T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ONTOLOJİLERİN OTOMATİK OLARAK EŞLEŞTİRİLMESİ

ALİ ALPEREN ÇOLAK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
YRD. DOÇ.DR. YUNUS EMRE SELÇUK

İSTANBUL, 2011

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ONTOLOJİLERİN OTOMATİK OLARAK EŞLEŞTİRİLMESİ

ALİ ALPEREN ÇOLAK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ PROGRAMI

DANIŞMAN
YRD. DOÇ.DR. YUNUS EMRE SELÇUK

İSTANBUL, 2011

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ONTOLOJİLERİN OTOMATİK OLARAK EŞLEŞTİRİLMESİ

Ali Alperen ÇOLAK tarafından hazırlanan tez çalışması 08.06.2011 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Tez Danışmanı

Yrd. Doç.Dr. Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi

Jüri Üyeleri

Yrd. Doç.Dr. Yunus Emre SELÇUK

Yıldız Teknik Üniversitesi _____

Prof. Dr. Oya KALIPSIZ

Yıldız Teknik Üniversitesi _____

Prof. Dr. Nadia ERDOĞAN

İstanbul Teknik Üniversitesi _____

ÖNSÖZ

Anlamsal bilgisayar ağları konusunda evrensel bilgiye küçük bir katkı sağlayabilmek amacıyla sıfırdan başlayarak çıktığımız bu yolda iki senelik bir uğraşın ve düzinelerce makalenin ince ince işlenmesiyle elde ettiğimiz çalışmamızı paylaşmaktan mutluluk duymaktayım. Bu çalışma süresince hep yanımda olan aileme, tüm konularda sadece bana destek olmakla kalmayıp, ailem gibi davranan değerli hocam Sayın Yunus Emre Selçuk'a teşekkür etmeyi bir borç bilirim. Ayrıca sistemin inşası, değerlendirilmesi ve gerekli platformlarda paylaşılması noktalarında yardımlarını esirgemeyen Sayın Christian Meilicke ve Cassia Trojahn Dos Santos'a teşekkür ederim.

Nisan, 2011

Ali Alperen ÇOLAK

İÇİNDEKİLER

Sayfa

SİMGE LİSTESİ

f	F-measure
p	Precision
r	Recall
o	Ontoloji
c	Konsept
r	İlişki
i	Örnek
k	Benzerlik Ölçütü
d	Veri Kaynağı

KISALTMA LİSTESİ

W3C	World Wide Web Consortium
OAEI	Ontology Alignment Evaluation Initiative
SEALS	Semantic Evaluation at Large Scale
RDF	Resource Description Framework
OWL	Web Ontology Language
OWL-DL	Web Ontology Language Description Logic
BSM	Base Similarity Matcher
PSM	Parametric String-based Matcher
Tf–Idf	Term Frequency–Inverse Document Frequency
VMM	Vector-based Multi-word Matcher
DSI	Descendant's Similarity Inheritance
UMLS	Unified Medical Language System
LMO	Linguistic Matching for Ontologies
VSM	Vector Space Model
GMO	Graph Matching for Ontologies
GOM	Generic Ontology Mapping
LOM	Large Scale Ontology Mapping
SOM	Semantic Ontology Mapping
SISF	Semantic Inductive Similarity Flooding
GUI	Graphical User Interface
LOM	Lucene Ontology Matcher
JWSL	Java WordNet Similarity Library
J&C	Jiang and Conrath Metric
DLD	Damerau-Levenstein Distance
LD	Levenstein Distance
TUM	Temel Uzaklık Matrisi
LAP	Linear Assigment Problem
LAPJV	Linear Assigment Problem Jonker Volgenant
LAPM	Linear Assigment Problem Mod

ŞEKİL LİSTESİ

Sayfa

ÇİZELGE LİSTESİ

Sayfa

ÖZET

ONTOLOJİLERİN OTOMATİK OLARAK EŞLEŞTİRİLMESİ

Ali Alperen ÇOLAK

Bilgisayar Mühendisliği Anabilim Dalı
Yüksek Lisans Tezi

Tez Danışmanı: Yrd. Doç.Dr. Yunus Emre SELÇUK

İnternet teknolojilerinin hızla geliştiği günümüzde webin üçüncü sürümünün temellerini oluşturan ontolojilerin eşleştirilmesinin önemi gittikçe artmaktadır. Ontoloji eşleştirme için geliştirilmiş olan OntoMatch yapısal olmayan eşleştirme problemlerinde oldukça başarılı sonuçlar elde etmektedir. Birinci bölümde tez ile ilgili genel bilgi verilmiş, ikinci bölümde Web 3.0 sürümü anlatılmış, üçüncü bölümde benzer çalışmalar ele alınmıştır. Geliştirilen uygulama dördüncü bölümde anlatılırken, uygulamanın sonuçları ve benzer çalışmalar ile karşılaştırmaları beşinci bölümü oluşturmaktadır.

Anahtar Kelimeler: Anlamsal ağlar, ontoloji, ontoloji eşleştirme

ABSTRACT

ONTOLOGY MAPPING

Ali Alperen ÇOLAK

Department of Computer Engineering
MSc. Thesis

Advisor: Assist. Prof. Dr. Yunus Emre SELÇUK

The importance of ontology mapping is increasing in parallel with Web 3.0, a familiy of technologies related to the Semantic Web aspect of the Internet. As a part of this thesis, an application named OntoMatch has been developed for ontology mapping problems and quite successful results have been obtained for non-structural mapping problems. The first three chapters gives overall information about the thesis domain: The first chapter gives a general introduction about the thesis, the second chapter explains the Web 3.0 version and the third section discusses similar studies. The developed application is described in the fourth section and finally, in the fifth chapter, the application results and comparisons with similar work are given.

Key words: Semantic Web, ontology mapping, ontology

BÖLÜM 1

GİRİŞ

1.1 Literatür Özeti

Güncel web sürümü Web 3.0'ın vaad ettiği işlevselliklerden olan kişiselleştirilmiş arama ve evrensel bilgi eşgüdümü gibi yeniliklerin sağlanabilmesi, anlamsal ağların temelini oluşturan ontolojilerin belli bir standardı olmasını gerektirmektedir. Ancak bu standardın sağlanamaması ve aynı konuda üretilmiş birçok ontoloji bulunması ontolojiler arasında eşleştirme yapılmasını büyük bir gereklilik haline getirmiştir.

1.2 Tezin Amacı

Ontoloji eşleştirme çalışmalarında istenilen iki farklı ontolojinin birebir eşlemesinin dayanılabilir bir zaman sınırlaması içerisinde bütün konseptler için yüzde yüz doğruluk oranı ile elde edilebilmesidir.

1.3 Orijinal Katkı

Ontolojilerin hiyerarşik konseptlerden oluşması ve bu konseptlerin veri ve nesne özellikleri içermesi, bire bir eşleme için farklı yollar kullanılabilmesini mümkün kılmıştır. Konseptler isimleri ve açıklamaları ile birbirine benzeyebilir ve bu benzerlik bir benzerlik matrisine dökülebilir. Kullanılan matris bir doğrusal atama problemi olarak düşünülüp, eşleştirme bu problemin çözümü olarak atanabilir.

BÖLÜM 2

WEB 3.0

İnsanlık web ile ilk tanıştığında, o henüz kullanıcıların üzerinde değişiklik yapamadıkları (sadece okuyabildikleri), kelime tabanlı arama yapabildikleri, her şeyin her şeye linkler ile bağlı olduğu bir bilgi portalı durumundaydı.

Zaman ilerledikçe insanın sosyal bir varlık olması sebebiyle; web de sosyalleşmeye başladı ve insanların sosyal ihtiyaçlarını karşılamaya çalıştığı bir mecra haline geldi. Kullanıcıların kendilerinin oluşturdukları web içeriklerini paylaşabilmeleri, her hangi bir konu üzerinde düşüncelerini anlatabilmeleri, web içeriklerini etiketleyebilmeleri ve bunun da ötesinde kendi şahsi bilgilerini profillerinde açıklayabilmeleri ile web şu anda koordine edilmeyi bekleyen bir şahsi bilgiler silosu görüntüsü arz etmektedir.

Bu dünya genelindeki birikimi derleyip toparlamak ve insanlığa yararlı bilgi çıkarımları sunmak amacıyla geliştirilen Web 3.0, şu anda en büyük teknolojik trendlerden biridir. Ekonomik olarak doğru kişiye doğru zamanda doğru yerde ve doğru iletişim aracıyla doğru ürünün ulaştırılmasından, yapılan web aramalarının kişiye özel anlamlandırılmış şekilde filtrelenmesine kadar içinde birçok yenilik barındıran bu web versiyonu şu an Yahoo, Google gibi devlerin kullandığı bir alt yapıdır.

İnternetin mucidi ve W3C yöneticisi Sir Tim Berners Lee tarafından 1999'da; "I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will

finally materialize.“ [1] sözleri ile açıklanan Semantik web, Web 3.0’ın en büyük parçasıdır [2]. Amit Agarwal’a göre ise Web 3.0, Semantik web ve kişiselleştirmeden oluşmaktadır [3].

2.1 Neden Web 3.0’a İhtiyaç Duyulmaktadır?

Verinin anlamının bilgisayarlar tarafından anlaşılabilmesi tezine dayanan Web 3.0’ın önemini bir örnek ile açıklayacak olursak; mesela bir aksiyon filmine gitmek ve ardından da balık yemek istiyoruz; bu durumda Google’ı açıp sinemaları arayıp, sinemalarda oynayan filmleri bulup, bu sinemalara yakın restoranları arayıp, restoranların menülerine bakıp, onlar hakkındaki yorumları okumamız gerekir ki bu da oldukça zaman alan bir işlemdir. Web 3.0 da ise “aksiyon filmi oynatan ve yakınında balık lokantası bulunan sinemalar” şeklinde bir arama yapıldığında arama makinesi bize yakınında bizim bütçemize uygun bir balık lokantası bulunan ve aksiyon filmi oynayan sinemayı getirir.

Rafine edilmiş ve kişiselleştirilmiş aramaların yanında, Web 3.0; PCTV alanında kullanıcının reklamını izlediği ürünün üstüne tıklayarak alabilmesi, buzdolabının evdeki süt bitince marketten sipariş verebilmesi vs. gibi hayatı kolaylaştırıcı uygulamaları da içinde barındırmaktadır.

Şu anda web üzerinde Terabaytlarca anlamlandırılmayı bekleyen veri vardır ve bu verilerin hepsinin tek bir veri tabanında anlamlı ve yapısal bir şekilde tutulup, bilgisayarların bu verilerden çıkarım yapabilmesi sağlandığında geriye sadece hayal etmek kalmaktadır [4].

Manoj Sharma’ya göre Web 3.0, “tamamen bütünleşmiş dünya” (totally integrated word) anlamına gelmektedir [5].

2.2 Küresel Eğilimler ve Web 3.0

Getirdiği yenilikler ve sunduğu kolaylıklar ile Web 3.0, ilk geliştirilmeye başladığı andan itibaren uygulama safhasında yerini almıştır. Şu anda bir çok site alt yapı olarak Web 3.0 kullanmaktadır.

Web 3.0 için küresel eğilimleri arama motorları, sosyal paylaşım siteleri ve multimedya mecraları kategorilerinde inceleyecek olursak;

Öncelikle Dbpedia[1], Wikipedia[2] 'daki bütün bilgileri ontolojiye taşımayı amaçlamaktadır.

2008'de "az ara çok anla"(search less understand more) sloganıyla ortaya çıkan bir arama motoru olan Evri[3] ; şu anda anlık haber arama motoru olarak hizmet vermekte ve yer, insan ve araçlar arası ilişkiler belirlenerek oluşturulmuş alt yapı sayesinde aramalara anlamlı bilgiler döndürülmektedir.

Google 2009'da hedefinin dünya üzerindeki tüm verileri indekslemek olduğunu açıklamıştır. Bu amacına ulaşmak için Metaweb'i kuran Google, Freebase isimli anlamlı bilgi veri tabanını internet kullanıcılarının yardımıyla oluşturmakta ve bu alt yapıyı Google News ve Google Squared bölümlerinde kullanmaktadır. Google gibi Swicki[4]'de, katılımcıların etiketlemeleri ile çalışan bir arama portalıdır.

Powerset doğal dil kullanan bir arama motoru iken 2008'de Microsoft tarafından satın alınıp Bing.com arama motorunun içeriğine eklenmiştir. Hakia[5] ise ontolojik, semantik, doğal dil kullanımlı bir arama motorudur. Bir şahıs arama motoru olan Wink[6] ise, kişilerin isim, telefon, adres veya fotoğraf bilgilerini bulmak için tasarlanmıştır.

[1] www.dbpedia.org (son ziyaret:09.02.2011)
[2] www.wikipedia.org (son ziyaret:09.02.2011)
[3] www.evri.com (son ziyaret:09.02.2011)
[4] www.eurekster.com (son ziyaret:09.02.2011)
[5] www.hakia.com (son ziyaret:09.02.2011)
[6] www.wink.com (son ziyaret:09.02.2011)

Sosyal paylaşım ortamları olarak Foaf[1] ve Google'ın Opensocial[2] platformları ile zamanında Web 2.0'ın önde gelen temsilcileri olan Facebook ve Twitter da günümüzde Web 3.0'ı kullanan mecralardandır.

Multimedya mecrası olarak; Getglue[3] sitesinde kullanıcılar sinema, müzik, tiyatro vs. gibi şeylere puan vermekte ve bunları tavsiye etmektedir. Bunun ile birlikte, bir video izleme sitesi Joost[4] ve bir çevrimiçi sinema izleme sitesi olarak da Netflix[5] de kullanıcılarına Web 3.0 teknolojilerini kullanarak hizmet vermektedirler.

2.3 Web 3.0'da Ontolojilerin Önemi

Dünya üzerinde bilgi namına ne varsa bütün her şeyin hiyerarşik bir yapıda temsil edildiği ve bu bilgilerinin bir biriyle bağlantılarının detaylı şekilde ele alındığı Web 3.0, asıl gücünü bu bilgilerin ve bağlantıların üzerinde çalıştırılan sorgulamaların çıkarım yapabilmesi özelliğinden almaktadır. Web 3,0'ın çıkarım yapabilmesi ise onun alt yapısını oluşturan ontolojiler sayesindedir. Ontolojiler Web 3,0'ın temel taşı, alt yapısıdır.

Tanım 2.1 Teorik olarak "paylaşılan bir kavramsallaştırmanın biçimsel ve net bir belirtimi" [40] şeklinde tanımlanan ontoloji, belli bir alandaki kavramların, bunların özelliklerinin ve birbirleri ile olan ilişkilerinin makinenin anlayabileceği şekilde kodlanmasından ibarettir[7].

2.4 Ontolojilerin Birleştirilmesinin Gerekliliği

Web 3.0 ilk planlandığında bütün bilgilerin devasa bir veri tabanında tutulması ve bütün internet mecralarının bu ana kaynağı kullanmaları düşünülmüştü ancak yeni teknolojinin hızla kullanıma geçmesi ve ontolojinin açık kaynak kodlu olması sebebiyle her mecranın kendisi için özel bir ontoloji geliştirmesi, aynı konuda birçok ontolojinin yazılmasına sebep olmuştur. Bu ontolojiler belli bir standarda uygun olarak geliştirilmediği için çığ gibi büyümesi gereken ontolojik alt yapı isteneni verememiştir.

[1] www.foaf-project.org (son ziyaret:09.02.2011)
[2] www.opensocial.org (son ziyaret:09.02.2011)
[3] www.getglue.com (son ziyaret:09.02.2011)
[4] www.joost.com (son ziyaret:09.02.2011)
[5] www.netflix.com (son ziyaret:09.02.2011)

Ontolojik alt yapının dünya genelinde tamamlanması, aynı veya yakın konulu ontolojilerin birleştirilmesi ile olabilecektir. Bunun için W3C kuruluşu her yıl bir yarışma[1] düzenlemektedir.

Yazılım geliştirme süreçleri ve CMMI gibi olgunluk modellerinin ayrı ayrı ontolojileri vardır. Bunlar arasında otomatik bir eşleştirme, bir yazılım evinin yazılım geliştirme süreçlerinin olgunluk modellerine ne oranda uyduğunun anlaşılabilmesini kolaylaştırır. [55]

[1] http://oaei.ontologymatching.org/ (son ziyaret:09.02.2011)

BÖLÜM 3

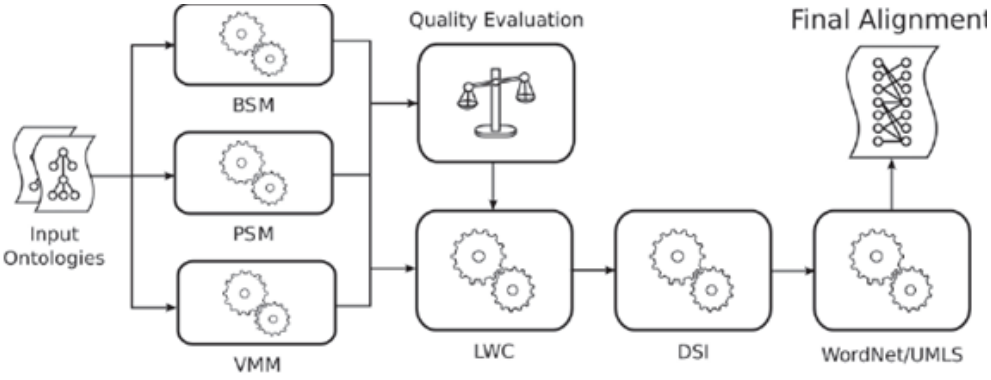
ÖNCEKİ ÇALIŞMALAR

Ontoloji eşleştirme, Web 3.0 da önemli bir alanı kapsaması sebebiyle bir çok araştırmacının ilgisini çekmiş ve bu problemin çözümü için kayda değer uygulamalar geliştirilmiştir. Bu tez çalışması kapsamında gerçeklenen OntoMatch ontoloji eşleştirme çözümü geliştirilmeden önce bu uygulamalar incelenmiş; sistemlerin teknik ve algoritmik detayları yazılı ve görsel bir şekilde aşağıda sunulmuştur. Bölümün sonunda araçlar çizelge şeklinde karşılaştırılmıştır.

3.1 Agreement Maker

Çok amaçlı geliştirilmiş arayüze sahip bir eşleştirme sistemi olan Agreement Maker [8]; otomatik olarak eşleme yapmakta, kullanıcıya eşleme için farklı stratejiler ve farklı parametreler seçme opsiyonu sunmaktadır. 2001 yılında geliştirilmeye başlanmış olan sistem, katmanlı bir mimariye sahiptir.

Bu sistem yedi farklı eşleme algoritmasının dört farklı katmanda kombine edilmesiyle üretilerek OAEI kampanyasına katılmıştır. Öncelikle üç farklı dizgi(string) tabanlı algoritma, girişi yapılan ontolojiler üzerinde çalışmaktadır. Bunlar Temel Benzerlik Eşleştirici (Base Similarity Matcher) (BSM) [9], Parametrik Dizgi Tabanlı Eşleştirici (Parametric String-based Matcher) (PSM) [10], ve Vektör Tabanlı Çoklu Kelime Eşleştirici (Vector-based Multi-word Matcher) (VMM) [10] algoritmaları olarak isimlendirilmektedir.



Şekil 3. 1 Agreement Maker işlem akışı

BSM: Temel bir dizgi karşılaştırma algoritmasıdır. Eşleşmeleri bulmak için kural tabanlı kelime gövdesi bulma algoritması (word stemming), gereksiz kelime silme algoritması (stop word removal) ve kelime normalizasyonu kullanır.

PSM: Değişim uzaklığı (edit distance) ve metin kesme (substring) ölçütlerini aşağıdaki gibi kombine eder.

$$\sigma(a,b) = 0.6 * substring(a,b) + 0.4 * edit\_distance(a,b) \tag{3.1}$$

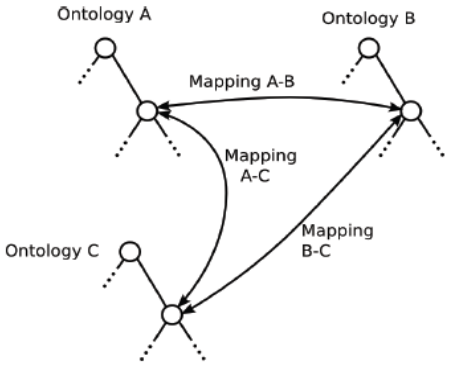
VMM: Her ontoloji konsepti için bir sanal dosya içerir. Bütün dizgileri TF-IDF vektörüne çevirir ve benzerliği kosinüs benzerliği ölçütü (cosine similarity measure) ile hesaplar.

LWC: (Linear Weighted Combination) Yukarıdaki üç eşleme algoritması paralel olarak çalıştıktan sonra sonuçlar bu katmanda anlamlandırılır. Her benzerlik algoritmasının ağırlığı lokal güvenlik kalitesi ölçütü (local confidence quality measure) ile otomatik olarak aşağıdaki gibi hesaplanır.

$$\sigma_{LWC}(a,b) = w_{BSM} * \sigma_{BSM}(a,b) + w_{PSM} * \sigma_{PSM}(a,b) + w_{VMM} * \sigma_{VMM}(a,b) \tag{3.2}$$

DSI: (Descendant's Similarity Inheritance) Konseptlerin atalarına göre eşlenmesi için kullanılan yapısal bir eşleme algoritmasıdır. ‘Eğer iki konsept yüksek benzerlik ile eşleşmiş ise bunların alt dalları da benzer olabilir’ mantığı ile çalışmaktadır.

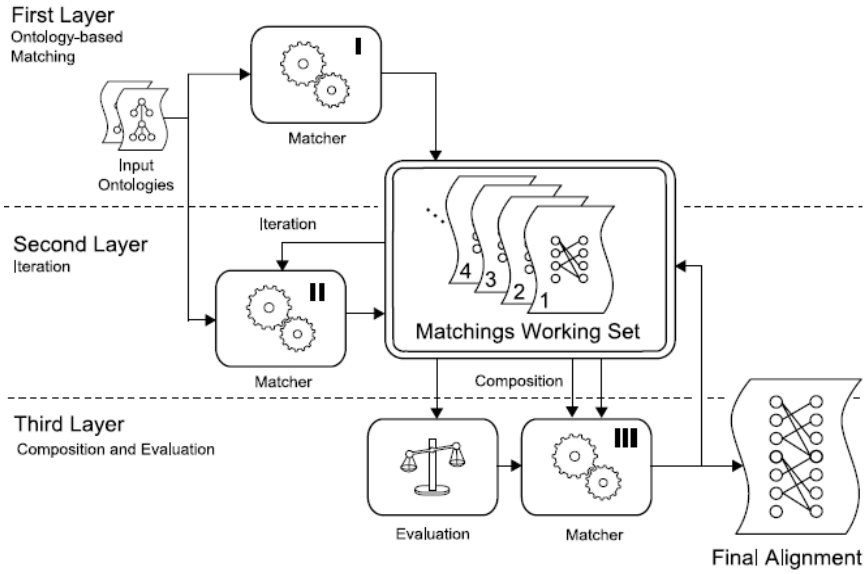
Son olarak da ontoloji konseptlerinin WordNet[17] ve UMLS[16] gibi kelime depolarındaki eş anlamlılarına göre eşleme yapılmaktadır. Yarışmaya özel olarak iki ontoloji arasındaki geçişkenliği bulan bir algoritma katmanı eklenmiştir.



Şekil 3. 2 Agreement Maker işlem akışı

A ve C benzer, B ve C benzer iken A ve B birbirine benziyor ise bu benzerliklerin reyting değeri bir arttırılmakta; benzemiyorsa bir azaltılmakta en son olarak da reyting değeri en yüksek benzerlikten en küçüğe sıralama yapılmakta ve bu bilgiler ile karşılaşılabilecek karmaşıklıklar çözülmeye çalışılmaktadır.

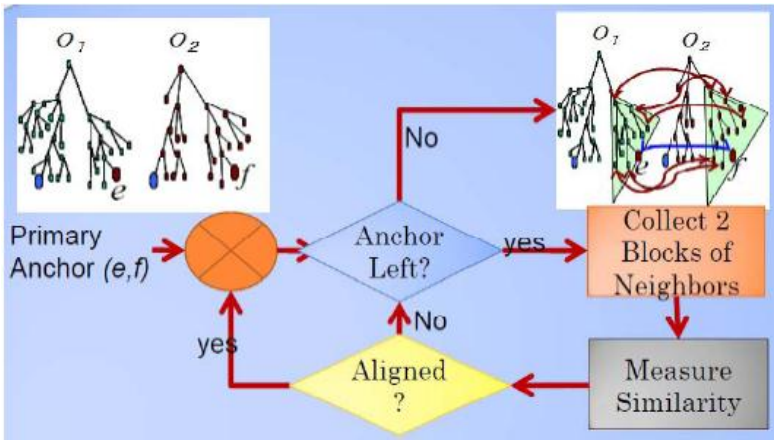
2009 kampanyasında genel sıralamada birinci, anatomi sıralamasında ise ikinci olan Agreement Maker 2010 OAEI kampanyası için biraz değiştirilmiş ve sistem mimarisi katmanlı bir yapıya kavuşmuş toplam başarımı artmıştır. Bu başarım artımı kelime ve vektör tabanlı eşleme algoritmalarındaki değişiklikten kaynaklanmaktadır. 2009da kullanılan BSM, PSM, LWC, VMM ve DSI algoritmalarına 2010da ASM (Gelişmiş dizgi karşılaştırma algoritması), GFM (Grup bulucu eşleme algoritması) ve IISM (İlerlemeli örnek yapısal eşleşleştirme algoritması) eklenmiştir.



Şekil 3.3 Agreement Maker yeni sistem mimarisi

3.2 Anchor-Flood

Böl-yönet tekniği ile ontoloji eşleme yapan Anchor-Flood [11] n*m adet karşılaştırma yapılmasının önüne geçmiştir ve diğerlerinden daha hızlı çalışmaktadır.



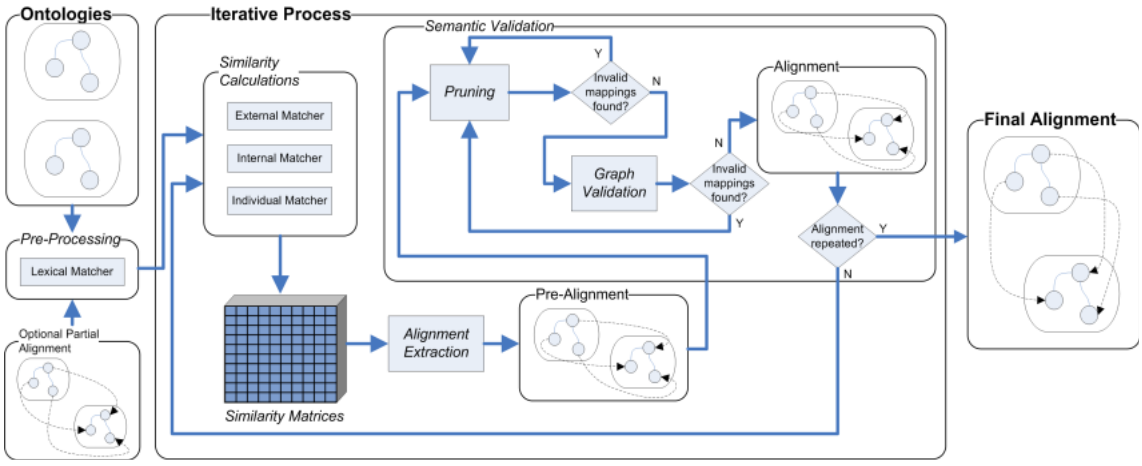
Şekil 3.4 Anchor Flood iş akışı

3.3 Aroma

Üç ana safhaya ayrılan Aroma'da [12]: (1) Önişlem safhası, her bir başlığı bir grup ifadeyle temsil eder; sınıflar ve özellikler gibi (2) ikinci safha, etiketler arasındaki kuralların ortaya çıkmasından oluşur; (3) işlem sonrası safha; sonuç eşleştirme doğruluğunu arttırma ve gereksiz eşleşmeleri elemeyi amaçlamaktadır.

3.4 Asmov

Nesne tipi özellikleri (Object Type Propety), veri tipi özellikleri (Data Type Property) ve örneklerin (Individual) de eşlemelerinin bulunduğu tam otomatik bir sistem olan Asmov [13][14], iki ontoloji arasındaki benzerliği söz dizilimsel özellik (id, label ve açıklama), dışyapı (ataları ve çocukları), içyapı [özelliklerin kaynak (domain) ve hedefleri (range), örnekleri] ve örnek benzerliği olmak dört özellik üzerinden hesaplamaktadır.



Şekil 3.5 Asmov iş akışı

Sözdizimsel çözümleyici (parser) olarak Jena ARP Parser [15] kullanmakta olan sistemde öncelikle UMLS Metathesaurus [16] veya WordNet [17] kelime hazineleri kullanılarak sözdizimsel eşleştirme yapılır. Eğer kelimeler bu hazinelerde bulunmazsa eşleme Levenshtein değişim uzaklığı ölçütü ile hesaplanarak yapılır. Bulunan benzerlik değeri 2 boyutlu matriste kaydedilir. Sonraki adımda ise hiyerarşik karşılaştırma yapılarak yanlış eşleşmeler elenmektedir.

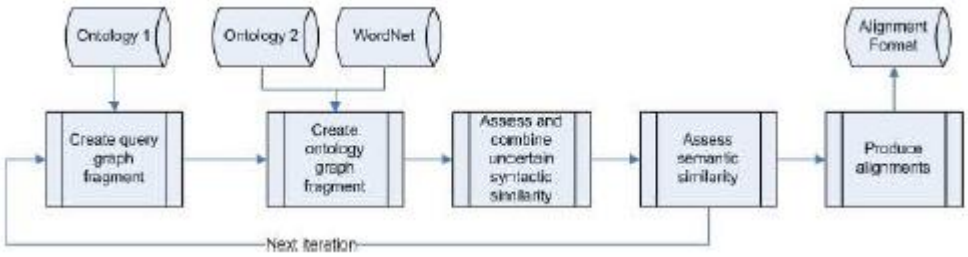
2010 kampanyasına sistemin örnek eşleme algoritmasını güçlendirerek, yapısal olarak benzer konseptleri çıkarım yöntemi ile bulan bir algoritmayı sisteme entegre ederek, büyük ontolojilerin eşleştirilmesi için disk yönetimini geliştirerek ve fonksiyonel özelliklerin etki alanlarını belirleyerek giren Asmov geçen yıla nazaran %2 oranında kendini geliştirmiştir.

3.5 DsSim

İlk Konsept ve özellik bazında karşılaştırma yapan DsSim[18], sonuç olarak seyrek benzerlik matrisi (sparse similarty matrix) oluşturmaktadır. Sistem eşleştirme yaparken örnekler ile ilgilenmemekte, kesin olmayan modelleme ve çıkarımlar için ise Dempster-Shafer'in delil teorisi [19] kullanmaktadır. DSSim, kullanıcı etkileşimli döngülü (iterative) bir sistemdir.

İş akışı şöyledir: Öncelikle ele alınan konseptin özellikleri çıkarılarak bir çizelge (graf) oluşturur. Sonraki adımda sözdizimsel olarak benzer konsept ve bunların eş anlamlılarından (ontoloji 2 den alınan) çizelge oluşturulur. Üçüncü adım olarak farklı

benzerlik algoritmaları ile kantitatif benzerlik değerleri bulunur ve bu değerler Dempster'in kombinasyon kuralı algoritması ile harmanlanır. Kombinasyonlardan çıkan sonuçlara göre çizelge ile ontoloji çizelgesi parçaları arasındaki en fazla güvenilirliği sağlayan eşlemeler kaydedilir.



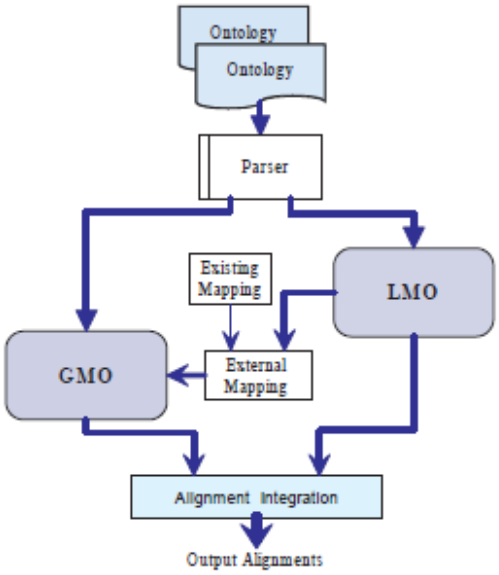
Şekil 3.6 DsSim algoritması

3.6 Falcon

Sözdizimsel çözümleyici olarak Jena kullanan otomatik bir sistem olan Falcon[19] , LMO isimli dilbilimsel bir eşleme algoritması ile GMO isimli bir çizelge eşleme algoritmasının birleşiminden oluşmaktadır. Sözcüksel yakınlık bir ölçüt olduğu gibi yapısal yakınlık da bir ölçüt olarak değerlendirilmektedir.

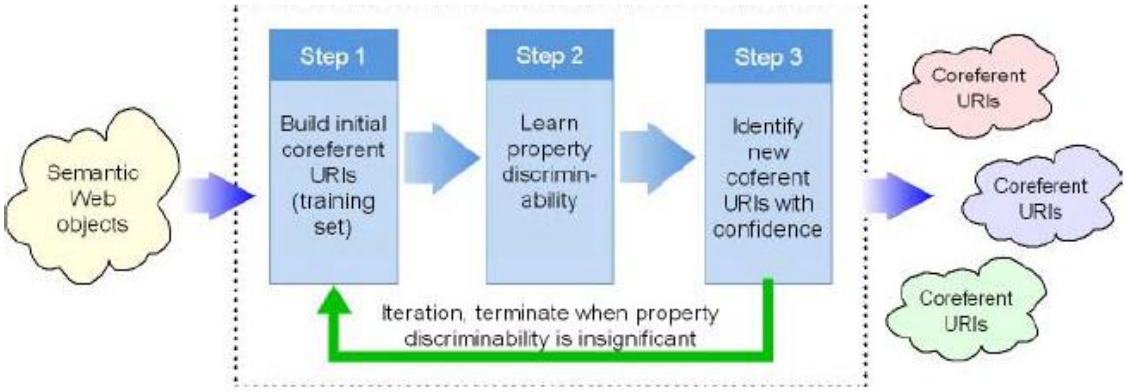
LMO (Linguistic Matching for Ontologies): Bu algoritmada hem sözdizimsel karşılaştırma hem de statik analiz yer almaktadır. Söz dizilimsel karşılaştırma da değişim uzaklığı ölçütü kullanılırken, statik analiz kısmında VSM (vector space model) kullanılmaktadır. VSM algoritmasında, verilen N adet sınıf içeren bir doküman N boyutlu bir vektör olarak düşünülmekte, her sınıf için çalışan sınıf ağırlıklandırma algoritması ile bu vektör doldurulmaktadır.

GMO (Graph Matching for Ontologies) algoritması ise iki sınıfın benzerlik değerini isim, açıklama, özellik benzerliklerinin toplamı olarak hesaplamaktadır.



Şekil 3.7 Falcon algoritması

2010 OAEI kampanyasına Object coref & Falcon ismiyle katılan araç, kendi kendini eğiten bir sisteme dönüşmüştür.



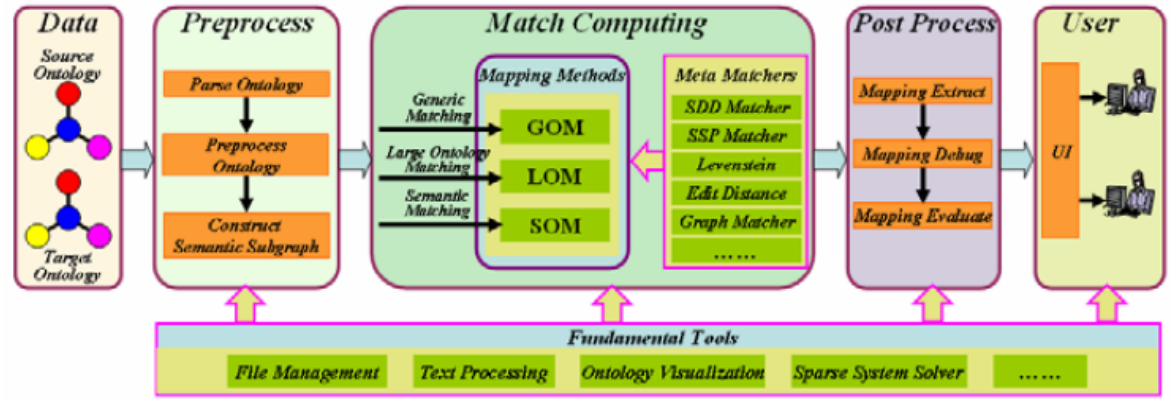
Şekil 3.8 Falcon sisteminin kendi kendini eğitmesi

Object coref burada varlıkların tanımlanarak kullanıldığını belirtmektedir. Varlık anlamsal ağ bulutundan süzülerek tanımlanmakta ve eşleştirme bu önbilgiye dayanılarak yapılmaktadır.

3.7 Lily

Dört ana fonksiyonun birleşmesinden oluşmuş bir sistemdir. (1) Generic Ontology Mapping yöntemi, (GOM), küçük boyutlu ontolojilerde genel eşleştirme yapmak için kullanılır (2) Large Scale Ontology Mapping yöntemi (LOM), büyük boyutlu

ontolojilerde eşleştirme yapmak için kullanılır (3) Semantic Ontology Mapping yöntemi (SOM), ontolojiler arasındaki anlamsal eşleştirme yapmak için kullanılır. Lily, semantik ilişkileri bulmak için arama motorları yardımıyla Web bilgisini kullanmaktadır. (4) Eşleştirmeler üzerinde hata ayıklama, eşleştirme sonuçlarını daha iyi olmasını sağlamaktadır.



Şekil 3.9 Lily algoritması

Lily [21] sisteminin iş akışı ise şöyledir:

Öncelikle anlamsal alt çizelge tekniği olarak adlandırılan böl-yönet tekniği ile ontolojiler anlamlı alt parçalara ayrılmaktadır. İkinci adımda GOM tekniği ile anlamsal alt çizelgeler sözlüksel ve yapısal olarak benzerliklerine göre ayrıştırılır. Üçüncü adımda LOM büyük boyutlu ontolojilerin eşlemesinde yer ve zaman tasarrufu sağlanması için ikililerin benzerliklerinde negatif ve pozitif değer verme için kullanılır. Dördüncü adımda SOM tekniğinde web üzerinde benzer ontolojiler aranıp bunlara göre eşleme yapılmaktadır. Son olarak tekrarlayan veya hatalı eşlemelerin elenmesi sağlanmaktadır. Sistem kullanıcı etkileşimi ile çalışmaktadır.

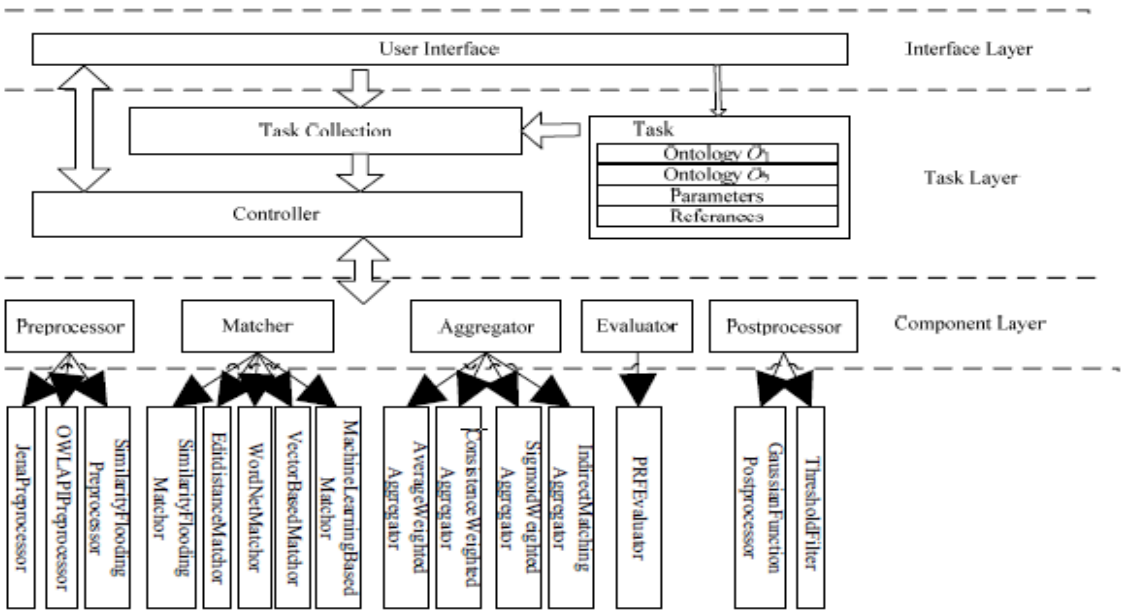
3.8 Rimom

Sistem, ontoloji içeriği ve özelliklerine göre farklı (sekiz adet) strateji belirleyip uygulayabilen bir eşleme sistemi çatısıdır. Rimom [22] altı aşamalı olarak çalışır. Öncelikle ontolojide konseptleri birbirinden en fazla ayırabilecek üç özellik seçilir. Seçilen özelliklere göre değişim uzaklıkları belirlenir ve bir strateji uygulanır. Doğrusal örnek çoğaltma (linear interpoltion) yöntemi ile eşleşen çiftler belirlenir. Sonrasında ise

yüksek yapısal benzerliğe sahip ontolojiler tekrar ele alınır. En son safhada da güvenilmez uyuşmalar silinir.

Kullanılan stratejiler ise: değişim ölçütü tabanlı strateji (benzerlik bulma), WordNet tabanlı strateji (eş anlamlı bulma), vektör benzerliği ölçütü tabanlı strateji, yol benzerliği (path- similarity) tabanlı strateji (kökten yaprağa yoldaki etiketlerin benzerliği: ele alınan konseptlerin kök konsepte (Thing) kadar olan atalarının başlıklarının benzerlik oranının hesaplanması), dinamik yol benzerliği tabanlı strateji, strateji kombinasyonu (1,2, ve 3. stratejilerin birlikte uygulanması), benzerlik yayılımı tabanlı strateji (ontolojiler benzer yapıdaysa eğer iki sınıf eşlenmiş ise bunların üst sınıflarının eşleşmesi diğerlerinin eşleşmesinden daha fazla olasılığa sahiptir) ve dolaylı eşleme stratejisidir.

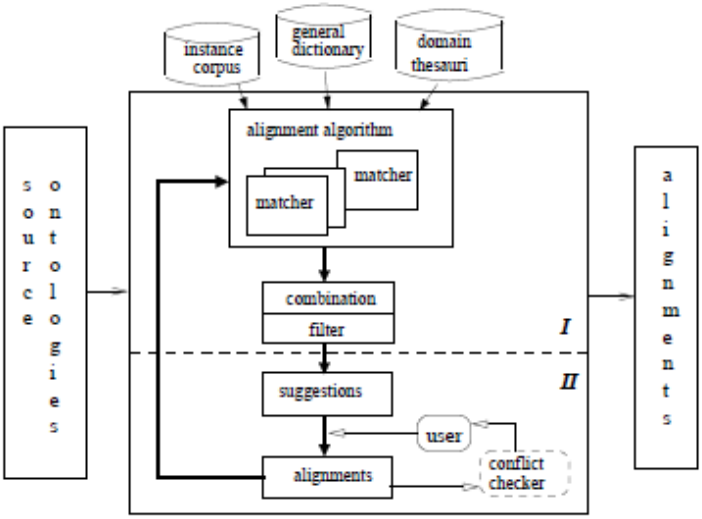
Arayüz katmanında Rimom kullanıcıya parameter belirleme ve/veya bileşen seçme işlemlerini kolaylaştırmak için grafik özellikli bir arayüz sunmaktadır. Görev katmanında eşleştirme görevleri için gerekli parametreler tutulmakta ve bileşen katmanının işleyişinin kontrolü yapılmaktadır. Bileşen katmanında ise beş farklı çalıştırılabilir bileşen grubu bulunmakta, kullanıcı belirli bir eşleştirme için bu bileşenlerden hangilerinin çalıştırılacağını seçebilmektedir.



Şekil 3.10 Rimom mimarisi

3.9 Sambo

Biyomedikal ontolojiler için özel olarak geliştirilmiştir.



Şekil 3.11 Sambo framework

Sambo [23] çatısı (framework) iki parçadan oluşur, birinci parça eşleme adaylarını belirlerken, ikinci parça final eşlemeleri ayarlar. Eşleme sistemi bir veya daha fazla algoritma içerebilir. Sistem beş adet eşleme algoritması içermektedir.

Bunlardan ikisi terminolojik eşleme, biri yapısal eşleme, biri bilgi alanı (domain) eşlemesi tabanlı bir diğeri ise öğrenme tabanlı eşleme algoritmalarıdır.

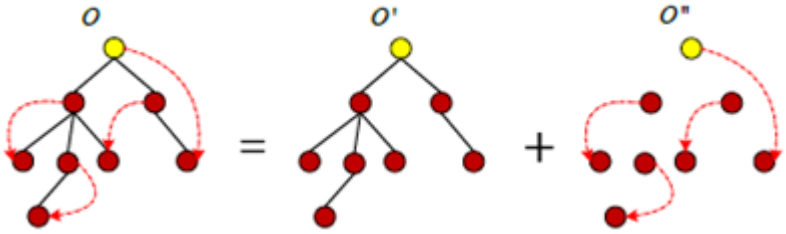
Terminolojik eşleme algoritmaları: sınıfların ve ilişkilerin isimleri ve eşanlamlıları ile çalışır. N-gram, değişim uzaklığı ölçütü ve linguistik algoritmaları uygulanmıştır.

Yapısal eşleme algoritması: ontolojiler üzerinde is-a ve has-a ilişkilerini temel alan bir algoritmadır. Girdi olarak işlenmiş veri alır. İşlenmemiş ontolojiler üzerinde çalışmaz. Girdi olarak benzerlik adayı eşler, özellikleri ve benzerlik değerleri verilmelidir.

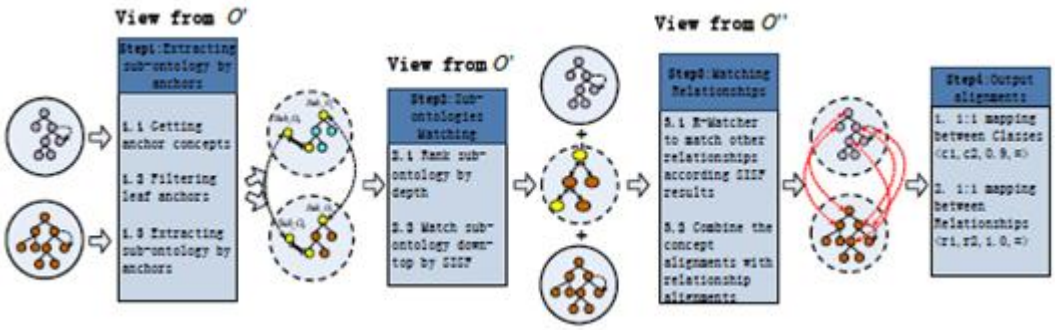
Alan bilgisi algoritması: Unified Medical Language System (UMLS) in içerdiği büyük kelime hazinesini kullanır. İki ontoloji arasındaki benzerlik UMLS'teki ilişkilerine dayanılarak hesaplanır.

3.10 Sobom

Üç farklı eşleştirme aracı içeren, böl ve yönet mantığı ile çalışan otomatik bir sistemdir. Ontolojiler alt ontolojilere ayrılarak işlemler gerçekleştirilir. Dilbilimsel eşleştirici I-Sub[26], yapı eşleştirici SISF (Semantik Inductive Similarity Flooding), -ki bu teknik, Anchor-Prompt [27] ve SF [28] algoritmasından esinlenmiştir- ve ilişki eşleştirici R-Matcher modüllerinden oluşur. Ayrıca, alt ontoloji oluşturucu bir algoritma SoE, I-Sub sonuçlarına göre alt-ontolojileri çıkarmak ve bunları sıralamak için SOBOM'a [24] dâhil edilmiştir. SOBOM yöntemi, tamamen diziseldir bu yüzden farklı eşleştiricilerin çalışma sıralarının değişmesi sonucu etkilemez.



Şekil 3.12 Sobomun oluşturulması



Şekil 3.13 Sobom iş akışı

3.11 TaxoMap

Başlıkları ve alt sınıf açıklamalarını, hiyerarşik yapıyı, eş anlamlıları ve çoklu başlıkları dikkate alan sistemde terminolojik teknikler yoğun olarak kullanılmıştır. Taxomap[29] büyük ontolojileri modüllere ayrılarak incelemektedir.

Sistemde ontoloji (C,Hc) şeklinde ele alınır. Burada C başlıklar ve alt sınıf başlıkları iken, Hc hiyerarşik yapıdır.

Denk konseptler, alt sınıf konsept ve anlamsal olarak ilişkili konseptleri bulmaya yönelik geliştirilmiştir. Sistemde sözdizimsel işlemler için TreeTagger[30] kullanılmıştır.

3.12 Prior+

Değişim uzaklığı ölçütü ve yapısal benzerlik hesaplamalarını kullanana Prior+[31], yapay sinir ağı tabanlı kısıt sağlama çözümlemesi içermektedir.

3.13 Xsom

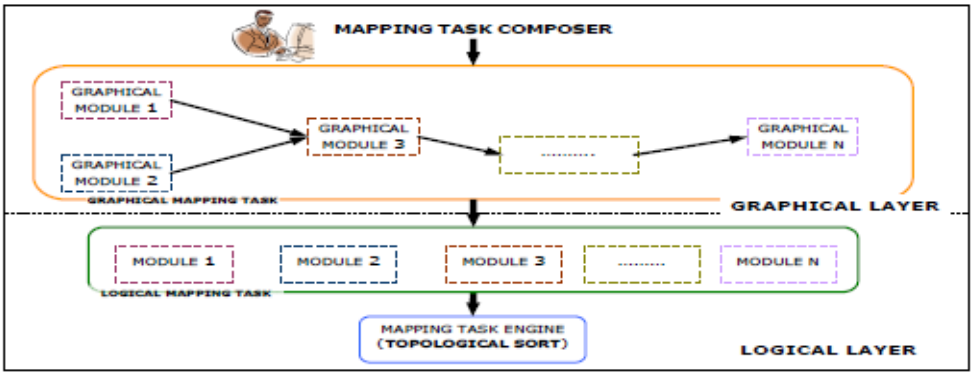
(eXtensible Smart Ontology Mapper) Teorik olarak Owl-DL eşlemesi için oluşturulmuştur ancak Xml eşlemelerinde bile kullanılabilecek esneklikte bir sistemdir. XSom [32] üç parçadan oluşmuştur: eşleme (matching), eşleştirme (mapping) ve tutarsızlık çözünürlüğü (inconsistency resolution). İlk bölümde birçok eşleme algoritması belirlenen stratejiye göre uygulanır. Çıktı olarak konsept-konsept, özellik-özellik veya örnek-örnek benzerliklerinin yer aldığı benzerlik eşleştirmesi oluşturulur. Bu eşleştirmeler eşleştirme alt sisteminde (mapping subsistem) tutulur. Eşleştirme alt sistemi bu eşleştirmelere farklı eşik (threshold) değerleri uygulayarak yapay sinir ağı algoritması için aday eşleşmeleri belirler.

Eşleme alt sistemi üç farklı teknikten oluşur.

Dil tabanlı; sözdizimsel modüldür. Jaro dizge benzerliği ve Levenshtein değişim uzaklığı ölçütünü kullanılır. Yapısal tabanlı; GMO algoritmasının geliştirilmiş bir sürümü kullanılmaktadır. Anlamsal tabanlı; Google tabanlı anlamsal algoritma kullanılmaktadır.

3.14 Ufome

Üç parçadan oluşan bir sistemdir. Bunlar: GUI, eşleştirme ve değerlendirmedir.



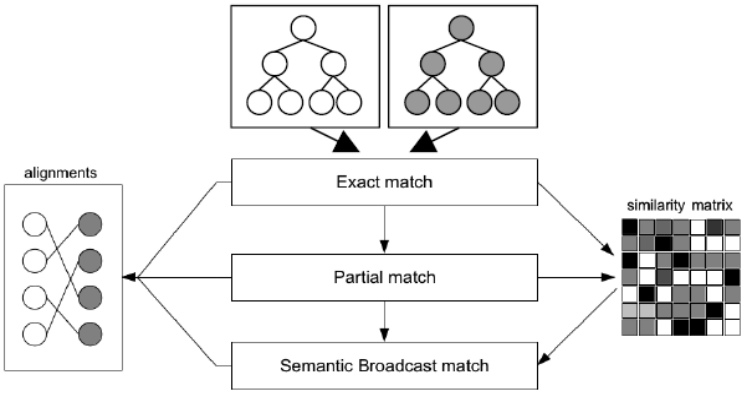
Şekil 3.14 Ufome algoritması

Gösterim (GUI) bölümünde Jena kullanılarak ontolojiler grafik arayüzde görselleştirilmekte, Eşleme bölümünde Lucene, String ve Wordnet olmak üzere üç farklı eşleyici algoritma kullanılmakta, Lucene[34] bölümünde ise Lucene Ontology Matcher (LOM) algoritması [35] implement edilmektedir.

Wordnet bölümü Java WordNet Similarity Library (JWSL) [36] ve Jiang and Conrath Metric (J&C) [37] kullanırken, String bölümü I-Sub [27], Jaro Winkler [38] ve Değişim uzaklığı ölçütü [39] algoritmalarını implement eder. Değerlendirme bölümü ise Değerlendirici, Karşılaştırıcı ve Performans Değerlendirici modüllerini içermektedir. Değerlendirici alt bölümü precision, recall ve f-measure değerlerini ölçmeyi, Karşılaştırıcı alt bölümü farklı stratejilerin precision, recall ve f-measure değerlerini karşılaştırmayı sağlamaktadır. Performans Değerlendirici alt sistemi ise Ufome'nin [33] çalışma zamanını iyileştirmek için geliştirilmiştir.

3.15 Blooms

Biyomedikal ontolojiler için özel olarak geliştirilmiş bir eşleştirme sistemi olan Blooms, kelime benzerliği ve benzerlik yayılımı ölçümlerini kombine ederek sonuca ulaşmaya çalışmaktadır. Olabildiğince otomatikleştirilmeye çalışılmış bir sistem olan Blooms, 2010 OAEI yarışmasına Agreement Maker sistemine integre edilerek katılmıştır.



Şekil 3.15 Blooms mimarisi

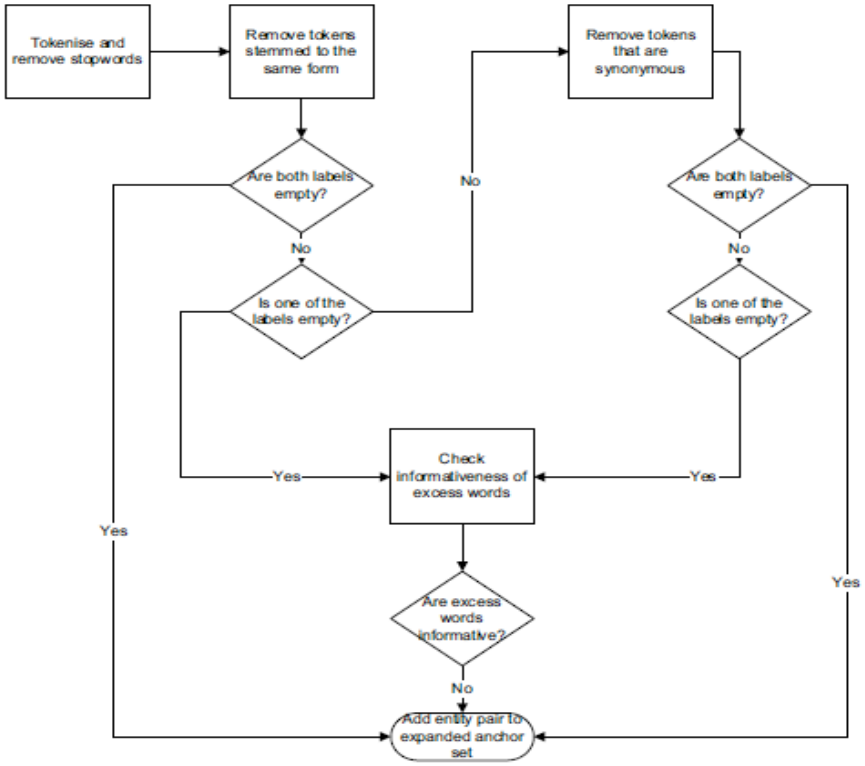
3.16 Codi

Örnekler, konseptler ve özellikler üzerinde çalışan olasılıksal bir sistem olan Codi, hem dışyapı hem de içyapı ile ilgilenmektedir. Araç, Markov mantık algoritmasını kullanmaktadır.

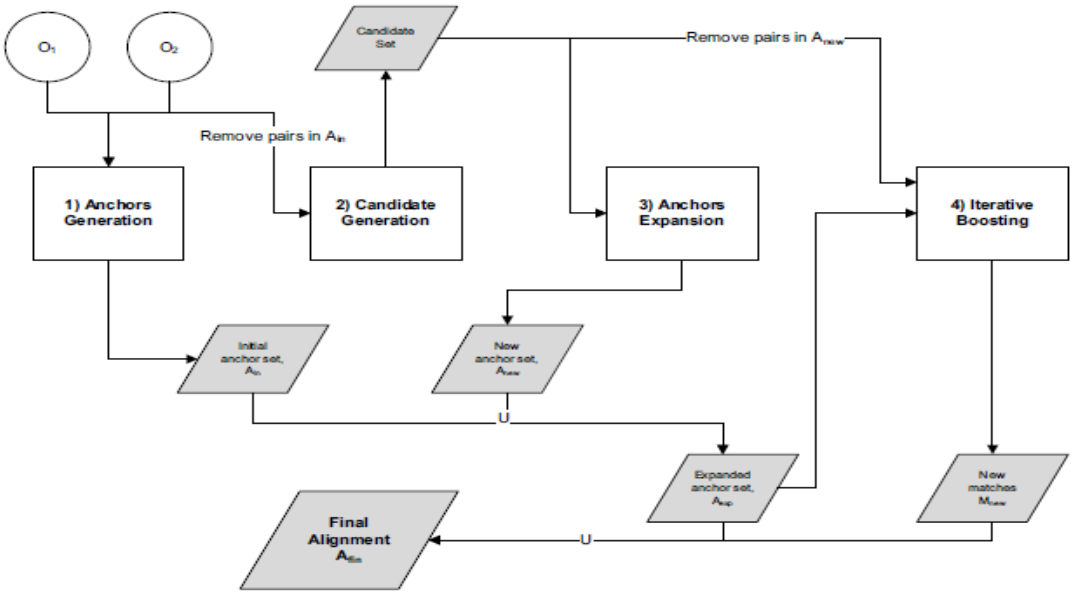
3.17 Eff2Match

Dört parçalı bir ontoloji eşleştirme aracı olan Eff2Match; ilk işlem olarak kelime eşleme yapmakta, ikinci işlem olarak VSM algoritmasını çalıştırmaktadır. Üçüncü işlemde ise term-removing algoritması çalıştırılmaktadır. Son işlem olarak bu eşlemelerden en fazla benzerliği veren eşleştirme sonuçları seçilmektedir.

İlk işlem safhasında konseptlerin sadece isim ve etiketleri kullanılarak birebir eşleme yapılmakta, bulunan sonuçlar bir hash tablosunda tutulmaktadır. Böylece karmaşıklık O(n1*n2)'den O(n1+n2)'ye indirilmektedir. İkinci işlem safhasında, konseptler daha önce eşlenmemiş varsayılmakta ve her konsept için isim, etiket ve açıklama bölümlerinden olmak üzere üçer adet VSM vektörü oluşturulmakta ve bunların benzerlikleri ölçülmektedir. Üçüncü aşamada konseptlerin etiketleri üzerinde terminolojik bir yaklaşımla birlikte bilişsel kelime eşleme algoritması (IWM) kullanılmıştır. Şekil 3.15de bu bölümde kullanılan algoritma verilmiştir.



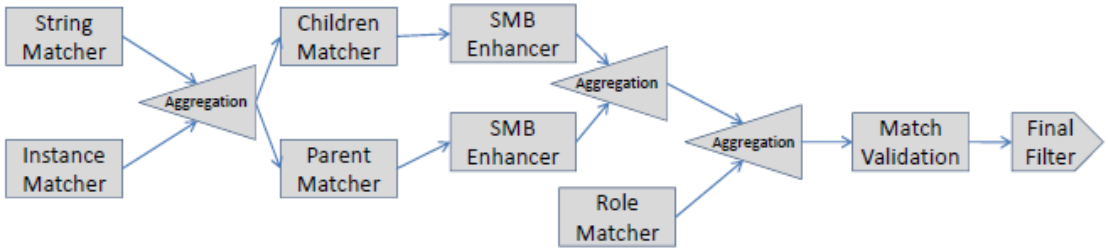
Şekil 3.16 Eff2match üçüncü bölüm algoritması



Şekil 3.17 Eff2match sistem yapısı

3.18 Gerome

Bir framework olan Gerome, İsrail teknik ve Alman Aachen üniversitelerinin ortak olarak geliştirdikleri bir ontoloji eşleştirme sistemidir. Genel rol tabanlı metamodel üzerine yapılandırılan sistem birçok farklı dil için kullanılabilmektedir. Geliştirilen SMB (Schema matching boosting) algoritması ile yapay zekâ bir tavsiye edici şeklinde kullanılmıştır. Gerome, Google ve Swoogle gibi kaynaklardan arka plan birikimi kullanmaktadır.



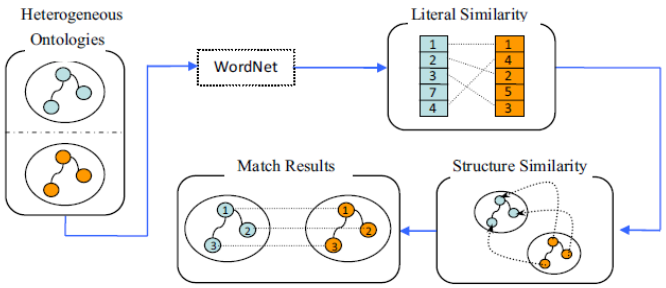
Şekil 3.18 Gerome eşleştirme stratejisi

3.19 MapPso

Ontoloji eşleştirme problemini bir optimizasyon problemi olarak gören ve ayrık parçacık sürü optimizasyonu (DPSO) temelli bir sistem olan MapPSO, kullanıcının isteğine göre konsept ve örnekler üzerinde ayrı ayrı eşleştirme yapabilmektedir.

3.20 Nbjlm

Nbjlm sistemi hem kelime benzerliği hem de anlamsal benzerliğin birlikte kullanıldığı hibrit bir ontoloji eşleştirme aracıdır. Hesaplamalarında Hash eşleştirme algoritması kullanmaktadır.



Şekil 3.19 Nbjlm sistem yapısı

Semantik benzerlik için her iki ontoloji birer ağaç olarak düşünülmüş, bu ağaçlar tek bir kök düğümde birleştirilmiş ve karşılaştırılan konseptlerin ve konseptlerin atalarının kök düğüme olan uzaklıkları hesaplanmıştır.

3.21 Karşılaştırma

Çizelge 3.1 ve 3.2 ile OAEI kampanyasına 2009 ve 2010 yıllarında katılan ontoloji eşleştirme araçlarının bir karşılaştırması verilmiştir; ilk tabloda araçların genel özellikleri, ikincisinde ise aracın algoritmik özellikleri bulunmaktadır.

Çizelge 3.1 de görüldüğü gibi araçlar, uygulama alanına (domain) özel geliştirilmiş bir uygulama olmaları açısından karşılaştırıldığında sadece Sambo ve Blooms araçlarının biyomedikal bir uygulama olduğu, diğer araçların ise genel kullanıma uygun olarak geliştirildiği görülmüştür. Parser olarak TaxoMap diğerlerinden farklı olarak TreeRagger; Codi ve Mappso ise OwlApi kullanırken, geri kalanlar Jena aracını kullanmış veya proje ekibi tarafından özel bir parser geliştirilmiştir. Burada bir istisna olarak Blooms aracı Agreement Maker sistemin entegre edilmiş bir algoritma olarak görülmüştür. Bu aracın giriş ve çıkış işlemleri Agreement Maker tarafından yapılmaktadır. Araçların çoğu kullanıcılar için grafik arayüz kolaylığı sağlamıştır. Çalışma şekli olarak araçlar yine çoğunlukla otomatik olarak çalışmaktadır: Kullanıcı sadece eşleştirilecek ontolojileri seçmekte, geri kalan işlemi ise program yapmaktadır. Etkileşimli çalışan araçlar ise kullanıcıya buldukları birebir eşlemenin doğruluğunu onaylatmakta veya kullanıcının bu eşlemeyi derecelendirmesini istemektedir. Parametre opsiyonu bulunan araçlar, kullanıcının girdi ve çıktı formatını değiştirebilmesi, eşleme için kullanılacak algoritmayı seçebilmesi veya kullanıcının eşleştirme alt değerini belirleyebilmesi desteğini sağlamaktadır.

Çizelge 3.1 Ontoloji eşleştirme araçlarının genel özelliklerinin karşılaştırılması

	Uygulama alanına özel	Parser	GUI	Çalışma Şekli	Parametre Opsiyonu
Agr Maker	Hayır	Özel	Evet	Otomatik	Evet
Anchor-Flood	Hayır	Jena		Otomatik	Evet
Aroma	Hayır	Jena	Hayır		Evet
Asmov	Hayır	Jena	Evet	Otomatik	Evet
DSSim	Hayır		Evet	Etkileşimli	
Falcon	Hayır	Jena		Otomatik	
Lily	Hayır	Özel	Evet	Otomatik	Evet
Rimom	Hayır			Otomatik	Evet
Sambo	Evet (Biyo medikal)		Evet	Otomatik	Evet
Sobom	Hayır	Jena	Hayır	Otomatik	
TaxoMap	Hayır	TreeTagger	Evet	Otomatik	Evet
PRIOR+	Hayır				
XSOM	Hayır	Jena	Evet	Otomatik	Evet
Ufome	Hayır	Jena	Evet	Etkileşimli	Evet
Blooms	Evet (Biyo medikal)	Agr Maker	Evet	Otomatik	Evet
Codi	Hayır	Owl Api	Hayır	Otomatik	Evet
Eff2Match	Hayır	Jena	Hayır	Otomatik	Evet
Gerome	Hayır	Jena	Hayır	Otomatik	Evet
Ln2r	Hayır	Jena	Hayır	Etkileşimli	Hayır
Mappso	Hayır	Owl Api	Hayır	Otomatik	Evet
Nbjlm	Hayır			Otomatik	

Çizelge 3.2 Ontoloji eşleştirme araçlarının algoritmik yapılarının karşılaştırılması

	Böl-Yönet	String Benzerliği	İç yapı	Dış Yapı	Örnek Benzerliği	Word Net & Umls	Eşleme Eleme	Farklı Eşleme Algoritmaları İçerme
AgrMaker		Evet	Evet	Evet	Evet	Evet		Evet
A-Flood	Evet	Evet			Evet			
Aroma	Hayır	Evet	Evet	Evet	Evet	Hayır	Evet	
Asmov	Hayır	Evet	Evet	Evet	Evet	Evet	Evet	Evet
DSSim		Evet	Evet			Evet		
Falcon		Evet	Evet	Evet	Evet			Evet
Lily	Evet	Evet	Evet	Evet	Evet	Hayır	Evet	Evet
Rimom		Evet	Evet	Evet	Evet	Evet	Evet	Evet
Sambo		Evet	Evet	Evet		Evet	Evet	Evet
Sobom	Evet	Evet	Evet	Evet	Evet			Evet
TaxoMap	Evet	Evet	Evet	Evet	Hayır	Hayır	Evet	Evet
PRIOR+		Evet	Evet	Evet				
XSOM	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet
Ufome	Evet	Evet	Evet	Evet	Evet	Evet	Evet	Evet
Blooms	Hayır	Evet	Hayır	Evet	Hayır	Hayır	Hayır	Hayır
Codi	Hayır	Evet	Evet	Evet	Evet	Hayır	Evet	Evet
Eff2match	Evet	Evet	Evet	Evet	Hayır	Hayır	Evet	Hayır
Gerome	Evet	Evet	Hayır	Evet	Evet	Swogle	Evet	Evet
Ln2r	Hayır	Evet	Evet	Evet	Evet	Hayır	Evet	Evet
Mappso	Evet	Evet	Evet	Evet	Evet	Hayır	Hayır	Evet
Nbjlm		Evet	Evet	Evet	Evet	Evet	Hayır	Hayır

Çizelge 3.2 de araçlar algoritmik yapı kıstasına göre karşılaştırılmıştır. Araçların sadece sekiz tanesi eşleştirme için böl-yönet algoritmaları kullanmakta, ontolojileri parça parça eşlemektedir; geri kalanlar ise bütün ontolojiyi hafızaya almayı seçmişlerdir. Benzerlik ölçütü olarak bütün araçların string benzerliğini kullanmakta olduğu dikkat çekmektedir. Dikkat çekici diğer bir nokta ise, az sayıda çalışma dışında içyapı ve dış yapı benzerliğinin yaygın kullanımıdır. İçyapı benzerliğini kullanmayan tek çalışma sadece Anchor-Flood olarak görülmüş, dış yapı benzerliğini kullanmayan çalışmalar olarak Anchor-Flood, Aroma ve DsSim tespit edilmiştir. Anchor-Flood diğer çalışmalardan farklı olarak nesne tipi özelliği ve veri tipi özelliğini hesaplarına katmamakta, üstelik hiyerarşik olarak benzerlikle alt dalların ve yaprakların benzerliğini arttırmamaktadır. Bu özellikleri ile Anchor-Flood 2009 OAEI kampanyasında anatomi testini 15 saniye gibi rekor bir zamanda eşleştirmeyi başarabilmiştir.

Çizelgede, örnekleri benzer konseptlerin birbirine daha benzer olacağı fikrine dayanan örnek benzerliği ölçütünün diğer benzerlik ölçütlerinden daha az kullanıldığı görülebilir. Çizelgedeki araçlardan sadece dördü bulduğu eşleştirmeleri bir kurala göre tekrar elden geçirip rafine etmemekte ve bulunduğu eşlemeleri eleme yapmadan sonuca iletmektedir. Üç araç dışında geri kalan bütün araçlar eşleştirme hesabı için bünyesinde farklı algoritma seçenekleri içermektedir

BÖLÜM 4

SAYISAL UYGULAMA

Felsefenin temel bir dalı olan metafizik kapsamında, varlıkların varoluşlarını, varlıkların sıradüzensel bir yapıda gruplandırılmasını ve birbirlerine olan benzerliklerine ve farklılıklarına göre yapılandırılmasını inceleyen bilim dalı olarak tanımlanan[1] Ontoloji; bilgisayar bilimlerinde paylaşılan bir kavramsallaştırmanın biçimsel ve net bir belirtimi [40] şeklinde ifade edilmektedir. Burada "paylaşılan" kelimesi; ontolojilerin, farklı uygulamalar ve topluluklar arasında yeniden kullanımı amaçladıklarını ve desteklediklerini belirtmekte iken "kavramsallaştırma" kelimesi; özel bir konu alanı ile sınırlı olmak şartıyla ontolojinin, insanların dünyadaki varlıklar üzerine nasıl düşündüklerinin soyut bir modeli olduğunu açıklamaktadır. "Biçimsel" kelimesi; ontolojinin, anlam tanımının biçimsel bir dille temsil edildiğini ve böylece tanım üzerindeki belirsizliklerin, farklı anlam çıkarma olasılıklarının ortadan kaldırıldığı vurgusunu yapmakta, "net bir belirtim" kelimesi ise soyut modeldeki kavramlara ve ilişkilere net isimler verildiği ve net tanımlar yapıldığını ifade etmektedir. [41]

$O=\{C_i,R_i,I_i,A_i\}$[2] şeklinde bir dörtlü ile gösterilebilecek olan ontolojide, C, sınıf veya konseptleri, R ilişkileri, I örnekleri (nesneleri), A ise özellikleri temsil etmektedir.

[1] http://en.wikipedia.org/wiki/Ontology (son ziyaret:09.02.2011)
[2] http://en.wikipedia.org/wiki/Ontology_alignment (son ziyaret:09.02.2011)

```
Namespace(p = <http://example.com/pizzas.owl#>)
    Ontology( <http://example.com/pizzas.owl#>
      Class(p:Pizza
            partialrestriction(p:hasBase someValuesFrom(p:PizzaBase)))
      DisjointClasses(p:Pizza p:PizzaBase)
      Class(p:NonVegetarianPizza complete
            intersectionOf(p:Pizza complementOf(p:VegetarianPizza)))
      ObjectProperty(p:isIngredientOf Transitive
        inverseOf(p:hasIngredient))
    )
```

Şekil 4.1 Bir ontoloji örneği

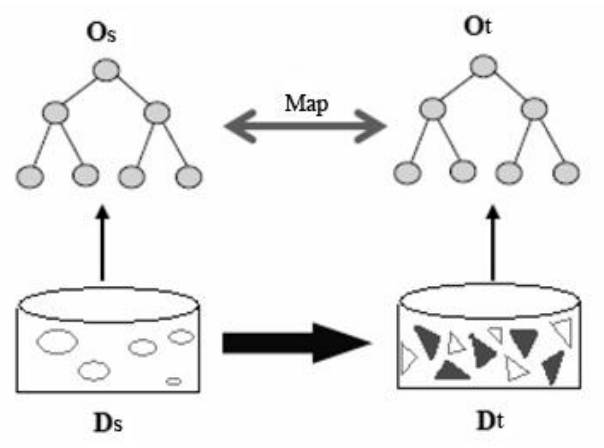
Şekil 4.1'de pizza ontolojisinin bir kısmı gösterilmektedir. Buna göre pizza sınıfı bazı değerlerini pizzaBase sınıfından almakta, vejetaryen ve vejetaryen olmayan pizzalar birbirinin tamamlayıcısı (tersi) durumunda bulunmakta ve pizzanın içerik (ingredient) isminde bir özelliği bulunmaktadır. isIngredientOf ve hasIngredient özellikleri birbirinin tersidir.

4.1 Ontoloji Eşleştirme

Ontoloji eşleştirme, aynı konu üzerine birçok ontolojinin geliştirilmesi, aynı olguyu veya nesneyi farklı yollardan anlatan ontolojilerin bir standarda ihtiyaç duyması ile ortaya çıkmış bir problemdir. Bir ontoloji eşleştirmesi aşağıdaki biçimde tanımlanır;

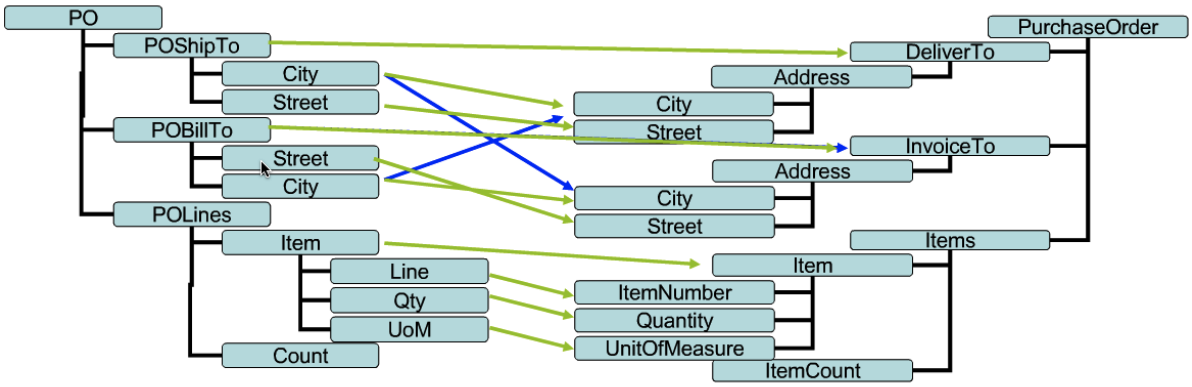
$$O_s \xrightarrow{M} O_t \Rightarrow M := \langle o_s, o_t, r, k \rangle \tag{4.1}$$

Burada O_s ve O_t aynı konu üzerinde geliştirilmiş ontolojileri, M eşleştirme fonksiyonunu, o_s ve o_t sırasıyla O_s ve O_t ontolojilerinin birer konseptini, r ilişki türünü (= vs.), k ise [0,1] benzerlik ölçütünü gösterir.



Şekil 4.2 Ontoloji eşleştirme

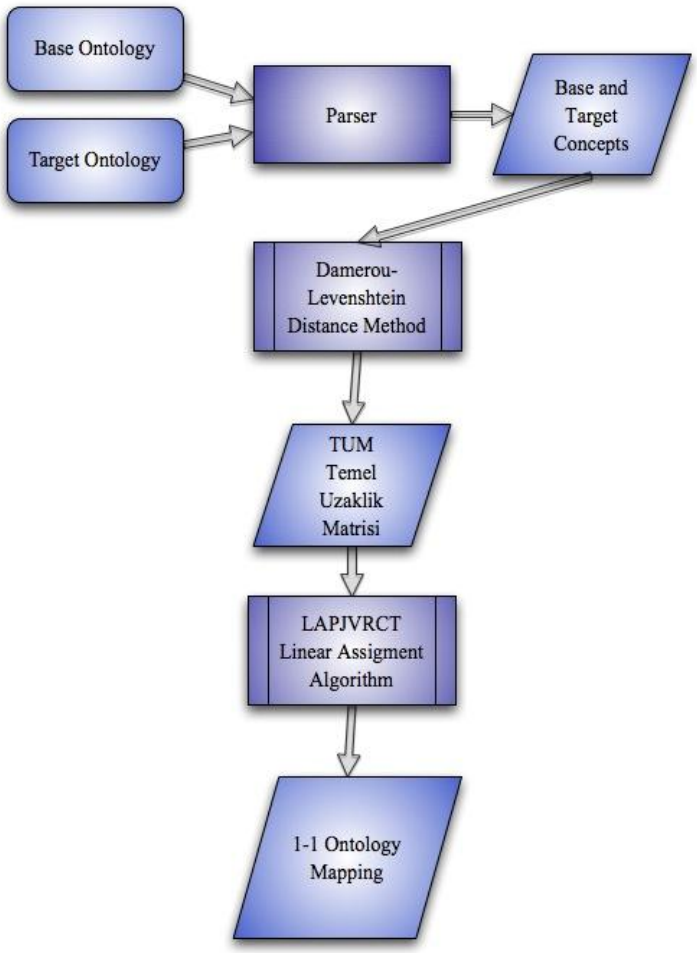
Şekil 4.2 ontoloji eşleştirmenin bir özetini vermektedir. D_s veri kaynağından oluşturulmuş O_s ontolojisi, D_t veri kaynağından oluşturulmuş Ot ontolojisi ile Map fonksiyonu tarafından karşılaştırılmakta ve birbirine en yakın konseptler eşleştirme için seçilmektedir. Şekil 4.3 bir eşleştirme örneğini göstermektedir.



Şekil 4.3 Bir ontoloji eşleştirme örneği

4.2 OntoMatch Ontoloji Eşleştirme Aracı

Ontoloji standardizasyonu için geliştirmiş olduğumuz eşleştirme yöntemi OntoMatch, girdi olarak aynı konulu iki ontoloji almakta ve çıktı olarak bu ontolojilerin eşleştirilmiş halini vermektedir.



Şekil 4.4 OntoMatch sistem mimarisi

OntoMatch, içerdiği algoritmaların dizge temelli olmaları sebebiyle öncelikle Rdf formatı ile verilen ontolojilerdeki konseptleri çözümlemektedir. Ontolojilerden biri kaynak ontoloji, diğeri hedef ontoloji olarak isimlendirilmekte, eşleştirme işlemi kaynaktan hedefe olacak şekilde hesaplanmaktadır. Bulunan sınıflar bir matrise aktarılmakta ve her kaynak sınıfın her hedef sınıfa olan uzaklığı ölçülmektedir. Bu ölçüm için Damerau-Levenstein [42] metriği kullanılmıştır.

Damerau-Levenstein Distance (DLD), iki dizge arasındaki uzaklığı ölçmek amacıyla, Frederick J. Damerau tarafından 1964'te ortaya konulmuş bir algoritmadır. Temel olarak, kaynak dizgenin hedef dizgeye dönüşmesi için yapılması gereken ekleme ve silme işlemleri sayısına iki dizge arasındaki uzaklık denir. DLD buna ek olarak dizge içinde karakter yer değiştirmesini de içermektedir [44]. Böylece diğer dizge uzaklık ölçme algoritmalarından daha iyi sonuçlar vermektedir. Kelime işlemcilerde kullanılan sözcük düzeltme, Google'da kullanılan "şunu mu demek istemiştiniz?" uygulamaları ve

Zemberek[1] bu algoritmayla çalışmaktadır. LD ve DLD aynı zamanda konuşma tanıma ve DNA işlemlerinde de kullanılmaktadır.

Diğer dizge tabanlı eşleştirme araçlarında LD (Levenstein Distance) kullanılırken, OntoMatch DLD kullanmaktadır. İki yöntem arasındaki farkı aşağıdaki örnek göstermektedir:

levenshtein('teusday', 'tuesday') = 2 ; levenshtein('teusday', 'thursday') = 2

damlev('teusday', 'tuesday') = 1 ; damlev('teusday', 'thursday') = 2[43]

Örnekte kullanıcı “tuesday” yazmak isterken yanlışlıkla “teusday” yazmıştır, LD yöntemi ile ölçüm yapıldığında bu dizgenin “tuesday” ve “thursday” dizgelerine uzaklığı aynı -2- olarak ölçüleceğinden, kullanıcının girdiği dizgenin anlamı anlaşılamamaktadır. Aynı dizge DLD yönteminde “tuesday” dizgesine daha yakın olarak hesaplandığından, doğru olarak anlaşılabilmektedir.

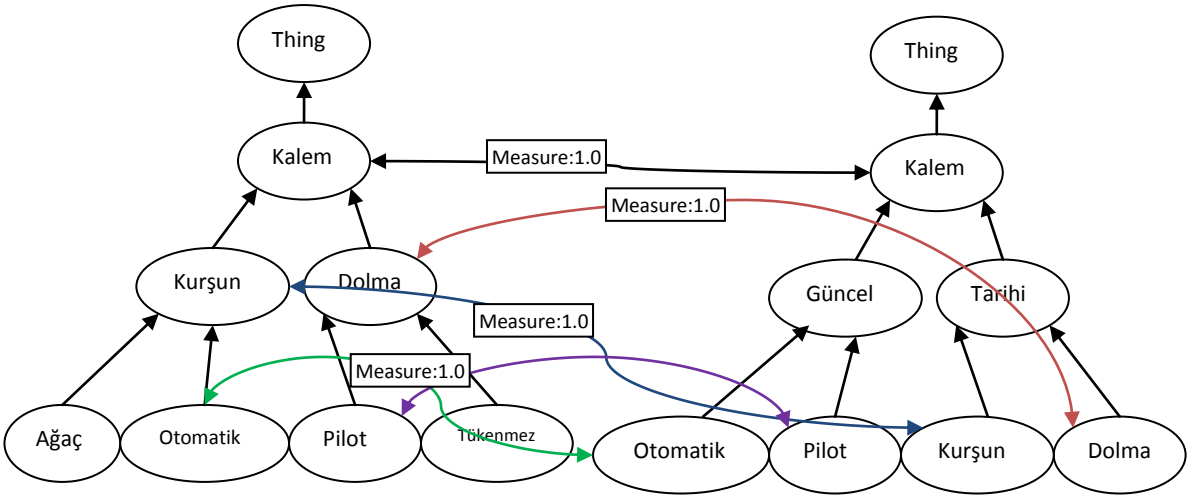
Uzaklık ölçüm işlemi sonucunda Şekil 4.5’te gösterilen temel uzaklık matrisi elde edilmektedir.

	C_{t1}	C_{t2}	C_{t3}	…..
C_{s1}	DLD(C_{s1},C_{t1})	DLD(C_{s1},C_{t2})	DLD(C_{s1},C_{t3})	
C_{s2}	DLD(C_{s2},C_{t1})	DLD(C_{s2},C_{t2})	DLD(C_{s2},C_{t3})	
C_{s3}	DLD(C_{s3},C_{t1})	DLD(C_{s3},C_{t2})	DLD(C_{s3},C_{t3})	
….				

Şekil 4.5 Temel uzaklık matrisi

Temel Uzaklık Matrisi (TUM) satırları kaynak ontolojinin sınıflarından, sütunları ise hedef ontolojinin sınıflarından oluşan bir karşılaştırma matrisidir. Matrisin hücrelerinde ilgili sınıfların dizge uzaklıklarının sonuçları bulunmaktadır.

[1] http://code.google.com/p/zemberek/ (Son ziyaret 10.02.2010)



Şekil 4.6 Örnek ontoloji eşleştirmesi

Şekil 4.6 da gösterilen örnek eşleştirme olsun. Bu eşleştirmeyi gerçekleştirmek için oluşturulacak olan Temel Uzaklık Matrisi (TUM) Şekil4.7 de belirtilmiştir.

	Kalem	Kurşun	Dolma	Ağaç	Otomatik	Pilot	Tükenmez
Kalem	0	5	5	5	5	5	5
Güncel	5	5	5	5	5	5	5
Tarihi	5	5	5	5	5	5	5
Otomatik	5	5	5	5	0	5	5
Pilot	5	5	5	5	5	0	5
Kurşun	5	0	5	5	5	5	5
Dolma	5	5	0	5	5	5	5

Şekil 4.7 Örnek TUM

Elde edilen matris (TUM) ağırlıklı iki parçalı bir çizelge şeklinde düşünüldüğünde, eşleme işleminin yapılması için bir Doğrusal Atama Probleminin (Linear Assigment Problem-LAP) çözülmesi gerekmektedir.

Doğrusal atama problemleri iki gruba ayrılmaktadır. Bunlar ikili eşleme ve azami eşleme olarak isimlendirilmektedir. İkili eşleme probleminin çözümü için iki üyenin eşleşmesi yeterlidir. Örneğin evlilik problemi, nehri geçmek isteyen kişilerin sallara

atanması gibi… Bu tip problemler en az kişinin açıkta kaldığı en fazla kişinin eşleştiği problemlerdir.

İkinci grup olan azami eşeleşme (maximum matching) problemleri ise LAPJV, Macar algoritması tarafından çözülmesi hedeflenen ve klasik eşleşmenin ötesinde, her eşleşmenin değer sahibi olduğu problemlerdir. Örneğin kişilerin iş bulması ama her işin ve kişinin farklı maaşlar alması halinde azami maaş (maximum maaş) değerine ulaşmak gibiBu problemde sadece eşleştirme yapmak yetersiz kalmakta, eşleştirmelerin değerinin maksimuma çıkarılması hedeflenmektedir. Bu problemin tersten okunuşu ise asgari maliyettir.

TUM üzerinde asgari maliyete yönelik çalışacak olan bir doğrusal atama problemi çözümü bize birbirine en yakın eşleştirmeleri verecektir.

LAP, gezgin satıcı problemi ve en kısa yol problemi gibi temel hesaplama optimizasyonu problemlerindendir. Bu problemlerde her sonucun bir ağırlığı vardır ve en düşük maliyetli sonuç aranmaktadır [45]. Genel olarak problem şöyle tanımlanır: Birçok ajan ve birçok iş vardır, herhangi bir ajan herhangi bir işi yapabilir ve her ajanın her işi yapması için bir maliyeti vardır. Her işin yapılması gereklidir fakat her ajanın çalışması gerekli değildir. Bütün işlerin yapılabilmesi için en düşük maliyetli çözüm aranmaktadır. Hungarian metodu [53], Simplex algoritması [54] gibi yöntemler bu problemin çözümü için geliştirilmiştir.

Macar (Hungarian) algoritması Munkers tarafından 1957 de geliştirilmiş $O(n^4)$ karmaşılığa sahip en çok bilinen doğrusal atama problemi çözümleriden biri olmasına rağmen karmaşılığı O(n) olan LAPJV algoritmasından oldukça yavaş kalmaktadır.

Geliştirilen sistemde R. Jonker and A. Volgenant'ın bulduğu LAPJV[46] algoritması kullanılmıştır. Jonker ve Volgenant bu çalışmalarında Dijkstra'nın en kısa yol metodunu seyrek ve yoğun matrislerde kullanmak için geliştirmişlerdir. LAP için 1955'te geliştirilen Hungarian metoduna sonradan geliştirilen sistemler performans açısından yetişememişlerdir. Şekil 4.6 da LAPJV ve diğer LAP algoritmalarının bir karşılaştırması verilmiştir[46]:

Cost range	Problem Size	LAP Algorithm ASSCT	LSAP	ASSIGN	LAPJV
1-100	50	51	32	22	15
	100	149	168	114	64
	150	283	535	256	179
	200	420	1363	520	323
1-1000	50	121	30	24	17
	100	637	165	123	77
	150	1447	456	364	225
	200	2217	850	665	406
1-10000	50	145	31	28	25
	100	1085	168	134	103
	150	3562	453	410	259
	200	6989	919	779	456

Şekil 4.8 Lapjv'nin diğer Lap algoritmaları ile karşılaştırılması(ms)

Şekil 4.6'da ASSCT algoritması, Carpaneto ve Toth[47] tarafından Hungarian metodu ile geliştirilmiş bir sistem, LSAP ise Dorhout'un geliştirdiği Tomizawa'nın algoritmasının bir versiyonudur [48, 49] ve Burkard ile Derigs tarafından Fortran diliyle yazılıp yayınlanmıştır [50]. Son olarak ASSIGN ise Bertsekas tarafından geliştirilmiş bir algoritmadır [51].

Aynı zamanda M. Dell'Amico ve P.Toth 2000 yılında bütün eşleştirme algoritmalarının üzerinde yapmış oldukları deneyde LAPJV-LAPM in en hızlı çalışan eşleme algoritması olduğu sonucuna ulaşmışlardır [52].

Geliştirilen sistemin öne çıkan özelliği ise; ontoloji eşleştirme için ontolojileri birer çizelge olarak ele alması, iki çizelgenin yakınlığını farklı bir metot ile bulup sonucu ağırlıklandırılmış bir çizelgede temsil etmesi ve eşleştirmeyi bu çizelgeden bilinen en iyi doğrusal eşleme yöntemi ile çıkarmasıdır. Doğrusal atama problemi çözümü çoğunlukla kare matrislerde uygulanmıştır ancak burada kaynak ontoloji ve hedef ontolojinin konsept sayısı farklı olacağından, matris dikdörtgendir.

Sistem, 2004'ten bu yana ontoloji eşleştirme araçlarının yarıştığı bir kampanya olan OAEI[1] (Ontology Alignment Evaluation Initiative) tarafından ontoloji eşleştirme sistemlerinin başarımının ölçülmesi için tasarlanan SEALS[2] (Semantic Evaluation at

[1] http://oaei.ontologymatching.org/ (son ziyaret:09.02.2011)
[2] http://www.seals-project.eu/ (son ziyaret:09.02.2011)

Large Scale) projesi kapsamında SEALS 2011'de yer alma amacıyla geliştirilmeye devam etmektedir.

Eşleştirme işlemi sonucunda bir eşleme dosyası çıktı olarak verilmektedir. Şekil 4.9 da örnek bir eşleştirme görülmektedir. kalem.owl ontolojisindeki kurşun sınıfı ile kalem2.owl ontolojisindeki kurşun sınıfı tam olarak eşleşmiştir. Eşleşme ölçütü "1" yani "tam"dır. Bu sınıfların aralarındaki benzerlik şekli ise eşitliktir.

```
<map>
        <Cell>
                <entity1 rdf:resource="http://kalem.owl#kurşun"/>
                <entity2 rdf:resource="http://kalem2.owl#kurşun"/>
                <measure rdf:datatype="xsd:float">1.0</measure>
                <relation>=</relation>
        </Cell>
</map>
```

Şekil 4.9 Örnek eşleştirme

BÖLÜM 5

SONUÇLAR VE ÖNERİLER

Bu bölümde öncelikle tez kapsamında gerçeklenen OntoMatch ontoloji eşleştirme çözümünün mevcut çalışmalar ile bağımsız bir kurum tarafından karşılaştırılması ile elde edilen sonuçlar verilecektir. Ardından da OntoMatch'in özellikle başarılı ve başarısız olduğu örnekler incelenecektir.

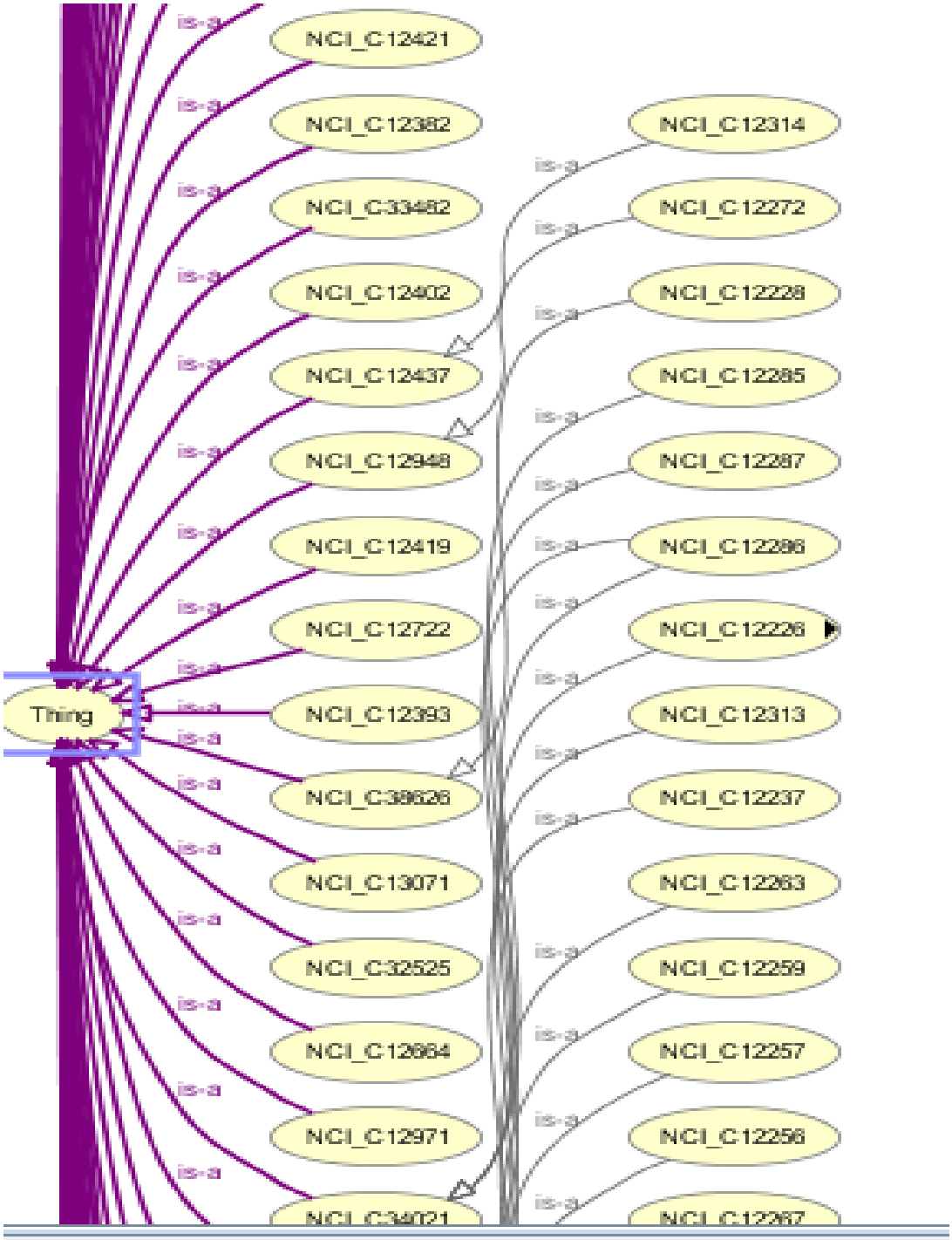
5.1 OAEI Ontology Alignment Evaluation Initiative

2004'ten bu yana ontoloji eşleştirme araçlarının yarıştığı bir kampanya olan OAEI, Fransız Inria Üniversitesi'nin önderliğinde dünya çapında birçok bilim adamının katılımıyla kurulmuş bir girişimdir. Ontoloji eşleştirme sistemlerinin başarımını ölçmeyi, sistemlerin başarım ve performansını karşılaştırmayı, algoritma geliştiricilerin arasındaki iletişimi güçlendirmeyi, eşleştirme sistemlerinin değerlendiren sistemleri geliştirmeyi ve ontoloji eşleştirme konusundaki çalışmalara yardımcı olmayı amaçlayan bu girişim, sistemleri dört farklı şekilde değerlendirmektedir.

Birinci bölüm "benchmark" bölümüdür. Bu bölümde 54 küçük test bulunmakta ve her test, eşleştirme sisteminin farklı bir becerisini ölçmektedir. Benchmark testleri yarışmaya katılacak sistemlerin kendilerini değerlendirebileceği testlerdir; referans eşleştirmeler ontolojiler ile birlikte yarışmadan önce verilmektedir.

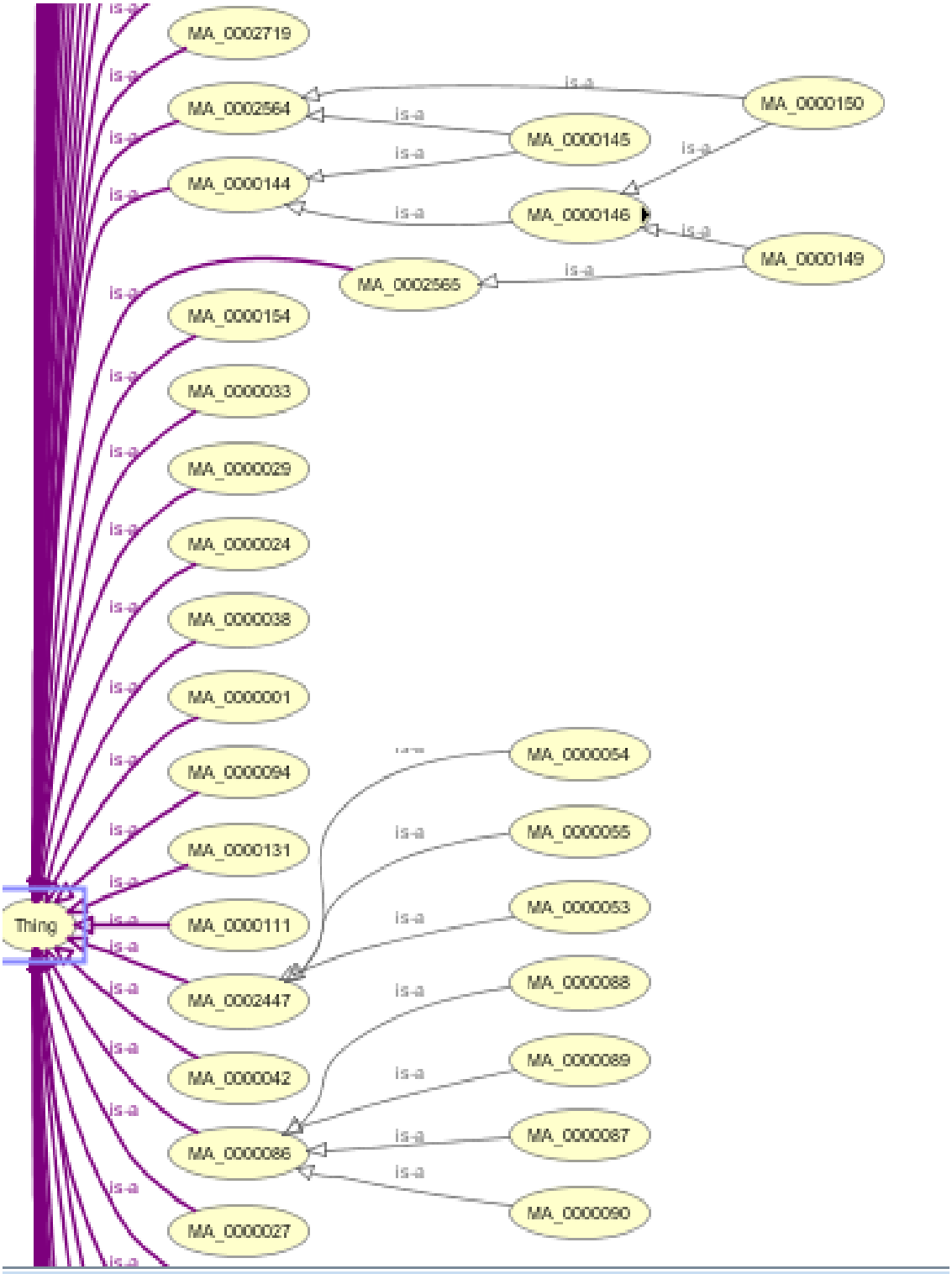
İkinci bölüm "anatomi" bölümüdür. Gerçek hayat veri setine dayalı olan bu bölümde insan ve fare anatomik ontolojileri verilmekte ve bu ontolojilerin eşleştirilmesi istenmektedir. İnsan anatomisi ontolojisi (Şekil 5.1) Ulusal Kanser Enstitüsü (NCI) tarafından geliştirilmiş olup 3304 adet sınıf içermekte, fare anatomisi ontolojisi (Şekil 5.2) ise 2744 adet sınıf içermektedir. OAEI, 2010 daki yarışmadan sonra Fare Geni

Araştırmaları (MGI) tarafından geliştirilmiş insan ve fare anatomisi ontolojisi eşleştirmesini sitesinde paylaşıma sunmuştur. Bu eşleştirme, yarışmada yer alan ontoloji eşleştirme çözümlerinin başarımını doğruluk açısından ölçmede kullanılmaktadır.



Şekil 5.1 İnsan ontolojisinin çok küçük bir bölümü

Şekil 5.1 ve 5.2 de anatomi bölümünde kullanılan insan ve fare anatomi ontolojileri verilmiştir. Bu ontolojilerde sınıflar birer “id” numarası ile temsil edilmekte, sınıf ismi olarak “id” bilgisi tutulmakta, sınıfın temsil ettiği varlığın ismi “label” olarak verilmektedir.



Şekil 5.2 Fare ontolojisinin çok küçük bir bölümü

Üçücü bölüm olan "conference", farklı gruplar tarafından geliştirilmiş 16 adet konferans organizasyonu ontolojisi içermekte, eşleştirme sistemleri bu ontolojiler üzerinde çalışmaktadır.

Dördüncü bölüm "directory" bölümüdür. Bu bölüm de ikinci bölüm gibi bir gerçek dünya senaryosudur. Kullanılan ontolojiler Google, Yahoo ve Looksmart sitelerinin ontolojileridir ancak bu bölümde yarışmacılardan verilmiş ontolojilerin hiyerarşik olarak tekrar yapılandırılması istenmektedir.

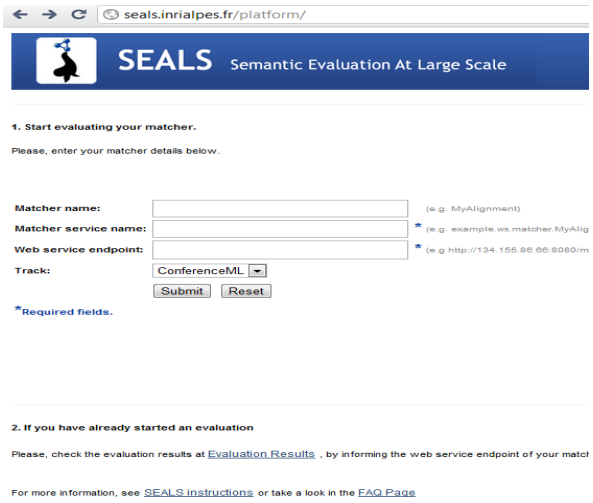
Beşinci bölüm "im@" olarak isimlendirilmekte örnek eşleştirme anlamına gelmektedir. Bu bölümde ise yarışmacılardan verilmiş farklı owl ve rdf formatlı dosyalardan örnekler temel alınarak eşleştirme yapılması istenmektedir.

Altıncı bölüm "vlcr" (very large crosslingual resources) bölümüdür. Bu bölümde yarışmacılar Thesaurus of the Netherlands Institute for Sound and Vision (GTAA) ontolojisi, New York Times konu başlıkları ontolojisi ve DBpedia ontolojisi gibi üç büyük ontoloji arasında eşleştirme yapmaya çalışmaktadır.

Her sene yapılan yarışma, kampanyanın web sitesinden duyurulmakta ve sonuç olarak sistemlerin detaylı anlatımı ile karşılaştırmalı başarım oranları paylaşılmaktadır.

5.2 Seals Platformu

Seals Platformu bir ontoloji eşleştirme çalışmaları için bir buluşma noktası olmak ile birlikte, OAEI yarışmasına katılan sistemlerin çevrimiçi olarak denendiği bir mecradır. Şekil 5.3 de görülen arayüz ile geliştirilen ontoloji eşleştirme web servisi seçilen test (track) için çevrimiçi olarak çalıştırılıp değerlendirilebilmektedir. Seals platformu, Seals komitesinin geliştirdiği bir projedir.



Şekil 5.3 Seals Platformu

5.3 Uygulama

Geliştirilen sistem, 64 bitlik mimariye sahip Intel Q9400 2.66 Ghz işlemci ve 4GB Ram bulunan bir bilgisayarda çalıştırılmış ve eşleştirme sonucu alınmıştır. Ontomatch sistemi öncelikle iki ontolojiyi almış, ontolojilerdeki konsept bilgilerini çıkarmış, TUM (temel uzaklık matrisi) anatomi testi için 2744*3304lük bir matris oluşturmuş ve işlemler bu matris üzerinde yapılmıştır. Bütün işlemler yaklaşık olarak 4 dakika sürmüştür.

5.4 Başarım Ölçütleri

Ontoloji eşleştirmesinin başarımı "precision, recall ve f-measure" değerleri ile ölçülmektedir.

$$precision = p = \frac{correct\_found\_mappings}{all\_found\_mappings} \tag{5.1}$$

$$recal = r = \frac{correct\_found\_mappings}{all\_possible\_mappings} \tag{5.2}$$

$$f-measure = f = \frac{2pr}{p+r} \tag{5.3}$$

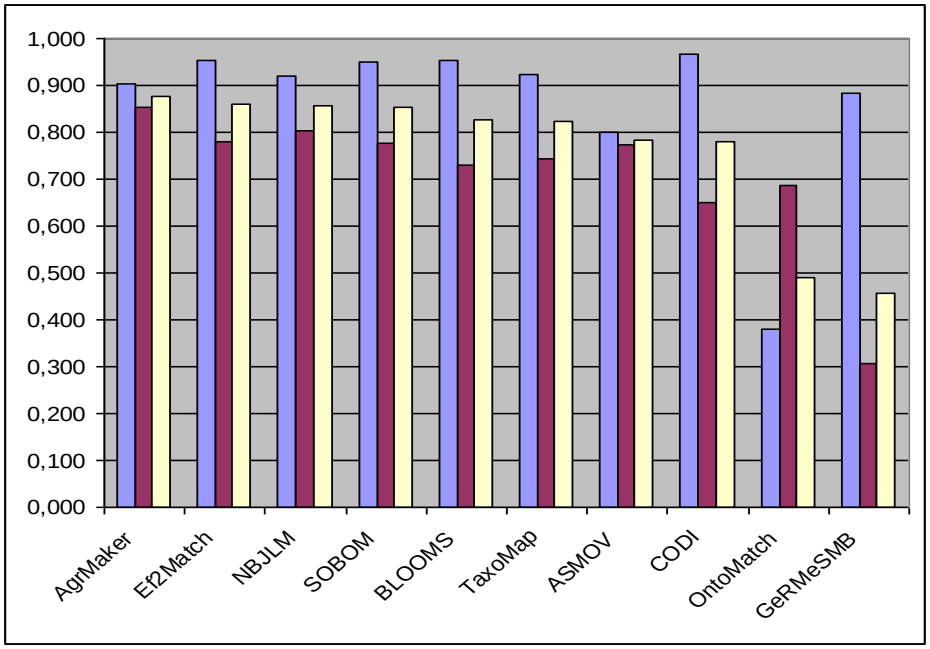
Precision ile bulunan eşleştirmelerin doğruluk oranı, recal ile tüm doğru eşleştirmeden ne kadarının bulunduğu ölçülmekte; iki ölçütün harmonik ortalaması ise f-measure değerini vermektedir. Üç ölçüt de [0,1] aralığında değer üretmektedir. Sonuç 1 değerine ne kadar yakın ise başarı da o kadar yüksek anlamındadır.

5.5 Başarım Karşılaştırmaları

Geliştirdiğimiz sistem gerçek dünya senaryosu olması bakımından anatomi testi üzerinde çalıştırılmış ve sonucu alınmıştır. Sistemin başarımı hem OAEI kampanyasının geliştirdiği bir araç olan AlignmentAPI kullanılarak tarafımızdan ölçülmüş hem de OAEI anatomi bölümü organizatörü Christian Meilicke tarafından farklı bir yöntem ile değerlendirilmiştir. İki şekilde de alınan başarım sonuçları birbirinin aynıdır. Başarım sonuçları OAEI kampanyası 2010 katılımcılarının sonuçları ile karşılaştırılmıştır. Çizelge 5.1 de sistemler f-measure değerine göre azalan şekilde sıralı olarak verilmiştir.

Çizelge 5.1 Oaei 2010 katılımcıları ve OntoMatch başarım sıralaması

Sistem	Precision	Recall	F-measure
AgrMaker	0.903	0.853	0.877
Ef2Match	0.955	0.781	0.859
NBJLM	0.920	0.803	0.858
SOBOM	0.949	0.778	0.855
BLOOMS	0.954	0.731	0.828
TaxoMap	0.924	0.743	0.824
ASMOV	0.799	0.772	0.785
CODI	0.968	0.651	0.779
OntoMatch	0.38	0.685	0.489
GeRoMe	0.884	0.307	0.456



Şekil 5.4 Oaei 2010 katılımcıları ve OntoMatch başarım sıralaması

Çizelge 5.1 ve Şekil 5.4'den anlaşılacağı gibi katılımcılardan 7 tanesi anatomi testinde, precision değerinde 0.9 barajını geçebilmiştir. OntoMatch en düşük precision değerine sahip sistem olmasına karşın recall değerinde GeRoMe ve CODI uygulamalarını geçmiştir Precision değeri geliştirilen sistemin arttırılması gereken en önemli özelliği olarak görülmüştür. Bulunan eşleştirmelerde doğruluk oranının arttırımı için OntoMatch'ın kullanmadığı treshould değeri belirleme kullanılabilir. Recall değeri olarak 0.8 barajını aşabilen sadece iki adet sistem vardır. Bu sonuca dayanarak tüm doğru eşlemelerin bulunabilmesi için daha çok mesafe kat edilmesi gerektiği söylenebilir. OAEI tarafından asıl başarım ölçütü olarak kabul edilen f-measure değerine göre; Word-Net, UMLS gibi kelime hazinelerini, iç ve dış benzerliği ve örnek benzerliği ölçütlerini kullanmamasına rağmen, OntoMatch 10 katılımcıdan 9. olmuştur. Değinilen ölçütlerin kullanılması ile OntoMatch'in başarımı yükseltilebilir.

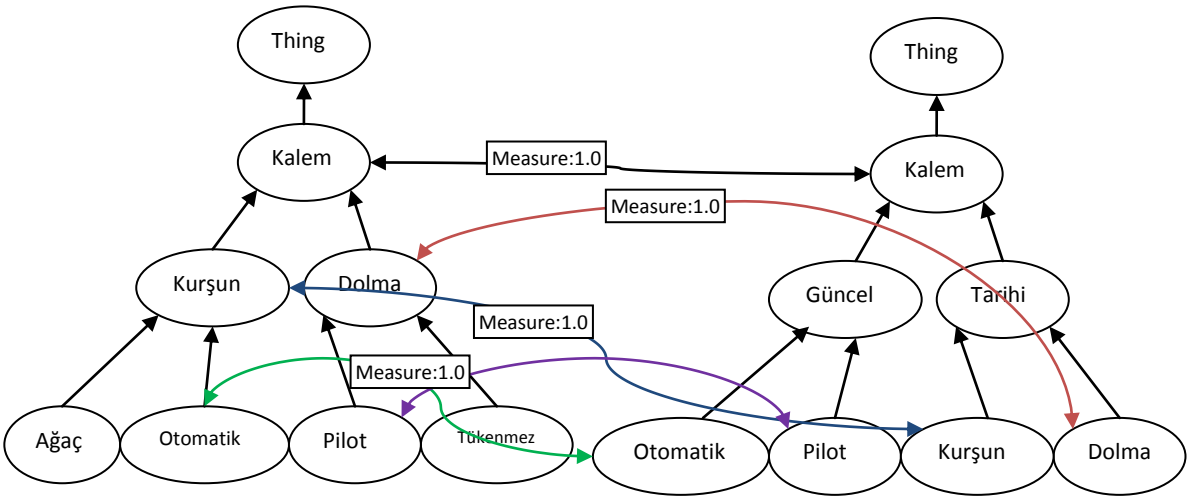
Çizelge 5.1 incelendiğinde, OntoMatch'in precision ölçütüne göre diğer katılımcılardan geride olduğu görülmektedir. Ancak OAEI'nin ölçümleri birden fazla ontoloji ve senaryo ile yapılmaktadır. Sistem girdi olarak owl formatında iki ontoloji almakta, çıktı olarak ise eşleştirme formatına uygun rdf uzantılı bir dosya oluşturmaktadır. OntoMatch'in en başarılı olduğu senaryolar Çizelge 5.2'de görülebilir:

Çizelge 5.2 Başarılı olunan senaryolar

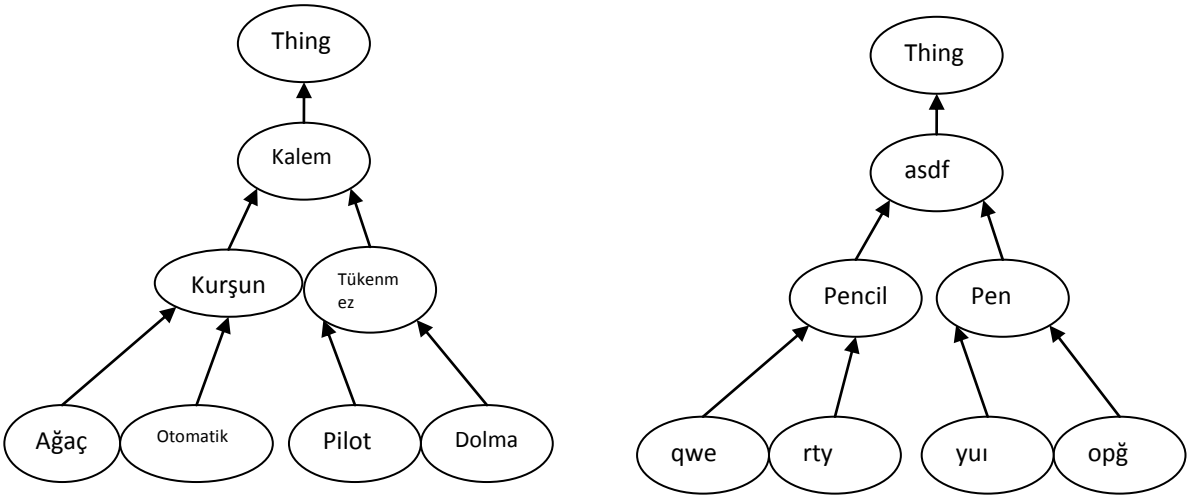
Test Adı	Precision	Recall	F-measure
101	0.857	0.989	0.918
103	0.857	0.989	0.918
104	0.857	0.989	0.918

Testlerde kullanılan ontoloji 33 konsept, 24 nesne özelliği, 40 veri özelliği, 56 konsepti belirli örnek ve 20 anonim örnek bulunduran bir bibliyografya ontolojisidir. 101 numaralı testte taban ve hedef ontoloji aynı ontolojiler iken, 103 numaralı testte hedef ontoloji taban ontolojinin owl:unionOf ve owl:oneOf gibi belirteçlerinin ve özellik tiplerinin çıkarılması ile elde edilmiş Owl Lite dilinde kodlanmış daha genelleştirilmiş bir halidir. 104 numaralı ontolojide ise ontoloji daraltılmış ve daha az bilgi verilmiş hali ile karşılaştırılmaktadır. Bu testlerde OntoMatch en iyi precision, recall ve f-measure değerini elde etmiştir.

Benchmark kategorisinde 100lü testler olarak adlandırılan bu ontolojilerin yanında 200lü testler olarak adlandırılan bir grup daha vardır. 100lü ontolojilerin aynı ontolojinin farklı derecelerde ayrıntılarını içermesine karşın 200lü ontolojiler aynı ontolojinin hiyerarşik yapısı korunarak konseptlerin isimlerinin anlamlı veya anlamsız kelimelerle değiştirilmesi ile elde edilmiştir. OntoMatch, ontolojiyi bir graf olarak ele aldığı için bu testlerde başarısız sonuçlar elde etmiştir. Örneğin Şekil 5.5 te görülen yapı için OntoMatch birebir eşlemeyi tam olarak yaparken, Şekil 5.6 te görülen yapı için eşleme yetersiz olmaktadır.



Şekil 5.5 Başarılı yapı



Şekil 5.6 Başarısız yapı

OntoMatch Şekil 5.6 eşlemesi için hiçbir sonuç üretmeyecektir. Farklı konulardaki ontoloji hiyerarşilerinin birbirine benzeme ihtimali de bulunmaktadır ve böyle bir durumda benzer hiyerarşilerin eşleştirilmesi hatalı olacaktır. Ancak aynı ontolojinin farklı konuşma dilleri için geliştirilmiş versiyonları arasında bir eşleme yapılmak istendiğinde, bu versiyonların hiyerarşileri birbiri ile tamamen benzeşecektir. Böyle bir gereksinim mevcut ise, eşleşme konusu tezde önerilen biçimden farklı bir yaklaşım ile ele alınmak zorundadır. OntoMatch hiyerarşik eşleme yaklaşımı kullanacak şekilde geliştirilmeye devam edilebilir.

KAYNAKLAR

[1] Berners-Lee, T. ve Fischetti, M. (1999). “Weaving the Web”, Harper, SanFrancisco.

[2] International Herald Tribune, Shannon V. "A 'More Revolutionary' Web", 18 Nisan 2011.

[3] Agarwal, A. "Web 3.0 Concepts Explained in Plain English", http://www.Labnol.org, 18 Nisan 2011

[4] Wikipedia Semantic Web, http://en.wikipedia.org/wiki/Semantic_Web, 18 Nisan 2011.

[5] Keen, A. "Web 1.0 + Web 2.0 = Web 3.0.", http://www.TypePad.com, 18 Nisan 2011.

[6] Gruber, T., (1993). "A Translation Approach to Portable Ontology Specifications", Knowledge Acquisition,5:199-220.

[7] Arvidsson F. ve Flycht-Eriksson A.,(2008). “Ontologies I”, http://www.ida.liu.se/~janma/SemWeb/Slides/ontologies1.pdf, 18 Nisan 2011.

[8] Cruz I. , Antonelli F. , Stroe C. , Keles U. ve Maduko A. ,(2009). “Using AgreementMaker to Align Ontologies for OAEI 2009: Overview, Results and Outlook”.

[9] Cruz I. ve Sunna W.,(2008). “Structural Alignment Methods with Applications to Geospatial Ontologies”, Transactions in GIS, special issue on Semantic SimilarityMeasurement and Geospatial Applications, 12(6):683.

[10] Cruz I, Antonelli F. ve StroeC ., (2009). “Efficient Selection of Mappings and Automatic Quality-driven Combination of Matching Methods.”, ISWC International Workshop on Ontology Matching. 2009, Chantilly -US.

[11] Seddiqui H. ve Aono M. , (2009). “Anchor-Flood: Results for Oaei-2009” .

[12] David J.,(2009). “Aroma Results for Oaei- 2009”.

[13] Yves R., Kabuka M., (2008). “Asmov Results for OAEI-2008” .

[14] Yves R. , Shironoshita P. ve Kabuka M.,(2009). ”Asmov Results for Oaei-2009”.

[15] Jena, HP Labs, http://www.hpl.hp.com/semweb/, 18 Nisan 2011.

[16] Unified Medical Language System, http://umlsks.nlm.nih.gov/, 18 Nisan 2011.

[17] WordNet, http://wordnet.princeton.edu/, 18 Nisan 2011.

[18] Nagy M., Vargas-Vera M. ve Motta E.,(2006). "DSSim Ontology Mapping with Uncertainty".

[19] Shafer, G. (1976). "A Mathematical Theory of Evidence", Princeton University Press.

[20] Jian N., Hu W., Cheng G. ve Qu Y., (2005). "FalconAO: Aligning Ontologies with Falcon".

[21] Wang P. ve Xu B., (2009). "Lily: Ontology Alignment Results for OAEI 2009".

[22] Zhang X, Zhong Q, Li J., Tang J., Xie G ve Li H., (2008). "RiMOM Results for OAEI 2008".

[23] Felzenszwalb, P.F. ve Huttenlocher, D.P. (2006). "Efficient Belief Propagation for Early Vision". International Journal of Computer Vision, 70, 1:41-54, Kasım 2006.

[24] Lambrix P, Tan H. ve Liu Q. ,(2008). "SAMBO and SAMBOdtf Results for the Ontology Alignment Evaluation Initiative 2008".

[25] Xu P.,Tao H., Zang T ve Wang Y.,(2009). "Alignment Results of SOBOM for OAEI 2009".

[26] Stoilos G.,Stamou G. ve Kollias S.(2005)." A String Metric for Ontology Alignment", 4th International Semantic Web Conference(ISWC'05). 6-10 Ekim 2005 Galway, İrlanda.

[27] Noy N.F. ve Musen M.A., (2001). "Anchor-PROMPT: Using Non-Local Context for Semantic Matching",IJCAI2001 Workshop on Ontology and Information Sharing, 4-5 Ağustos 2001, Seattle, USA.

[28] Melnik S., Molina H. G. ve Rahm E., (2002). "Similarity Flooding: A Versatile Graph Matching Algorithm", 18th Int'l Conf. Data Eng. (ECDE'02), 3-8 Nisan 2002, Atlanta, USA.

[29] Hamdi F, Zargayouna H., SafarB. ve Reynaud C., (2008). "TaxoMap in the OAEI 2008 Alignment Contest".

[30] Schmid H., (1994). "Probabilistic Part-of-Speech Tagging Using Decision Trees", 4.International Conference on New Methods in Language Processing, 1994, Stuttgart.

[31] Mao M. ve Peng Y.,(2007). "The PRIOR+: Results for OAEI Campaign 2007" .

[32] Curino C., Orsi G. ve Tanca L., (2007). " X-SOM Results for OAEI 2007".

[33] Pirrò G. ve Talia D.,(2006). "UFOme: A User Friendly Ontology Mapping Environment".

[34] Lucene- The Apache Lucene Project, http://lucene.apache.org, 19 Nisan 2011.

[35] Pirrò, G. ve Talia, D.,(2007). “An approach to Ontology Mapping based on the Lucene Search Engine Library”, 1st International Workshop on Semantic Web Architectures For Enterprises (SWAE ’07), 4 Ekim 2007, Regensburg, Germany.

[36] Java WordNet Similarity Library (JWSL) and the Similarity Experiment http://grid.deis.unical.it/similarity, 19 Nisan 2011.

[37] Jiang, J. ve Conrath, D.,(1997).” Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy”, ROCLING X, 1997, Taiwan.

[38] Winkler, W. E.,(1999). “The State of Record Linkage and Current Research Problems”.

[39] Levenshtein, I.V.,(1996). “Binary Codes Capable of Correcting Deletions, Insertion and Reversals”.

[40] Gruber, T.R.,(1993).” A Translation Approach to Portable Ontology Specifications”, Knowl. Acquis., 5:199-220.

[41] Uschold, M. ve Gruninger, M.,(2004). “Ontologies and Semantics for Seamless Connectivity”, SIGMOD, 3:58-64.

[42] Damerau,F.J.,(1964). “A Technique for Computer Detection and Correction of Spelling Errors”, Communications of the ACM.

[43] Levesthtein Uzaklık Ölçütü, http://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance, 20 Nisan 2011.

[44] Damerau Levenshtein Uzakllık Ölçütü, http://blog.lolyco.com/sean/2008/08/27/damerau-levenshtein-algorithm-levenshtein-with-transpositions/ , 20 Nisan 2011.

[45] Doğrusal Atama Problemi, http://en.wikipedia.org/wiki/Linear_assignment_problem, 20 Nisan 2011.

[46] Jonker R. ve Volgenant A.,(1986). “A Shortest Augmenting Path Algorithm for Dense and Sparse Linear Assignment Problems”. Computing by Springer-Verlag, 38:325-340.

[47] Carpaneto,G. veToth,P.,(1980). ”Algorithm548 Solution of the Assignment Problem”, ACMTransactions on Mathematical Software 6:104-111.

[48] Dorhout, B., (1973), “Het Lineaire Toewijzings Probleem: Vergelijking van Algorithmen”.

[49] Dorhout, B. (1975).”Experiments with Some Algorithms for the Linear Assignment Problem”.

[50] Burkard,R.E.ve Dcrigs,U.,(1980). “Assignment and Matching Problems : Solution Methods with Fortran Programs”, NewYork: Springer 1980.

[51] Bertsekas,D.P.,(1981). “A New Algorithm for the Linear Assignment Problem”, MathematicalProgramming 21: 152-171.

[52] Dell’amico, M. Ve Toth P.,(2000).” Algorithms and Codes for Dense Assigment Problems, The State of Art”.

[53] Hungarian Algoritması, http://en.wikipedia.org/wiki/Hungarian_algorithm, 20 Nisan 2011.

[54] Simplex Algoritması, http://en.wikipedia.org/wiki/Simplex_algorithm, 20 Nisan 2011.

[55] Liao L., Qu, Y. ve Leung H., (2005). "A Software Process Ontology and Its Application", ISWC 2005 Workshop on Semantic Web Enabled Software Engineering, 6-10 Ekim 2005 Galway, İrlanda.

EK-A

DLD Algoritması

```
int DamerauLevenshteinDistance(char str1[1..lenStr1], char str2[1..lenStr2])
  // d is a table with lenStr1+1 rows and lenStr2+1 columns
  declare int d[0..lenStr1, 0..lenStr2]
  // i and j are used to iterate over str1 and str2
  declare int i, j, cost

  for i from 0 to lenStr1
     d[i, 0] := i
  for j from 1 to lenStr2
     d[0, j] := j

  for i from 1 to lenStr1
     for j from 1 to lenStr2
        if str1[i] = str2[j] then cost := 0
                             else cost := 1
        d[i, j] := minimum(
                             d[i-1, j  ] + 1,    // deletion
                             d[i  , j-1] + 1,    // insertion
                             d[i-1, j-1] + cost   // substitution
                           )
        if(i > 1 and j > 1 and str1[i] = str2[j-1] and str1[i-1] = str2[j]) then
           d[i, j] := minimum(
                             d[i, j],
                             d[i-2, j-2] + cost   // transposition
                           )
         end for
   end for

  return d[lenStr1, lenStr2]
```

LAPJV Algoritması

```
package matcher;
// n must be lower than m !!!!
public class LapJvRct {
public static int[] Lapjv(int m,int n,int[][]c){
int inf= 30000;
   int cnt, f, f0, h, i, i1, j, j1, j2 = 0, k, u1, u2, last = 0, low, min = 0, up;
   int[]    col, d , free, pred, v, x, y ;


   x=new int[n];
   free=new int[n];
   col=new int[m];
   y=new int[m];
   v=new int[m];
   d=new int[m];
   pred=new int[m];

   for (i=0;i<n;i++){ x[i]=0; free[i]=i;}

   for (j=0;j<m;j++){ col[j]=j; y[j]=0; v[j]=0;}

   cnt=0;
   f=n;                  // Each row is still unassigned
```

```
do{
    k=0; f0=f; f=0;
    while (k < f0)
    {
      i=free[k];
      k=k+1;
      u1=c[i][0]-v[0];
      j1=0;
      u2=inf;
      for (j=1;j<m;j++)
      {
        h=c[i][j]-v[j];
        if (h<u2)
        if (h>=u1) {u2=h; j2=j;}
           else {u2=u1; u1=h; j2=j1; j1=j;}
      }
      i1=y[j1];
      if (u1<u2) v[j1]=v[j1]-u2+u1;
            else if (i1>0){ j1=j2; i1=y[j1]; }
      if (i1>0){ if (u1<u2) {k=k-1; free[k]=i1; x[i1]=0;}
             else { f=f+1; free[f]=i1; x[i1]=0;}}
      x[i]=j1; y[j1]=i;
    }
    cnt=cnt+1;
  }while (cnt<2);
//-------------------------------------------------------
  f0=f;
  for (f=0;f<f0;f++)     //{ Find augmenting path for each unassigned row }
  {
    i1=free[f]; low=0; up=0;
    for (j=0;j<m;j++){ d[j]=c[i1][j]-v[j]; pred[j]=i1;}
   cnt=0;
    do{
      if (up==low)       //{ Find columns with new value for minimum d }
      {
        last=low-1; min=d[col[up]]; up=up+1;
        for (k=up;k<m;k++)
        {
          j=col[k]; h=d[j];
          if (h<=min)
          {
            if (h<min) {up=low; min=h;}
            col[k]=col[up]; col[up]=j; up=up+1;
          }
        }
        for (h=low;h< up-1;h++)
```

```
        {
          j=col[h];
          if (y[j]+cnt==0) {
          for (k=0;k<last;k++)//                { Updating of column prices }
          {
            j1=col[k];
            v[j1]=v[j1]+d[j1]-min;
          }
          do{                          //        { Augmentation }
            i=pred[j]; y[j]=i; k=j; j=x[i]; x[i]=k;
          }while (i!=i1);
         cnt=1; f=f0;
         break;

          }
        }
      }  //{ up=low }
      if (cnt==0){
      j1=col[low]; low=low+1; i=y[j1]; //             { Scan a row }
      u1=c[i][j1]-v[j1]-min;
      for (k=up;k<m;k++)
      {
        j=col[k];
        h=c[i][j]-v[j]-u1;
        if (h<d[j])
        {
          d[j]=h; pred[j]=i;
          if (h==min)
          if (y[j]+cnt==0) {
          for (k=0;k<last;k++)//                { Updating of column prices }
          {
            j1=col[k];
            v[j1]=v[j1]+d[j1]-min;
          }
          do{                          //        { Augmentation }
            i=pred[j]; y[j]=i; k=j; j=x[i]; x[i]=k;
          }while (i!=i1);
         cnt=1; f=f0;
          }
                else {col[k]=col[up]; col[up]=j; up=up+1;}
        }
      }}
      }while (cnt==0);
    } // { for f }

  return x;
}}
```

ÖZGEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı	: Ali Alperen ÇOLAK
Doğum Tarihi ve Yeri	: 23.10.1981 / Çanakkale
Yabancı Dili	: İngilizce
E-posta	: alperen@ce.yildiz.edu.tr

ÖĞRENİM DURUMU

Derece	**Alan**	**Okul/Üniversite**	**Mezuniyet Yılı**
Lisans	Bilgisayar Mühendisliği	Yıldız Teknik Üniversitesi	2007
Lise	Konya Atatürk SML		1999

İŞ TECRÜBESİ

Yıl	**Firma/Kurum**	**Görevi**
2011 -	YTÜ Bilgisayar Mühendisliği Böl.	Araştırma Görevlisi
2008- 2011	Türkay Bilişim	Yazılım Mühendisi
2001-2002	Genç Yazılım	Programcı

Proje

1. OntoMatch Ontoloji Eşleştirme Çözümü
2. Spatiotemporal Data Modeling in PostgreSql for Future Queries
3. Online Dinamik Dershane Otomasyon Sistemi