Olay Tespiti Problemi için Akan Veri İşleme Platformu Kullanımı: Avantaj ve Kısıtların İncelenmesi (Using a Stream Processing Platform for Event Detection: Advantages and Limitations)

Özlem Ceren Şahin[1], Nesime Tatbul[2], ve Pınar Karagöz[1]

[1] ODTÜ Bilgisayar Müh. Bölümü, Ankara, Türkiye
{e1746668,karagoz}@ceng.metu.edu.tr
[2] Intel Labs ve MIT, Cambridge, ABD
tatbul@csail.mit.edu

Özet. Sosyal medya ve sosyal ağlar, günümüzde, bilgi ve haber paylaşımı için yoğun olarak kullanılan ortamlardır. Hava durumu, trafik yoğunluğu, kaza ve benzeri beklenmedik olay ve durumların duyurulması için konvansiyonel basın yayın organlarından çok daha hızlı paylaşım sağlamaktadır. Bu nedenle sosyal medya mesajlarından olay tespiti yoğun olarak çalışılan bir araştırma konusudur. Bu çalışmamızda olay tespiti problemi için akan veri işleme platformu Apache Storm'un kullanılabilirliğini ve performansını inceledik. Olay tespiti için literatürde kullanılan iki tekniği ele aldık. Her iki alternatif yöntem Apache Storm üzerinde kodlandı. Ara veri saklama ortamı olarak Apache Cassandra kullanıldı. İlk olay tespiti yöntemi mesajlarda geçen kelimelerin sıklığını takip eder, sıklık oranı ani artış gösteren kelimelerin bir olayı işaret ettiği varsayımına dayanır. İkinci yöntem kümeleme tabanlı bir yöntemdir ve hiyerarşik kümeleme algoritmalarının olay tespiti problemi için uyarlanmış bir versiyonunu kullanır. Her iki yöntem için Apache Storm'un sunduğu özelliklerin nasıl kullanıldığı detaylandırılarak, sağlanan kolaylıklar ve kısıtlamalar irdelenmiştir. Buna ek olarak, deneysel analiz amaçlı olarak oluşturulan sistemin simülasyon amaçlı nasıl kullanıldığı konusunda da bilgi sunulmaktadır.

Abstract. Social media and social networks are now used extensively for information and news sharing. They provide ability to convey news and information much faster than conventional media for sharing weather conditions, traffic accidents and other unexpected events and situations. For this reason, event detection from social media messages is an intensively studied research topic. In this work, we examined the usability and performance of a streaming data processing platform, the Apache Storm, for event detection problem. We used two techniques used in the literature for event detection. Both alternatives are coded on Apache Storm. Apache Cassandra is used as the intermediate data storage medium. The first method of event detection is based on tracking the frequency of the

words in the messages, and it is based on the assumption that the words with a sudden increase in the frequency rate indicate an event. The second method is a clustering-based method and it uses a version of the hierarchical clustering algorithms adapted for the event detection problem. We elaborated on the use of the features provided by Apache Storm for both methods, and discussed the facilities and limitations provided. In addition, information on how to use the system created for simulation purposes for experimental analysis is also provided.

Anahtar Kelimeler: Olay tespiti · Akan veri işleme platformu · Gerçek zamanlı analiz

Keywords: Event detection · · Stream processing platform · Real-time analytics

1 Giriş

Sosyal medya ve mikroblog hizmetlerinin yükselişi, internette paylaşımın etkin bir yolu olarak başladı ve özellikle akıllı telefonların, tabletlerin, vb. yaygın kullanımıyla, sosyal medya erişilebilir büyük bir veri kaynağı haline geldi. Günümüzde sosyal medya insanların düşüncelerini ifade ettiği ve güncel sorunlara tepki verdiği ana platform durumundadır. İnsanlar etraflarında meydana gelen olayları veya durumlarını yayınlamak için sıklıkla sosyal medyayı kullanmaktadırlar. Bu durum, sosyal medyanın olay tespit ve analizinde önemli bir yer almasına olanak vermektedir.

Twitter, günde yaklaşık 340 milyon tweet yayınlayan 300 milyondan fazla aktif kullanıcıya sahip en popüler mikroblog sosyal ağ hizmetidir. Tweet adı verilen mesajların çoğunukla erişilebilir olması nedeniyle akademik çalışmalarda yoğun olarak Twitter verileri kullanılmaktadır [5],[12], [14], [7]. Bugüne kadar Twitter verileri akademik çalışmalarda depremleri, felaketleri, politik konuları, trafiği, vb. tespit etmek için kullanılmıştır. Bununla birlikte, sosyal medyayı kullanarak olayları tespit etmek hala aktif ve popüler bir araştırma problemidir.

Literatürdeki çalışmalarda, *olay*, belirli bir zamanda ve yerde gerçekleşen ve kısa zamanda dikkat çeken bir etkinlik olarak tanımlanır [5]. Bunu takiben, *olay tespiti*, haber ya da mesaj içeriklerini kullanarak, meydana gelen olayları hızla tespit etmeyi amaçlar [4].

Bu çalışma, olay tespitini bir akış işleme problemi olarak ele almaktadır. Düşük gecikmeli, yüksek verimli akış işleme platformları yıllardır kullanılmaktadır ve olgun seviyede birçok platform mevcuttur [3]. Bu platformlar, olay tespit çalışmaları için altyapı sağlama potensiyeli taşımaktadır. Akış işleme platformları, sınırlı sayıda araştırmada olay tespit uygulamaları ile kullanılmıştır [10], [16]. Ancak bu çalışmalarda platformların kullanım detayları sunulmamış, artı ve eksileri irdelenmemiştir.

Çalışmamızda hızlı artış tespiti (burst detection) mantığına dayanan iki olay tespit yöntemini ele aldık. İlk yöntem, mesajlarda geçen kelimelerin sıklığının takibine dayalıdır, sıklık oranı ani artış gösteren kelimelerin bir olayı işaret ettiği

varsayımına dayanmaktadır. İkinci yöntem kümeleme tabanlı bir yöntemdir ve olay tespit problemi için uyarlanmış bir hiyerarşik kümeleme algoritması kullanır. Bu bildiride Apache Storm'un sunduğu özelliklerin olay tespit yöntemlerinin kodlanmasında nasıl kullanıldığı detaylandırılarak, sağlanan kolaylıklar ve kısıtlar irdelenmiştir. Buna ek olarak, deneysel analiz sırasında, oluşturulan sistemin üzerinde akış simülasyonun nasıl yapıldığı konusunda da bilgi sunulmaktadır.

Bildirinin içeriği şöyledir. Kısım 2'de, benzer çalışmalar özetlenmiştir. Kısım 3'te çalışmada kullanılan temel teknolojiler olan Apache Storm ve Twitter API hakkında özet bilgi verilmektedir. Olay tespit yöntemleri ve Storm üzerinde nasıl oluşturuldukları Kısım 4'de detaylı olarak anlatılmaktadır. Kısım 5'te durum çalışması ve yapılan değerlendirmeler verilmektedir. Kısım 6'da genel bir değerlendirme ile bildiri sonlandırılmaktadır.

2 İlgili Çalışmalar

Bilgi erişimi konusundaki çeşitli çalışmalarda veri kaynağı olarak Twitter kullanılmaktadır [8], [11], [4], [13], [15]. Örneğin, [4]'de tweetler kullanılarak olay tespiti yapan farklı olay tespit teknikleri sınıflandırılmıştır. [5]'deki çalışmada Twitter verileri üzerinden ağ analizi kullanarak kullanıcı ilişkileri ve etkileşimleri aracılığıyla olay tespit yöntemlerinin iyileştirilmesi ele alınmıştır.

Literatürde, çalışmamızla benzer şekilde olay tespiti problemi için akan veri işleme platformu kullanan oldukça sınırlı sayıda çalışma bulunmaktadır. [10] ve [16]'da sunulan çalışmaların her ikisi de platform olarak Storm kullanmıştır. Ancak, bu çalışmalarda problemin farklı yönleri ele alınmıştır. [10]'deki makalede aynı ekip tarafından daha önce önerilen bir algoritmaya bir uzantı olarak yeni bir dağıtık anahtar kelime bölümleme şeması kullanılarak birden fazla düğüme ölçekleme üzerine odaklanılmıştır.[16]'daki çalışmada ise, Storm platformu üzerinde k-means kümeleme algoritması kullanan bir çözüm önerilmiştir.

3 Temel Teknolojiler

Bu kısımda, çalışmamızda kullandığımız Twitter API ve Apache Storm hakkında bilgi sunulmaktadır.

3.1 Twitter API

Twitter, herkese açık mesajların (tweet) belirli bir yüzdesini paylaşan REST ve Streaming API'ler sunmaktadır. REST API ile mesajlar, kullanıcılar, konumlar veya Twitter verilerinin diğer nitelikleri hakkında bilgi talebi gönderilebilir. Yanıtlar JSON veya XML formatlı objeler olarak iletilir. Öte yandan, Streaming API, istenen bir kritere göre filtrelenebilen Twitter verisi akışı sağlar.

Bu çalışmada, Twitter Streaming API'sinin Java kütüphanesi olan *Twitter4j* [2] kullanıldı. Twitter4j'nin konum filtresi özelliği ile, ABD ve Kanada dışındaki ülkelerden gelen mesajları filtreledik. Deneyler ve değerlendirme aşamasında kullanmak üzere, streaming API'den gelen mesajlar Apache Cassandra veritabanına

kaydedildi. Bütün deneyler, toplanan bu verilerin 7 günlük kısmını içeren aynı mesaj grubu üzerinde çalıştırıldı. Veri toplama aşamasında konum filtresi dışında başka bir filtre kullanılmadı. Anahtar kelime tabanlı filtreleme yapılmadığı için tespit edilecek olaylar için bir kısıtlama uygulanmadı. Bu sayede veri içinde farklı tipte olayların yer alması ve veri çeşitliliği sağlandı.

3.2 Apache Storm

Akan veri işlemede performans kısıtlamalarını iyileştirmek amacıyla Nathan Marz ve BackType [1] ekibi tarafından oluşturulan ve Twitter tarafından alındıktan sonra açık kaynak haline gelen Apache Storm, gerçek zamanlı dağıtık akan veri işleme sistemi olarak kullanılmaktadır [6] [9]. Storm, farklı programlama dilleri ile kullanılabilir. Storm'un kullanım alanlarından bazıları gerçek-zamanlı analiz, online makine öğrenmesi ve dağıtık uzaktan işlem çağrısıdır (distributed remote procedure call).

Çalışmanın başlangıcında, gerçek zamanlı dağıtık bilgi işleme platformları gözden geçirilerek, çalışma amacına en uygun iki aday olarak Apache Storm ve Apache Spark seçildi ve karşılaştırmalı olarak incelendi. Gerek Apache Storm gerekse Apache Spark, klasik işleme ve tasarımı gerçek zamanlı dağıtık bir sistem olarak tanımlayabildiğinden, gerçek zamanlı olarak iş zekası ve analitiği uygulamak için tüm gereksinimleri sağlamaktadır. Ancak Spark genel amaçlı dağıtık bir bilgi işleme platformu iken Storm akış odaklı bir dağıtık hesaplama platformudur. Bu nedenle, çalışmada Apache Storm kullanılmıştır.

Storm'da üç çeşit temel yapıtaşı bulunur: *spout (musluk)*, *bolt (cıvata)* ve *topoloji*.

- Storm, bir hesaplamada akışların kaynağı olarak *spout* adı verilen yapıları kullanır. Spout, Kafka veya RabbitMQ gibi bir dağıtık mesajlaşma sisteminden verileri okuyabilir veya kendi akışını Twitter Streaming API'si veya Apache Cassandra gibi bir veritabanı kullanarak oluşturabilir.
- Storm, herhangi bir sayıdaki giriş akışını işlemek ve herhangi bir sayıda yeni çıktı akışını üretmek için *bolt* adı verilen yapıları kullanır. İşlevler, filtreler, akış katıştırmaları, toplu akışlar, veri tabanları ile iletişim gibi hesaplama mantığının çoğu boltlarda uygulanır.
- Storm, spout ve bolt'lar arasındaki bağlantı ve veri akışını *topoloji* adı verilen bir ağ yapısı şeklinde tanımlar. Topoloji, karmaşık ve çok aşamalı bir akış hesaplamasıdır.

Her spout ve bolt, topoloji içinde çeşitli görevler yürütür. Her görev, bir iş parçacığına karşılık gelir. Akış gruplamaları, verinin bir görev kümesinden başka bir görev kümesine nasıl gönderileceğini tanımlar. Her spout ve bolt için paralel görev sayısı geliştirici tarafından belirlenir.

Apache Storm beş farklı gruplama türü sunmaktadır:

- Karışık gruplama: Bu gruplandırma türünde, akış, bolt görevleri arasında rastgele ve eşit şekilde dağıtılır. Dağıtım Apache Storm tarafından yürütülür.

– Tüm gruplama: Bu gruplama türü akışı tüm bolt görevlerine aktarır.
– Alana göre gruplama: Bu gruplandırma çeşidi ise akışı, kullanıcı tarafından belirtilen bir alana göre dağıtır.
– Genel gruplama: Bu tür gruplandırma tüm akışı tek bir göreve toplar.
– Doğrudan gruplama: Bu özel bir gruplandırma türüdür. Bu gruplamada veri grupları kullanıcılar tarafından belirli bir bolt görevine aktarılır. Bu nedenle, veri gruplarını görevler arasında dağıtmak, geliştiricinin tercihine bağlıdır.

4 Metot

Bu bölümde, olay tespiti için kullandığımız *anahtar kelime tabanlı olay tespiti*, ve *kümeleme tabanlı olay tespiti* teknikleri için oluşturulan Apache Storm topolojileri ve topolojilerin bileşenleri ayrıntılı bir şekilde anlatılmakta ve tartışılmaktadır. Olay tespiti tekniklerinden önce her iki teknikte de ortak olarak kullanılan konum tabanlı farklı akışlar oluşturma ve veritabanı kullanımı konuları hakkında bilgi verilmektedir.

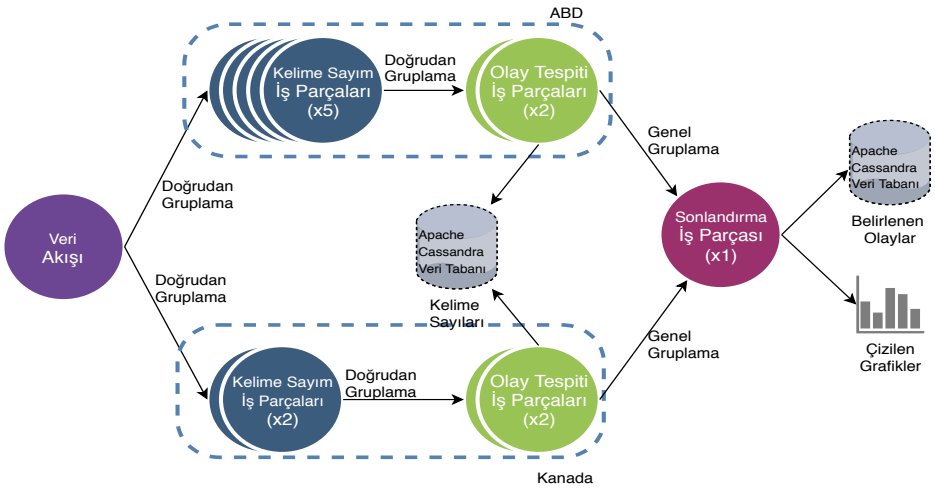
Vurgulanması gereken önemli bir nokta her iki olay tespiti yönteminde de verilerin belli zaman penceresi içinde kalan bloklar şeklinde işleniyor olmasıdır. Bloklar arasında zamansal örtüşme bulunmaz, zaman sırasına göre birbirini takip eder. Burada amaç, ardışık bloklar arasındaki değişimlerin takibi ile olay tespiti yapılmasıdır. Her bir turda topoloji bir mesaj bloğunu işler. Zaman pencereleri (bloklar) akmaya devam ettiği sürece işlem turları devam eder. Zaman penceresinin büyüklüğü veri işleme ihtiyacına göre belirlenebilir. Deneylerimizde pencerler 6 dakikalık bloklar olarak belirlenmiştir.

4.1 Konum Tabanlı Farklı Akışlar Oluşturma

Tüm dünya üzerinde kullanılan Twitter verileri, dünya çapında bilinen bir sanatçı tarafından duyurulan yeni albüm gibi global bir olayı ya da yerel bir etkinliği içerebilir. Bu nedenle, gelen tweetler ülkeyi, şehri veya tüm dünyayı ilgilendiren olaylar şeklinde etiketlenebilir. Twitter'da konum verisi mevcut ve erişime açıksa, Twitter API üzerinden bu bilgiye ulaşılabilir. Bu çalışmada Twitter Streaming API'nın konum filtreleme seçeneği kullanılarak, Kanada ve ABD'den gönderilen tweet akışı üzerinde çalışılmıştır. Önerilen sistemde, iki farklı ülke aynı işleme adımlarıyla iki paralel akışa bölünmüştür ve veri hacimlerindeki farklardan dolayı farklı paralelizm sayılarına sahiplerdir. Şekillerde görüldüğü gibi, (Şekil 1 ve Şekil 2), akış kaynağı, tweetin gönderileceği boltu, konum bilgilerine bağlı olarak belirleyebilir. Bu sayede, olaylar gerçekleştiği konum bilgileriyle tespit edilir. Bu çalışmanın sadece konum filtrelemesi açısından Kanada ve ABD'den gönderilen mesajlara odaklanmasına rağmen, farklı ülkeler için yeni işleme hattının entegre edilmesi oldukça kolaydır.

4.2 Veritabanı Kullanımı

Gerek akan veri parçalarının, gerekse ara çıktıların saklanması için veri saklama alanı gereksinimi oluşmaktadır. Çalışmamızda bu ihtiyaç için Apache Cassandra kullanılmıştır. Apache Cassandra, ölçeklenebilirlik ve yüksek kullanılabilirlik



Şekil 1: Anahtar Kelime Tabanlı Olay Tespiti Topolojisi

sağlayan bir NoSQL veritabanıdır. Veriler çoklu düğümlere kopyalandığından, sistem hata toleransı sağlar. Apache Cassandra'nın sisteme entegre edildiği iki kullanım durumu vardır. Birincisi, işlem sonunda tespit edilen olaylar veya tweetler gibi sistemde kullanılan veya sistemin oluşturduğu verileri depolamaktır. İkincisi, durum bilgisi içeren akış işlemesi sağlamak için kelimelerin veya küresel kümelerin sayısı gibi mevcut blok içinde oluşturulan veriyi depolamaktır. İş parçalarının mevcut blok için işini bitirip bitirmediği veya sürecini tamamlamak için ne kadar sürdüğü gibi durum bilgileri de Cassandra'da saklanır.

4.3 Anahtar Kelime Tabanlı Olay Tespiti Yöntemi

Anahtar kelime tabanlı olay tespiti yönteminin Apache Storm topolojisi, Şekil 1'de sunulmaktır. Bu yöntemdeki ana adımlar şöyledir: Spout'tan bir bir blok kapsamında gelen her tweet kelimelere ayrılarak ön işleme tabi tutulur. Sonraki bolt'ta kelime sayılarındaki artış takip edilerek o turdaki olay ifade eden kelimeler belirlenir.

Bu yöntemde iki parametre kullanılır:

- Tf-Idf_Artış_Oranı: Bu parametre, bir kelimenin olay olarak tespit edilmesi için son iki tur arasındaki tf-idf değerinin artış oran eşiğini tanımlamak için kullanılır. Örneğin, bu parametre 10 olarak belirlenmiş ve bir kelimenin son iki turdaki tf-idf değerleri sırasıyla 0.001 ve 0.015 ise, bu anahtar kelimenin bir olayı ifade ettiği tespit edilir ($0.015/0.001 > 10$).
- Kelime_Sıklık_Eşiği: Bu parametre, bir sözcüğü çok rastlanan bir kelime olarak varsaymak için kullanılan eşiği tanımlar. Sadece çok rastlanan kelimeler anahtar kelime tabanlı olay tespit algoritmasına tabi tutulur.

Yöntemdeki her adımı aşağıdaki gibi detaylandırabiliriz.

Kelime Sayma Boltu. Akıştan gelen mesajlar ilk olarak kelimelere bölünür ve kelimeler ilgili bolta saymak üzere gönderilir. Kelime sayma boltunun iş parçalarının temel görevi o turda geçen kelimelerin sayılarını belirlemektir. Kelimelerin sayısı, kelimenin bir olayı temsil edip edemeyeceğine karar vermek için kullanılır. Performans, büyük veri analizi için önemli olduğundan, nadir geçen kelimeler bu boltta elenir ve ilerleyen işlemlere tabi tutulmaz. Nadir geçen kelimeleri belirlemek için Kelime_Sıklık_Eşiği parametresinde tanımlanan eşik kullanılır. Bu ön eleme, mevcut turda en sık kullanılan kelimeleri tanımlamakta ve nadir kelimeler için gereksiz hesaplamaları önlemektedir.

Olay Tespit Boltu. Bu boltta son iki tur için her kelimenin tf-idf değeri hesaplanır. Bir kelimenin bir olayı temsil edip etmediğine tf-idf değerleri kontrol edilerek karar verilir. Tf-idf değerinin hesaplanması için Denklem 1, Denklem 2 ve Denklem 3'te verilen formüller kullanılır.

$$\mathrm{tf}(t,d) = \frac{f_{t,d}}{|\{t' \in d\}|} \tag{1}$$

- $f_{t,d}$: t kelimesinin d dokümanında kaç kez geçtiğini gösterir.
- $|\{t' \in d\}|$: d dokümanında bulunan toplam kelime sayısı.

$$\mathrm{idf}(t,D) = log\frac{N}{1 + |\{d \in D : t \in d\}|} \tag{2}$$

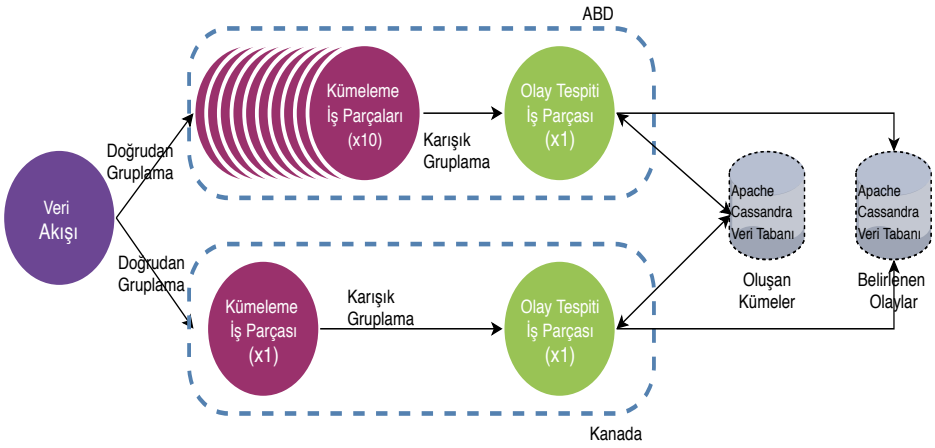
- N: Sistemde bulunan toplam doküman sayısı $N = \{|D|\}$
- $|\{d \in D : t \in d\}|$: t kelimesinin geçtiği toplam doküman sayısı.

$$\text{tf-idf}(t,d,D) = \mathrm{tf}(t,d) \cdot \mathrm{idf}(t,D) \tag{3}$$

Sonlandırma Boltu. Bu bolt iki basit görev yerine getirir. Olayları temsil eden kelimelerini Apache Cassandra veri tabanına kaydeder, ve her olay kelimesinin son 10 tur içerisindeki sayılarını gösteren çizgi grafiği çizer.

4.4 Kümeleme Tabanlı Olay Tespiti Yöntemi

Bu yaklaşım, her bir blok içindeki tweet'lerin kümenlenmesi (clustering) ve kümelerdeki büyümenin takibi adımlarına dayanmaktadır. Yaklaşım, ani büyüme gösteren tweet kümelerinin bir olay ifade ettiği fikrine dayanmaktadır. Kümeleme işlemi kelime vektörlerin kosinüs benzerliklerine dayalı çalıştığı için, spout, tweetleri kelime vektörlerine çevirir ve bir sonraki bolta gönderir. Tweet vektörleri, bir tweet içindeki kelimeleri normalize edilmiş ağırlıklarıyla birlikte içerir. Örneğin "RIP Muhammed Ali RIP" tweet'inin vektörü {"RIP": 0.5, "Muhammed": 0.25,



Şekil 2: Kümeleme Tabanlı Storm Topolojisi

"Ali": 0.25 } şeklinde gösterilir. Bu vektör temsili, tweetler ve kümeler arasındaki kosinüs benzerliğini hesaplamak için kullanılır.

Kümeleme tabanlı olay tespiti yöntemi için oluşturulan Apache Storm topolojisi Şekil 2'de sunulmaktadır. Topoloji, kümeleme ve olay tespiti boltlarından oluşmaktadır.

Kümeleme Boltu. Bu bolt, kosinüs benzerliği kullanarak tweet'leri kümelere atar. Verimlilik açısından belli bir eşik değerinin altındaki sayıda mesaj içeren kümeler silinir. Performans iyileştirmesi için iki aşamalı bir kümeleme kullanılır. Bu bolt, yalnızca ilk aşamadaki, *yerel kümeleme* adımından sorumludur. Bir turun başlangıcında iş parçalarında hiçbir yerel küme yoktur. Her iş parçası, akış kaynağı tarafından dağıtılan tweet vektörleriyle kendi kümelerini oluşturur ve kümeleri günceller. İşlemin sonunda her kümeleme iş parçası, küme listesini, değerlendirme için bir sonraki bolta aktarır. Küme ataması için bu bolt, tweet vektörü ile mevcut yerel kümeler arasındaki kosinüs benzerliğini hesaplar ve kosinüs benzerliği belirtilen eşikten daha yüksekse, tweet o kümeye atanır. Her küme vektörü buna göre güncellenir. Herhangi bir tweet için kosinüs benzerlik kısıtlaması karşılanmazsa, tweet vektörü için yeni küme oluşturulur. Turun sonunda, veri akışı sırasında oluşturulan tüm kümeler, olay tespit boltuna gönderilir.

Olay Tespit Boltu. Bu bolt, her zaman bloğunun sonunda etkinleştirilir. Küme boltunun her bir iş parçası, yerel küme listesini olay tespit boltuna gönderir ve olay tespit boltu tüm iş parçalarından gelen bu görev listelerini biriktirir. Önceki boltun her bir iş parçasının yerel küme listeleri geldiğinde, olay tespit boltu yerel kümelerin değerlendirmesini başlatır. Bu değerlendirme iki adımdan oluşur:

– Yerel Küme Değerlendirmesi: Bu bolt ilk olarak farklı iş parçaları tarafından oluşturulan yerel kümeleri birleştirir. Birleştirme işlemi, iki yerel kümenin

temsil vektörünün kosinüs benzerliği belirtilen eşiğe eşit veya daha yüksekse gerçekleşir. Birleştirme işlemi sırasında, her kelimenin ağırlığı yeniden hesaplanır ve iki küme vektörü teke indirilerek güncellenir.

– Global Küme Değerlendirmesi: Yerel değerlendirmeden sonra, veritabanından mevcut kümelerin listesi alınır. Bu kez, iş parçalarından gelip birleştirilmiş yerel kümeleri, veritabanı tarafından tutulan *global kümelerle* karşılaştırır. Bu adımda, her bir kümenin kosinüs benzerliği her bir yerel küme için tek tek hesaplanır ve gerekli durumda global küme yerel küme ile birleştirilerek güncellenir. Güncellenen global kümenin büyüme oranı, Denklem 4 kullanılarak hesaplanır. Büyüme oranı belirtilen eşiği sağlarsa, geçerli tur için olay olarak işaretlenir. Son adım olarak, son 3 turda aktif olmayan global kümeler performans için elimine edilir ve veritabanından silinir.

$$\text{kume\_buyume\_orani(C)} = \frac{|\{t_{eklenen} \in C\}|}{|\{t_{hepsi} \in C\}|} \tag{4}$$

- $|\{t_{eklenen} \in C\}|$: son turda C kümesine eklenen tweet sayısı.
- $|\{t_{hepsi} \in C\}|$: C kümesindeki toplam tweet sayısı.

5 Durum Çalışması ve Değerlendirmeler

5.1 Veri Kümesi ve Çalıştırma ortamı

Geliştirilen olay tespit yöntemlerini, 31 Mayıs 2016 - 7 Haziran 2016 tarihleri arasındaki bir hafta içinde toplanmış olan yaklaşık 12 milyon tweet içeren veri kümesi üzerinde uyguladık. Daha önce bahsedildiği üzere, coğrafi konum filtrelemesi kullanılarak ABD ve Kanada'dan gönderilen mesajlar toplandı. Bunun dışında bir filtreleme kullanılmadığı için veri kümesi farklı tipte olaylar içermektedir. Tüm deneyler 3.2 GHz i5 işlemcili, 16 GB hafıza içeren MacOS versiyon 10.13.3 bilgisayar üzerinde çalıştırılmıştır.

5.2 Akış Simülasyonu

Deneyler sırasında, farklı yöntem ve konfigürasyonlar arası karşılaştırma yapabilmek için aynı veri üzerinde akış simülasyonu yapma ihtiyacı bulunmaktadır. Bu amaçla, akış kaynağı (spout) olarak tanımlanmış olan Apache Cassandra veritabanına kaydedilen veriler, 6 dakikalık zaman pencereleri (bloklar) halinde, tweet'lerin zaman sırasına uygun olarak çekilir. Bir blok içindeki mesajların akışı bittiğinde, spout bir sonraki bloğun akışını hemen başlatmaz. Bir sonraki blok, geçerli zaman bloğunun tüm işlemleri tamamırılmıştırlanana kadar askıya alınır. Sistemin güvenilirliği ve olay tespitinin doğruluğu için zaman blokları arasında askıya alma işlemi gereklidir; çünkü, bir sonraki blok hemen başlatılırsa, o anda işlenmekte olan bloktaki kelimeler ile bir sonraki blokta işlenecek olan kelimeler birbirine karışır ve yanlış olay tespitlerine neden olur. Bu nedenle, bloklar arası askıya alma işlemini sağlayacak bir akış protokolü tanımlamak gerekir. Apache

Storm bu amaç için bir hazır bir yapı sağlamaz, dolayısıyla bu probleme çözüm olarak çalışmamızda iki farklı yaklaşım tanımladık. Birinci yaklaşımda, turlar arasında, spout, yapılan deneylerle belirlenen bir süre kadar uyutularak bekletilir. İkincisinde ise Storm tarafından tanımlanan doğrudan gruplama tekniğini kullanarak mevcut turun bitip bitmediği kontrol edilir.

İlk yaklaşımımızda, görevler arasında veri dağıtımı Apache Storm tarafından karma gruplama ve alan gruplamaları ile yürütülür. Farklı turların karışmasını önlemek için turlar arasında uyku aralıkları kullanılır. Bu yaklaşımda, verinin Storm'un kendi programlama mekanizması tarafından dağıtılması daha verimli bir dağıtım ve işlem süresi sağlamasına rağmen, turlar arasında kullanılan uyku tamponu bu avantajı dezavantaja dönüştürmektedir. Bunun sebebi uyku tamponu süresinin en uzun süren tura göre seçilmesidir. Turların işleme süresi gün içerisindeki 6 dakikalık blokların veri hacimlerinin farklı olmasından dolayı büyük farklılıklar göstermektedir. Bu durum eylemsiz geçen büyük zaman aralıklarına neden olmaktadır. Özetle; spout, turlar arasında yapılan deneyler tarafından belirlenen bir süre kadar bekletilir ve belirlenen bu süre, her bir tur için tüm görevlerin işlemlerini bitirmesi için yeterli bir süre olarak seçilmiştir.

İkinci yaklaşımda ise, Storm'un dağıtım ve programlama mekanizmasının kontrolünün tam olarak geliştiriciye bırakıldığı, doğrudan gruplama tekniği kullanılır. Bu yaklaşımda veriler, görevlerin uygunluğu kontrol edilmeden, sırayla her bir göreve tek tek dağıtılır. Dolayısıyla verilerin dağıtımındaki verimlilik ilk yaklaşım kadar iyi sağlanmamaktadır, çünkü Storm doğrudan gruplama tekniği için görevlerin doluluğunu kontrol etme seçeneğini sunmamaktadır. Öte yandan, bu yöntemin avantajı, turun bittiği anı algılama yeteneğine sahip olunmasıdır. Böylece işlem yapılmadan geçen uyku aralıkları ile zaman kaybedilmemektedir. Performansta sağladığı artış sebebiyle, deneylerde ikinci yaklaşım kullanılmıştır.

5.3 Değerlendirmeler

Yöntemlerin kodlanmasında ve çalıştırılmasında gözlemlerimiz şöyle özetlenebilir:

- Sunulan yöntemlerin akan veri platformu üzerinde geliştirilme eforu için formal bir efor metriği kullanılarak ölçüm yapılmadı. Ancak daha önceki çalışmalarımızda platform kullanmadan geliştirdiğimiz kümeleme tabanlı olay tespiti çözümümüz [13] ile karşılaştırdığımızda, Storm'un iş akışı tanımlama ve iş parçalarını dağıtma konusunda sağladığı yeteneklerin yöntemleri kodlama eforunu oldukça hafiflettiğini gözlemledik.
- Apache Storm'un yerel bir veritabanı sunmaması ve işlemsel (transactional) destek ihtiyacı bulunması kısıtlamalar getirmektedir. İşlemsel destek özellikle kümeleme tabanı teknikte, tweet'lerin kümeleme atanması sırasında gerekli olmaktadır. Çalışmamızda bu ihtiyacı lokal ve global kümeleme aşamaları ile karşılamaya çalıştık.
- Deneyler sırasında ihtiyacımız olan akış simülasyonu için akış kaynağı olan spout'u veritabanı olarak tanımlayabilmek kolaylık sağlamaktadır. Bununla birlikte, deneylerden daha çabuk sonuç alabilmek için, bir haftalık tweet akışını daha hızlı oynatmak gerekmektedir. Gerçek akışta, 6 dakikalık pencere

boyunca tweet'ler biriktirilikten bir önceki pencerenin tweet'leri bu zaman zarfında işlendiği için tweet blokları arasında ek bir senkronizasyona gerek olmamaktadır. Ancak simülasyon sırasında hem art arda gelen bloklardaki tweet'lerin birbirine karışmaması, hem de mümkün olduğu kadar akışı hızlandırma gereği vardır. Storm, bu ihtiyaca yönelik bir mekanizma sunmadığı için mevcut yapılarla çözüme gidilmiştir.
- Storm üzerinde geliştirilen iki yöntemi karşılaştırdığımızda geliştirme eforu açısından, kelime tabanlı yöntem daha basit adımlar içerdiği için çok daha verimlidir. Adımların basit olması çalışma zamanında da verim sağlamaktadır (saniyede 1200 tweet işlenmektedir). Öte yandan, kümeleme tabanlı yöntem gerek geliştirme eforu, gerekse çalışma süresi açısından daha geride kalmaktadır (kümeleme tabanlı yöntemde saniyede 300 tweet işlenebilmektedir).
- Çalışmamızın odak noktası olay tespiti doğruluğu olmamakla birlikte, bu konuda alınan sonuçlar özetle şöyledir: Veri 20 farklı olay içermektedir. F-measure ölçüsüne göre kelime tabanlı yöntem %62, kümeleme tabanlı yöntemse %70 başarı oranı ile olay tespiti yapabilmiştir. Kümeleme tabanlı yöntemde özellikle geri çağırma (recall) oranı %100'e varmaktadır. Beklenen şekilde anahtar kelime tabanlı yöntemde sonuçların daha zor yorumlandığını ve kümeleme tabanlı yöntemin daha net ve başarılı sonuçlar ortaya koyduğunu söyleyebiliriz.

6 Sonuç

Çalışmamızda olay tespiti problemi için, akan veri işleme platformu olan *Apache Storm*'un kullanımını inceledik. Olay tespiti için literatürde kullanılmış olan artış tespitine dayalı iki farklı yöntemi Storm üzerinde geliştirdik. Her iki yöntem için de akan veriyi belli zaman uzunluğundaki pencereler (bloklar) halinde işledik. İlk yöntem ardışık turlar arasında tweet'lerde geçen kelimelerin sayılarındaki artışın takibine dayanmaktadır. İkincisinde ise tweet'ler kümelenmekte, ve küme büyüklüğündeki artış takip edilmektedir. Her iki yöntem için algoritmaların adımları Storm boltları olarak geliştirilerek tüm algoritma bir Storm topolojisi olarak tanımlandı. Akan veri işleme platformu olarak Apache Storm'un, olay tespiti problem için kullanışlı yetenekler sağladığını ve geliştirme eforunu azalttığını gözlemledik. Geliştirmede zorlandığımız iki nokta göze çarpmaktadır. Birincisi deneylerde simülasyon yapma ihtiyacı sırasında veri akış hızını kontrol edecek ve zaman bloklarını ayrık tutacak hazır bir yapının bulunmamasıdır. İkincisi de Apache Storm içinde yerli bir veritabanı bulunmaması ve özellikle kümeleme tabanlı yöntemde işlemsel (transactional) ihtiyaçların karşılanmasında zorluk yaşanmasıdır. Takip eden çalışmalarda, artış takibi tabanlı yöntemlerin yanı sıra, akan veri işleme platformlarının, daha farklı olay tespiti ve tahmini çözümleri için kullanımı üzerine odaklanılması olasıdır.

References

1. Backtype website. http://www.backtype.com/, accessed: 2018-04-03

2. Twitter for java website. http://twitter4j.org/en/index.html, accessed: 2018-04-03
3. IEEE Data Engineering Bulletin, Special Issue on Next-Generation Stream Processing (2015)
4. Atefeh, F., Khreich, W.: A Survey of Techniques for Event Detection in Twitter. Computational Intelligence **31**(1), 132–164 (2015)
5. Cordeiro, M., Gama, J.: Online Social Networks Event Detection: A Survey. In: Michaelis, S., Piatkowski, N., Stolpe, M. (eds.) Solving Large Scale Learning Tasks. Challenges and Algorithms, Lecture Notes in Computer Science, vol. 9580, pp. 1–41. Springer, Cham (2016)
6. Foundation, A.S.: Apache Storm. http://storm.apache.org
7. Gulisano, V., Jerzak, Z., Voulgaris, S., Ziekow, H.: The DEBS 2016 Grand Challenge. In: ACM International Conference on Distributed and Event-based Systems (DEBS). pp. 289–292 (2016)
8. Java, A., Song, X., Finin, T., Tseng, B.: Why we twitter: understanding microblogging usage and communities. In: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis. pp. 56–65. ACM (2007)
9. Marz, N.: A storm is coming. https://blog.twitter.com/2011/storm-coming-more-details-and-plans-release, accessed: 2018-04-03
10. McCreadie, R., Macdonald, C., Ounis, I., Osborne, M., Petrovic, S.: Scalable Distributed Event Detection for Twitter. In: IEEE International Conference on Big Data. pp. 543–549 (2013)
11. Milstein, S., Chowdhury, A., Hochmuth, G., Lorica, B., Magoulas, R.: Twitter and the Micro-Messaging Revolution: Communication, Connections, and Immediacy – 140 Characters at a Time (An O'Reilly Radar Report) (2008)
12. Mokbel, M.F., Magdy, A.: Microblogs Data Management Systems: Querying, Analysis, and Visualization (Tutorial). In: ACM SIGMOD International Conference on Management of Data (SIGMOD). pp. 2219–2222 (2016)
13. Ozdikis, O., Karagoz, P., Oğuztüzün, H.: Incremental Clustering with Vector Expansion for Online Event Detection in Microblogs. Social Network Analysis and Mining **7**(1), 56 (2017)
14. Ozdikis, O., Senkul, P., Oguztuzun, H.: Semantic expansion of hashtags for enhanced event detection in twitter. In: Proceedings of the 1st International Workshop on Online Social Systems. Citeseer (2012)
15. Ozdikis, O., Senkul, P., Oguztuzun, H.: Semantic Expansion of Tweet Contents for Enhanced Event Detection in Twitter. In: International Conference on Advances in Social Networks Analysis and Mining (ASONAM). pp. 20–24 (2012)
16. Wang, Y., Xu, R., Liu, B., Gui, L., Tang, B.: A Storm-Based Real-Time Micro-Blogging Burst Event Detection System. In: Wang, X., Pedrycz, W., Chan, P., He, Q. (eds.) Machine Learning and Cybernetics, Communications in Computer and Information Science, vol. 481, pp. 186–195. Springer (2014)