Çoklu Veri Depo Ortamını Kullanan Alana Özgü Varlık Bağlama Yaklaşımı

A Domain Specific Entity Linking Approach Consuming Multistore Environment

Emrah İnan[1], Burak Yönyül[1], Fatih Tekbacak[2]
[1]Bilgisayar Mühendisliği Bölümü, Ege Üniversitesi, İzmir, Türkiye
{emrah.inan, burak.yonyul}@ege.edu.tr
[2]Bilgisayar Mühendisliği Bölümü, Adnan Menderes Üniversitesi, Aydın, Türkiye
ftekbacak@adu.edu.tr

***Özetçe*—Web üzerindeki verilerin çoğu yapısal olmayan bir halde bulunmaktadır ve bu nedenle makinelerin işleyebileceği bir yapıya dönüştürülmesi gerekmektedir. Dolayısıyla yapısal olmayan bu verilerin öncelikle gereksinime göre yapısallaştırılması ve kullanım durumlarını dikkate alarak farklı veri modellerinde saklanması uygun olacaktır. Gereksinimler ve çeşitleri arttıkça tek bir yöntem hepsini çözmede yetersiz kalmaktadır. Buna göre farklı saklama ihtiyaçlarını karşılayan tek bir saklama teknolojisinin kullanılması da uygun olmayacaktır. Farklı tipte şemaya sahip depoların (store) birlikte ve bütünleşik olarak yönetilmesi veritabanı literatüründe multistore ve polystore (çoklu depo) olarak ifade edilmektedir. Bu çalışma kapsamında Varlık Bağlama problemi üzerinde durularak veriler yapısallaştırılacak ve bu veriler farklı veri modellerinde bütünleşik bir ortamda yönetilecektir. Son olarak bütünleştirilmiş bu büyük veri ortamı sorgulanacak ve yöntem belirlenerek incelenecektir.**

Anahtar Kelimeler—büyük veri; çoklu depo; sorgulama; varlık bağlama; veri bütünleştirme.

***Abstract*—Most of the data on the web is non-structural, and it is required that the data should be transformed into a machine operable structure. Therefore, it is appropriate to convert the unstructured data into a structured form according to the requirements and to store those data in different data models by considering use cases. As requirements and their types increase, it fails using one approach to perform on all. Thus, it is not suitable to use a single storage technology to carry out all storage requirements. Managing stores with various type of schemas in a joint and an integrated manner is named as "multistore" and "polystore" in the database literature. In this paper, Entity Linking task is leveraged to transform texts into well-formed data and this data is managed by an integrated environment of different data models. Finally, this integrated big data environment will be queried and be examined by presenting the method.**

Keywords—big data; multi store; querying; entity linking; data integration.

I. GİRİŞ

Günümüz bilgi sistemlerinde işlenen veri hacminin her geçen gün artmasıyla ölçeklenebilirlik gereksinimi doğmuştur. Bu gereksinime göre çok büyük miktarlardaki veriyi performans kaybı olmadan saklayıp sorgulayabilmek önem kazanmıştır. Bu ihtiyaçtan ötürü her biri farklı kullanım amacına sahip No-SQL veri modelleri ortaya çıkmıştır. Bu çalışma bağlamında farklı veri modelleri Varlık Bağlama probleminde doğan çoklu veri modeli yönetimi için kullanılacaktır. "Her soruna uygun tek bir çözüm yoktur" ("No one size fits all") ilkesinden yola çıkılarak farklı yapıdaki verilere ihtiyaç duyulan Varlık Bağlama sisteminde tek bir veri modeli yaklaşımının kullanılması uygun olmayacaktır. Çok çeşitli kalıcılık yaklaşımına göre bir veri yönetim sisteminde farklı gereksinimleri karşılayan farklı veri modellerinin bir arada ve bütünleşik biçimde yönetilmesi gerekmektedir. Bu gereksinim de veri bütünleştirme problemini doğurmaktadır. Farklı ve özel şemalardaki verinin bütünleşik yönetilip sorgulanması amacıyla çoklu depo sistemleri (multistore systems) ortaya çıkmıştır. Bu çalışma kapsamında, çoklu depo ortamına gelecek sorguların yüksek performansta işletilmesi ve ayrıca sistemin eşzamanlı ve yoğun şekilde gelen sorguları ölçeklenebilir ve yüksek verimlilikte (high throughput) yönetilmesi hedeflenmektedir. Çalışmanın ikinci bölümünde literatür taraması yapılmıştır. Üçüncü bölümde ise çoklu veri depolarını kullanan varlık bağlama probleminin tanımı yapılarak dördüncü bölümde yöntem belirtilmiştir. Son bölümde çalışma özetlenerek gelecek çalışmalardan bahsedilmiştir.

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

II. LİTERATÜR ÖZETİ

Çalışma kapsamında Varlık Bağlama problemi üzerinde durularak veriler yapısallaştırılacak ve bu veriler farklı veri modellerinde bütünleşik olarak yönetilecektir. Bu bölümde Varlık Bağlama problemi ve çözümüne ilişkin yöntemler ile çoklu veri modeli yönetimiyle ilgili çalışmalar incelenecektir.

Varlık Bağlama yönteminde öğreticili öğrenme, öğreticisiz öğrenme ve bilgi tabanı kaynaklarını kullanan yöntemler vardır [1]. Bu çalışma kapsamında Bilgi Tabanı kullanan yöntemler üzerine çalışılacaktır. Varlık Bağlama problemi çözümü için literatürde en çok kullanılan bilgi kaynaklarından biri Vikipedi kaynağıdır. Vikipedi tabanlı ilk çalışmalar Tanımlı Varlık Çözümleme ve Vikifikasyon (Wikification) olmak üzere iki alt görevden oluşmaktadır. Tanımlı Varlık Çözümleme [2], [3] metin üzerindeki bahsin Vikipedi'den elde edilen sözlük ile çözümlenmesini amaçlarken Vikifikasyon metindeki kelime parçalarının kapsama göre en uygun Vikipedi sayfasıyla bağlanmasını amaçlamaktadır. Bu yaklaşım seçilen Vikipedi sayfalarının kendi aralarındaki global benzerliğini yoksayarak sadece yerel özelliğe dayanmaktadır. Yerel benzerliğe dayanan yaklaşımlarda her bir bahis için verilen cümle ayrı ayrı çözümlenirken, global benzerliğe dayanan çalışmalar verilen cümledeki bütün bahislerin birbirleriyle uyumlu çözümlenmesini amaçlar. Örnek olarak verilen cümlede "Michael Jordan" bahsi bilimadamı olan yerine basketbolcuyu etiketliyorsa aynı cümledeki "Bulls" bahsi de Vikipedi'de sayfalar arasındaki bağ yapısından benzerlik hesaplanarak "Chicago Bulls" basket takımıyla bağlanır [4].

Global benzerliği kullanan Cucerzan, Vikipedi sayfalarındaki kelime kümesinden oluşan kapsam bilgisini de çözümleme sürecine dahil etmiştir [3]. Milne ve Witten [5] daha sonra bağlanacak varlıklara ait Vikipedi sayfalarının birbiriyle olan bağlantı yapısından hesaplanan anlamsal benzerliği ön plana çıkararak aday kelime anlamının çözümlemesini incelemiştir. Bu yöntem Vikipedi bağ yapısını kullanarak iki aday varlığın özelleşmiş bu bağ yapısındaki kendilerine gelen ve kendilerinden çıkan bağların benzerliğinden faydalanmaktadır. Bu açıdan bakıldığında alana özgü Vikipedi bağ yapısının daraltıldığı durumlarda yöntemin başarısı olumsuz etkilenmektedir. Kulkarni ve arkadaşları [6] ise daha sonra sadece yerel ya da sadece global benzerliğe dayanan yaklaşımlarda karşılaşılan sorunların üstesinden gelmeyi amaçlamıştır. Uzun metinlerdeki varlıkların bağlanması yöntemi, çok kapsamlı hesaplamalara dayandığından dolayı zaman bakımından hızlı sonuç üretememektedir.

Varlık Bağlama ve Kelime Anlamı Çözümleme için geliştirilen Babelfy [7] çoklu dil destekli bir uygulama olup BabelNet [8] kaynağını kullanmaktadır. BabelNet büyük ölçekli çoklu dil desteği veren ansiklopedik sözlük ve anlamsal ağ altyapısı olarak WordNet, Open Multilingual WordNet ve Wiktionary'den faydalanırken; tanımlı varlık isimlerinin bağlanmasında Vikipedi, Wikidata kaynaklarından yararlanmaktadır. YAGO3 [9] bilgi tabanı YAGO [10] bilgi tabanının çoklu dil destekli hali olarak WordNet ve Vikipedi kaynaklarından faydalanmıştır. UWN [11] ve YAGO2 [12] bilgi tabanları YAGO3 gelişim sürecine katkı veren çalışmalardır. Babelfy ve AIDA uygulamalarında görüldüğü gibi mevcut durumda çalışmalar önce geniş kapsamlı büyük bir Bilgi Tabanı elde ettikten sonra verilen girdi metnine göre bu Bilgi Tabanı'nı daraltarak en yoğun alt çizgeden çözümleme yapmayı amaçlamaktadırlar.

Veri Bütünleştirme farklı kaynaklardaki verinin bir araya getirilerek, bu verinin kullanıcıya birleştirilmiş bir biçimde sunulması problemine odaklanan alandır. İlişkisel veri tabanlarının ortaya çıkışından beri çalışılan bir konu olmakla birlikte, veritabanı sistemleri varlığını sürdürdükçe de çalışılmaya devam edecektir. Bu yaklaşımı teorik yönleriyle inceleyen Lenzerini [13] temel olarak; bir veri bütünleştirme uygulamasının modellenmesi, veri bütünleştirmede sorgu çalıştırma, tutarsız veri kaynaklarıyla başa çıkma ve sorgular üzerinde çıkarsama konuları üzerinde durmuştur. Veri bütünleştirme sistemleri global şema (global schema) ve kaynaklar kümesi (set of sources) olmak üzere iki temel bileşeni baz alan bir mimari üzerine kurulmaktadır. Yazılım mimarisi açısından Local As View (LAV) ve Global As View (GAV) olmak üzere iki ana yaklaşım bulunmaktadır. Lenzerini'nin çalışmasında bu iki yaklaşım; modelleme, sorgu işletimi, tutarsız kaynakların ele alınması ve sorgu çıkarsama açılarından değerlendirilmiş ve birbiri ile kıyaslanmıştır.

LAV yaklaşımında global şema, kaynaklardan bağımsız olarak tanımlanır ve global şema ile kaynaklar arasındaki ilişkiler her bir kaynak global şemanın görüntüsü (view) olarak tanımlanarak kurulur. LAV yaklaşımı global şemanın sabit ve iyi tanımlanmış olduğu durumlarda (bir kurumsal model veya ontoloji olarak) avantajlıdır. Global şema değişmediğinden yeni bir kaynak sisteme kolayca eklenebilir ve sistem kolayca genişletilebilir. Yeni kaynağın diğer kaynaklar ile bağlantısı global şema üzerinde tanımlanır. Sorgu işletimi view'lar üzerinden eşlemelerin tahmin edilmesi ile yapılır ve tam (exact) sonuç garanti edilmez. Verinin bütünleştirilmesi global şemadaki eşlemelere göre sorgu işlemcisi tarafından ele alınır.

GAV yaklaşımında ise global şema, kaynaklar cinsinden ifade edilerek oluşturulur. GAV yaklaşımı kaynakların sabit olduğu ve sorgu işletimlerinin yoğun olduğu veri bütünleştirme sisteminde etkilidir. Sorgu işletimi LAV yaklaşımına göre daha basittir. Global şema ile yerel şemalar arasındaki sorgu ve veri dönüşümü, sarmalayıcılar (wrappers) aracılığıyla gerçekleştirilir. Her alt sorgu ilişkili olduğu kaynak üzerinde işletilip sonuçları sarmalayıcılar tarafından dönüştürüldükten sonra birleştirilerek ana sorgu sonucu elde edilir. Sorgu, sarmalayıcılar tarafından ele alındığından dönüşüm

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

önceden bellidir ve tam sonuç garanti edilir. Ancak sisteme yeni kaynak eklemek zordur, çünkü yeni eklenen şema için veriyi ve sorguyu dönüştürecek bir sarmalayıcı yazılması gereklidir.

Halevy ve arkadaşları [14] yaptıkları çalışma kapsamında veri bütünleştirme çalışmalarını derlemişlerdir. Burada veri bütünleştirme kavramı; şema eşlemelerinin yaratılması, uyarlanabilir sorgu işleme, XML'in kullanımı, model yönetimi, uçtan uca veri yönetimi ve yapay zekâ yönleriyle değerlendirilmiş ve bu alanlarda yapılan çalışmalara değinilmiştir. Ayrıca veri bütünleştirme endüstrisinin çalışma alanları ve karşılaşabilecekleri ölçeklenebilirlik, performans, kapsam ve yapay zekâ araçlarının kullanımı gibi zorluklar tartışılmıştır. Çalışmada ayrıca LAV yaklaşımı ile yeni bir kaynağın ekleme kolaylığı ve kaynak şemalar üzerinde kısıtların kolaylıkla tanımlanabilmesi avantajları vurgulanmıştır. Değinilen bir diğer önemli nokta gelecek çalışmalar kısmında bahsedilen veri uzaylarının "ihtiyacın kadar kullan" ilkesine dayanan veri yönetim şeklidir. Buna göre şema eşlemeleri kapsamlı olarak düşünülmeden her bilgi kaynağı birer servis olarak sunulmakta ve sistem kullanılmaya başlandıktan sonra ihtiyaca göre şema eşlemeleri eklenebilmektedir.

Çoklu depo bütünleştirme için ön çalışmaları ve ilk gerçekleştirimi yapılan CloudMdsQL [15] sorgu dili fikrini Kolev ve arkadaşları [16] hayata geçirmiştir. Bu çalışmada, tasarlanan sorgu dili ve bu dile özel geliştirilen sorgu motoru tüm detayları ve işleyişiyle ele alınmıştır. Öncelikle veri modellerine ve sorgu dilindeki temel yapılara değinilmiştir. Daha sonra sorgu motorunun yapısı ve temel bileşenleri anlatılmıştır. Dağıtık halde master-slave yapıda çalışan sorgu motorundaki düğümler veri ve sorgu planı değiş tokuşu yapabilmektedir ve sorgu işletimi yöntemlerinden bağlı birleşimi (bind join) esas almaktadır. Sorgu dilinin özgünlüğü ise hem SQL sorgularını hem de veri depolarına (data stores) özgü sorguları bir arada çalıştırabilmesidir. Python dilinde yazılan sorgu derleyicisi depolar arası bütünleştirimi Python'da yer alan "geçici soyut tablo ifadeleri" (named-table-expression) yardımı ile sağlamaktadır. Sorgu işletiminde öncelikle sorgu yorumlanarak ağaç yapısında öncül sorgu işletim planı oluşturulmaktadır. Maliyet modeline (cost-model) veya üst veriye dayanarak alternatif sorgu planları oluşturulup en iyisi seçilmekte ve ilk işçi (worker) düğüme iletilerek bağlı birleşim yöntemiyle ve sırasıyla ilgili depoların sarmalayıcı fonksiyonları kullanılarak işletilmektedir. Tasarlanan sorgu dilinin ve sorgu motorunun çalıştığını göstermek için üç adet deponun bulunduğu bir kullanım durumu yaratılmıştır. Buna göre ilişkisel veritabanında bilim insanlarının bilgileri, belge veritabanında yayın ve yayınlar ile ilgili yorum bilgileri, çizge veritabanında ise bilim insanlarının birbirleriyle olan arkadaşlık bilgileri tutulmaktadır. Değerlendirme (evaluation) kısmında ayrıca sorgu motorunun gerçekleştirim yapısı, veri kümelerinin içerdiği veri miktarı ve seçicilik değerleri, değerlendirme ortamının kurulum ayarları da dahil olmak üzere tüm ayrıntılar açıkça anlatılmıştır. Eniyileştirmelerin etkisini göstermek için beş adet sorgu ve her sorgu için de üç farklı sorgu işletim planı oluşturulmuş ve sonuçlar sorgu işletim süreleri cinsinden grafiksel olarak gösterilip yorumlanmıştır. Yazarların bu araçla ilgili birçok çalışması bulunmaktadır [16-19].

Duggan ve arkadaşları [20] birden fazla veri modelini kapsayan bir bilgi yönetimi ihtiyacına vurgu yaparak yeni bir birleştirilmiş veritabanı (federated database) bakış açısı olan çoklu depo (polystore) yaklaşımını ortaya atmışlardır. Intel Science ve Technology Center for Big Data gruplarıyla ortak çalışılarak, MIMIC-II [21] hastane veri kümesi üzerinde farklı veri modellerini kapsayan bir kullanım durumu oluşturulmuş ve tasarlanan BigDAWG isimli çoklu depo sistemi prototipi bu kullanım durumu üzerinde denenmiştir. Bu yeni yaklaşımda üç hedef üzerine odaklanılmıştır. Bunlardan ilki saklanacak nesneler için "yer şeffaflığı"dır (location transparency) ki böylece kullanıcılar birden çok veri yönetim sistemini kapsayan bildirimsel (declarative) sorgular atabilmektedir. Çalışma kapsamında ortaya atılan "bilginin adası" (island of information) soyutlaması tek bir sorgu dili ile erişilen saklama motorlarını (storage engine) ifade etmektedir. İkinci hedef "anlamsal bütünlük"tür (semantic completeness) ve öyle ki yeni bir saklama motoru çoklu depoya eklendiğinde, bu saklama motorlarıyla ilgili herhangi bir işlevsellik kaybedilmeyecektir. Üçüncü hedefte ise kullanıcılar belirli bir arka-uçta (back-end) saklanan nesnelere farklı adalar üzerinden erişebileceklerdir. Örneğin; fiziksel olarak belirli bir arka uçta tutulan bir veriye hem dizi (array) hem de ilişkisel (relational) adalar üzerinden erişilebilecektir. Çoklu depo içerisindeki bir ada, yer şeffaflığını her saklama motoru için yazılan bir dolgu (shim) ile desteklemektedir. Çalışma kapsamında tasarlanan sorgu mekanizmasında ilgili alt sorgunun çalıştırılacağı ada SCOPE anahtar kelimesi ile belirlenmekte ve bir başka alt sorguyla ilişkili birleşim (join) operasyonlarının gerçekleştirilmesi için ise CAST anahtar kelimesi kullanılarak arka-uçlar arasında nesnelerin kopyalanması sağlanmaktadır. Tasarlanan sistemdeki özgün yanlardan biri de şemalar hakkında bir üst-veriye sahip olmayı gerektirmeden kara kutu (black box) yaklaşımıyla sorguların işletileceği arka-uçların belirlenmesi ve sorgu eniyileştirmesinin sağlanmasıdır. Bunun için kara-kutu yaklaşımında üç farklı çalışma modu ile profilleme yapılmaktadır. Eğitim (Training) modunda sorgu tüm saklama motorları üzerinde çalıştırılıp süreleri kaydedilmektedir. Eniyileşmiş (Optimized) modda sorgu en kısa süreyi veren saklama motoru üzerinde işletilmektedir. İyimser (Opportunistic) modda ise yeni saklama motorlarının kaynakları uygun duruma geldiğinde saklanan alt-sorgular bunlar üzerinde işletilmektedir. Böylece kara-kutu yaklaşımıyla sistem üzerinde alt-sorgu bazında dinamik profilleme yapılarak, her alt sorgunun işletileceği en uygun saklama motorunun belirlenmesi sağlanmış olmaktadır.

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

Bir diğer çalışmada [22] ölçeklenebilir ve kendi kendini düzenleyebilen bir çoklu depo sistemi tasarlamışlardır. ESTOCADA ismini verdikleri ve altyapısında birçok veri modelini destekleyebilen bu çoklu depo sisteminin temelinde veri bütünleştirme yöntemlerinden LAV (local as view) ve view tabanlı yeniden yazma (view-based rewriting) yatmaktadır. Yerel sorgu işletimi, alt sorgu sonuçları toplanarak oluşturulan bir alt veri kümesi (view) üzerinde yapılmaktadır. Ayrıca birleşik (conjunctive) sorgularda ve/veya sorgu planı oluşturulurken çalıştırılacak olan sorgu, çalıştırılacağı deponun diline ve kısıtlarına uydurularak yeniden düzenlenmektedir (reformulation under constraints). Sistemde veri kümesi parçalama (dataset fragmentation) yöntemi uygulanarak ölçeklenebilirlik sağlanmaktadır. Yerel depoların performansı, sorgulama ve saklama olmak üzere iki kısıta göre değerlendirilip bir sorgu için en iyi çalıştırma planı oluşturulmaktadır. Zaman içerisinde bu planlar ve çalışma süreleri kaydedilip, gerektiğinde sistem tarafından ilgili sorgu parçasının verisinin bir başka depoya yüklenmesi (örneğin verinin ilişkisel veritabanından alınıp sütun veritabanına taşınması) işlemi gerçekleştirilmektedir. Böylece sistemin kendi kendini düzenleme yeteneğini kazanmasının yanında ölçeklenebilirliği de artmış olmaktadır.

Bondiombouy ve Valduriez [23] sunmuş oldukları araştırma raporunda (research report) ilk olarak çoklu veritabanı sistemlerinde (multi-database) sorgu işletimi yöntemlerine değinmişlerdir. Çoklu depo sistemlerini sorgulama yöntemleri açısından yerel depolara (native stores) bağımlılıklarına göre "gevşek bağımlı", "sıkı bağımlı" ve "melez" olarak sınıflandırmış ve her sınıfa uyan üçer çalışma tanıtmışlardır. Bu çalışmalardan gevşek bağımlı olarak; BigIntegrator [24], Forward [25] ve QoX [26]; sıkı bağımlı olarak; Polybase [27], HadoopDB [28], Estocada [22]; melez olarak ise; SparkSQL [29], BigDAWG [20], CloudMdsQL [16], [18], [19] sistemleri örnek olarak seçilmiştir. Anlatılan çalışmaların mimarisi, sorgu işletimi ve eniyileştirme yöntemleri açıklanmıştır.

III. PROBLEM TANIMI

Varlık Bağlama ve Çoklu Veri Modeli Bütünleştirme ayrı ayrı yoğun olarak çalışılan araştırma alanlarıdır. Ancak literatürde bu alanların beraberce incelendiği ve birbirlerinin çözümünü kolaylaştırmak için kullanıldığı çalışma bizim gördüğümüz kadarıyla bulunmamaktadır. Bu çalışma bağlamında Varlık Bağlama yönteminden doğacak farklı ve çoklu veri modeli gereksinimi dolayısıyla iki çalışma alanının beraberce incelenmesi uygun görülmüştür.

Geliştirilecek Varlık Bağlama yönteminin kapsam benzerliği hesaplamaları belge deposunda saklanacaktır. Bilgi tabanı kullanılarak gerçekleştirilecek yöntemdeki varlıkların kendi aralarındaki benzerliğinin ölçülmesi için çizge deposunda saklanan alana özgü bilgi tabanından yararlanılacaktır. Ayrıca, kelime sıklıkları anahtar-değer deposunda tutularak atıflarla aday varlıklar arasındaki benzerlik hesaplamasında kullanılacaktır.

Bu çalışma kapsamında çoklu depo ortamına gelecek sorguların yüksek performansta işletilmesi, ayrıca sistemin eşzamanlı ve yoğun şekilde gelen sorguları ölçeklenebilir ve yüksek verimlilikte (high throughput) yönetebilmesi için bir yöntem önerilmektedir.

IV. YÖNTEM

Bu çalışmada Varlık Bağlama probleminin çözümü ve bu problemde Çoklu Veri Modeli Bütünleştirme yönteminin geliştirilmesi için sinema filmleri alanında kullanım senaryosu gerçekleştirilmektedir. Öncelikle Vikipedi kaynağındaki sinema kategorisindeki film, yönetmen, oyuncu ve tür gibi bilgileri içeren sayfalar ayrıştırılarak (parsing) gereksinime uygun veri modellerinde saklanacaktır. Örnek olarak filmlere ait Vikipedi sayfalarının metni, belge veritabanı olan MongoDB deposunda tutulacaktır. Her Vikipedi sayfasındaki en sık geçen kelimeler ve bu kelimelerin bulunduğu sayfalar Redis anahtar-değer veritabanında saklanacaktır. Vikipedi bağ yapısı ve kavramlara ait çizge ise Neo4J veri tabanında RDF modeline uyumlu olacak şekilde tutulacaktır.

Alan bağımlı varlık bağlama yöntemi, atıf-varlık benzerliği ve varlık-varlık ilintililiği hesaplanması olmak üzere iki alt yöntemden oluşmaktadır. Belirli bir alana bağlı metinler ilk önce atıfların tespit edilmesi (Mention Detection) aşamasında işlenmektedir. Bu adım, sonraki alt bölümde Atıf-Varlık Benzerliği adımı içinde değerlendirilecektir. Bu çalışmanın özgün yöntemleri Varlık-Varlık ilintililiği (Entity-Entity Relatedness) adımında gösterilmiştir. Son adımda ise anlam karmaşıklığı çözümlenen atıfların alan bağımlı bilgi tabanındaki varlıklarla bağlanmış etiketli metni elde edilmektedir.

Bu çalışmada atıf-varlık benzerliği hesaplanması için öncelikle metinlerdeki atıfların etiketlenmesi gerekmektedir. Şekil-1'de gösterildiği gibi alana bağlı bilgi tabanındaki örnekler (instance) alfabetik sırada atıflar (bahis) sözlüğünde saklanır. Daha sonra bu sözlükten hızlı arama yapılabilmesi için Apache Solr aracı (Elastic Search aracının kurulum maliyetinden dolayı test edilmesi ileriki aşamalara bırakılmıştır) ile indekslenir.

Verilen metin kelimelere ayrılarak en fazla altılı kelime gruplarında (TagMe [30] çalışması örnek alınarak) n-gram listesi oluşturulur. Son adımda oluşturulan n-gram listesi indekslenmiş atıflar (bahis) sözlüğünde sorgulanarak atıfların etiketlenmesi sağlanır. Kelime grupları eşleşirken en uzun dizi eşleşmesi örnek alınmaktadır. Örneğin, "Wicker", "Park" ve "Wicker Park" atıfları içinden karakter boyutu en uzun olan ve diğer iki kelimeyi de içeren "Wicker Park" atıfı seçilmektedir. Girdi metninde etiketlenmiş her bir atıf için aday varlıklar çıkarılarak bu varlıklardan atıfa bağlam açısından en benzeri bulunmaktadır. En benzer atıf-varlık

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

Şekil 1. Atıfların belirlenmesi için geliştirilen altyapı

çiftini bulmak için varlıklara ait Wikipedia sayfalarındaki içeriklerden oluşturulan kelime sıklıkları ve bu kelimelerin sıklık sayıları dikkate alınarak Jaccard ölçümü kullanılmaktadır.

$$J(L_m, L_e) = \frac{L_m \cap L_e}{L_m \cup L_e} \quad (1)$$

Yukarıdaki formülde atıfın bulunduğu girdi metninden oluşturulan kelime sıklık listesi (L_m) ile aday varlığa ait Wikipedia sayfasından elde edilen sıklık listesi (L_e) için Jaccard hesaplanmaktadır. Sırasıyla atıf ve varlık listelerinin her bir listedeki kelime sıklıklarının toplam sıklığa bölünmesi ile Jaccard benzerlik değeri elde edilmektedir. Bu değerlerin kesişimlerinin birleşime bölünmesi ile 0-1 aralığında bir değer her bir atıf-varlık listesi için hesaplanmaktadır. Bundan sonraki aşamada, 1 değerine en yakın atıf-varlık çifti belirlenmektedir.

Varlık Bağlama yaklaşımları içinde henüz kullanılmamış olan varlık-varlık ilintililiğinin hesaplanması için RDF2VEC [31] çalışması Sinir Ağı Dili (Neural Language Model) modellerini RDF çizge izdüşümlerine (embeddings) uyarlamıştır. Bu dil modellerinin ana varsayımı, metinlerdeki kelime dizilerinde birbirine yakın olan kelimelerin anlamsal olarak daha ilintili olmalarıdır. Bu çalışma, kelime dizileri yerine RDF modelindeki varlıklar ve varlıklar arasındaki ilişkilerden oluşan varlık-ilişki dizileri tanımlamaktadır. Böyle bir yaklaşımın RDF çizge verisine uygulanması için öncelikle çizgenin varlık-ilişki dizilerine dönüştürülmesi gerekmektedir. Böylece bu varlık-ilişki dizileri, Sinir Ağı Dili modellerinde eğitilerek RDF çizgesindeki her bir varlığın Gizil Özellik Uzayı (Latent Feature Space)'ndaki nümerik değerler vektörü şeklinde temsili sağlanmaktadır. RDF2VEC ile Anlamsal İlintililiğin Ölçülmesi için adımlar aşağıdaki gibi listelenmiştir:

1. RDF çizgelerinin seçilen strateji (Weistfeiler-Lehman Subtree RDF Graph Kernels, graph walks) ile varlık-ilişki dizilerine (entity-relation sequences) çevrilmesi
2. Varlık-ilişki dizileri kullanılarak Sinir Ağı Dil Modeli (Neural Language Model) yöntemlerinden (CBOW, Skip-Gram) biri ile modelin kurulması
3. Sinir Ağı Dil modelinden yararlanarak iki varlığın anlamsal ilintililiğinin Softmax fonksiyonu kullanılarak hesaplanması

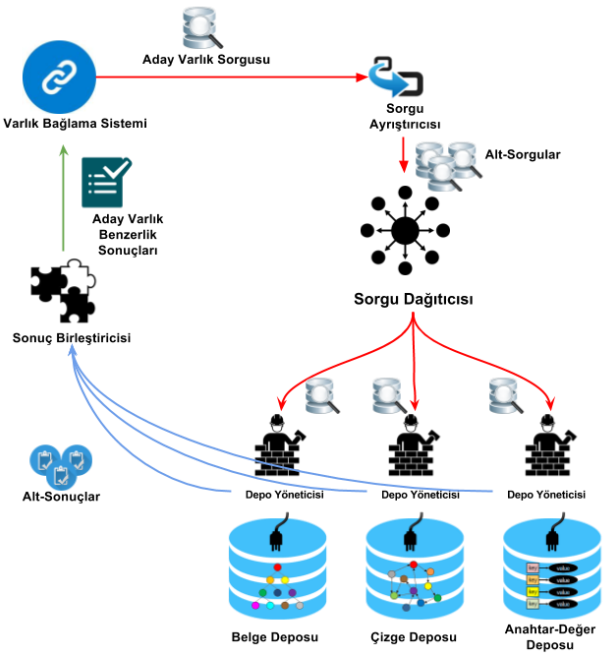
Böylece iki varlığın anlamsal ilintililiği Softmax fonksiyonu kullanılarak bir varlığın diğer varlığın bağlamında olma olasılığı olarak hesaplanmaktadır. Bu şekilde ilintililiği 1'e en yakın çıkan varlık çiftinin birbirine en çok ilintili olduğu anlaşılmaktadır.

Çoklu Depo ortamının taslak olarak mimarisi Şekil-2'de görülmektedir. Varlık Bağlama kapsamında sistemin işleyişi bu mimari model üzerinden anlatılacak olursa; öncelikle aday varlıkların benzerliklerini hesaplamak amacıyla yollanan aday varlık sorgusu dağıtılarak farklı veri depoları üzerinden işlem yapılır.

Sorgu dağıtıcısı ile Anahtar-Değer Deposu'nda saklanan aday varlıklara ait kelime sıklık listesi (L_m) çekilecektir. Aynı anda belge deposunda saklanan aday varlıkların her biri için çekilmiş Wikipedia sayfalarından elde edilmiş sıklık listesi (L_e) yine mevcut anahtar-değer deposundan sorgulanacaktır. Böylece bu iki listeden mevcut atıf ve aday varlıklar arasında Jaccard benzerlikleri hesaplanacaktır.

Geliştirilecek Varlık Bağlama yönteminin RDF2VEC modeline dayanan yöntemi için alana özgü bilgi tabanı çizge deposunda tutulacaktır. Sorgu dağıtıcısı üzerinden depo yöneticisine ulaşan aday varlık alt-sorguları RDF2VEC modeline uygun sonuçları döndürecektir. Daha sonra bu sorgular Softmax fonksiyonu için girdi olacaktır ve aday varlıklar arasındaki ilintililik hesaplamasında kullanılacaktır.

Sisteme gelen ana sorgu "Sorgu Ayrıştırıcısı" birimi tarafından ayrıştırılarak ilgili yerel depolarda çalıştırılmak



Şekil 2. Çoklu depo ortamı mimarisi

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

üzere alt sorgulara ayrılır ve "Sorgu Dağıtıcısı" birimine iletilir. Sorgu Dağıtıcısı, gelen alt sorguların çalıştırılacağı depoyu bularak, her alt sorguyu ilgili olduğu deponun yönetiminden sorumlu "Depo Yöneticisi" birimine iletir. Depo Yöneticisi, kendisine gönderilen alt-sorguyu ilgili yerel depo üzerinde çalıştırır ve sonucu ortak veri modeline (RDF) dönüştürür. Dönüştürülen alt sorgu sonuçları ana sorgu sonucunun oluşturulması için "Sonuç Birleştiricisi"ne gönderilir. Sonuç Birleştiricisi; hash join yöntemini uygulayarak, alt sorgu sonuçlarını birleştirip ana sonucu oluşturur ve sonucu Varlık Bağlama sistemine iletir.

V. SONUÇ

Bu çalışma kapsamında Varlık Bağlama problemi üzerinde durularak veriler yapısallaştırılacak ve bu veriler farklı veri modellerinde bütünleşik olarak yönetilecektir. Gelişmiş Bilgi Tabanı kaynaklarının ortaya çıkmasıyla Varlık Bağlama probleminde bu bilgi tabanlarının kullanımı giderek yaygınlaşmıştır. Bu açıdan bakıldığında Varlık Bağlama yönteminde kullanılacak tanımlı varlıkların çizge veri modelinde saklanması uygun olacaktır. Ek olarak tanımlı varlıklara ait dokümanlar belge veritabanında tutulacaktır. Bu dokümanlardaki kelime sıklıkları ve kelimelerin geçtiği metinler anahtar-değer veri tabanında saklanacaktır. Bu ortamın yönetilmesi için de çoklu depo ortamının geliştirilmesi gereksinimi ortaya çıkmıştır. Çoklu depo ortamı, Varlık Bağlama yönteminin gerçekleştirimi için sorgulanma ve veri modeli bütünleştirilmesi işlemlerinde kullanılacaktır. Gelecek çalışmalarda bu sorgu yönetiminin başarımı ve Varlık Bağlama yönteminin diğer yöntemlerle karşılaştırılması yapılacaktır.

TEŞEKKÜR

Yrd. Doç. Dr. Fatih Tekbacak bildirinin yazımı esnasında Alanya Alaaddin Keykubat Üniversitesi (ALKÜ)'nde görev yapmakta olup ALKÜ'ye bu süreçte verdiği destekten dolayı teşekkür ederiz.

KAYNAKÇA

[1] W. Shen, J. Wang, J. Han, "Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions", IEEE Transactions on Knowledge and Data Engineering, 27(2), 443-460, February 2015.

[2] R. Bunescu, M. Pasca, "Using Encyclopedic Knowledge for Named Entity Disambiguation", Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06), Trento, Italy, 9-16, 2006.

[3] S. Cucerzan, "Large-Scale Named Entity Disambiguation Based on Wikipedia Data", Proceedings of Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007), Prague, Czech Republic, 708-716, 2007.

[4] L. Ratinov, D. Roth, D. Downey, M. Anderson, "Local and Global Algorithms for Disambiguation to Wikipedia", Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT'11), Portland, Oregon, 1375-1384, June 19-24, 2011.

[5] D. Milne, I. H. Witten, "Learning to Link with Wikipedia", Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM '08), Napa Valley, California, USA, 509–518, October 2008.

[6] S. Kulkarni, A. Singh, G. Ramakrishnan, S. Chakrabarti, "Collective Annotation of Wikipedia Entities in Web Text", Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'09), Paris, France, 457-466, June 28 - July 1, 2009.

[7] A. Moro, A. Raganato, R. Navigli, "Entity Linking Meets Word Sense Disambiguation: A Unified Approach", Transactions of the Association for Computational Linguistics, 2, 231-244, 2014.

[8] R. Navigli, S. P. Ponzetto, "BabelNet: The Automatic Construction, Evaluation and Application of A Wide-Coverage Multilingual Semantic Network", Artificial Intelligence, 193, 217-250, December 2012.

[9] F. Mahdisoltani, J. Biega, F. M. Suchanek, "YAGO3: A Knowledge from Multilingual Wikipedias", 7th Biennial Conference on Innovative Data Systems Research (CIDR 2015), Asilomar, California, USA, January 4-7, 2015.

[10] F. M. Suchanek, G. Kasneci, G. Weikum, "YAGO: A Core of Semantic Knowledge Unifying WordNet and Wikipedia", Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, Canada, 697-706, May 8-12, 2007.

[11] G. de Melo, G. Weikum, "UWN: A Multilingual Lexical Knowledge Base", Proceedings of the ACL 2012 System Demonstrations, Jeju Island, Korea, 151-156, July 2012.

[12] J. Hoffart, F. M. Suchanek, K. Berberich, E. L. Kelham, G. de Melo, G. Weikum, "YAGO2: Exploring and Querying World Knowledge in Time, Space, Context and Many Langauges", Proceedings of the 20th International Conference Companion on World Wide Web (WWW 2011), Hyderabad, India, 229-232, 2011.

[13] M. Lenzerini, "Data Integration: A Theoretical Perspective", Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, Madison, Wisconsin, 233-246, June 3-5, 2002.

[14] A. Halevy, A. Rajaraman, J. Ordille, "Data Integration: The Teenage Years", Proceedings of the 32nd International Conference on Very Large Databases (VLDB'06), Seoul, Korea, 9-16, September 12-15, 2006.

[15] C. Bondiombouy, B. Kolev, O. Levchenko, P. Valduriez, "Integrating Big Data and Relational Data with A Functional Sql-like Query Language", International Conference on Database and Expert Systems Applications, Valencia, Spain, 170-185, September 2015.

[16] B. Kolev, P. Valduriez, C. Bondiombouy, R. Jiménez-Peris, R. Pau, J. Pereira, "CloudMdsQL: Querying Heterogeneous Cloud Data Stores with A Common Language", Distributed and Parallel Databases, 34(4), 463-503, 2015.

[17] C. Bondiombouy, B. Kolev, O. Levchenko, P. Valduriez, "Multistore Big Data Integration with CloudMdsQL", Transactions on Large-Scale Data-and Knowledge-Centered Systems XXVIII, 9940, 48-74, 2016.

[18] B. Kolev, C. Bondiombouy, O. Levchenko, P. Valduriez, R. Jimenez-Péris, R. Pau, J. Pereira, "Design and Implementation of the CloudMdsQL Multistore System", 6th International Conference on Cloud Computing and Services Science (CLOSER), 1, Roma, Italy, 352-359, April 2016.

[19] B. Kolev, C. Bondiombouy, P. Valduriez, R. Jiménez-Peris, R. Pau, J. Pereira, "The CloudMdsQL Multistore System", Proceedings of the 2016 International Conference on Management of Data (SIGMOD'16), San Francisco, California, USA, 2113-2116, June 26 - July 01, 2016.

[20] J. Duggan, A. J. Elmore, M. Stonebraker, M. Balazinska, B. Howe, J. Kepner, S. Madden, D. Maier, T. Mattson, S. Zdonik, "The BigDAWG Polystore System", ACM Sigmod Record, 44(2), 11-16, June 2015.

[21] M. Saeed, M.Villarroel, A. T. Reisner, G. Clifford, L. W. Lehman, G. Moody, T. Heldt, T. H. Kyaw, B. Moody, R. G. Mark, "Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A Public Access Intensive Care Unit Database", Critical Care Medicine, 39(5), 952-960, 2011.

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.

[22] F. Bugiotti, D. Bursztyn, A. Deutsch, I. Ileana, I. Manolescu, "Invisible Glue: Scalable Self-Tuning Multi-Stores", Conference on Innovative Data Systems Research (CIDR), Asilomar, United States, January 2015.

[23] C. Bondiombouy, P. Valduriez, Query Processing in Multistore Systems: An Overview, RR-8890, INRIA Sophia Antipolis - Méditerranée, 2016.

[24] M. Zhu, T. Risch, "Querying Combined Cloud-Based and Relational Databases", International Conference on Cloud and Service Computing (CSC 2011), Hong Kong, China, 330-335, December 12-14, 2011.

[25] K. W. Ong, Y. Papakonstantinou, R. Vernoux, "The SQL++ Semi-structured Data Model and Query Language: A Capabilities Survey of SQL-on-Hadoop, NoSQL and NewSQL Databases", CoRR, abs/1405.3631, April 2014.

[26] A. Simitsis, K. Wilkinson, M. Castellanos, U. Dayal, "Optimizing Analytic Data Flows for Multiple Execution Engines", Proceedings of the 2012 International Conference on Management of Data (SIGMOD'12), Scottsdale, Arizona, USA, 829-840, May 20-24, 2012.

[27] D. J. DeWitt, A. Halverson, R. Nehme, S. Shankar, J. Aguilar-Saborit, A. Avanes, M. Flasza, J. Gramling, "Split Query Processing in Polybase", Proceedings of the 2013 International Conference on Management of Data (SIGMOD'13), New York, USA, 1255-1266, June 22-27, 2013.

[28] A. Abouzeid, K. Bajda-Pawlikowski, D. Abadi, A. Silberschatz, A. Rasin, "HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads", Proceedings of the VLDB Endowment, 2(1), 922-933, August 2009.

[29] M. Armbrust, R. S. Xin, C. Lian, Y. Huai, D. Liu, J. K. Bradley, X. Meng, T. Kaftan, M. J. Franklin, A. Ghodsi, M. Zaharia, "Spark SQL: Relational Data Processing in Spark", Proceedings of the 2015 International Conference on Management of Data (SIGMOD'15), Melbourne, Victoria, Australia, 1383-1394, May 31- June 4, 2015.

[30] P. Ferragina, U. Scaiella, "TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities)", Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10), Toronto, Canada, 1625-1628, 2010.

[31] P. Ristoski, H. Paulheim, "RDF2Vec: RDF Graph Embeddings for Data Mining", 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, 498-514, October 17-21, 2016.

Bu çalışma ASYU-2017 Konferansında sunulmuş ve sadece özet olarak
Bildiri Özetleri Kitabı (ISBN: 978-605-84722-8-0)'nda yayımlanmıştır.