da ne zaman popülerliğini yitirdiği gözlemlenebilir.

	Kural-1		Kural-2		Kural-3		Kural-4	
Pencere	Frekans	Pencere No	Frekans	Pencere No	Frekans	Pencere No	Frekans	Pencere No
10K	4	{274,371,499,522}	4	{50,87,337,555}	0	-	0	-
20K	1	{54}	1	{25}	0	-	0	{223}
30K	3	{91,97,123}	1	{29}	1	{55}	0	-
40K	3	{68,101,132}	2	{2,84}	1	{29}	1	{111}
50K	4	{54,58,81,106}	3	{18,21,67}	2	{33,39}	1	{22}
60K	5	{45,67,78,87,88}	1	{15}	0	-	1	{74}
70K	1	{53}	2	{1,15}	1	{28}	0	-
80K	4	{50,60,64,65}	1	{11}	1	{4}	0	-
90K	2	{41,57}	2	{10,12}	0	-	0	-
100K	4	{21,37,40,51}	1	{9}	1	{3}	1	{4}

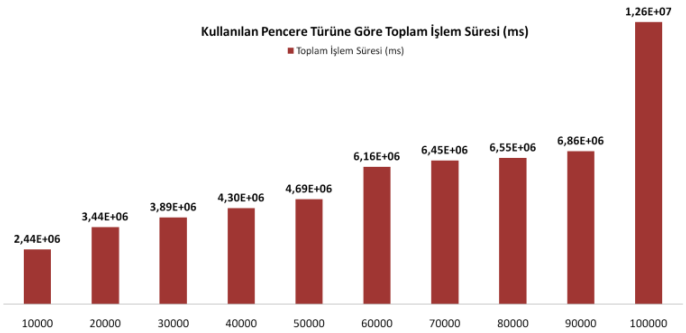
Şekil 6 Tüm pencerelerde görülen (1-2) ve görülmeyen (3-4) popüler kurallar.

Şekil 7'de 10K-100K arağında seçilen pencerelere göre veri işlem süreleri gösterilmektedir. 100K penceresinin dışındaki

tüm pencelerde düzenli bir işleme zamanı olarak gözlemlenmektedir. 10K boyutundaki pencerelerin bir senelik veriyi toplam işleme süreleri ~40 dk. (0.045ms x 100k x 1s/1000ms x 570 pencere) iken, bu süre Şekil 8'de 100K'lık pencerelerde ~3.5 saate ulaştığı gözükmektedir.



Şekil 7 Zıplayan pencere türlerinin çalışma süreleri.



Şekil 8 Zıplayan pencere türüne göre toplam işlem süresi.

100K'lık pencerenin son pencerelerinde işlem süresi olağandışı bir şekilde değişmektedir. Bu değişimin M, N ve w değerlerine direk bağlı olarak bir sonuç vermektedir. Ayrıca bir pencerenin ortama işlem süresinin pencere boyutu arttıkça arttığı gözlenilmiştir. Bu yüzden, büyük veri kümelerinin daha küçük parçalara bölünerek işlenmesi gerekmektedir. Böylelikle eldeki kaynaklarla işlenemeyecek büyüklükteki verilerin bile, gerçek-zamanda daha kısa sürede işlenebileceği gösterilmiştir. Ancak deneyler boyunca, yapılan pencere seçimlerinden 10K'dan küçük 100K'dan büyük pencerelerde hesaplama karmaşıklıkları yüzünden herhangi bir sonuç elde edilememiştir. Büyük pencereli hesaplar IBM HS22 Blade sunucusu üzerinde kullanılan 18GB'lık hafızadan daha fazlasına gereksinim duyulmaktadır. Bu da gerçek-zamanlı olarak yapılacak işlemelerde pencere boyutunu en uyun şekilde seçilmesi gerektiğini göstermektedir.

VI. SONUÇ VE ÖNERİLER

Bu çalışmada akan veriler üzerinden gerçek-zamanlı İKM ve çevrimiçi/çevrimdışı kuralların karşılaştırmalı analizi yapılmıştır. Elde edilen ilişkisel kuralların zaman içerisinde değişimleri analiz edilmiş, böylece başlarda popüler olan bir kuralın sonlarda popülerliğini kaybettiği veya analizin başlarında gözükmeyen ilişkisel kuralların sonlarda oluşmaya başlayabileceği gözlemlenmiştir. Bu sonuç ile çevrimdışı yöntemlerle yapılan analizlere göre doğru zamanlarda en ilgili sonuçlar elde edileceği gibi, elde edilen kurallara göre dinamik ve uyarlamalı iş zekası sistemleri geliştirilebileceği aşikardır.

Bu makalede veri akışları üzerinde kural çıkarımının gerçeklenme detaylarını ve başarım bulgularını paylaştık. Değişik pencere boyutları ve LastFM veri kümesini kullandık. İleriki araştırmalarımızda çıkarılan kuralların ardışık-desen madenciliği için otomatik olarak sisteme kaydı, ön işlem aşamalarının da OİD sorgularıyla birlikte gerçek-zamanlı olarak yapılması gibi çeşitli çalışmalara yer verilecektir.

KAYNAKÇA

[1] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. "Aurora: A new model and architecture for data stream management", *VLDB Journal*, 12(2):120–139, August 2003.

[2] Apache Hadoop Project, http://hadoop.apache.org/

[3] I. Ari and O. F. Celebi, "Finding Event Correlations in Federated Wireless Sensor Networks", *In Proc. Federated Wireless Sensors and Systems Workshop (FedSenS)*, *In Conj. with IWCMC*, 2011

[4] C. Borgelt, An Implementation of the FP-growth Algorithm, ACM Workshop of Open Source Data Mining Software, (OSDM), pages 1-5, 2005.

[5] EsperTech Inc. Event Stream Intelligence, http://www.espertech.com/

[6] C. Giannella, J. Han, J. Pei, X. Yan, P. S. Yu; Mining Frequent Patterns in Data Streams at Multiple Time Granularities; *Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT*; 2003.

[7] LastFM, http://www.dtic.upf.edu/~ocelma/MusicRecommend-ationDataset/ lastfm-1K.html

[8] Weka ML, http://www.cs.waikato.ac.nz/~ml/weka/

[9] P. S. Yu, Y. Chi: Association Rule Mining on Streams. Encyclopedia of Database Systems (2009)

[10] Y. Zhu, D. Shasha. "Statstream: Statistical monitoring of thousands of data streams in real time." VLDB, 2002.

[11] I Ari, E. Ölmezoğulları, H. Sözer, Application of Combinatorial Testing Techniques in CEP Engines, IEEE-SİU2012

[12] Pang-Ning Tan, Introduction to Data Mining, Ch6. Association Analysis: Basic Concepts and Algorithms, 25 Mart 2006

[13] IBM Infosphere Stream Mining Toolkit, http://www-01.ibm.com/software/data/infosphere/streams/

[14] R. Agrawal, and S. Ramakrishnan. "Fast algorithms for mining association rules." Proc. 20th Int. Conf. VLDB. Vol. 1215. 1994.

[15] Ismail Ari, Erdi Olmezogullari, Omer Faruk Celebi, Data Stream Analytics and Mining in the Cloud, IEEE DaMiC 2012

[16] What is Big Data? http://www-01.ibm.com/software/data/bigdata/

[17] Sergey Melnik , et al, Dremel: interactive analysis of web-scale datasets, Proc. of the VLDB Endowment, v.3 n.1-2, 2010