TURGUT ÖZAL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ve BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

EĞİTİM ARAMA MOTORUNDA SORGU ÖNERME

DOKTORA TEZİ

Hazırlayan
İbrahim Bahattin VİDİNLİ

Tez Danışmanı
Yrd. Doç. Dr. Rıfat ÖZCAN

Ankara-2016

(Boş sayfa)

TURGUT ÖZAL ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK ve BİLGİSAYAR MÜHENDİSLİĞİ
ANABİLİM DALI

EĞİTİM ARAMA MOTORUNDA SORGU ÖNERME

DOKTORA TEZİ

Hazırlayan
İbrahim Bahattin VİDİNLİ

Tez Danışmanı
Yrd. Doç. Dr. Rıfat ÖZCAN

Ankara-2016

Turgut Özal Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmasında;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,

- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,

- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,

- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,

- kullanılan verilerde herhangi bir tahrifat yapmadığımı,

- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı beyan ederim.

...../...../.....

İ. Bahattin VİDİNLİ

ONAY

İ. Bahattin VİDİNLİ tarafından hazırlanan “Eğitim Arama Motorunda Sorgu Önerme” başlıklı bu çalışma, ... tarihinde yapılan savunma sınavı sonucunda (oybirliği/oyçokluğu) ile başarılı bulunarak jürimiz tarafından Elektrik ve Bilgisayar Mühendisliği dalında doktora tezi olarak kabul edilmiştir.

Yrd. Doç. Dr. Rıfat ÖZCAN

Doç. Dr. Yıldıray YALMAN

Doç. Dr. İsmail Sengör ALTINGÖVDE

Doç. Dr. Hasan Şakir BİLGE

Yrd. Doç. Dr. Enver KAYAASLAN

ÖNSÖZ

Bu tezde günümüzde yaygın kullanımı olan internet arama motorlarının sorgu önerme özelliklerinin geliştirilmesine yönelik çalışmalar yapılmış, gelişmeler sağlanmış ve önerilerde bulunulmuştur.

Bu doktora çalışmasının yapılmasında gerekli ve en güzel imkanları sunan Turgut Özal Üniversitesi'ne, ilgili çalışmalar için cesaretlendiren ve desteklerini esirgemeyen Prof. Dr. İsmail ERTÜRK'e, doktora çalışması süresince kıymetli destekleri, yönlendirmeleri ile çalışmaların bu seviye gelmesini sağlayan tez danışmanım Yrd. Doç. Dr. Rıfat ÖZCAN'a, tez izleme komitesindeki diğer üyeler olan ve çok değerli önerileri ve yönlendirmeleri ile doktora çalışmasının başarıya ulaşmasında büyük katkıları olan Doç. Dr. Yıldıray YALMAN'a ve Doç. Dr. İ. Sengör ALTINGÖVDE'ye, doktora ders sürecinde kıymetli bilgilerini paylaşan Prof. Dr. İsmail AVCIBAŞ, Doç. Dr. Ünal GÖKTAŞ, Doç. Dr. Vedat KIRAY ve Yrd. Doç. Dr. Mehmet CANTÜRK'e, desteklerinden ötürü TÜBİTAK'a ve yoğun çalışma temposu süresince desteklerini esirgemeyen eşime ve tüm aileme teşekkürlerimi sunarım. Bu çalışma TÜBİTAK 113E065 numaralı projesi (Arama Teknolojilerinin İlk ve Orta Öğretim Öğrencilerine Göre Optimizasyonu) kapsamında desteklenmiştir.

ÖZET

VİDİNLİ, İ. Bahattin. Eğitim Arama Motorunda Sorgu Önerme, Doktora Tezi, Ankara, 2016.

İnternet ortamındaki ve daha geniş anlamıyla dijital ortamdaki veriler devasa boyutlarda olup günden güne veri miktarı artmaktadır. Artan bu verilerde istenen bilgiye ulaşmak için arama motorları çok önemli bir işlev görmektedir. Arama motorları, kullanıcıların işine yarayabilecek yardımcı bilgiler de sunmaktadır ki bunlardan biri sorgu önermelerdir. Bu tezde, arama motorlarının sorgu önerme işlevlerini geliştirici yöntemler önerilmiş, gerçeklemeler yapılmış, deneyler yapılarak sonuçlar ölçülmüş ve deney sonuçlarının istatistiksel değerlendirmeleri yapılmıştır.

Sorgu önermeye yönelik olarak bir sorgu karşılaştırma yöntemi ve problem indirgemesi önerilmiştir. Sorgu önerme algoritmalarının birleştirilerek ve geliştirilerek kullanılabileceği bir sorgu önerme çerçeve mimarisi önerilmiştir. Bu mimari içinde kullanılabilecek ve sorguların çeşitli özelliklerini (oturum, yol frekansları v.b.) kullanan sorgu önerme/sıralama algoritmaları geliştirilmiş ve mimari içine entegre edilmiştir. İlk ve orta öğretim öğrencilerinin kullandığı gerçek bir eğitim arama motorunun sorgu kayıtları veri seti olarak kullanılmıştır. Sorguların eğitimle ilgili özelliklerinin de (sorgunun dersi ve sınıfı) kullanıldığı bu algoritmalar ve mimari gerçekleştirilmiş ve rastgele seçilen sorgu kümesi için sorgu önermeleri üretilmiştir. Bunlar değerlendiriciler vasıtasıyla başarım oranını ölçmek suretiyle değerlendirilmiştir. Bu deney ve değerlendirme sonuçları kullanılarak başarım artışları hesaplanmış, elde edilen sonuçlar istatistiksel metriklerle teyid edilmiştir.

Yapılan çalışma ve deneyler neticesinde yeni önerilen algoritma ve yöntemlerin literatürde bilinen bir sorgu önerme yöntemine göre belirgin bir şekilde ilerleme kaydettiği gösterilmiştir.

Anahtar Sözcükler: *Arama motoru, Sorgu önerme, Sorgu tamamlama*

ABSTRACT

VİDİNLİ, İ. Bahattin. Query Suggestion on Educational Search Engines, Doctorate Thesis, Ankara, 2016.

The amount of data in Internet and digital media is huge. Search engines play a very important role in this increasing data by enabling us to reach the information we need. While search engines are presenting users the links of datas that they search for, they may also show users some extra information such as query suggestions.

In this thesis, methods and algorithms that improve the query suggestion functionality of search engines are suggested, designed, implemented, tested and results measured, verified by statistical methods.

A query comparison method and a reduction of problem of query suggestion is suggested. A query suggestion framework is suggested, designed and implemented where many query suggestion methods and algorithms can be merged and combined in it. Query suggestion algorihtms designed and integrated into this query suggestion framework. These algorithms uses some of features of queries (such as session, path frequencies etc.) in a search engine query log, including some educational features (such as course, grade). Query log of a real educational search engine used as data set, which used by K12 students. Random queries are selected and query suggestion algorithms are run to produce query suggestions. These suggestions are evaluated by human assessors and scores of different algorithms are measured. The results of evaluation are also verified by statistical methods. The results of tests, evaluation and statistical verifications show that, new suggested framework and methos achive better results compared to previous similar work in this field.

Keywords: *Search engine, Query suggestion, Query recommendation, Query autocompletion*

İÇİNDEKİLER

ÖNSÖZ.....i

ÖZET.....ii

ABSTRACT.....iii

İÇİNDEKİLER.....iv

SİMGELER VE KISALTMALAR.....viii

TABLOLAR.....ix

ŞEKİLLER.....x

GİRİŞ.....1

a) Arama Motorlarında Sorgu Önerme Problemi – Motivasyon (Motivation).....2

b) Katkılar.....4

c) Ana hatlar.....5

TEMEL BİLGİLER ve İLGİLİ ÇALIŞMALAR.....6

1.1. TEMEL BİLGİLER.....6

1.1.1. BİLGİ ERİŞİMİ (INFORMATION RETRIEVAL).....6

1.1.2. WEB BİLGİ ERİŞİMİ (WEB INFORMATION RETRIEVAL).....7

1.1.3. İNTERNET ARAMA MOTORLARINA GİRİŞ.....7

1.1.3.1. İnternet Arama Motorlarının Çeşitleri.....8

1.1.4. ARAMA MOTORLARINDA SORGU ÖNERME İLE İLGİLİ TEMEL KAVRAMLAR.....9

1.1.4.1. Sorgu, İlk Sorgu, Sorgu Önerme, Arama Sonuçları, Doküman/URL 9

1.1.4.2. Sorgu Kayıtları (Query Logs).....10

1.1.4.3. Sorgu-Tıklama Grafiği (Query Click Graph, Query-URL Bipartite Graph).....11

1.1.4.4. Doküman/Url Bilgileri ve İçerikleri (Content).....12

1.1.4.5. Çapa Metni (Anchor Text).....13
1.1.4.6. Kullanıcı Bağlamı ve İlgili Bilgileri.....13
1.2. SORGU ÖNERMEDE YAKLAŞIMLAR VE İLGİLİ ÇALIŞMALAR.....13
TEORİ ve YÖNTEM.....18
2.1. SORGU ÖNERME PROBLEMİNİN İNDİRGENMESİ.....18
2.2. SORGU ÖNERME ÇERÇEVE YAPISI/MİMARİSİ.....21
2.2.1. Sorgu Seçimi/Bulunması Aşaması.....25
2.2.2. Sıralama Öncesi Genel Kontroller.....26
2.2.3. Sıralama Aşaması.....28
2.2.3.1. Alt Katmanda Bir Özelliğe Göre Skorların Belirlenmesi.....29
2.2.3.2. Çerçeve Yapı'da Sıralama.....30
2.2.3.2.1. Sıralama Birleştirme Yöntemleri (Rank Aggregation Methods). .31
2.2.3.2.1.1.Ağırlıklı Skor Temelli Birleştirme Yöntemleri (Weighted Score Based Aggregation).....32
2.2.3.2.1.2.Sıra Temelli Birleştirme Yöntemleri (Rank Based Aggregation) 35
2.2.3.2.1.2.1.Borda Count.....35
2.2.3.2.1.2.2.Weighted Borda Count.....36
2.2.3.2.1.2.3.Simple Voting.....37
2.2.3.2.1.2.4.Weighted Voting.....37
2.2.3.2.2. Sıralama Birleştirme Yöntemlerinin Seçimi.....37
2.3. ÖNERİLEN/GERÇEKLEŞTİRİLEN SORGU ÖNERME/SIRALAMA ALGORİTMALARI.....39
2.3.1. Hitting Time.....40
2.3.2. Oturum Bilgisini Kullanan Algoritmalar.....42
2.3.2.1. Oturum Sayısı Algoritması.....43
2.3.2.2. Oturum Yakınlığı Algoritması.....44
2.3.3. Yol Frekans Değeri Algoritmaları.....47
2.3.3.1. Grafik İçerisindeki Tıklama Yolu.....47
2.3.3.2. Yol Frekanslarının Kullanıldığı Algoritmalar.....49

2.3.4. Yol Frekansları Algoritmasının Hitting Time Algoritmasından Farkı.....53
2.3.5. Eğitimle İlgili Özellikleri Kullanan Sorgu Önerme Algoritmaları.....54
2.3.5.1. Sınıf Benzerliği.....54
2.3.5.2. Ders Bilgisi.....56
2.3.6. Karma (Hibrid) Algoritmalar.....58

DENEYLER, BULGULAR ve SONUÇLARI.....59
3.1. DENEY ORTAMI ve VERİ KÜMESİ.....59
3.1.1. Deneyde Kullanılan Araçlar.....59
3.2. GELİŞTİRİLEN YAZILIM(LAR).....60
3.2.1. Web Servis.....63
3.3. DENEY PROSEDÜRÜ.....64
3.3.1. Verilerin Ön İşlemesi ve Veritabanına Kaydedilmesi.....64
3.3.2. Sorgu Oturumlarının Oluşturulması.....65
3.3.3. Sorguların Seçilmesi ve Sorgu Önermenin Simülasyonu.....65
3.3.4. Gerçeklenip Test Edilen Algoritmalar.....66
3.3.5. Hibrid Metodlar için Parametre Belirleme.....68
3.3.5.1. Cross-Validation Tekniği ile Parametre Belirleme.....68
3.3.5.2. Manüel Parametre Belirleme.....70
3.3.6. Sorgu Önermelerin Değerlendirilmesi.....70
3.4. BULGULAR VE DENEY SONUÇLARI.....72
3.4.1. Değerlendirici Mutabakatları.....72
3.4.2. Yeni Algoritmaların Performansları.....73
3.4.2.1. Ortalama Başarım Ölçümü.....73
3.4.2.2. NDCG (Normalized Discounted Cumulative Gain) Ölçümü.....80
3.4.3. İstatistiksel Testler.....82
3.5. İleride Yapılabilecek Çalışmalar.....84
3.5.1. Yeni Sorgu Özellikleri.....84
3.5.2. Yeni Sorgu Karşılaştırma Yöntemlerinin Bulunması.....84
3.5.3. Mimaride Yapılabilecek İyileştirmeler.....84
3.5.4. İmla Denetimi Modülü.....85

3.5.5. Veri Kalitesinin Artırılmasına Yönelik Çalışmalar.....85
3.5.6. Performans İyileştirmeleri.....85

SONUÇ.....86

KAYNAKLAR.....87

SİMGELER VE KISALTMALAR

IP: Internet Protocol *(İnternet haberleşme protokolü)*

ID: Identification *(Kimlik bilgisi)*

QS: Query Suggestion *(Sorgu Önerme)*

LO: Learning Object (Eğitim Nesnesi, Dokümanı)

SÖ: Sorgu Önerme

HT: Hitting Time

BFS: Breadth First Search

DFS: Depth First Search

HT-BFS: Hitting Time BFS

HT-DFS: Hitting Time DFS

PF3-BFS: Path Frequency-3 BFS *(Yol Frekansları-3 BFS algoritması)*

PF4-BFS: Path Frequency-4 BFS *(Yol Frekansları-4 BFS algoritması)*

Hyb1: Hybrid-1 BFS *(Hibrid/Karma-1 BFS algoritması)*

Hyb2-Bo: Hybrid-2 Borda Count *(Hibrid/Karma-2 BFS Borda algoritması)*

Hyb3-Wbo: Hybrid-3 Weighted Borda Count *(Hibrid/Karma-3 BFS Weighted Borda algoritması)*

Hyb4-WV: Hybrid-4 Weighted Voting *(Hibrid/Karma-3 BFS Weighted Voting algoritması)*

TABLOLAR

Tablo 1: Borda Count Puanlama Formülü Hesaplama Örneği.....36
Tablo 2: Weighted Borda Count Puanlama Formülü Hesaplama Örneği.....37
Tablo 3: Örnek Sorgu Önermeleri ve Skorları.....38
Tablo 4: Yol Frekansları Algoritmalarının Çeşitleri.....51
Tablo 5: Deneyde ve Tez Yazımında Kullanılan Araçlar.....60
Tablo 6: Sorgu Seçimi ile İlgili Kriterler.....66
Tablo 7: Cross-Validation Tekniğinin Uygulanması İçin Verinin Bölünmesi.....69
Tablo 8: Değerlendirmede Kullanılan Puanlamalar.....72
Tablo 9: Kappa Mutabakatının Değerlendirilmesi.....73
Tablo 10: Kısaltma ve Yöntem Açıklamaları.....74
Tablo 11: Sorgu Önerme Algoritma Çiftlerinin Eşli t-test Sonuçları.....83

ŞEKİLLER

Şekil 1: Arama Motorlarında Sorgu Önermeye Örnekler.....3
Şekil 2: İlk Sorgu ve Sorgu Önermelerine Örnek.....10
Şekil 3: Sorgu-Tıklama İkili Grafiği (Query URL Bipartite Graph).....12
Şekil 4: Sorguların Karşılaştırılması Yöntemi.....19
Şekil 5: Sorgu Önerme Çerçeve Mimarisi.....24
Şekil 6: Örnek: Bir Ağacın DFS ile Gezilme Aşamaları.....25
Şekil 7: Örnek: Bir Ağacın BFS ile Gezilme Aşamaları.....26
Şekil 8: Sorgu Önerme Çerçeve Mimarisi Sıralama Katmanı.....29
Şekil 9: Oturum Yakınlığı Algoritması Temsili Gösterim.....45
Şekil 10: Geliştirilen Programın Örnek Çıktısı.....61
Şekil 11: Geliştirilen Programın Örnek Çıktısı.....62
Şekil 12: Sorgu Önerme Değerlendirme/Anket Arayüzü.....71
Şekil 13: Sorgu Önerme Algoritmalarının Ortalama İlgililik Skorları.....75
Şekil 14: Yeni Algoritmaların temel Hitting Time DFS'e Göre Ortalama İlgililik Skor Artış Yüzdeleri.....76
Şekil 15: Sorgu Önerme Algoritmalarının Sık, Orta ve Nadir (Head, Torso, Tail) Sorgular için Ortalama İlgililik Skorları.....78
Şekil 16: Sık, Orta ve Nadir (Head, Torso, Tail) Sorgular için Yeni Yöntemlerin Ortalama İlgililik Performans Artış Yüzdeleri.....79
Şekil 17: Algoritmaların NDCG Performansları.....81
Şekil 18: Hitting Time-DFS Temel Metoduna Göre Yeni Algoritmaların Performans Artışı (NDCG Metriğine Göre).....81

GİRİŞ

Dijital haberleşme, bilgisayar ve İnternet çağının yaygınlaşması ile beraber bilgi üretimi ve bilgiye erişim imkanları oldukça artmıştır. Bilgi üretiminin kolaylaştığı ve yaygınlaştığı, geniş kitlelerin kolayca ve kimi zaman da farkında olmadan bilgi ürettiği günümüzde İnternet üzerindeki bilgi kaynakları belki de tahmin edilemeyecek kadar artmıştır. Devasa boyutlara ulaşan bilgi kaynaklarında bilgiye erişim ve çıkarım büyük önem arz etmektedir. Özellikle internet ortamındaki bilgi kaynaklarına kullanıcıların kolaylıkla erişebilmesini sağlamak için genel anlamda "İnternet arama motorları" diye adlandırılan araçlar geliştirilmiştir. İnternet arama motorları, İnternet üzerindeki kaynaklara çok daha pratik olarak ulaşmayı sağlayan İnternetin olmazsa olmazları haline gelmiştir. Arama motorları, Bilgi Erişimi biliminin bir alt bilimi olan Web Bilgi Erişimi'nde bir araç olarak değerlendirilebilir.

İnternet arama motorları temel olarak, İnternet üzerindeki içeriklerin kullanıcılar tarafından kolayca bulunabilmesini sağlayabilmek için web sayfalarını kaydederler. Daha sonra kullanıcılar, arama motoru arayüzünü kullanarak istedikleri bilgi ya da internet sayfasına ulaşmaya çalışırlar. Kullanıcılar arama motoruna, aradıkları şeye dair bazı anahtar kelimeler (sorgu) verirler. Bu girdiye göre arama motoru, internet adresleri listesi ile beraber bu adreslere dair bazı bilgileri (özet v.b.) gösterebilir.

İnternetin yaygınlaşması ile ortaya çıkan arama motorları, kullanıcıların işlerini bir hayli kolaylaştırmış, internet adreslerinin ezberlenmesi veya sık kullanılanlara kaydedilmesi ihtiyacını da ortadan kaldırmıştır. Bu o kadar yaygınlaşmıştır ki, kullanıcılar artık en sıradan, en bildikleri adreslere bile arama motorlarını kullanarak erişir hale gelmişlerdir.

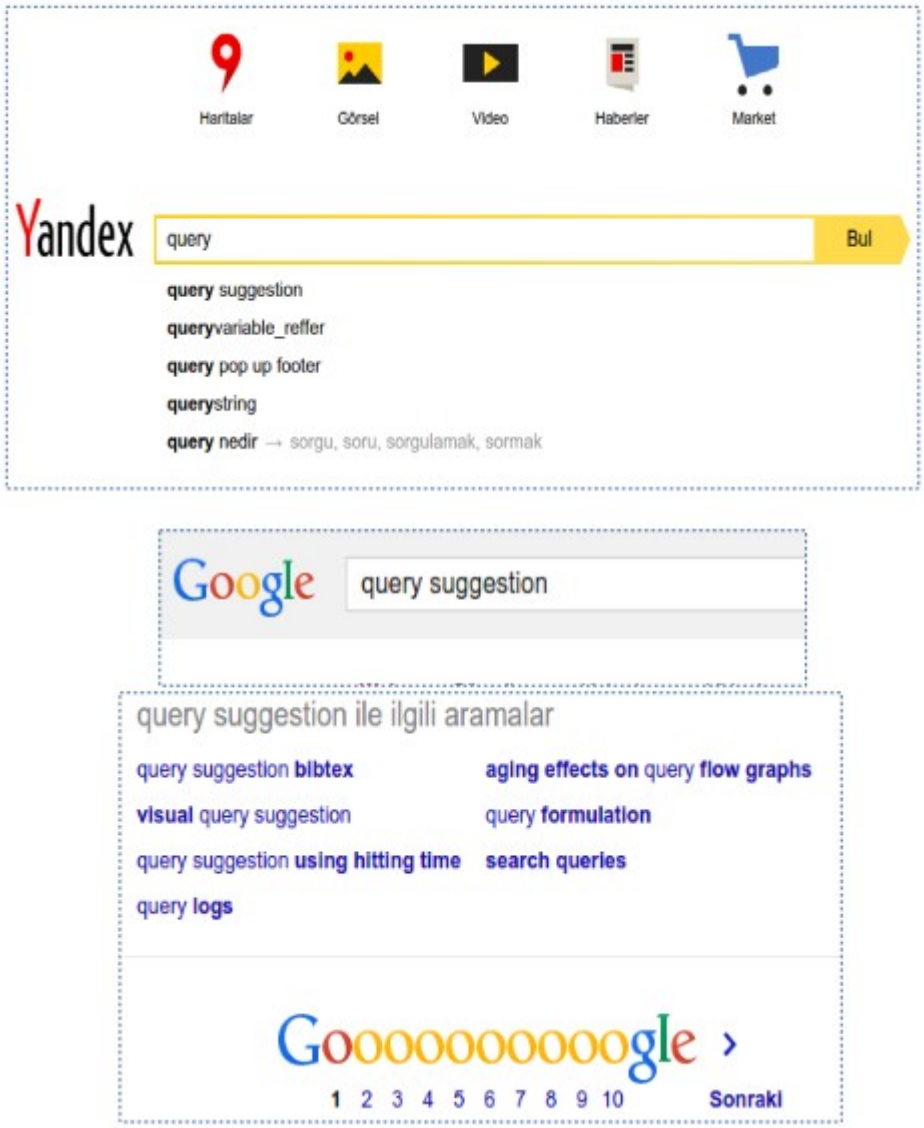
Zaman içerisinde İnternet arama motorlarının kullanıcılara sunduğu bilgiler, ihtiyaçlar ve kullanım alışkanlıklarına bağlı olarak gelişmiş ve çeşitlenmiştir. Kullanıcının aradığı bilgi ile ilgili internet sayfalarının gösterilmesine ek olarak,

aranılan konu ile ilgili olabilecek alternatif sorguların kullanıcıya sunulması faydalı olabilir. Bir bilgi boyutu olarak da adlandırılabilecek bu alternatif sorgular "sorgu önerme" *(query suggestion)* olarak adlandırılabilir. Arama yapılan konu ile ilgili "sorgu önerme"lerinin yapılması, kullanıcının arama deneyimini geliştirebileceği gibi hedefe daha kısa sürede ulaşmasını da sağlayabilecektir.

a) Arama Motorlarında Sorgu Önerme Problemi – Motivasyon *(Motivation)*

Kullanıcıların arama motorlarındaki yaptıkları aramalar çoğunlukla kısa, yazım hatalı ya da eksik olabilmektedir (Cui, Wen, Nie, & Ma, 2003). Bu yüzden, kullanıcılar aradıkları sonuçları bulmakta zorlanabilmekte ya da yanlış yönlere gidebilmektedir. Öğrenciler, gençler ve çocuklar aradıkları bilgi için sorgu oluşturmakta zorluk çekebilmektedirler. Bu duruma bir çözüm olması açısından, arama motorlarında (birinci boyut bilgi olarak görebileceğimiz) normal arama sonuçlarına ek olarak, kullanıcıya "şunu da arayabilirsiniz" ya da "şunu mu aramak istediniz" şeklinde sorulara cevap olarak başka aramalar önerilmesi faydalı görülmüştür. Bu sayede kullanıcının kısıtlı bilgisine rağmen, alternatif kaynaklara da ulaşma imkanı doğabilmektedir. Kullanıcıların aradığı bilgi hakkında sınırlı bilgi ya da anahtar kelimeye sahip olduğu düşünülürse, bu oldukça faydalı olacaktır.

Örneğin, kullanıcı "ikizkenar üçgenler" araması yaptığında, bununla ilgili web sayfaları listelenirken, alternatif aramalar olarak "matematik ikizkenar üçgenler", "eşkenar üçgenler", "ikizkenar üçgenlerde açılar" gibi aramaların kullanıcıya sunulması faydalı olabilir. Şekil 1'de hali hazırda kullanılan sorgu önerme arayüzlerinin bazı çeşitlerine örnekler verilmiştir. Söz konusu şeklin bir kısmı, literatürde "sorgu tamamlama" *(query autocompletion)* olarak bilinen yardımcı araç olup, daha çok yazılmaya başlanılan sorgunun devamının tahmin edilmesine dayanır.



Şekil 1: Arama Motorlarında Sorgu Önermeye Örnekler

Sorgu önerme aynı zamanda, gerek arama motorunun kullanıcı ihtiyacını anlamaya yönelik, gerekse kullanıcının işini kolaylaştırmaya yönelik bir "yardımcı araç *(assistive feature)*" olarak da görülebilir (Parikh, Singh, & Sundaresan, 2013). Arama motorları, kullanıcı isteklerini karşılamak ve kullanıcının bilgi ihtiyaçlarını, niyetlerini anlamak için bu tür yardımcı araçlar kullanabilirler.

b) Katkılar

Sorgu önerme hakkında yapılan bu tez çalışması ile bu alandaki yöntemler üzerinde fikir yürütülmüş, geliştirme, gerçekleştirme ve deneyler yapılarak aşağıda özetlenen belirgin gelişmeler sağlanmıştır:

- "Sorgu Önerme" problemi yeniden tanımlanmış ve iki sorgunun karşılaştırılmasına indirgenmiş, basitleştirilmiştir.
- Modüler ve genişletilebilir bir sorgu önerme çerçeve mimarisi önerilmiştir. Bu mimari ile yeni ve birden çok sorgu önerme algoritması kolayca eklenebilir ve birleştirilebilir.
- Literatürdeki genel amaçlı bir sorgu önerme algoritmasının performansı, ilk ve orta öğretim öğrencileri tarafından kullanılan gerçek bir eğitim arama motoru sorgu kayıtları üzerinde ölçülmüştür ve algoritmada yapılan bazı değişikliklerle performansı arttırılmıştır, yeni algoritmaların performanslarıyla karşılaştırılmıştır.
- Birden çok sorgu önerme algoritmasının pratik ve modüler bir şekilde birleştirilebileceği ve bunun performansı artıracağı gösterilmiştir.
- Sorguların oturum *(session)*, tıklama sayısı, frekansı v.b. özellikleri yanında eğitimle ilgili sorgunun sınıfı ve dersi özelliklerini kullanan yeni sorgu önerme algoritmaları geliştirilmiştir. Ayrıca sorgu kayıtlarından elde edilen sorgu ve tıklanan belgeler grafiğindeki yolları dikkate alan özgün yol frekans algoritmaları önerilmiştir. Geliştirilen algoritmalar önerilen çerçeve mimari içine entegre edilmiş ve kullanılmıştır.
- Yeni algoritmaların birleşimi olan hibrid algoritmalar önerilmiş ve bunlar bahsedilen çerçeve mimari içine entegre edilmiş; hibrid algoritmalarda skor temelli birleştirmenin daha başarılı olduğu gösterilmiştir.
- Yeni algoritmaların gerçeklemesi yapılmış, bunlarla deney çalışması gerçekleştirilmiş, önerilen yeni algoritmaların temel alınan algoritmalara göre

başarım artışları ölçülmüş ve yapılan ölçümler çeşitli metriklerle doğrulanmıştır.

c) Ana hatlar

Bu çalışmanın ana hatları, sorgu önerme probleminin çözümüne yönelik Teori ve Yöntem önerisi ile bu yöntemlere dayanılarak gerçekleştirilen Deneyler'den oluşmaktadır. Bu kapsamda, tez çalışması şu bölümlerden oluşmaktadır: Temel Bilgiler ve İlgili Çalışmalar bölümünde sorgu önerme konusunda gerekli olan temel bilgilere yer verilmiş ve literatürde daha önce yapılan çalışmalardan bahsedilmiştir. Teori ve Yöntem bölümünde öncelikle sorgu önerme probleminin nasıl indirgendiğinden ve yeniden tanımlandığından bahsedilmiştir. Sorgu önerme probleminin modüler, pratik ve genişletilebilir bir mimaride ele alındığı, sorgu önerme çerçeve mimarisi hakkında bilgi verilmiştir. Ayrıca, gerçekleştirilen ve önerilen sorgu önerme algoritmaları sunulmuştur. Deneyler bölümünde, gerçekleştirilen algoritmalarla oluşturulan deney ortamından, bunların değerlendirilmesinden ve istatiksel sonuçlarından bahsedilmiştir. Son bölümde çalışma sonuçlandırılmış ve ilerde yapılabilecek araştırmalar hakkında fikirler ve öneriler sunulmuştur.

BÖLÜM I

TEMEL BİLGİLER ve İLGİLİ ÇALIŞMALAR

Bu bölümde arama motorlarında sorgu önerme problemi hakkında temel bilgiler verilmiş ve bu konuda literatürde yapılmış çalışmalar hakkında bilgi verilmiştir. Önerilen ve yeni geliştirilen yöntemlere geçmeden önce ilgili kavramlar ve konu hakkında bilgi verilmesi daha önce yapılan çalışmaların incelenmesi faydalı olacaktır.

1.1. TEMEL BİLGİLER

Arama motorları temel olarak "Bilgi Erişimi *(Information Retrieval)*" biliminin bir parçası olarak görülebilir. İlerleyen bölümlerde, konu ile ilgili temel bilgiler ve kavramlar verilmiştir.

1.1.1. BİLGİ ERİŞİMİ *(INFORMATION RETRIEVAL)*

Bilgi erişimi, çok geniş anlamlar ihtiva edebilir ve herhangi bir ortamda yer alan veriye erişim, söz konusu verinin alınmasıdır. Veri alınacak ortamlar çok çeşitli olabilir. Dijital ortamlar (elektronik depolar, bilgisayar diskleri/sunucuları, çeşitli sinyal verileri v.b.), ya da fiziki ortamlar (kütüphanelerdeki kitaplar, fiziki depolarımızdaki kayıtlar, yazılı veriler v.b.) bunlara örnek olarak verilebilir. Söz konusu veri alındıktan sonra içerdiği veri işlenerek çeşitli formatlara dönüştürülebilir ve yeni bilgiler elde edilebilir. Bu geniş anlamla düşünüldüğünde örneğin, cebimizdeki cüzdandan bir kredi kartını alıp üzerindeki numaranın okunması bile bir Bilgi Erişimi olarak görülebilir (Christopher D. Manning, Prabhakar Raghavan, & Hinrich Schütze, 2009).

Özellikle dijital teknolojilerin ve bilgisayar ağlarının yaygınlaşması ile bilgiye

erişim çok önemli bir hal almış ve sürekli kullanılır olmuştur. Hayatımızın hemen her anında, dijital/bilgisayar ortamlarında üretilen verilere erişim ve bunların işlenmesi ile çok büyük boyutlarda bilgi oluşmuş ve bu bilgilerin erişilmesi, işlenmesi, değerlendirilmesi çok önemli hal almıştır.

1.1.2. WEB BİLGİ ERİŞİMİ *(WEB INFORMATION RETRIEVAL)*

Web Bilgi Erişimi, özellikle İnternet (web) üzerindeki verilere erişim kastedilerek kullanılan, Bilgi Erişiminin bir alt konusu olarak görülebilir. Bir üstteki bölümde anlatıldığı üzere, Bilgi Erişimi'nin alanı (çeşitlilik açısından) çok geniştir. Halbuki Web Bilgi Erişimi'nde hedef veriler İnternet üzerinde bulunmaktadır. İnternet üzerindeki verilerin elde edilerek depolanması, işlenmesi ve üzerinde çeşitli işlem ve sorgular yapılması ile bu verilerden faydalanmak mümkün olmaktadır. İnternet Arama Motorları bunun en belirgin örneğidir. Arama motoru olmayan, ancak internet üzerindeki verileri alarak bir şekilde depolayan çeşitli servisler de (web crawler, web arşivleyici v.b.) buna dahildir.

İnternet Arama Motorları, Web Bilgi Erişimi'nin en belirgin ve yaygın örneğidir. Bu, aynı zamanda, internet üzerindeki verilere teknik ya da bilim insanı olmayan normal kullanıcıların da erişimini mümkün ve kolay kıldığından dolayı, web bilgi erişimine ayrı bir boyut kazandırmıştır.

1.1.3. İNTERNET ARAMA MOTORLARINA GİRİŞ

İnternet (dünya çapında ağ, world wide web) 90'larda önemli bir bilgi kaynağı olduktan bir süre sonra arama motorları, internetteki bilgiye erişim kolaylığını artırmak amacıyla ortaya çıkmaya başlamıştır. Kullanıcılara ilgili web sayfalarının listesini gösterebilmek amacıyla, arama motorları, internet üzerindeki kaynakları gezinerek içerdiği bilgilerin özetini ve indeksini oluşturur. Kaydedilen ve indekslenen bu veriler kullanılarak, kullanıcıların ihtiyaçlarına uyan bilgi kaynakları

gösterilir. Daha sonraları arama motorları kullanıcılara, sordukları bilgiyle ilgili ek bilgiler, veriler sunmaya başlamışlardır. Kullanıcının sorduğu ilk sorguyla ilgili olabilecek diğer "sorgular" da bu ek bilgilerden olup, genellikle "sorgu önerme" *(query suggestion)* olarak adlandırılırlar.

Arama motorlarını kullanan kullanıcıların arama alışkanlıklarını ve davranışlarını incelemek önem arz etmektedir (Jansen & Spink, 2006). Kullanıcıların yaptığı sorgular çoğunlukla kısa, yetersiz ve yanlış yazılabilmektedir (Cui et al., 2003). Kullanıcılar arama yaptığı konuda yeterince bilgiye de sahip olmayabilirler. Özellikle çocuklara ve öğrencilere yönelik arama motoru tasarımı yapıldığında bu durum önem arz etmektedir (Torres, Hiemstra, Weber, & Serdyukov, 2012). Bu tarz genç ve öğrenciler aradıkları bilgi ile ilgili sorgularını hazırlamakta, yazmakta ya da değiştirmekte zorlanmaktadırlar (Babuscu & Özcan, 2014). Bu sebeple, arama motorlarının kullanıcılara ilgili olabilecek ek bilgiler sunmaları oldukça faydalı olabilir.

1.1.3.1. İnternet Arama Motorlarının Çeşitleri

İnternet üzerinde arama imkanı sunan başlıca arama motoru çeşitleri şunlardır:

1. Genel amaçlı web arama motorları,
2. Dikey arama motorları (bir amaca yönelik),
    a) Ticari ürün (emlak, oto, ikinci el v.b.) arama motorları,
    b) Fiyat arama motorları,
    c) İş/eleman arama motorları,
    d) Eğitim arama motorları,
    e) Her tür eşyalar/cihazlar için arama motorları (internete bağlı tüm eşyaların arama motoru, webcam, ev, cihaz v.b. *(internet of things)*; Örnek: shodan.io),

3. Meta arama motorları

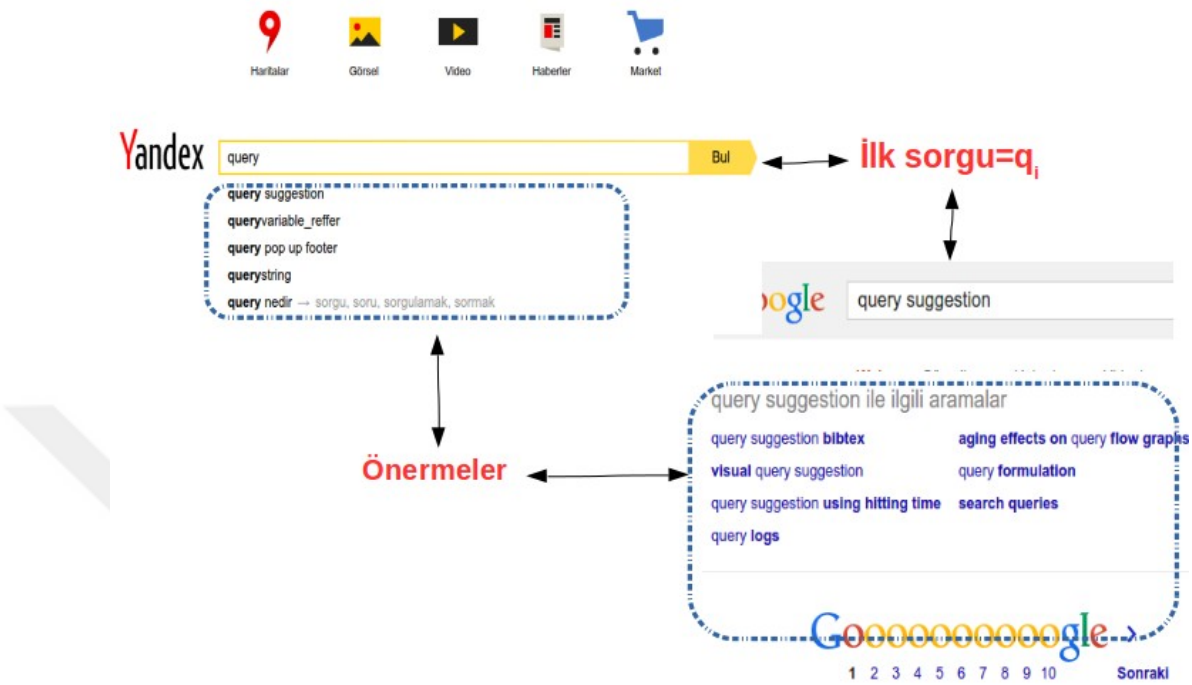
1.1.4. ARAMA MOTORLARINDA SORGU ÖNERME İLE İLGİLİ TEMEL KAVRAMLAR

Arama motorlarında sorgu önerme probleminde başvurulan temel bazı kavramlar örnek olarak aşağıda verilmiştir. Burada verilen bilgi kaynaklarına ek olarak, kullanılan arama motorunun üretebileceği başka veriler de kullanılabilir.

1.1.4.1. Sorgu, İlk Sorgu, Sorgu Önerme, Arama Sonuçları, Doküman/URL

Kullanıcının arama motorunda yazarak arama yaptığı metne "sorgu" adı verilir. Kullanıcılar aradığı konu hakkında tam bilgi sahibi olmasa bile arama motorlarına birtakım anahtar kelimeler yazarak "sorgu"lar yaparlar. Örneğin bir kullanıcı "uçak bileti" ifadesini arama motoruna yazarak bu konuda bilgi sahibi olmak ister. Bir arama motorunun birincil fonksiyonu olarak, kullanıcıya aradığı konu ile ilgili bilgiler içeren doküman/URL bağlantı bilgileri sunulur ki bunlara "arama sonuçları" denir.

Birçok kullanıcının girdiği sorgular, arama motorunun veri deposunda kaydedilerek sorgu kayıtlarını oluşturur. Bir kullanıcı açısından düşünüldüğünde, o kullanıcının girdiği sorgu "ilk sorgu" $(q_i=q_{initial})$ olarak değerlendirilebilir. Bu ilk sorgu'dan yola çıkılarak, kullanıcının ilgisini çekebilecek, işini kolaylaştırabilecek başka sorgular arama motoru tarafından (örneğin bu tez çalışmasında önerdiğimiz gibi) kullanıcıya gösterilir. Arama motoru tarafından kullanıcıya gösterilen bu ek sorgulara "sorgu önerme" adı verilir. Sorgu önermeler çoğunlukla, (bir kullanıcının girdiği) ilk sorgu'dan yola çıkılarak, diğer kullanıcıların da yaptığı sorgulardan da yararlanılarak yapılır. İlk sorgu ve sorgu önermeleri Şekil 2'de örnek halinde gösterilmiştir. Bu konuda ilerleyen bölümlerde bilgi verilmiştir.



Şekil 2: İlk Sorgu ve Sorgu Önermelerine Örnek

Kullanıcının bir sorgu yapmasını müteakip arama motoru tarafından kullanıcıya hem arama sonuçları hem de sorgu önermeleri gösterilir (başka bilgiler de sunulabilir). Kullanıcıya gösterilen arama sonuçları direk doküman bilgisi/linki olabileceği gibi, başka internet site adresi de olabilir. Her iki durumda da kullanıcıya gösterilen arama sonucuna URL denebilir. Kullanıcı bu arama sonuçlarından birisine tıkladığında söz konusu "sorgu"dan yola çıkarak, tıkladığı doküman/URL'ye gitmiş olur ve o sorgu ile doküman/URL arasında bir çeşit ilişki oluşmuş olur.

1.1.4.2. Sorgu Kayıtları *(Query Logs)*

Kullanıcılar bir arama motorunu kullandıklarında, yaptıkları tüm aramalar ve hatta tüm hareketler arka plandaki işlemler tarafından çoğunlukla kayıt altına alınır. Kullanıcının aradığı terimler, tıklamaları, sırayla ve hangi zamanda hangi URL/dokümana tıkladıkları, mümkün olduğu kadar hedef URL/dokümanda ne kadar kaldıkları, kullandıkları bilgisayar ve internet bağlantısının özellikleri (IP v.b.) hep kayıt altına alınabilir. Bu kayıtlara kısaca "sorgu kayıtları" *(query logs)* diyebiliriz.

Çoğunlukla kullanıcılar tarafından yapılan sorgulardan daha fazla bilgi kaydedilir.

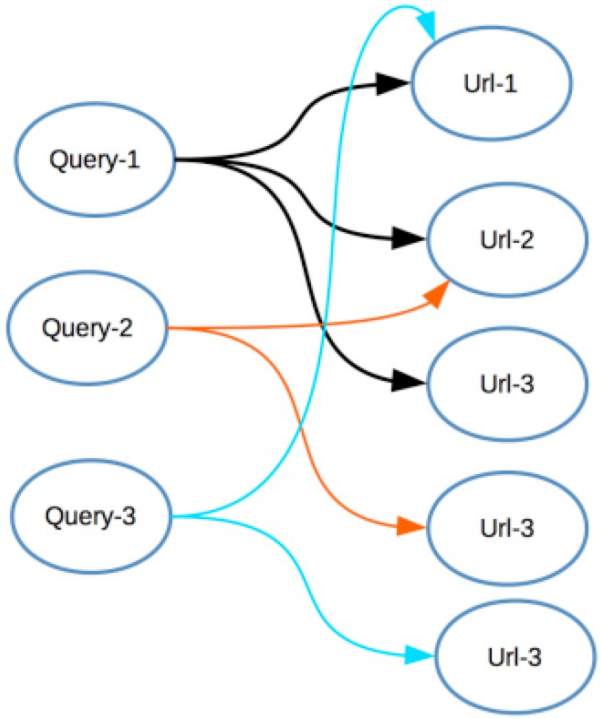
Sorgu kayıtları, söz konusu arama motoru ve sistemi tarafından çeşitli formatlarda, kayıt dosyası *(log)* ya da veritabanı (yapısal veya NoSQL) formatlarında kaydedilebilir. Daha sonra bu sorgu kayıtları, çeşitli işlemlerde ve araştırmalarda kullanılabilir ki, sorgu önerme de bunlardan biridir. Örneğin fiyat/ürün arama motorlarındaki uygun "ürün önerme" de yine bu sorgu kayıtları vasıtasıyla gerçek zamanlı veya diğer şekillerde yapılabilir.

1.1.4.3. Sorgu-Tıklama Grafiği *(Query Click Graph, Query-URL Bipartite Graph)*

Sorgu kayıtları üzerinde çalışan sorgu önerme algoritmalarının önemli bir bölümü bu sorgu kayıtlarını bir tıklama grafiğine dönüştürerek kullanırlar. Bu sayede sorgu ve URL/dokümanların görselleştirilmesi ve aynı zamanda ilişki kurularak algoritmalar üzerinde kullanılması daha pratik hale gelir. Şekil 3'de böyle bir grafik gösterilmiştir. Bu grafikte sorgulardan oluşan bir düğüm grubu, tıklanan URL/dokümanlardan oluşan bir düğüm grubu ile bunları birbirine bağlayan bağlantılar bulunmaktadır. Böyle bir grafiğe literatürde "sorgu-URL ikili grafiği" *(query-URL bipartite graph)* denmekte olup, bu tez çalışmasında kısaca "sorgu tıklama grafiği" olarak da kullanılmıştır. Sorgu tıklama grafiğinde genellikle sorgular sol tarafta, URL/dokümanlar sağ tarafta bulunur ve bunları birbirine bağlayan bağlantılar/yollar bulunmaktadır. Sorgu tıklama grafiğinde soldaki sorgulara ve sağdaki URL/dokümanlara "düğüm" de denir *(node)*. Düğümleri birbirine bağlayan bağlantılara da "kenar" *(edge)* denir.

Bir kullanıcı bir sorguyu arama motorunda arayıp onun sonuçlarına tıklayınca, o sorgudan kullanıcının tıkladığı URL ya da dokümana doğru bir bağlantı oluşur. Bu şekilde sorgu düğümleri ile URL/doküman düğümleri arasında bağlantılar oluşur. İlk başta basit gibi görünen bu bağlantılar, tüm grafik oluşturulduğunda çok anlam kazanır ve sorgular, URL/dokümanlar ve bunların kendi içinde bir takım ilişkilere işaret ederler. Oluşturulan bu grafiğin incelenmesi, değişik şekillerde gezilmesi ile

sorgu ve URL/dokümanlar arasında çeşitli bağıntılar ortaya çıkarılabilmektedir. Bölüm 2 ve Bölüm 3'te bu sorgu tıklama grafiği ve ilişkiler kullanılarak var olan ve geliştirilen algoritmalardan bahsedilmiştir.



Şekil 3: Sorgu-Tıklama İkili Grafiği (Query URL Bipartite Graph)

1.1.4.4. Doküman/Url Bilgileri ve İçerikleri *(Content)*

Arama motorları veri depolarına ekledikleri dokümanların ve web sitelerinin içeriklerini ve bazı meta-data'larını kaydederler. Meta-data, çoğunlukla "bilgi hakkında bilgi" *(data about data)* olarak adlandırılmakta olup, hangi bilginin nerede olduğunu tanımlar. Arama motorları, genellikle gezindikleri *(crawl ettikleri)* dokümanlar hakkında elde edebildikleri tüm meta-data ve içerikleri kaydederler. Bu veriler daha sonra kullanıcıya arama sonuçlarının ve sorgu önermelerinin sunulmasında (sonuçlarda sayfa özetinin verilmesi gibi) kullanılır.

1.1.4.5. Çapa Metni *(Anchor Text)*

İnternetteki sitelerin bağlantılarında, bağlantının içeriğini tanımlayan çoğunlukla kısa ama tanımlayıcı bir metin kullanılır ve buna çapa metni *(anchor text)* denir. Çapa metni, çoğunlukla verilen linkin içeriğini tanımladığından dolayı ilgili sayfalar hakkında (o sayfanın içeriği çekilmemiş olsa bile) bir fikir verebilir. Bu sebeple arama motorları tarafından gerek arama sonuçlarının sunulmasında gerekse sorgu önermelerin gösterilmesinde kullanılabilmektedir.

1.1.4.6. Kullanıcı Bağlamı ve İlgili Bilgileri

Arama motoru kullanıcısının yaş, sınıf, ülke, IP adresi, ID'si, bağlamı *(user context)* gibi kullanıcı hakkında çeşitli bilgiler sunan veriler de arama motorları tarafından kullanılabilmektedir. Bu bilgiler, sunulacak içeriklerin (arama sonuçları, sorgu önermeler v.b.) kişiselleştirilmesinde, daha uygun içeriklerin sunulmasında kullanılmasıyla, daha başarılı bir deneyim ortaya çıkabilmektedir. Örneğin belirli bir yaşın altındaki kullanıcıya onun anlayamayacağı içeriklerin sunulması çok faydalı olmayacaktır.

1.2. SORGU ÖNERMEDE YAKLAŞIMLAR VE İLGİLİ ÇALIŞMALAR

Sorgu önerme literatüründe çalışmalar temel olarak iki bölüme ayrılabilir (Bhatia, Majumdar, & Mitra, 2011). Yakın zamanlardaki çalışmaların birçoğu (Arora & Duhan, 2013; Baeza-Yates, Hurtado, & Mendoza, 2005; Bordogna, Campi, Psaila, & Ronchi, 2012; Cao et al., 2008) sorgu önermelerini üretebilmek için sorgu kayıtlarını kullanmaktadır. Diğer bazı çalışmalar (Bhatia et al., 2011) bir sorgu kaydına ihtiyaç duymaz ve web sayfalarının içeriklerinde analiz yaparak sorgu önermeleri üretmeye çalışmaktadır.

Sorgu önermelerini üretmek için kullanıcı bağlam (context) bilgilerini kullanan çalışmalar da vardır (Cao et al., 2008; Huang, Chien, & Oyang, 2003). Kullanıcı

bilgileri (yaş, cinsiyet, kullanıcıadı, IP adresi, araçları v.b.s), kullanıcının girdiği önceki sorgular ve elde edilebilecek diğer bazı bilgiler kullanıcı bağlam bilgisi olarak düşünülebilir.

Sorgu önerme ilk başlarda "sorgu kümeleme" ve "terim kümeleme" ("Query Clustering", "Term Clustering") şeklinde ortaya çıkmıştır (Lewis & Croft, 1990). Daha sonraki çalışmalar (Fonseca, Golgher, De Moura, & Ziviani, 2003; Wang & Zhai, 2008) sorgu kayıtlarını *(query log)* inceleyerek sorgular arasındaki ilişki kurallarını tesbit etmeye çalışmaktadırlar. Birçok çalışma (Baeza-Yates et al., 2005; Mei, Zhou, & Church, 2008; Wen, Nie, & Zhang, 2001) ilgili sorguları ve dokümanları belirlemek için arama kayıtlarından sorgu-tıklama ikili grafiğini *(bipartite graph)* oluşturmaya ve sorgular arasındaki bağlantıları ve ilgileri belirlemeye çalışmaktadırlar. Burada oluşturulan ve kullanılan sorgu-tıklama ikili grafiği, yapılan sorguları temsil eden bir grup düğüm, tıklanan URL ya da dokümanları temsil eden diğer bir grup düğüm ve sorgudan URL/dokümana doğru bir tıklama olması durumunda arasındaki bağlantıdan oluşmaktadır. Bu tez çalışmasında bundan sonra bu grafiğe "sorgu tıklama grafiği" *(query click graph)* denilmiştir.

"Hitting Time" algoritması (Mei et al., 2008) bu grafiği kullanarak ilgili sorguları bulan yöntemlere önemli bir örnektir. Bu algoritma kullanıcının ilk girdiği sorgu ile (sorgu önermede kullanılacak) aday sorgular arasındaki geçiş ihtimallerini hesaplayarak sorgu önerme yapmaya çalışmaktadır. Bu ihtimali hesaplarken, sorgular ile URL/dokümanlar arasındaki tıklama frekanslarını kullanmaktadır. Bu algoritma, buradaki çalışmamızın sonuçlarını karşılaştırmakta kullandığımız ve gerçekleştirdiğimiz bir yöntem olup, detayları Bölüm 2.3.1'de anlatılmıştır.

Sorgu tıklama grafiğine alternatif olarak, (Boldi et al., 2008) sorgu kayıtlarını tarayarak "sorgu akış grafiğini" (query-flow graph) önermiştir. Bu grafik sorgular arasında ağırlığı olan bağlantılar içermektedir. Buradaki ağırlık fonksiyonu sorgular arasındaki içerik, oturum ve zamanla ilgili özelliklerden oluşturulmaktadır. Sonraki bir çalışmasında (Boldi, Bonchi, Castillo, Donato, & Vigna, 2009) sorgu önerme için,

bu sorgu akış grafiği üzerinde rastgele yürüyüşe *(random walk)* dayalı bir yöntem önermektedirler.

(Wen et al., 2001) sorguları içeriklerine ve tıklanan dokümanlara göre kümelemekte ve bu kümeleri de ilgili sorguları bulmakta kullanmaktadır. Sorgu kayıtları sorgu-tıklama bilgilerinden daha fazla bilgi de içerebilir. Sorgu sıklığı/frekansı, sorgu zamanı, sonuç sayıları v.b. ek bilgiler de sorgu kayıtlarında bulunabilir ve sorgu önermede kullanılabilir (Silvestri, 2010). Bu tez çalışmasında da bu özelliklerin birçoğu kullanılmıştır.

Kümeleme yöntemi *(clustering)* (Meng, Huang, & Gu, 2014) çalışmasında da kullanılmıştır. Söz konusu çalışma, benzer sorguları seçmek için sorguların terimlerini ve tıklanan dokümanların özelliklerini kullanmış, aynı zamanda bu iki özelliği/yöntemi bir katsayı ile lineer olarak basitçe bir arada kullanmanın yolunu da önermiştir.

Sorgu önermede önemli diğer bir metod da içerik analizidir. (Bhatia et al., 2011; Kraft & Zien, 2004) çalışmasında sorguların, dokümanların, doküman özetlerinin veya çapaların *(anchor text)* içerikleri benzer sorguların bulunmasında kullanılmıştır. Bu yöntemler aynı zamanda, sorgu tıklama kayıtlarının olmadığı durumlarda da kullanılabilir.

Sorgu önermenin diğer temel bir çeşidi de "sorgu genişletme"dir *(query expansion)*. Kullanıcının girdiği sorguya ilgili bazı terimlerin eklenmesi birçok durumda faydalı olabilir (Cui et al., 2003) ve diğer yöntemlere göre uygulaması daha basit olabilir.

Sorgu önerme, kullanıcıların kendileri tarafından yapılan "sorgu düzeltme"nin diğer bir çeşidi olarak da görülebilir. Diğer bir tabirle, sorgu önerme, arama motoru tarafından otomatik olarak yapılan ve kullanıcıya hazır şekilde sunulan bir "sorgu düzeltme" *(query reformulation)* olarak değerlendirilebilir.

Literatürde sorgu düzeltmeler "genelleştirme *(generalization)*", "özelleştirme *(specialization)*", "hata düzeltme *(error correction)*" ve "paralel hareket *(parallel*

move)" v.b. diye adlandırılabilecek farklı kategorilere ayrılmıştır (Silvestri, 2010). Bu sorgu düzeltme tiplerine "sözlük eşleştirme *(vocabulary match)*", "muğlaklığı giderme *(disambiguation)*" gibi kategorileri ekleyenler de olmuştur (Parikh et al., 2013). Yapılan sorgu tiplerinin araştırıldığı bir çalışmada (He et al., 2009) bunlara ek olarak "Tekrarlanan sorgular" ve "Eşanlamlı kullanımı" eklenmiştir. Bu çalışmalar çoğunlukla, kullanıcıların sorgu düzeltme ihtiyaçlarını anlamak ve çözümler üretmek adına yapılmıştır, zira kullanıcıların önemli bir kısmının sorguları değiştirme/düzeltme ihtiyacı hissettiği görülmüştür. Örneğin, büyük bir sorgu kaydının analizi neticesinde, kullanıcıların %46'sının sorgularını düzelttiği/değiştirdiği gözlenmiştir (Jansen & Spink, 2006). Kullanıcıların sorgularını nasıl düzelttiğine dair Bayesian ağına dayalı bir olasılık yaklaşımı çalışması da yapılmıştır (Lau, Tessa, Horvitz, 1999).

Genç kullanıcıların, çocukların ve öğrencilerin arama motorlarını kullanma alışkanlıklarına yönelik yeni çalışmalar (Torres, Weber, & Hiemstra, 2014; Usta, Altingovde, Vidinli, Ozcan, & Ulusoy, 2014) ve bu gençlere yönelik yeni teknik geliştirme çalışmaları vardır (Torres et al., 2012). Daha önce bahsedildiği gibi, genç kullanıcılar sorguları değiştirmede ve düzeltmede zorluk yaşmaktadırlar (Babuscu & Özcan, 2014) ve bu kullanıcılar için sorgu önerme ve sorgu tamamlama mekanizmaları daha büyük bir ihtiyaçtır. Bu tez çalışmasında, genellikle 5 ila 8. sınıflar tarafından kullanılan bir eğitim arama motorunun verileri kullanılmıştır.

(Torres et al., 2012) tarafından yapılan çocuklara yönelik sorgu önerme çalışmasının bu alanda ilklerden olduğu bilinmektedir. Öğrencilerin bilgiye erişim ihtiyaçlarına yönelik olarak yapılan bir çalışmada (Yılmazel, 2011) sorgu kayıtlarını baz alan bir sorgu önerme yöntemi, Türkiye'deki üniversite öğrencilerinin kullandığı uzaktan eğitim içeriklerinde kullanılan bir arama motorunda kullanılmıştır. Bir oturumda ne kadar beraber arandığına göre sorgular arasındaki benzerliklerin hesaplandığı (Apache Mahout aracı ile) işbirlikçi filtrelemeyi *(collaborative filtering)* temel alan bir yaklaşım kullanılmıştır. Söz konusu çalışmada, sadece sorgulara ait oturum bilgileri kullanılmış, eğitimle ilgili özellikler kullanılmamıştır. Bu tez çalışmasında ise, sorgu kayıtları, oturum bilgileri, kullanıcı bilgileri ve

eğitimle ilgili özelliklerin kullanıldığı algoritmalar ve bu algoritmaların beraberce kullanılmasına imkan veren modüler bir çerçeve mimari önerilmiştir.

BÖLÜM II

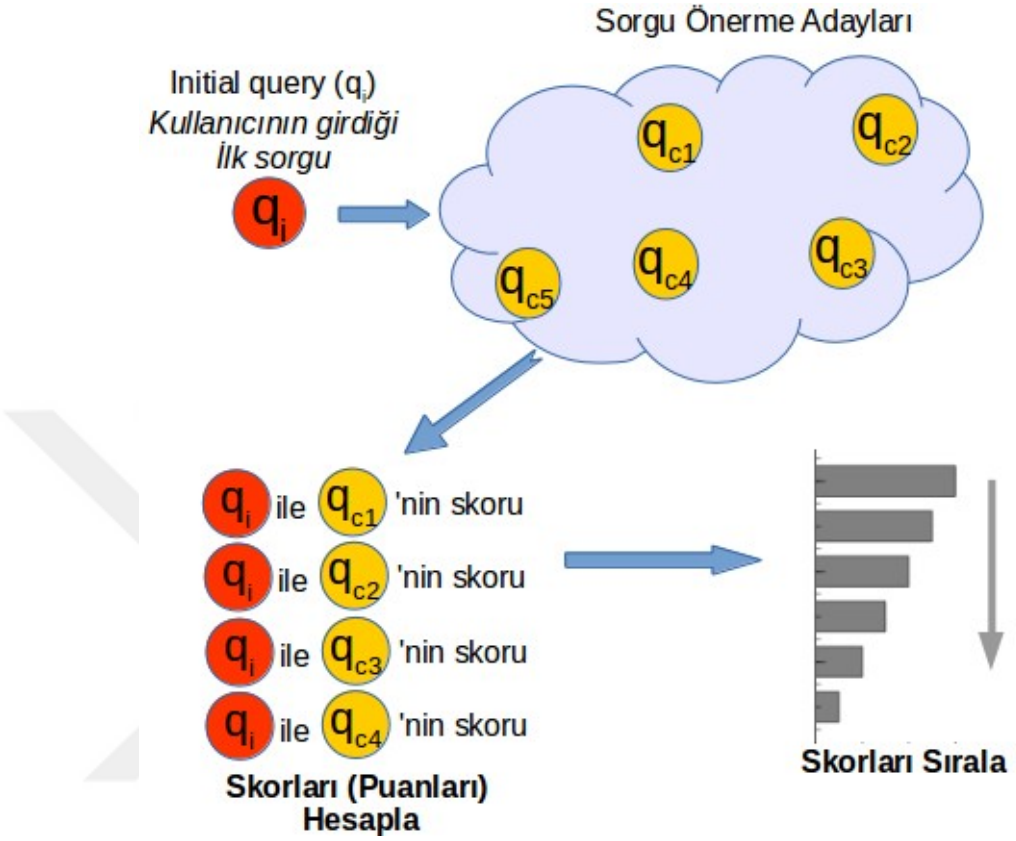
TEORİ ve YÖNTEM

2.1. SORGU ÖNERME PROBLEMİNİN İNDİRGENMESİ

Mevcut çalışmalar, belirgin algoritma ve yöntemler Bölüm 1'de incelenmiştir. Bu algoritmalara bakıldığında, yerine göre çok büyük olabilen sorgu-URL ikili grafiğinden yola çıkarak veya belgelerin/sorguların içerik analizini yaparak birtakım sorgu önermelerinin üretilmeye çalıştığı görülecektir. Bu tarz yaklaşımların hepsinde, üretilmesi beklenen sorgu önermelerinin verilen giriş bilgilerinden (sorgu-URL ikili grafiği, içerik v.b.) yola çıkılarak algoritmaya has yaklaşım ile belirli bir yöntem ve adım olmaksızın, çoğu zaman da karmaşık bir şekilde üretildiği görülmektedir. Bu algoritmaların kendi içinde bir yöntemi olsa da, sorgu önerme probleminin çözümüne yönelik belirgin, açık ve basit *(straightforward)*, genel geçer bir yöntemleri yoktur.

Yapılan çalışmalarda da hazırlık anlamında çeşitli algoritmalar üretilmiş ve bu sırada, sorgu önerme algoritmalarının geliştirilmesinde kullanılabilecek genel geçer bir problem indirgemesinin faydalı olacağı değerlendirilmiştir. Buna göre, sorgu önerme problemi, sadece "iki sorgunun karşılaştırılması" problemine indirgenebilir. Binlerce ya da milyonlarca sorgu kayıtlarına, sorgu-URL ikili grafiğine göre düşünüldüğünde, sadece iki sorgunun benzerliğini aramak ve ölçmeye çalışmak, iki sorguyu karşılaştırmak açık bir şekilde problemin kolaylaştırılmasını sağlamaktadır.

Şekil 4'de, söz konusu sorgu karşılaştırma yöntemi görselleştirilmiştir.



Şekil 4: Sorguların Karşılaştırılması Yöntemi

Bu yaklaşımın şu avantajları vardır:

- Sorgu önerme problemi, açık bir şekilde "iki sorgunun (ilk sorgu ve aday sorgu) karşılaştırılması" problemine indirgenmektedir,
- İki sorgu birçok basit, açık ve düzgün *(straightforward)* metodlar sayesinde karşılaştırılabilir,
- Birden çok sorgu karşılaştırma metodu kolaylıkla birleştirilebilir, (iki sorgu karşılaştırılırken, özellikleri ölçülürken)
- Bu yaklaşım kolayca geliştirilebilir bir mimaridedir,
- Bu yaklaşımla, var olan ve yeni metodların detaylarının gözlenmesi, hata

ayıklaması ve geliştirilmesinin daha kolay olması beklenir,

Bu problem indirgemesi sayesinde, bir geliştirici sadece iki sorgunun karşılaştırılmasına kafa yoracaktır. Şu da unutulmamalıdır ki, bu karşılaştırma sorguların önermede sıralanması içindir ve her zaman iki sorgunun benzerliği/ilgililiği anlamına gelmeyebilir, karşılaştırılacak sorguların diğer özellikleri de ölçülebilir. Örneğin aday sorguların ilk sorguya çok benzer olması durumunda, farklılaştırma *(diversification)* arzu edilen bir şeydir ve bu sefer ilgililik/benzerlik değil, çeşitli özellikleri ile farklılık ölçülebilir ve karşılaştırılabilir. Bu yöntem, sorgu önermedeki "farklılaştırma *(diversification)*" amacına yönelik olarak kullanılabilir.

Bu yöntem, diğer bilim dallarında da sıralamanın zaman zaman zor/karışık olduğu durumlarda ve problemlerde kullanılabilir. Literatürde "Sinanoğlu İndirgemesi" olarak bilinen yöntemde de (Sinanoglu, 1988), benzer şekilde bir problem/teoremin indirgenmesi amaçlanmıştır ki, bu tür problem indirgemeleri kendi alanlarında çok faydalı olabilmektedir.

Bu yaklaşım, (eklemeli sıralama gibi *(insertion sort)*) sayıların sıralı listedeki sayılarla karşılaştırıldığı sayı sıralama algoritmalarına da benzer olarak görülebilir. Bu tez çalışmasında, bu basit karşılaştırma yönteminin arama motorlarındaki sorgu önerme probleminde uygulanabileceği gösterilmiştir. Sorgu karşılaştırmanın nasıl yapıldığına, hangi özelliklerin kullanılabildiğine dair detaylı bilgiler, algoritma önerilerinin ve gerçeklemelerinin yapıldığı Bölüm 2.3'de verilmiştir.

Bu tez çalışmasındaki algoritmalarımızda temel olarak bu yaklaşım kullanılmış ve sorgu önerme algoritmalarının gerek tasarımında gerekse gerçekleştirilmesinde oldukça verimli olduğu gösterilmiştir. Bu yaklaşımla, sorgu önerme algoritmalarının geliştirilmesi ve birleştirilerek hibrid olarak kullanılması da mümkündür. Bu karşılaştırma metodu, sonraki bölümde önerilen çerçeve mimari yapıda da kullanılmıştır.

2.2. SORGU ÖNERME ÇERÇEVE YAPISI/MİMARİSİ

Bu bölümde, önerdiğimiz ve çalışmalarımızda başarıyla kullandığımız sorgu önerme mimari yapısı anlatılmıştır.

Bir önceki bölümde, sorgu önerme problemine en temelde bir yaklaşım sergilenmiş ve problemin indirgenmesine yönelik bir önermede bulunulmuştu. Bu, sorgu önerme problemine farklı bir bakış açısı ile yaklaşma ve çözümüne yönelik olarak basitleştirici bir önerme idi. Yaptığımız çalışmalarda, bu problem indirgemesine ek olarak sorgu önermenin daha sistematik, daha planlı ve geliştirilebilir yapılabilmesi için genel bir çerçeve yapının uygulanmasının faydalı olacağı görülmüş ve bu yönde çalışmalar yapılmıştır.

Sorgu önermenin geliştirilebilmesi için modüler, sistematik, basit ve aynı zamanda bir o kadar da efektif bir çerçeve yapı önerilmiştir. Bu çerçeve yapı, farklı sorgu önerme algoritmalarının da birlikte entegre edilmesiyle kolayca geliştirilebilecek şekilde modüler bir mimaride olup sorgu önerme probleminin çözümünü de kolaylaştırmakta, sistematikleştirmektedir. Farklı algoritmalar bu mimari içerisinde kolaylıkla kullanılabilir ve bu sayede de farklı algoritmalar birleştirilebilir veya karşılaştırılabilir.

Bu mimari, kısaca ana iki aşamadan oluşmaktadır: seç ve sırala. Bu iki ana aşama, sorgu önerme adaylarının seçilmesi ve sıralanmasıdır. Bu ana aşamalara göre nispeten daha küçük bazı aşamalar sorgu önermenin kalitesini artırmak için eklenebilir. Seçme sonrası genel kontrol aşaması ve sıralama sonrası son kontrol aşamaları bu amaçla eklenmiştir. Önerilen mimari yapı şu aşamaları içermektedir:

1. **Aday sorgu önermelerini belirle/seç (Ana/majör aşama)**
2. Genel Kontroller (İsteğe bağlı aşama)
3. **Seçilen sorgu önermelerinin bir veya daha fazla algoritma ile sıralanması (Ana aşama)**
4. Son kontroller (İsteğe bağlı aşama)

a) Sorgu değiştirme, genelleştirme ve farklılaştırma (Refinement, generalization, diversification) (isteğe bağlı aşama)

b) Yeniden sıralama, son işlemler (isteğe bağlı aşama)

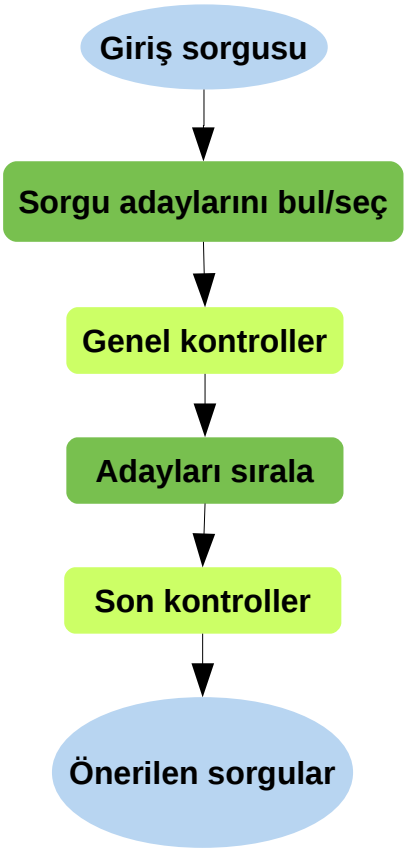
Bu yapının oluşturulmasında etken olan sebepler ve oluşan fikirler şu şekilde sıralanabilir:

- Sıralamadan önce, sorguların seçilmesi aşamasının ayrı bir adımda yapılması ve diğerinden ayrı düşünülmesinde fayda bulunmaktadır. Sorgu adaylarının seçilmesinde, grafik gezinme algoritmalarından DFS *(Depth First Search, Derinlemesine Arama)* veya BFS'den *(Breadth First Search, Genişlemesine Arama)* istifade edilebilir. Bu aday sorgusu seçme adımı, diğer adımlardan tamamen ayrı, farklı bir adımdır. Bu adımın amacı, muhtemel sorgu önermelerinin araştırılarak bulunmasıdır. En genel anlamda, eldeki tüm sorgu önermeleri (tüm giriş kümesi), aday sorgular olarak ele alınabilir *(aday sorgular=tüm sorgular)*, ancak bu durum çok yüksek işlem gücü gerektirecektir.
- Genel kontroller aşaması, majör bir aşama olmamakla birlikte, aday sorgular arasında olabilecek hatalı ve faydasız sorguların temizlenmesinde kullanılabilir. Çok kısa sorguların, çok uzun sorguların veya yanlış yazılan sorguların temizlenmesi buna örnek olarak verilebilir. Kullanıcının sorduğu ilk sorguya çok benzeyen sorgular da burada elenebilir.
- Hazırlanan “aday sorguların” sıralanması, sonraki majör aşamadır. Bu aşamada, şimdiye kadarki bilinen yöntemler kullanılabileceği gibi, yeni yöntemler de kullanılabilir. Bölüm 2.1'de bahsettiğimiz “sorgu karşılaştırma” ve Bölüm 2.3'deki teknikler gibi geliştirdiğimiz yeni yöntemler bu aşamada kullanılmıştır. Birden fazla metodun bu aşamada beraberce kullanılması da mümkün ve önerilmektedir. Çalışmalarımızda eğitim arama motorlarının özelliklerini (ders, sınıf v.b.) kullanan ve diğer sorgu özelliklerini kullanan

farklı yöntemler bu aşamada bir arada (hibrid bir şekilde) kullanılmıştır.

- Sıralama aşamasından sonra, en azından şimdilik majör bir aşama olmayan bir adım da, sorgu değiştirme, genelleştirme ve farklılaştırma gibi prosedürlerdir. Sıralama aşamasından sonra, normalde sıralamada en üst sıralarda bulunan aday sorgular kullanıcıya gösterilir. Ancak kullanıcıya göstermeden hemen önce, bahsedilen değiştirme, genelleştirme ve farklılaştırma işlemlerini, aynı zamanda son kontrolleri yapmak ve gerekirse yeniden sıralamak faydalı olabilmektedir.
- Oluşturulan bu çerçeve yapının en önemli bir özelliği, “Sorgu Önerme” için gerekli olan adımları parçalara bölmesi ve daha kolay çözülebilir problemler haline getirmesidir. Bu sayede eğitim arama motorlarında, sorgu loglarında bulunabilecek hemen her tür özellik, kolaylıkla sorgu önerme'de kullanılabilecektir.
- Sorgu sıralama aşamasında, “sorgu karşılaştırma” tekniği önerilmektedir. Bu teknik sayesinde, sıralama aşaması da kendi içinde kolaylaşmaktadır. Aynı zamanda, eğitim arama motorundaki özelliklerin birleştirilerek veya tek başına kullanılabilmesi mümkün olmaktadır. Çalışmalarımızda sorgu karşılaştırma yöntemi kullanılmıştır.

Şekil 5'de, çerçeve mimarimin genel hatları özet olarak gösterilmiştir. Önerdiğimiz sorgu önerme mimari yapısının bu aşamalarının detayları ilerleyen alt bölümlerde anlatılmıştır.

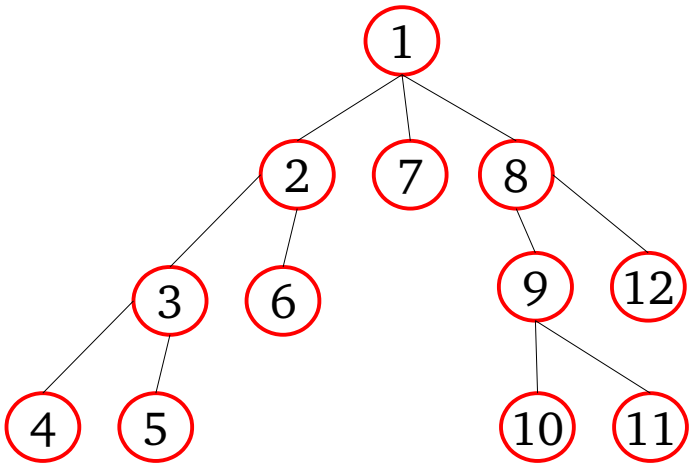


Şekil 5: Sorgu Önerme Çerçeve Mimarisi

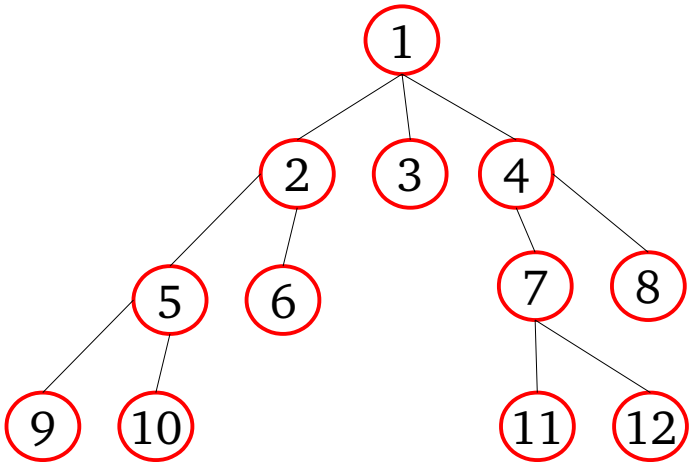
2.2.1. Sorgu Seçimi/Bulunması Aşaması

Bu aşama, sorgu önerme mimarimizin birinci ana aşamasıdır. DFS, BFS gibi bilinen yöntemlerle yapılabilir. Burada amaç, sorguların sıralanması aşamasında kullanılacak olan ya da, sorgu önermesi yapılacak olan "aday önermelerin" belirlenmesi, bulunmasıdır. Aday önermelerin bulunması tamamen ayrı bir aşama olup başka yeni yöntemlerle de belirlenebilir. Sorgu önerme mimarimizde baz alınan ve hedeflenen bir yaklaşım da "kavramların ayrılması"dır *(Separation of Concerns/Concepts)*. Bu yaklaşım sebebiyle, sorgu adaylarının seçilmesi aşaması diğer aşamalardan ayrı değerlendirilebilir.

Bu aşamada yeni yöntem önerilmemiş, var olan ikili ağaç gezinme yöntemlerinden DFS *(Depth First Search)* ve BFS *(Breadth First Search)* kullanılmıştır. Her ikisi de grafik teorisinde grafik gezinme yöntemleridir. Sorgu-url grafiği üzerinde yapılacak bir DFS veya BFS gezinme, aday sorguların bulunmasına imkan verecektir. Bir ağacın DFS ile gezinmesi Şekil 6'da, BFS ile gezinmesi Şekil 7'de örnek olarak gösterilmiştir.



Şekil 6: Örnek: Bir Ağacın DFS ile Gezilme Aşamaları



Şekil 7: Örnek: Bir Ağacın BFS ile Gezilme Aşamaları

Her ne kadar bu tez çalışmasında sorgu seçimi/bulunması için başka yöntemler kullanılmamışsa da, içerik analizi ya da var olan sorgu terimlerinin kullanılması ile yeni aday sorgularının önerilebileceği düşünülmektedir. Yapılan çalışmalarda BFS ile gezinme yönteminin ilgili sorguların bulunmasında daha başarılı olabileceği gösterilmiştir.

Sorgular DFS veya BFS ile seçilmek üzere grafik gezinirken, kullanıcının sorduğu ilk sorgu ile dersi bilgisi farklı olan sorgular listeye alınmamıştır. Bu işlem ilk sorgu ile muhtemel adayın ders bilgisi tesbit edilebiliyorsa yapılmış, ikisinden herhangi birinin dersi tesbit edilemiyorsa, aday sorgulara eklenmiştir. Ders bilgisinin tesbiti ile ilgili bilgi Bölüm 2.3.5.2'de detaylı bilgi verilmiştir.

2.2.2. Sıralama Öncesi Genel Kontroller

Yapılan çalışmalarda, kullanıcıların eğitim arama motoruna girdikleri sorgulardan bir kısmının hatalı, çok uzun ya da çok kısa olduğu görülmüştür. Bu sebeple, bir önceki ana adımda çeşitli yöntemlerle seçilen sorgu önerme adaylarının

sıralanmaya geçilmesinden önce, bazı temel kontrollerin yapılmasında fayda bulunmaktadır. Bu genel kontrol aşamasında, anormal olarak görülen bu sorguların sıralamaya geçmeden önce elenmesi amaçlanmıştır. Bu aşama majör bir aşama olmamakla birlikte, nihai sorgu önermelerin kalitesini yükselttiği görülmüştür. Burada bahsedilen kontroller, yapılacak daha fazla test, deney ve gözlemle geliştirilebilir, artırılabilir.

Yapılan çalışmalarda aşağıdaki temel kontrollerin yapılması faydalı görülmüş ve uygulanmıştır:

1. Çok sayıda kelime barındıran ya da çok büyük sorgular elenmiştir,
   a) 8 kelimeden uzun sorgular,
   b) 110 karakterden uzun sorgular,
   c) Bir kelimede 60'dan fazla karakter olan sorgular (çoğu hatalı sorgudur),
2. Çok küçük sorgular elenmiştir (3 karakterden küçük olanlar). Bunların genellikle hatalı sorgular olduğu görülmüştür,
3. Sorulan ilk sorgunun bir alt kümesi olan sorgular elenmiştir. Örneğin, kullanıcı “matematik üçgen şekiller” sorduysa, “matematik üçgen” şeklinde gelen sorgu önermeleri elenmiştir,
4. Çok genel sorgular, genel anlam ifade edenler elenmiştir.
   a) Bunun için bir “genel terimler” dizisi oluşturulmuş, bu diziye şu sorgular eklenmiştir: ['konu anlatımı', 'özet', 'vitamin oyun', 'sınıf', 'konular', 'tarih', 'sorular', 'soru', 'vitaminde oyunlar', 'konu anlatımı', 'konu anlatımları', 'konu özetleri', 'anlatımlı soru çözümleri', 'matematik oyunları', 'eğitici oyunlar', 'interaktif etkinlik', 'etkinlik', 'oyunlar', 'oyun', 'oyun oyna', 'fen', 'türkçe', 'turkce', 'matematik', 'sosyal bilgiler', 'sosyal', 'fen bilimleri', 'fen ve teknoloji', 'alıştırma', 'canlandırma', 'daha', 'araştırma', 'konu', 'etkileşimli', 'interaktif etkinlikler', 'alıştırmalar', 'interaktif', "vitamin'de ara", 'boy', 'int', 'anasayfa', 'ana sayfa', 'ana sayfa', 'ana sayfa', 'oyu', 'fen

oyun', 'ders', 'test çöz', 'test', 'mat', 'oyunla', 'özet', 'fen ve', 'test çöz', 'çalışmalar', 'fen bilgisi', 'konu anlatım', '].

b) Bu genel sorgular, yapılan çalışmalarda tesbit edilen ve çok genel veya anlamsız sonuçlar üretebilen sorgulardır.

5. İlk sorulan sorgunun dersi belliyse, sorgu adaylarındaki farklı derse ait sorgular elenmiştir. Bu sayede, eğitimle ilgili bir sorguda, farklı dersin önermelerinin kullanıcıya gösterilmesi önlenmiştir.

   a) Örneğin, kullanıcı “matematik üçgen açılar” sorduysa, kendisine tarihle ilgili sorguların önerilmesinin büyük ölçüde önüne geçilmiştir.

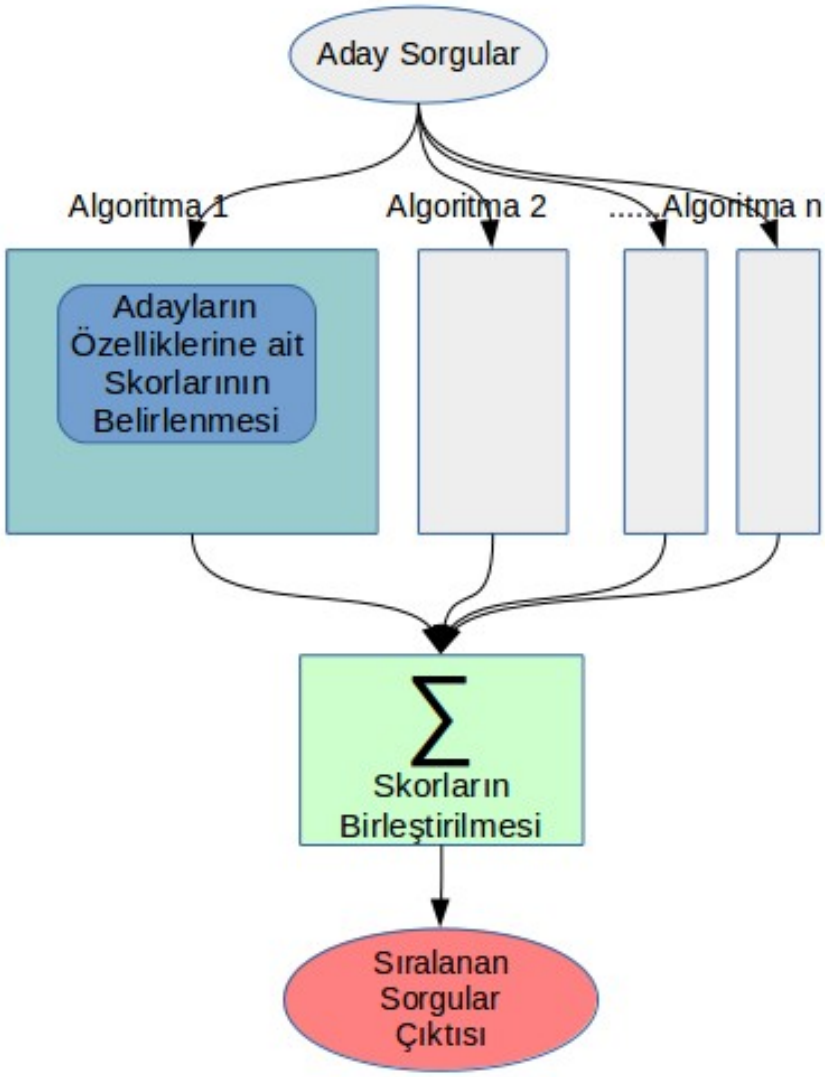
   b) Eğer ilk sorgu veya aday sorguların ders bilgisi belirlenemediyse, o zaman bu eleme yapılmamıştır.

Yapılan bu çalışma ile birtakım hatalı ya da çok genel sorguların nasıl elenebileceği gösterilmiş olup, buna benzer kural ve işlemler, yapılacak testler ve deneylerle arttırılabilir.

2.2.3. Sıralama Aşaması

Çerçeve mimarinin sıralama aşaması, olmazsa olmaz ana bir aşama olup, bu aşamaya gönderilen aday sorgu önermelerinin bir veya daha fazla algoritma kullanılarak sıralanması amaçlanmıştır. Bu aşamada birden çok algoritmanın skorları/sıraları birleştirilebilmektedir.

Öncelikle alt metodlara göre skorlar belirlenmeli, daha sonra bunların çeşitli yöntemlerle sıralaması yapılmalıdır. Sıralama aşamasının genel yapısı Şekil 8'de gösterilmiş, sıralamaya ait ilerleyen bölümlerde detaylar verilmiştir.



Şekil 8: Sorgu Önerme Çerçeve Mimarisi Sıralama Katmanı

2.2.3.1. Alt Katmanda Bir Özelliğe Göre Skorların Belirlenmesi

Çerçeve yapı'da sıralama yapılabilmesi için, öncelikle sorgulara yönelik çeşitli özelliklerin skorlarının belirlenmesi gerekmektedir. Skorlar belirlendikten sonra, bir sonraki bölümde anlatıldığı üzere bir veya daha fazla alt algoritma birleştirilerek bir sıralama yapılabilir. Sorgulara ait sorgu frekansı, oturum sayısı (Bölüm 2.3) v.b.

çeşitli özelliklerin skorları sorgu kayıtlarından direk olarak ya da çeşitli hesaplamalarla belirlendikten sonra, bir sonraki bölümde anlatıldığı üzere birleştirilebilecektir. Bu tez çalışmasında, kullanılabilecek özellikler ve bunların hesaplanma yöntemleri Bölüm 2.3'de daha detaylı anlatılmıştır.

2.2.3.2. Çerçeve Yapı'da Sıralama

Bu bölümde, önerilen "Sorgu önerme çerçeve yapısının" sıralama aşaması anlatılmıştır. Çerçeve yapının bir aşaması olan sıralama katmanı, diğer bölümlerde (Bölüm 2.2.3.1, Bölüm 2.3 v.b.) anlatılan diğer sıralama algoritmalarını kullanarak birleştirmektedir. Bu sebeple, alt katmanlardaki sıralama algoritmalarının detayları bu bölümde gösterilmemiş, çerçeve yapının sıralama aşamasında alt katmanlardaki sıralamaların nasıl birleştirildiği anlatılmıştır.

Sıralama aşamasının alt basamakları şöyle özetlenebilir:

1. Sorgu önerme adaylarını alt katman algoritmalarına göndererek skorlarını belirle veya sıralamalarını yaptır (bir önceki bölümde anlatılan aşama),
2. Alt katmandan gelen sıralama/skor sonuçlarını V_1, V_2... şeklinde vektörler halinde al.
3. Gelen V_1, V_2... vektör verilerini, Bölüm 2.2.3.2.1'de gösterilen sıralama birleştirme algoritmalarından biriyle birleştirerek nihai sıralamayı oluştur.

Alt katmanlarda (algoritmalarda) yapılan sıralamalar V_1, V_2 ... şeklinde vektörlerde saklanarak buraya gönderilir. Örneğin, kullanıcının "matematik üçgen şekiller" ilk sorgusunu (q_i=$q_{initial}$) eğitim arama motorunda girdiğini varsayalım. Bu durumda, bu katmana gelen aday sorgu önermeleri ve bunlara karşılık üretilen sıralama skorları vektörleri aşağıdaki gibi olabilir (bir alt metod için örnektir, başka şekillerde de vektörler oluşturulabilir):

$$\text{Aday Sorgular} = \begin{bmatrix} \text{matematik şekiller} \\ \text{üçgen şekiller} \\ \text{üçgenler matematik} \\ \text{üçgenler açılar} \end{bmatrix} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\Rightarrow V_1 = \begin{bmatrix} q_1 ile\, q_i nin\, benzerlik\, skoru \\ q_2 ile\, q_i nin\, benzerlik\, skoru \\ q_3 ile\, q_i nin\, benzerlik\, skoru \\ q_4 ile\, q_i nin\, benzerlik\, skoru \end{bmatrix} = \begin{bmatrix} Sq_1 \\ Sq_2 \\ Sq_3 \\ Sq_4 \end{bmatrix}$$

Burada belirtilen V_1 vektörü, alt katmandaki tek bir sıralama algoritması tarafından üretilen bir vektördür. Ana sıralama aşaması, birden fazla sıralama algoritmaları içerebilir. Birden fazla sıralama algoritması tarafından üretilen sıralama skorları, V_1, V_2, V_3 gibi vektörlerde ayrı ayrı saklanabilir. Bu durumda, ayrı ayrı hesaplanan sıralama vektörleri aşağıdaki Bölüm 2.2.3.2.1'deki "sıralama birleştirme yöntemleri *(rank aggregation methods)*"nden biriyle birleştirilebilir.

Birden fazla algoritmanın birleştirildiği bu yönteme, hibrid sıralama da diyebiliriz. Bu sıralama yöntemi ve çerçeve yapı ile, birden fazla, hatta çok sayıda sıralama algoritmasının kolayca ve efektif bir şekilde birleştirilmesi mümkündür. Aynı zamanda, farklı kişiler ya da gruplar tarafından geliştirilebilecek olan sıralama algoritmalarının hem karşılaştırılması hem de ortak bir çerçeve yapıda kullanılması mümkün hale gelmiştir. Bu sayede, "Sorgu Önerme" konusunun tamamına hakim olmayanların da bu konuya algoritma üretmeleri mümkün hale gelmiştir.

2.2.3.2.1. Sıralama Birleştirme Yöntemleri *(Rank Aggregation Methods)*

Sıralama birleştirme yöntemleri en azından iki kategoriye ayrılabilir; skor temelli birleştirme ve sıra temelli birleştirme (Renda & Straccia, 2003). Başka sıralama birleştirme yöntemleri bu aşamada de önerilebilir ya da kullanılabilir. Bu çalışmamızda bu iki çeşit (toplamda 4 tane) birleştirme yöntemleri gerçeklenmiş ve test edilmiştir.

2.2.3.2.1.1. Ağırlıklı Skor Temelli Birleştirme Yöntemleri *(Weighted Score Based Aggregation)*

Bu çalışmamızda, farklı algoritmalardan (alt metodlardan) elde edilen skorların birleştirilmesine, bu sayede farklı metodların başarımlarının kullanılmasına çalışılmıştır. Bunun için akla gelen ilk ve en basit hali şudur (dezavantajları aşağıda belirtilecektir):

$$\text{Birleştirilmiş Sıralama Vektörü} = V_1 + V_2 + \ldots + V_n = \sum_{i=1}^{\text{algoritma sayısı}} V_i$$

Yapılan çalışmalarda farklı alt algoritmalardan gelen skorların aralıklarının birbirinden çok farklı olduğu gözlenmiştir. Örneğin bir algoritma 0-10 arasında değer üretebilirken, bir başkası 0-1000 arası değer üretebilir. Bu durumda, birden fazla algoritma vektörünün toplanması anlamını yitirecek, küçük aralıklı algoritmanın bir etkisi kalmayacaktır. Bu sebeple, her bir alt algoritmadan gelen skor vektörünün normalizasyonunun alınması düşünülmüştür; bu durumda söz konusu birleştirme formülü şu hal alacaktır:

$$\text{Birleştirilmiş Sıralama Vektörü} = Norm(V_1) + Norm(V_2) + \ldots + Norm(V_n)$$

$$= \sum_{i=1}^{\text{algoritma sayısı}} Norm(V_i)$$

Aynı zamanda yapılan testlerde ve gözlemlerde, bazı alt algoritmaların diğerlerine göre daha başarılı olabileceği gözlenmiş, alt algoritmaların başarım oranlarını dikkate alabilmek için her bir algoritmaya, kendi başarım oranına göre bir katsayı verilebileceği değerlendirilmiştir. Normalizasyonu alınmış her bir vektörün aynı zamanda bir katsayı ile yeniden ölçeklenmesi ile, başarılı olduğu bilinen algoritmaların sonuç üzerindeki etkisinin artırılması hedeflenmiştir. Bu durumda birleştirilmiş skorların, farklı algoritmalardan elde edilen skorların ağırlıklı olarak toplanması ile, aşağıdaki formülle elde edilebileceği önerilmiş/kullanılmıştır:

$$\text{Nihai Sıralama Vektörü} = k_1 \cdot Norm(V_1) + k_2 \cdot Norm(V_2) + \ldots + k_n \cdot Norm(V_n)$$

$$= \sum_{i=1}^{\text{algoritma sayısı}} k_i \cdot Norm(V_i)$$

Burada belirtilen Norm fonksiyonu, her bir sıralama alt algoritmasından gelen değerleri 0 ile 1 arasında yeniden ölçekleyen bir fonksiyon olup, şöyle formülize edilebilir:

$$Norm(V) = \{ \frac{x}{max(V)} \mid \forall x \in V \}$$

Farklı sıralama algoritmalarından gelen verilerin yeniden ölçeklenmesi ile, hem bu alt algoritmalar birbirleriyle karşılaştırılabilir olmuş, hem de nihai sıralama tek bir alt-algoritma sonucu ile fazla etkilenmemiştir. Özellikle bazı alt sıralama algoritmalarının üretebileceği çok yüksek ya da hatalı değerler göz önüne alınırsa, bu durum önemli bir normalizasyon sebebi olarak ortaya çıkmaktadır. Yapılan bu işlem ile, tek bir algoritmadaki hata ya da hatalı/yanlı değer sebebiyle genel sıralama fazla bozulmamaktadır. Bazı algoritmaların farklı giriş verilerinde farklı tepkiler verebileceği düşünülürse, bunun faydası daha iyi anlaşılmaktadır. Yaptığımız çalışma ve testlerde (Bölüm 2.3, Bölüm 3 ve diğerleri), farklı sıralama algoritmalarının farklı veri kümelerinde, sorgu tipine, sorgu loglarına göre farklı başarı oranları yakaladığı görülmüştür. Yapılan bu normalizasyon çalışması ile, bu farklar da yumuşatılmakta, hatalar azaltılmaktadır.

Bazı algoritmalardaki yüksek skorlar, logaritmik bir normalizasyon uygulayarak da giderilebilir. Bu durumda, söz konusu formül şu hal alır:

$$Nihai\ \ Sıralama\ \ Vektörü = k_1 \cdot Norm(\log(V_1)) + k_2 \cdot Norm(\log(V_2)) + \ldots + k_n \cdot Norm(\log(V_n))$$

$$= \sum_{i=1}^{algoritma\ sayısı} k_i \cdot Norm(\log(V_i))$$

Tüm skor vektörlerinin logaritmik olması da şart değildir. Bu, alt sıralama/değerlendirme/skorlama algoritmalarının ürettiği veriye göre değişir. Eğer

üretilen veriler çok değişkenlik arz ediyorsa, normalizasyon fonksiyonuna ek olarak logaritmik skala uygulamak faydalı olabilir. Yapılan testlerde, çok değişik sonuçlar üretebilen sıralama algoritmaları için logaritmik hesaplama yöntemi uygulanmıştır.

Bu formülde kullanılan k_1,k_2 … katsayıları, söz konusu algoritmalardan bazılarının etkisini bilerek ve belli ölçü içerisinde artırmak veya azaltmak için kullanılabilir. Varsayılan olarak bu katsayılar 1 alınabilir. Yapılan testlerde, bazı algoritmaların sonuçlarda daha etkili olduğu gözlendiğinden dolayı katsayıları yüksek tutulurken, bazı algoritmalar için düşük tutulmuştur. Katsayılar aynı zamanda, bir sıralama algoritması havuzu oluştuğunda aynı kod altyapısı ile algoritmaların bazılarını sıfırlayarak yani etkisini yok ederek sistemin denenmesini ya da kullanılmasını mümkün hale getirmişlerdir. Örneğin Bölüm 3.3'deki çalışmalarda ilk başlarda kullanılan katsayılar toplam 13 adet sıralama algoritması için [-3, 2, 2, 1, 0.5, 0.5, 1, 2, 0, 0, 1, 4, 8]'dir. Bunların detayları Bölüm 3.3'de verilmiştir.

Bölüm 3.3'deki testlerde bazı katsayılar belirlenmiştir. Ancak bazı algoritmaların farklı giriş kümeleri üzerinde daha iyi sonuçlar üretebildiği, ama aynı algoritmaların farklı durumlarda aynı başarıyı elde edemediği gözlenmiştir. Yani aynı ve sabit katsayılar kümesinin, tüm sorgu önermeleri için kullanılması her zaman mantıklı olmayabilir. Bu sebeple, oluşturulacak "Eğitim arama motoru sorgu önermelerinin (SÖ)" bir kısmının bir katsayı seti tarafından belirlenmesi, diğer bir kısmının ise başka bir katsayı seti tarafından belirlenmesi faydalı olabilecektir. Örneğin ilk 8-10 sorgu önermesi için bir katsayı seti, kalan 10-15. sorgu önermesi için de farklı bir katsayı kullanılabilir. Bu şekilde, kaliteli sorgu önermelerinin ilk 15 sorgu önermesinde (SÖ) bulunması şansı artırılmış olmaktadır. Bununla ilgili geliştirilen ve önerilen formül şöyledir:

$$\textit{Birinci grup SÖ için Nihai Skor Vektörü} = FSV_1 = \sum_{i=1}^{\textit{algoritma sayısı}} k_{1i} \cdot Norm(V_i)$$

$$\textit{İkinci grup SÖ için Nihai Skor Vektörü} = FSV_2 = \sum_{i=1}^{\textit{algoritma sayısı}} k_{2i} \cdot Norm(V_i)$$

(logaritmik hesaplamanın Norm fonksiyonu içine taşındığı varsayılmıştır.)

Buradaki k_{1i} katsayıları birinci grup katsayılar, k_{2i} katsayıları da ikinci grup katsayılardır. FSV_1 vektörü, ilk 8-10 sorgu önermenin (SÖ) seçilmesi için kullanılacak, FSV_2 vektörü de kalan 2-6 sorgunun seçilmesi için kullanılacaktır (Toplam 10-15 SÖ gösterileceği varsayılırsa). Eğer her iki kümede de var olan SÖ'ler varsa, onların birleştirilerek tek bir küme gösterilmesi faydalı olacaktır. Bu yöntem ile, farklı katsayılardan oluşan iki farklı hibrid metod birleştirerek daha sağlıklı sonuçlar elde edilebilir. Bu yöntemin faydası olabileceği düşünülmekle birlikte, yapılan çalışma ve testlerde bu şekilde iki ayrı katsayı kümesi kullanılmamış, ileride yapılabilecek bir çalışma olarak bırakılmıştır.

2.2.3.2.1.2. Sıra Temelli Birleştirme Yöntemleri *(Rank Based Aggregation)*

Bu birleştirme yöntemlerinde, alt yöntemlerin oluşturduğu sonuçların skorları yerine, sıralamaları (rank) kullanılır. Bu tez çalışmasında, Borda Count, Weighted Borda Count ve Weighted Voting metodları ile birleştirmeler gerçeklenmiş ve testleri yapılmıştır.

2.2.3.2.1.2.1. Borda Count

Borda Count metodu (Borda, 1781; Dwork, Kumar, Naor, & Sivakumar, 2001) alt yöntemlerin sonuçlarını, her bir sonucu, bulunduğu pozisyona göre ağırlıklandırarak birleştiren bir metoddur. Her bir alt metodun sonuçlarının puanı/skoru hesaplama örneği ve formülü Tablo 1'de gösterilmiştir. Her bir alt metoddaki sonuçların puanı bu tabloya göre hesaplanır, sonra bu puanlar toplanarak, toplam puana göre tekrar bir sıralama yapılarak nihai sonuç kümesi oluşturulur. Bu metod, yapılan çalışma ve testlerde Hibrid-2-Borda adıyla gösterilmiştir.

Tablo 1: Borda Count Puanlama Formülü Hesaplama Örneği

Alt Metod Sıralaması	Sorgu Önerme Adayları	Formül (n=aday sayısı)	Puan
1	query 1	n	5
2	query 2	n-1	4
3	query 3	n-2	3
4	query 4	n-3	2
5	query 5	n-4	1

Tablo 1'de gösterilen formülün daha genel hali şöyle ifade edilebilir:

$$Puan_i = \text{Sonuç Sayısı} - sira_i + 1$$

2.2.3.2.1.2.2. Weighted Borda Count

Bu, bir önceki Borda Count metodunun hafif değiştirilmiş halidir (Aslam & Montague, 2001). Borda Count metoduna ek olarak, her bir alt metod ayrıca bir katsayı ile ağırlıklandırılarak hesaplanır. Bu, Bölüm 2.2.3.2.1.1'deki ağırlıklı skor temelli birleştirme yöntemine benzer şekilde katsayılarla yapılır. Bu yöntemin hesaplama örneği ve formülü Tablo 2'de gösterilmiştir. Bu yöntem de denenmiş, başarımı ölçülmüş, sonuçlar Bölüm 3'de verilmiştir. Bu tez çalışmasında verilen figür ve tablolarda bu yöntem “Hibrid-3-Weighted Borda Count” olarak isimlendirilmiştir.

Tabloda örneği gösterilen formülün daha genel hali şöyle ifade edilebilir:

$$Puan_i = k \cdot (\text{Sonuç Sayısı} - sira_i + 1)$$

burada k=alt metodun katsayısıdır.

Tablo 2: Weighted Borda Count Puanlama Formülü Hesaplama Örneği

Alt Metod Sıralaması	Sorgu Önerme Adayları	Alt Metod katsayısı (ağırlığı)	Formül (n=aday sayısı)	Puan	Örnek k=2 için Puan
1	query 1	k	k*n	k*5	10
2	query 2	k	k*(n-1)	k*4	8
3	query 3	k	k*(n-2)	k*3	6
4	query 4	k	k*(n-3)	k*2	4
5	query 5	k	k*(n-4)	k*1	2

2.2.3.2.1.2.3. Simple Voting

Simple Voting metodu, diğerlerine göre çok daha basit bir metod olup, bir sorgu adayı, alt metodların sonuçlarında sadece varsa bir puan verilir. Yani sorgu adayının alt metodun sonuçlarındaki sırası dikkate alınmaz. Bu yöntem, bizim çalışmalarımızda kullanılmamıştır.

2.2.3.2.1.2.4. Weighted Voting

Bir önceki Simple Voting yönteminin biraz gelişmiş hali olup, her alt metoda ayrıca bir ağırlık verilir. Çalışmalarımızda bu yöntem de test edilmiştir.

2.2.3.2.2. Sıralama Birleştirme Yöntemlerinin Seçimi

Bu çalışmanın amacı, sıralama birleştirme yöntemlerini ölçmek ya da kıyaslamak değildir. Ancak çerçeve yapı içerisindeki hibrid metodlarda birden fazla alt metodun birleştirilebilmesi için uygun bir birleştirme yönteminin seçilmesi gerekmektedir. Yapılan ilk testlerde, skor temelli birleştirme yönteminin performansı gözlenmiştir. Skor temelli birleştirmenin faydası literatürde de belirtilmiştir; (Renda & Straccia, 2003) çalışmasında, bir meta arama motorunda skor temelli birleştirme metodlarının daha başarılı olduğu belirtilmiştir.

Skor temelli birleştirme yönteminin, sıra temelli birleştirme yönteminden daha başarılı olduğu söylenebilir. Skorların var olduğu, ya da hesaplanabildiği durumlarda skor temelli birleştirme yönteminin daha başarılı olduğu deney sonuçlarında da görülmüştür. Sadece, skorların hesaplanamadığı, alt metodlardaki sıralamaların elde var olduğu durumlarda sıra temelli birleştirme tercih edilmelidir. Skorların bulunduğu/hesaplanabildiği durumlarda ise, daha hassas hesaplamalar yapılabildiğinden dolayı, skor temelli birleştirme tercih edilmelidir. Önerilen sorguların skorları, ayrı bir bilgi boyutu olarak da görülebilir. Tablo 3'de görülen sıralı listeyi göz önüne alalım,

Tablo 3: Örnek Sorgu Önermeleri ve Skorları

Sıra	Sorgu Önermesi	Skor (bir metoda göre)
1	American airline	0.9
2	Airline tickets	0.2
3	Airline bus	0.1

Bu sorgu önermelerinin değerlendirilmesi, başka bir metodla birleştirilmesi için sadece sıralamalar kullanılırsa, ikinci sıradaki sorgu önerme adayının değeri birinci sıradakine çok yakındır. Ancak, bu sorgu önermelerinin skorları varsa (ya da hesaplanabiliyorsa) o zaman ikinci sıradaki önermenin değeri birinci sıradakinden çok daha azdır. Eğer sadece bir tane metod kullanılıyorsa, o zaman bunun bir önemi yoktur, zira ister sıralama pozisyonu ile, ister skor ile bakılsın sıralama değişmemektedir. Ancak birden fazla alt metodun birleştirilmesi söz konusu ise (özellikle çok sayıda metod varsa) alt metodlardaki sorgu önermelerinin skorları, nihai sıralamayı çok etkilediklerinden dolayı çok önem arz eder hale gelmektedir. Bu fikirden yola çıkılarak çerçeve yapımızda “sorguların karşılaştırılması” yöntemi önerilmiş ve sorgu önermelerinin skorları öncelikli olarak kullanılmıştır. Bölüm 3'te gösterildiği üzere skor temelli birleştirme yöntemi ile daha başarılı sonuçlar elde edilmiştir.

2.3. ÖNERİLEN/GERÇEKLEŞTİRİLEN SORGU ÖNERME/SIRALAMA ALGORİTMALARI

Yapılan çalışmalarda, eğitim arama motorunda kullanılabilecek olan sorguların özellikleri araştırılmış, üzerinde algoritmalar geliştirilmiş ve bulunan bu algoritmalar önerilen Sorgu Önerme (SÖ) Çerçeve Yapısı'nda kullanılmıştır. Bulunan ve var olan sorgu özellikleri, detayları ilgili bölümlerde belirtilen yöntemlerle "değer"lendirilmiş, rakamsal olarak ifade edilmiştir. Rakamsal olarak bulunan sorgu özelliklerinden yola çıkılarak sorgu karşılaştırmaları yapılmış ve nihai olarak da Sorgu Önerme amaçlı sıralamalar üretilmiştir.

Sorguların karşılaştırılması için, sorgulara bir değer atfedilmesi, rakamsal olarak sorguların belirtilmesi gerekmektedir. Tek başına bir sorgunun rakamsal bir değeri olabileceği gibi, önerilecek sorgu ile (SÖ ürettiğimiz) ilk sorgunun arasındaki bir karşılaştırma değeri de kullanılabilir. Sorgulara rakamsal olarak değer vermek için eğitim arama motoru kayıtlarından aşağıdaki özellikler çıkarılmış ve kullanılmıştır:

1. Sorguların Hitting Time skorları, (Mei et al., 2008)
2. Sorguların oturum sayıları *(session count)*,
3. Sorguların oturumlardaki yakınlıkları *(session proximity)*,
4. Sorguların tıklanma sayıları *(number of clicks)*,
5. Sorguların sorgulanma miktarı *(query count)*,
6. İki sorgunun (öğretimdeki) sınıf bilgisinin benzerliği *(grade similarity)*,
7. Sorgu sonuçları sayısı (Örneğin, çok nadir ya da yanlış bir sorgu, çok az veya sıfır sonuç döndürebilir),
8. Kaç kullanıcının o sorguyu tıkladığı *(query click count)*,
9. Tıklanan url/dokümanda harcanan zaman *(dwell time)*,
10. Sorgudan sorguya geçişteki yol frekansları (Bölüm 2.3.3) *(path-frequency*

scores).

Bu özelliklerin bazıları müteakip bölümlerde daha detaylı anlatılmıştır. Bu algoritmaların çoğunda Bölüm 2.2'de bahsedilen çerçeve yapı ve "sorgu karşılaştırma" tekniği kullanılmıştır. Sorgu değerleme ve karşılaştırma, Bölüm 2.1'de belirtildiği gibi ilk sorgu *(q_i=q initial)* ile aday sorgular arasında yapılmıştır. Sorgu karşılaştırmada, yukarıda bahsedilen özellikler ile, bulunacak başka karşılaştırma yöntemleri de kullanılabilir. İlerleyen bölümlerde, bu çalışma kapsamında karşılaştırmada kullanılan yöntemler anlatılmıştır.

Elimizde bir "Aday Sorgu Önermeleri" bulunduğundan dolayı, her bir algoritmanın Sorgu karşılaştırma değer vektörü şöyle olabilir:

$$Aday \quad Sorgular = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\Rightarrow Değer \quad Vekötrü = V = \begin{bmatrix} q_1 ile \;\; q_i \;\; sorgusunun \;\; karşılaştırması \\ q_2 ile \;\; q_i \;\; sorgusunun \;\; karşılaştırması \\ q_3 ile \;\; q_i \;\; sorgusunun \;\; karşılaştırması \\ q_4 ile \;\; q_i \;\; sorgusunun \;\; karşılaştırması \end{bmatrix} = \begin{bmatrix} Sq_1 \\ Sq_2 \\ Sq_3 \\ Sq_4 \end{bmatrix}$$

Burada bulunan vektör, Bölüm 2.2.3'de belirtilen nihai sıralamanın hesaplanmasında kullanılabilecektir. Bu yöntemin örnek ve detayı ilerleyen bölümlerde verilmiştir.

2.3.1. Hitting Time

Bu alt bölümdeki gerçekleştirmeler, tamamen ilgili çalışmada (Mei et al., 2008) önerilen yöntemler baz alınarak yapılmış, geliştirilen diğer yöntemlerle karşılaştırmak ve desteklemek için kullanılmıştır.

Söz konusu çalışmada belirtilen modelde, V_1 adındaki sorgular kümesi, V_2 adındaki URL/dokümanlar kümesi ve bunları bağlayan E adındaki kenarlar

kümesinden oluşan bir G grafiği tanımlanmıştır. Eğer V_1 kümesindeki bir i sorgusundan V_2 kümesindeki k dokümanına/url'sine bir tıklama varsa, i'den k'ya bir "kenar" vardır denilir ve bu kenara i'den k'ya yapılan tıklama sayısınca bir sayısal değer verilerek w(i,k) diye gösterilir (w=weight, ağırlık, frekans, sayı). Bu şekilde, tüm eğitim arama motoru logu kullanılarak, ya da en azından bir kısmı kullanılarak bir G grafiği oluşturulur. Bu grafikte, hangi sorgulardan hangi dokümanlara tıklandığı ve ne kadar tıklandığı bilgisi bulunacaktır.

Kullanıcının eğitim arama motoruna girdiği sorgudan (q_i=q initial) yola çıkılarak ve G grafiği DFS (Depth First Search, derinlemesine dolaşım) metoduyla gezilerek bir alt-grafik oluşturulur. Alt grafik oluşturulurken, önceden belirlenen, limitli bir sayıdaki sorgu alınır (aksi takdirde sonsuz ya da çok büyük bir alt-grafik oluşacaktır). Bu durumda, söz konusu makale/çalışmaya göre, V_1 kümesindeki bir i sorgusundan V_2 kümesindeki bir j dokümanına gitme/ulaşma ihtimali şu şekilde tanımlanmıştır:

$$p_{ij} = \sum_{k \in V_2} \left(\frac{w(i,k)}{d_i} \frac{w(k,j)}{d_k} \right)$$

veya, fonksiyonel olarak:

$$p_{ij} = \sum_{k \in V_2} \left(\frac{w(i,k)}{d_i} \frac{w(k,j)}{d_k} \right) = \sum_{k \in V_2} F(i,j,k)$$

$$burada: \; F(i,j,k) = \frac{w(i,k)}{d_i} \frac{w(k,j)}{d_k}$$

bu formüllerdeki d_i ve d_k şöyle tanımlanmıştır:

$d_i = \sum_{j \in V_2} w(i,j)$ yani: i'inci sorgu'dan çıkan ve tüm dokümanlara giden tıklamalar toplamı,

$d_k = \sum_{i \in V_1} w(i,k)$ yani: tüm sorgulardan çıkan ve k'ıncı dokümana giden tıklamalar toplamı,

V_1: sorgular kümesi,

V_2: doküman/URL kümesi

w(i,k): i'inci sorgudan k'ıncı doküman/url'ye giden tıklama toplamı, $i \in V_1, k \in V_2$

Söz konusu çalışma/makalede önerilen hesaplamaya göre, V_1 kümesindeki her bir sorgu için şu yinelemeli hesaplama yapılır:

$$h_i(t+1) = \left(\sum_{j \neq s} p_{ij} h_j(t) \right) + 1$$

s: ilk sorgu (q_i)

Bu yinelemeli hesabı, belirli bir döngü adedince çalıştırdıktan sonra, bulunan değerler küçükten büyüğe doğru sıralanarak kullanıcıya sunulur. Burada, küçük değerliler en üstte gösterilir.

Bu hesaplamalar, söz konusu makalede önerilen hesaplamalar olup, çalışmalarımızda karşılaştırma yapmak ve hibrid metodda kullanılmak üzere gerçeklemesi yapılmıştır. Söz konusu makalenin başında bu hesap normalde olasılık hesaplamaları ile yapılmış, ancak bu hesaplardaki matris boyutları V_1 ve V_2 ile doğru orantılı olacağından ve çok yüksek iş gücü gerektireceğinden dolayı basitleştirilerek yukarıdaki yinelemeli basit hesap yöntemi önerilmiştir.

Normalde orijinal makalede grafikte gezinme yöntemi olarak DFS kullanılmıştır. Yapılan çalışmalarımızda ise, hem DFS hem de BFS yöntemleriyle çalışılmış ve test edilmiş, sonuçları da Bölüm 3'deki test sonuçlarında gösterilmiştir. Bu test sonuçlarında, yeni geliştirilen yöntemlerin başarım oranları görülmektedir.

2.3.2. Oturum Bilgisini Kullanan Algoritmalar

Bu algoritmalar, eğitim arama motorunun sorgu loglarından üretilen oturum bilgilerini kullanmaktadır. Sorgu kayıtlarındaki aynı kullanıcıdan gelen ve aralarında en fazla 30 dakika süre bulunan sorgular bir oturum kabul edilmiştir. Oluşturulan oturum bilgileri veritabanında yeni bir tabloya kaydedilerek burada belirtilen

algoritmalarda kullanılmıştır. Oturum bilgisi bir defalığına tüm kayıtlar için üretilmiş ve sonrasında veritabanından hazır olarak kullanılmıştır.

2.3.2.1. Oturum Sayısı Algoritması

Bu algoritmada, ilk sorgu ile aday sorguların her birinin aynı oturum içinde var olduğu oturumlar sayılır. Yani, her bir aday sorgu için, o sorgu ile ilk sorgunun beraberce içinde bulunduğu oturum sayısı bulunmalıdır. Daha önce belirtildiğine benzer şekilde, dört tane aday sorgu olduğunu varsayalım (normalde çok daha fazla olur). Bu durumda şu şekilde bir "aday sorgular" matrisi ve Değer Vektörü oluşturulur:

$$Aday\ Sorgular = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

$$\Rightarrow Değer\ Vektörü = V = \begin{bmatrix} q_1\ ve\ q_i\ beraber\ var\ olan\ oturum\ sayısı \\ q_2\ ve\ q_i\ beraber\ var\ olan\ oturum\ sayısı \\ q_3\ ve\ q_i\ beraber\ var\ olan\ oturum\ sayısı \\ q_4\ ve\ q_i\ beraber\ var\ olan\ oturum\ sayısı \end{bmatrix} = \begin{bmatrix} Sq_1 \\ Sq_2 \\ Sq_3 \\ Sq_4 \end{bmatrix}$$

Bu vektör, direk olarak Bölüm 2.2'de belirtilen çerçeve yapıda ve hesaplamasında kullanılabilir. Normalde, aday sorgu sayısı, işlem kapasitesine de bağımlı olarak 4'den çok daha yüksek olur. Yapılan çalışmalarda ve testlerde 300 olarak alınmıştır. Bu aynı zamanda, var olan eğitim arama motoru logu boyutu ile de doğru orantılıdır. Daha detaylı bilgi Bölüm 3'de verilmiştir.

Algoritma sözde kodu şöyledir:

```
sessions ← determine_user_sessions_from_query_logs()
FOR query IN Candidate_Query_Suggestions (CQS)
      // İçinde q_initial ve query ikisi birden bulunan oturumları bul
      count ← 0
```

```
        FOR session IN Sessions:
                IF query and q_initial IN session:
                        count ← count + 1
                END IF
        END FOR
        list ← list + count
END FOR
SORT resulting list in decending order
RETURN TOP N RESULTS
```

Bu algoritma, bir sonraki bölümde anlatılan oturum yakınlığı *(session proximity)* algoritmasına göre çok daha basit olup, sadece oturumlardaki ortak sorgu adetlerini sayar. Oturum yakınlığı algoritması kullanıldığında, burada anlatılan oturum sayısı algoritması daha az hassas ve gereksiz kalmaktadır. Bu metod, sorgu önerme çerçeve yapısının ve sorgu karşılaştırma tekniğinin kullanılışını en temel olarak göstermek amacıyla test edilmiş ve gösterilmiştir. Burada, her bir aday sorgu için rakamsal bir değer üretilmektedir. Önerilen sorgu karşılaştırma tekniğinde buna benzer şekilde aday sorguları ile ilk sorgunun rakamsal karşılaştırmaları baz alınmaktadır.

2.3.2.2. Oturum Yakınlığı Algoritması

Bu algoritma, bir önceki bölümde bahsedilen oturum sayısı algoritmasının biraz geliştirilmiş ve hassaslaştırılmış halidir. Öncekinde iki sorgunun (aday sorgu ve ilk girilen sorgu) aynı oturum içerisinde geçmesi aranırken, bunda ise aynı zamanda bu iki sorgu arasındaki uzaklık ölçülmekte, böylece sorguların birbirleriyle olan benzerlik ya da ilgileri tahmin edilmeye çalışılmaktadır. Aynı oturum içerisinde birbirine daha yakın olan iki sorgunun daha ilgili olabileceği varsayımından hareket edilmiştir.

Şekil 9'da, oturum yakınlığı algoritmasının ana fikri temsili olarak gösterilmiştir. Aşağıda da algoritmanın sözde kodu verilmiştir. Söz konusu şekildeki

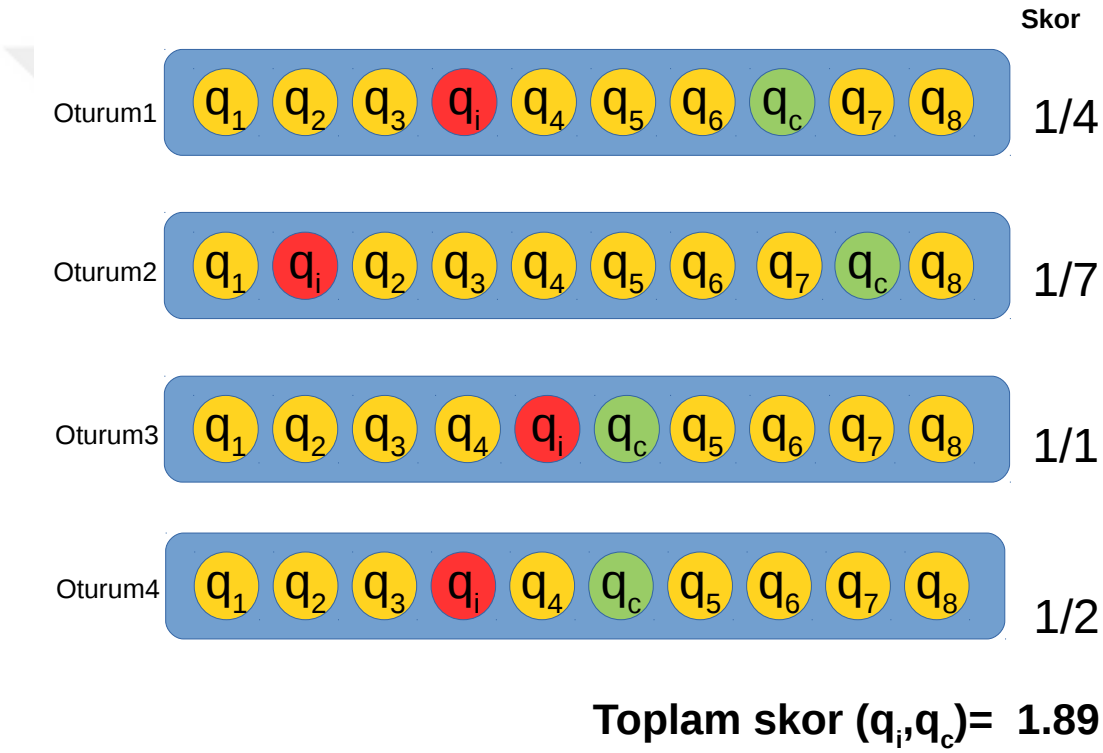
kısaltmalar şunları ifade etmektedir:

q_1,q_2,q_3, ... : İlgili oturumdaki sorgular (q_i ve q_c'yi de içerir)

q_i ($q_{initial}$): Kullanıcının arama motoruna girdiği ilk sorgu,

q_c: Sorgu önerme adaylarından birisi

(Bu hesap, her sorgu önerme adayı için yapılmaktadır)



Şekil 9: Oturum Yakınlığı Algoritması Temsili Gösterim

Algoritmanın sözde kodu şöyledir:

```
list ← empty list []
Sessions ← determine_user_sessions_from_query_logs()
FOR query IN Candidate_Query_Suggestions (CQS)
       score ← 0
       FOR session IN Sessions:
              IF session have both q_initial and query in it:
                     distance ← Find distance of q_initial and query in this
```

```
session
                    score ← score + 1/distance
            END IF
        END FOR
        list ← list + score
END FOR
SORT resulting list consisting of score elements
RETURN TOP N RESULTS
```

Oturum yakınlığı algoritmasını aşağıdaki hesaplamalarla da ifade edebiliriz:

Sus=Set of User Sessions (Kullanıcı oturumları kümesi)

S=A session in Sus (bu kümedeki bir oturum)

Sqsc=Set of Query Suggestion Candidates (sorgu önerme adayları kümesi)

Sqsc içerisindeki her bir sorgu için, yani her bir sorgu önerme adayı için şu değerleme/skor hesaplanır:

$$Oturum\ \ Yakınlığı\ \ Değeri(q) = \sum_{\substack{k=1 \\ q_i \neq q \\ q_i \in S_k \wedge q \in S_k}}^{oturum\ \ sayısı} \frac{1}{pos(S_k, q_i) - pos(S_k, q)}$$

Burada, pos(S_k,q) ifadesi, q sorgusunun S_k oturumu içerisindeki pozisyonu, konumunu ifade etmektedir.

Elde edilen oturum yakınlığı değerleri ile elde edilen vektör, büyükten küçüğe doğru sıralanıp en üstteki n değer alınırsa, bu metodla sorgu önerme sonuçları elde edilmiş olur. Aynı zamanda, bu vektör, Bölüm 2.2.3'de anlatıldığı üzere diğer yöntemlerin vektörleriyle de belli katsayılar kullanılarak birleştirilerek de sorgu önermeleri elde edilebilir.

Yapılan çalışmalarda, oturum yakınlığı ve oturum sayıları algoritmalarından gelen değerlerin önemli bir kısmının sıfır olduğu (yani $q_{initial}$ ile sorgu önerme adaylarının aynı oturum içerisinde geçme ihtimalinin düşük olduğu) gözlenmiş,

ancak bu yöntemlerin diğer yöntemlerle beraber kullanılması durumunda başarımı artırdığı görülmüştür.

Bu yöntem, yeterli miktarda oturum bilgisi var olduğunda makul sonuçlar üretebilmektedir. Genelde normal beklenen sonuçlar üretmekle beraber, örneğin kullanıcının yanlış olan bir sorgusunu düzeltmesi gibi durumlarda hatalı önerme yapabilmektedir. Hatalı önerme ile doğru olan önerme aynı oturum içerisinde geçtiğinden dolayı, – diğer bazı şartlara da bağlı olarak – biraz hatalı çalışabilmektedir. Bu sorunu aşabilmek için, diğer yöntemlerle birlikte kullanım bir çözüm olabilir.

2.3.3. Yol Frekans Değeri Algoritmaları

Kullanıcıların arama ve sonuçlara tıklamaları query-URL grafiği şeklinde gösterilmektedir (Bölüm 1.1.4.3). Bu bölümdeki yöntemler, kullanıcılar bu grafikte (bir anlamda) gezindikçe ortaya çıkan yol ve frekansları (tıklama sayılarını) kullanmaktadır. Bu bölümlerde tıklama yolu ve geliştirilen ilgili algoritmalar hakkında bilgiler verilecektir.

2.3.3.1. Grafik İçerisindeki Tıklama Yolu

Bir kullanıcı eğitim arama motorunda bir "q_1" sorgusu yapıp bir arama sonucuna (URL/LO, Eğitim Dokümanı, *Learning Object*) tıkladığında, q_1'den LO_1'e doğru bir yol oluştuğu varsayılır. Başka bazı kullanıcılar da q_2 sorgusunu arayıp LO_1 eğitim dokümanına tıklayabilir, ki bu durumda da q_2 ile LO_1 arasında bir yol-ilişki oluşur. Aynı eğitim dokümanına farklı iki sorgu ile ulaşıldığından dolayı, bu durum q_1 sorgusu ile q_2 sorgusu arasında bir ilişki oluşturmaktadır. Burada oluşan tıklama "yolu" şöyle gösterilebilir:

$$q_1 \rightarrow LO_1 \rightarrow q_2$$

Aynen q_1 sorgusundan q_2 sorgusuna geçildiği gibi, q_2 sorgusundan da benzer

şekilde q_3 sorgusuna geçilebilir. Bu durum şöyle gösterilebilir:

$q_1 \rightarrow LO_1 \rightarrow q_2 \rightarrow LO_2 \rightarrow q_3$

Buradaki gösterimde (notasyonda) tıklanan eğitim dokümanı bilgisini atlarsak, kısaca şöyle de gösterilebilir:

$q_1 \rightarrow q_2 \rightarrow q_3$

Buna "tıklama yolu" diyebiliriz. Bu tıklama yolunda üç tane düğüm *(nod)* iki tane de kenar *(edge)* vardır. Bu bilgiler ışığında q_1 ile q_3'ün belirli bir dereceye kadar ilişkisi vardır denebilir. Tıklama yolu üzerindeki bu sorguların birbirleriyle ilişkisi aynı zamanda kaç defa tıklandıklarıyla, yani frekanslarıyla da ilgilidir. Ne kadar çok tıklanmışsa, o kadar çok ilişki vardır. Ancak, bu yolun uzunluğu ne kadar uzunsa da o kadar ilişki azalır. Eğer tıklama sayılarını parantez içerisinde gösterirsek ve iki farklı yol var olduğunu düşünürsek, yukarıdaki yol/gösterim şöyle bir hal alır:

yol 1: q_1 (2)$\rightarrow$ q_2 (3)$\rightarrow q_3$

yol 2: q_4 (12)$\rightarrow$ q_5 (15)$\rightarrow q_6$

Bu iki yol bilgisinden, q_4 sorgusunun q_6'ya olan ilişkisinin q_1'in q_3'e olan ilişkisinden/benzerliğinden çok daha fazla olduğu açıkça anlaşılmaktadır (aynı yol uzunluğu olduğu varsayıldığında). Bu durum tartışılabilir. Örneğin navigasyon amaçlı çok genel sorgularda bu durum biraz değişebilir. Benzer şekilde, yol uzunluğunun sorgular arasındaki ilişkisine etkisini anlamak için şöyle bir örnek verebiliriz: Şu şekilde iki tıklama yolu düşünelim:

yol 1: q_1 (10)$\rightarrow$ q_2 (10)$\rightarrow q_3$

yol 2: q_4 (10)$\rightarrow$ q_5 (10)$\rightarrow q_6$ (10)$\rightarrow q_7$

Burada, q_1 sorgusunun q_2'ye olan ilişkisinin, $q_4 \rightarrow q_7$ ilişkisinden daha kuvvetli olduğu söylenebilir. Zira q_4'den q_7'ye giderken daha fazla atlama noktası vardır (benzer tık sayıları olduğu varsayıldığında).

Bu temel gösterim ve bilgi bizi "yol frekansları" ile ilgili ilerleyen bölümlerdeki algoritmaların oluşturulmasını sağlamıştır.

2.3.3.2. Yol Frekanslarının Kullanıldığı Algoritmalar

Eğitim arama motoru sorgu kayıtlarından üretilen oturum bilgileri üzerinde yapılan çalışmalarda "yol-frekans" bilgilerinin kullanıldığı dört farklı algoritma geliştirilmiştir. Hesaplamalar aynı tıklama yolu üzerinde, farklı şekilde yapılmıştır. Buradaki amaç, ilk başta kullanıcının girdiği sorgu ile aday sorgu önermesi arasındaki ilişkiyi ölçebilmektir.

Yol frekansları üzerinde bu algoritmaların çalıştırılabilmesi için, öncelikle her bir aday sorgu için, ilk sorgu ile aday sorgu arasındaki tüm "yol"ların belirlenmesi gerekir. Bu amaçla, kullanıcı tıklamalarının kaydedildiği kayıtlardan istifade edilmiştir. Hesaplama yapılan her bir yolun başında $q_{initial}$, sonunda ise aday sorgu önermesi vardır (tersi de aynı sonucu döndürür).

Şu şekilde bir tıklama yolu *(click-path)* olduğunu varsayalım:

$q \text{ –}(f_1)\text{→ LO –}(f_2)\text{→ } q_{next}$ (f_1,f_2: tıklama sayısı/frekansı)

Burada bir ilk sorgu, tıklanan bir eğitim dokümanı *(LO: Learning Object)* ve aynı eğitim dokümanına tıklaması var olan başka bir sorgu (q_{next}) vardır. Bu gösterimi, eğitim dokümanı bilgilerini atılarak şu şekilde kısaltılmıştır:

$q \text{ –}(f)\text{→}q_{next}$; $f=(f_1+f_2)/2$

Eğer birden fazla aşama varsa:

$q_i \text{→ } q_1 \text{ → } q_2 \text{ → } q_c$ ($q_i=q_{initial}$, q_c=*Candidate Query Suggestion*=Sorgu Önerme Adayı)

Burada dört sorgu arasında üç tane yol parçası, dolayısıyla da üç tane tıklama frekansı vardır.

Daha somut bir örnek vermek gerekirse, kullanıcıların şöyle bir sorgu kaydına sahip olduğunu ve bu yol bilgisinin sorgu kayıtlarından çıkarıldığını varsayalım:

açılarına göre üçgenler —(4.5)→ üçgen çizimi —(23.5)→ üçgen çeşitleri —(5.0)→ matematik noktaların birbirine göre uyumu —(3.5)→ paralel iki doğru

Buradaki tıklama frekansları değerleri şöyledir: [4.5, 23.5, 5.0, 3.5]

Verilen bu ön bilgilere göre, q_i ile q_c arasındaki ilişki, ikisi arasındaki tıklama yolunun frekanslarıyla ölçülebilir. İkisi arasındaki ilişki değeri (rakamsal değer) frekans/tıklama sayısı ile doğru orantılı iken, iki sorgu arasındaki yolun uzunluğu ile (doğrusal veya üslü olarak) ters orantılıdır. Bu iki durum ayrı ayrı ele alınmıştır: yol ile doğrusal ters orantılı olması ve yol ile üslü ters orantılı olması. Bu hesaplamalar şu önsezi ve varsayımla yapılmıştır:

- kullanıcılar yol üzerinde ne kadar tıklarsa, o kadar q_i ile q_c ilgilidir,
- tıklama yolu ne kadar kısa ise, q_i ile q_c o kadar ilgilidir.

Bu varsayımlar ve çıkarımlardan yola çıkılarak önerilen ve oluşturulan algoritma formülleri Tablo 4'de verilmiştir. İlk sorgudan aday sorgu önermesine giden yol sayısı birden fazla olabilir. Bu nedenle algoritmalar bu çoklu-yolu kullanabilir veya kullanmayabilir. Sadece tek yolu kullanan algoritmalar, grafiğin gezilmesi sırasında bulunan ilk yolu kullanmaktadır.

Aynı zamanda, yol üzerindeki her bir basamağın hesaplamalardaki ağırlığı, frekans ile işlem yaparken farklı da olabilir. Bu sayede ilgili olan sorgular daha belirgin bulunabilmektedir.

Tablo 4: Yol Frekansları Algoritmalarının Çeşitleri

Yol-Frekans Algoritma nosu #	Çoklu Yol kullanıyor mu?	Yol parçalarının katsayısı-ağırlığı	Formül
Algoritma 1	Hayır	Sabit=1	$Skor_1 = \frac{\sum_{j=0}^{len(path)} Fr(path(j))}{len(path)}$
Algoritma 2	Hayır	Sabit=1	$Skor_2 = \frac{\sum_{j=0}^{len(path)} Fr(path(j))}{(len(path))^2}$
Algoritma 3	Evet	Yol parçasının pozisyonuyla ters orantılı=2^{-j}	$Skor_3 = \sum_{i=1}^{number\ of\ paths} \frac{\sum_{j=0}^{len(path_i)} Fr(path_i(j)) \cdot 2^{-j}}{len(path_i)}$
Algoritma 4	Evet	Yol parçasının pozisyonuyla ters orantılı=2^{-j}	$Skor_4 = \sum_{i=1}^{number\ of\ paths} \frac{\sum_{j=0}^{len(path_i)} Fr(path_i(j)) \cdot 2^{-j}}{(len(path_i))^2}$

Tablo 4'de kullanılan:

path(j): Yol'un j'inci parçasını,

Fr(path(j)): Yol'un j'inci parçasının frekansı, yani tıklanma sayısını göstermektedir. Örneğin yukarıdaki örnekte q_i ile q_l arasındaki frekans 4.5'dir.

Birinci ve ikinci algoritmalar her bir yol parçasına sabit “bir” ağırlığını vermiştir. Halbuki üçüncü ve dördüncü algoritmalar, yol parçasının pozisyonuyla ters orantılı olarak, her birisine farklı bir ağırlık vermiştir. Bu durumda, yolun başlangıcındaki sorguya yakın olan yol parçalarının katsayısı, ağırlığı daha fazladır; örneğin, yolun başlangıcındaki 20 tıklama, sonraki pozisyonlardaki 20 tıklamadan

daha ağırlıklıdır. Aynı metod, ilk iki algoritmaya da uygulanabilir ve yeni iki algoritma elde edilebilir, test edilebilir. Yapılan çalışmalarda, eğitim arama motorundan elde edilen loglar ve yollar vasıtasıyla, bir sorgudan diğerine geçme ihtimalleri bu dört yöntemle hesaplanarak test edilmiş, en son deney çalışmasında da üç ve dördüncü yöntem üzerinden diğer yöntemler karşılaştırılmış, sonuçlar Bölüm 3.4'de gösterilmiştir. Bu hesaplamalara göre;

Yukarıdaki örnek için frekans değerleri: [4.5, 23.5, 5.0, 3.5]

Algoritma 1'in sonuç skoru/değeri: 4.484375

Algoritma 2'nin sonuç değeri=Algoritma 1'in değeri / uzunluk =1.12109375

Sorgu-doküman grafiğinde *(Query-Url graph)* bir sorgudan diğerine doğru gezinirken, izlenen yol uzadıkça sorgular arasındaki ilgi değerinin azalacağı sezgisel olarak, açıkça anlaşılmaktadır. Bu sebeple, ikinci algoritmanın birinci algoritmadan daha iyi olduğu anlaşılmaktadır.

Aşağıda bir sorgudan diğerine giden çoklu yol örneği görülmektedir:

“açılarına göre üçgenler” sorgusundan “geniş açı” sorgusuna giden “tüm yollar”

Yol: 0

açılarına göre üçgenler —(4.5)→ üçgen çizimi —(23.5)→ üçgen çeşitleri —(5.5)→ geniş açı

Frekanslar: [4.5, 23.5, 5.5]

Yol Değeri (3. alg.): 5.875

Yol Değeri2 (4. alg.): 1.9583

Yol: 1

açılarına göre üçgenler —(4.5)→ üçgen çizimi —(2.0)→ geniş açı

Frekanslar: [4.5, 2.0]

Yol Değeri (3. alg.): 2.75

Yol Değeri2 (4. alg.): 1.375

Yol Toplam Değeri (3. alg.): 8.625

Yol Toplam Değeri2 (4. alg.): 3.33

2.3.4. Yol Frekansları Algoritmasının Hitting Time Algoritmasından Farkı

Yol frekansları ve Hitting Time algoritmaları sorgu-URL tıklamalarını kullanarak çalışmaktadır. Bu bölümde, bu algoritmaların aynı veri üzerinde nasıl farklı çalıştığı anlatılmıştır. Deneylerde incelenen Hitting Time algoritması bir sorgudan diğerine "geçiş ihtimallerini" hesaplamaya çalışmaktadır. Örneğin, query i'den (q_i), k URL'sine doğru hareket edildiğini varsayalım (Bölüm 2.3.1'de olduğu gibi). Bu durumda, i sorgusundan k URL'sine geçiş ihtimali şu şekilde hesaplanır:

$$\text{Geçiş İhtimali} = \frac{w(i,k)}{d_i}$$

burada d_i şu şekilde hesaplanır:

$d_i = \sum_{j \in V_2} w(i,j)$ yani: i'inci sorgu'dan çıkan ve tüm dokümanlara giden tıklamalar toplamı,

w(i,k): i'inci sorgudan k'ıncı doküman/url'ye giden tıklama toplamıdır.

Buna göre, i sorgusundan çıkan tüm tıklamalar hesaba katılmaktadır, ki ihtimal hesabı bunu gerektirmektedir. Bu tez çalışmasında önerilen yol frekansları metodunda ise, ihtimaller kullanılmamakta, bunun yerine i sorgusundan k URL'sine giden tıklamalar/yollar kullanılmaktadır (varsa birden fazla yollar kullanılarak). Tıklama miktarları, ilk sorguya uzaklığa göre de ağırlıklandırılmakta, ilk sorguya

yakın olan sorgulara daha yüksek skor verilmektedir.

2.3.5. Eğitimle İlgili Özellikleri Kullanan Sorgu Önerme Algoritmaları

Bu bölümde, sorguların eğitimle ilgili özelliklerini kullanan algoritmalardan bahsedilmiştir.

2.3.5.1. Sınıf Benzerliği

Burada bahsedilen sınıf, öğrencinin okuldaki sınıfıdır, makine öğrenmedeki sınıflandırmada kullanılan sınıf değildir.

Bu algoritmalar, arama sonuçlarında gösterilen dokümanların (LO: *Learning Object*) sınıf bilgilerini kullanır. İki sorgu arasındaki benzerlik/ilginin tesbiti için arama kayıtları kullanılarak her bir sorgu için bir arama simüle edilir. Arama sonuçlarındaki sınıf bilgisi alınır. Benzer sınıf bilgileri bulunan sorgular benzer kabul edilir ve bir değer/skor verilir. Bu şekilde tüm aday sorgular orijinal sorgu ile karşılaştırıldığında, en yüksek skorlu olanlar seçilebilir. Bu yöntem tek başına çok yeterli olmasa da, diğer yöntemlerle birlikte kullanıldığında başarılı sonuçlar verebilir ve aynı zamanda ilgisiz sorguların elenmesinde kullanılabilir.

Sorgu önerme çerçeve yapımızda olduğu gibi ilk girilen sorgu ve aday sorgular bu işlem için giriş olarak alınır. Aşağıda "sınıf benzerliği için" sözde koda benzer basamaklar gösterilmiştir:

1. İlk sorgu için arama simülasyonu yap ve çıkan dokümanların sınıflarını belirle
2. Her bir aday sorgu için:
   a) Arama simülasyonu yap ve çıkan dokümanların sınıflarını belirle, Bir doküman birden fazla sınıf ile eşleştirilmiş olabilir, yani bir doküman birden fazla sınıfta kullanılabilir.
   b) Her bir sınıf için kaç tane doküman listelendiğini belirle,

c) Aday sorgu ile ilk sorgunun aynı sınıfa ait doküman ortaklıklarını belirlemek için: sayılarını çarp ve bu çarpımları topla. Bu şekilde, daha fazla ortak sınıf içeren sorgular daha yüksek skora sahip olacaktır.

2. Çıkan sonuçları büyükten küçüğe doğru sırala, en üstteki n değeri al.

Sözde kod şu şekildedir:

```
list ← empty list []
grades_initial ← İlk sorgu için arama simülasyonu yap, ilk sorgunun
sınıflarını belirle
FOR grade IN grades_initial:
       count_i(grade) ← sınıf listesinde "grade" olan dokümanları say
FOR query IN Aday Sorgu önermeleri
       Grades ← "query" için arama simülasyonu yap, sınıfları belirle
       sum(query) ← 0
       FOR grade IN Grades:
              count2(grade) ← sınıf listesinde "grade" olan dokümanları say
              mult ← count2(grade) * count_i(grade)
              sum(query) ← sum(query) + mult
       END FOR
       list ← list + sum(query)
END FOR
list ← list listesini büyükten küçüğe doğru sırala
RETURN list listesinden en üstteki n eleman
```

Bu sözde kodda anlatılan yöntem formülize edilirse, şöyle ifade edilir:

S_{LOq}= q sorgusu yapıldığında gösterilen eğitim dokümanları kümesi

S_{qsc}=Sorgu önermeleri kümesi *(Set of Query Suggestion Candidates)*

q_i = Kullanıcı tarafından ilk başta girilen sorgu $(q_{initial})$

S_{LOqi}= q_i $(q_{initial})$ sorgusu yapıldığında gösterilen eğitim dokümanları kümesi

Sqsc'de bulunan her sorgu için şu hesaplama yapılır:

$$Rank(q, q_i) = \sum_{k=1}^{ng} \left(\sum_{j=1}^{nr} F(q,k,j) \cdot \sum_{j=1}^{nr} F(q_i,k,j) \right)$$

where,

$$F(q,k,j) = \begin{cases} 1 & \text{Eğer q sorgusunun j'inci arama sonucundaki dokümanın k sınıfı varsa} \\ 0 & \text{Aksi halde} \end{cases}$$

Her bir arama sonucundaki eğitim dokümanının birden fazla sınıf bilgisi olabileceği unutulmamalıdır. Burada,

ng=Muhtemel sınıf sayısı *(number of grades)*

nr=Arama sonuçları sayısı (q ve q_i için, hangisi büyükse) *(number of results)*

Bu hesaplama, iki sorgu arasında ne kadar ortak sınıf bilgisi olduğunu gösteren sayısal bir değer oluşmasını sağlayacaktır. Bu sayede, Bölüm 2.2.3'de belirtilen sıralama metodu kullanılabilecektir. İki sorgu daha fazla ortak sınıfa sahip olduğunda bu sayısal değer daha büyük olacaktır. Bu sayede, bu algoritmanın farklı sınıflara ait sorguları büyük oranda elemesi beklenebilir.

2.3.5.2. Ders Bilgisi

Eğitim arama motorları, kullanıcının kaptığı arama ve tıklamaların eğitimle ilgili bilgilerini de genellikle sorgu kayıtlarında barındırmakta ve veri olarak sunmaktadır. Eğitimle ilgili bilgiler tıklanılan ders, sınıf, konu v.b. olabilir. Bir doküman'ın (LO) ders bilgisi kayıtlı olabilir, bilinebilir. Ancak, kullanıcının yaptığı bir sorgunun ders bilgisi direk olarak bulunmamakta olup, bu bilginin doküman veya diğer verilerden elde edilmesi gerekmektedir.

Yapılan çalışmalarda, eğitim dokümanlarının ait olduğu ders bilgisi de kullanılmıştır. Kullanıcının tıkladığı dokümanların ders bilgisi kullanılarak, kullanıcının yazdığı sorgunun ait olduğu, olabileceği ders tahmin edilmeye çalışılmıştır. Sorguların ders bilgisi elde edildiğinde, ilk sorgu ile aday sorguların dersleri karşılaştırılarak skorlama ya da aynı dersten sorguların gösterilmesi yoluna

gidilebilir.

Yapılan çalışmalarda, ilk sorgunun ders bilgisi ve aday sorgunun ders bilgisi tesbit edilebildiği zaman, dersleri farklı olanlar sorgu önerme sonuçlarından çıkarılmıştır. Sorgu önerme sonuçlarından çıkarma, en son sıralama aşamasında, genel kontroller aşamasında yapılabileceği gibi, grafik oluşturma aşamasında da yapılabilir. Yapılan son çalışmalarda, grafik oluşturulurken dersi farklı olanların elenmesi yoluna gidilmiştir. Bu şekilde, birbiriyle ilgisiz, dersi tamamen farklı olan sorguların önerilmemesine çalışılmıştır. Eğer sorguların dersi tesbit edilemezse herhangi bir eleme yapılmamıştır.

Bir sorgunun ders bilgisini tesbit eden sözde kod şu şekildedir:

```
output ← “”
lo_courses ← Aramayı simüle et, arama sonuçlarındaki ders bilgilerini al
course_counts ← ders isimlerinin bulunduğu ilk başta sıfır olan isimli dizi
(named array)
FOR lo IN lo_courses:
       course_counts[lo ders adı]+= 1/(lo sırası)
       // sonuçlarda gösterim sırasına göre üsttekilere daha yüksek puan ver.
END FOR

IF course_counts büyüklüğü=0:
       output ← “”
ELSE IF course_counts büyüklüğü=1:
       output ← course_counts içindeki ilk elemanın dersi
ELSE:
       total ← tüm ders sayılarını topla
       FOR count IN course_counts:
              IF (count/total)>0.8:
                     //toplam sayının %80ine sahip olan ders, aradığımız derstir.
                     output ← bu dersin adı
```

```
RETURN output
```

Bu fonksiyonun çıktısı boş olabilir ve bu, ders bilgisinin belirlenemediği anlamına gelir. Bu durumda, ilgili algoritmalarda ders bilgisi kullanılmayacak, göz önüne alınmayacaktır. Yukarıdaki sözde kodda anlatılan yöntem formülize edilirse, şöyle ifade edilebilir:

$$Destek_c(q)=\sum_{j=1}^{nr} G(q,c,j)$$

burada,

$$G(q,c,j)=\begin{cases}\frac{1}{j} & \text{q sorgusu için j'inci sıradaki doküman/url c dersi ile ilgiliyse}\\ 0 & \text{Aksi takdirde}\end{cases}$$

ve nr arama sonucu sayısını ifade etmektedir (number of results)

$$\text{Destek Oranı}_c(q)=\frac{Destek_c(q)}{\sum_{c=1}^{nc} Destek_c(q)}$$

burada nc ders sayısını ifade etmektedir (number of courses)

Bu destek oranı *(support ratio)* bilgisine göre, belirli bir oranın üstündeki ders, q sorgusunun dersi olarak kabul edilebilir. Yapılan çalışmalarda 0.8 oranı bunun için kullanılmıştır. Yani, arama sonuçlarındaki dokümanların derslerinin yaklaşık %80 ve daha fazlası hangi derse ait ise, kullanıcının girmiş olduğu sorgu da o derse aittir olarak kabul edilmiştir. Hem kullanıcının girdiği sorgu, hem de aday sorgular için bu şekilde, her sorgunun dersi belirlenmiştir.

2.3.6. Karma (Hibrid) Algoritmalar

Bölüm 2.2.3'de belirtildiği üzere, birden fazla algoritma sıralama skorları, yapılacak testlerle belirlenecek katsayılarla normalize edilerek hibrid algoritmalar oluşturulabilir. Hibrid algoritmalar hakkında detaylı bilgi Bölüm 2.2.3'de verilmiştir.

BÖLÜM III

DENEYLER, BULGULAR ve SONUÇLARI

3.1. DENEY ORTAMI ve VERİ KÜMESİ

Önceki bölümlerde anlatılan sorgu önerme çerçeve yapısı ve ilgili algoritmalar, sorgu kayıtları kümesinden rastgele bazı sorgular seçilerek denenmiştir. Bu rastgele sorgulardan üretilen sorgu önermeleri kaydedilip, doğruluk ve kalitelerine göre değerlendirilmiştir. Veri kümesi, testler-deneyler ve değerlendirme süreci ilerleyen bölümlerde detaylı olarak anlatılmıştır.

Sebit firmasına ait Vitamin eğitim platformundaki eğitim arama motorunun arama kayıtları *(query log)* kullanılmıştır. Arama kayıtları, kullanıcılar tarafından yapılan aramaları, bu aramalarda tıklanan eğitim dokümanlarını ve eğitim dokümanları hakkında sınıf, ders gibi bazı ek bilgileri içermektedir.

Gerçekleştirilen algoritmalar tamamen sorgu-doküman grafiğini kullandığından dolayı, sorgu ve tıklama kayıtları büyük öneme sahiptir. Var olan ve yeni geliştirilen/geliştirilebilecek algoritmaların başarım oranlarının bu veri kümesinin boyutu ve kalitesinin artmasıyla daha da artacağı tahmin edilmektedir.

Kullanılan verilerde 325,241 tanesi tıklanmış 2,028,395 adet sorgu kaydı, 6,520 adet eğitim dokümanı, 52,635 adet benzersiz sorgu, 52,713 kullanıcı ve 150,214 adet (içinde birden fazla arama olan) oturum bilgisi vardır. Bu veriler söz sonusu arama motorunun Aralık-2013 ile Mayıs-2014 arasındaki sorgu kayıtlarından elde edilmiştir.

3.1.1. Deneyde Kullanılan Araçlar

Deneylerin gerçekleştirilmesinde kullanılan belli başlı araçlar Tablo 5'de

belirtilmiştir.

Tablo 5: Deneyde ve Tez Yazımında Kullanılan Araçlar

Amaç / Nerede Kullanıldığı	Araç
Verilerin alındığı orijinal veritabanı	Mongo DB
Verilerin işlemede kullanıldığı ve aktarıldığı ilişkisel veritabanı	Mariadb (Mysql eşleniği)
Yazılımların gerçeklendiği dil	Python
Değerlendirme web arayüzü yazılımı	Python
P-test yapmak için kullanılan Python kütüphanesi	Scipy kütüphanesinden Stats modülü
Deneylerin gerçeklendiği işletim sistemi	Ubuntu 15.10 ve Redhat (ayrı ayrı ikisinde de gerçeklenmiştir)

3.2. GELİŞTİRİLEN YAZILIM(LAR)

Tasarımı ve hesaplamaları yapılan ve sözde kodları hazırlanan algoritmalar python dili kullanılarak gerçekleştirilmiştir. Algoritmalar kendi başlarına çalışabildikleri gibi, önerilen sorgu önerme çerçeve yapısı ile de kullanılabilmektedirler.

Geliştirilen yazılım, interaktif olarak veya direk çıktı verecek şekilde fonksiyonel olarak çalışabilmektedir. İnteraktif olarak çalıştığında, giriş sorgusunu alarak, sorgu önermelerini hesaplamakta, sorgu önermeleriyle beraber önerilen her bir sorgunun tüm algoritmalardaki skorlarını ekrana yazarak kontrol ve değerlendirme, hata ayıklama *(debug)* imkanları sunmaktadır. Örnek iki program çıktısı Şekil 10 ve Şekil 11'da verilmiştir (aradaki *debug* ve bilgi çıktıları atlanmış, sonuç kısmı gösterilmiştir).

```
12:01:32 ht4 bitti../301
12:01:32 alg1_3_2 bitti
12:01:32 sort_combined3.2: adaylar sayısı: 297
Eleme öncesi aday uzunluk1: 297
Eleme öncesi aday uzunluk: 297 2 3
Eleme sonrası aday uzunluk*: 90
Kullanılan metodlar ve katsayıları: (Toplam 13 adet metod, top değeri:25.0)
Metod: hitting time skorları                  katsayisi: -3    toplama orani: -0.12
Metod: session proximity                      katsayisi: 2     toplama orani: 0.08
Metod: tık sayıları (logaritmik)              katsayisi: 2     toplama orani: 0.08
Metod: sorgu sayilari (logaritmik)            katsayisi: 1     toplama orani: 0.04
Metod: sınıf benzerlikleri (logaritmik)       katsayisi: 0.5   toplama orani: 0.02
Metod: result sayıları (logaritmik)           katsayisi: 0.5   toplama orani: 0.02
Metod: user sayıları (logaritmik)             katsayisi: 1     toplama orani: 0.04
Metod: Dwell time (logaritmik)                katsayisi: 2     toplama orani: 0.08
Metod: Tık/Sorgu oranı                        katsayisi: 1     toplama orani: 0.04
Metod: yeni3 = multiple (path(j)*2^-j)/len    katsayisi: 4     toplama orani: 0.16
Metod: yeni4 = multiple (path(j)*2^-j)/len^2  katsayisi: 8     toplama orani: 0.32
Orjinal (Önermesi hazırlanacak) Sorgu: "bedir savaşı"  Dersi: "Sosyal Bilgiler"
Sno  AnaSkor  Sorgu Önerme                    Dersi            Hitting SessProx Tik# Sorgu# SinifBen Result# User# dwellT  Yeni1 Yeni2 Tik/Sor Yeni3 Y4  #Path
 1)  14.525 "ipek yolunda türkler"            Sosyal Bilgile     9.890   0.52  7.78   10.44  10.34   5.931  6.931   7.57    7.0    7.0   0.16   22.6  13.5    3
 2)  13.746 "hendek savaşı"                   Sosyal Bilgile     9.505   0.50  2.81    4.32   6.54   2.402  2.000   8.70    5.5    5.5   0.35   25.4  12.9    4
 3)  12.816 "islam tarihi"                    Sosyal Bilgile     9.833   0.09  5.55    8.87  10.16   6.683  4.755   7.31    3.0    3.0   0.10   26.0  11.7    4
 4)  11.531 "ilk türk islam devletleri"       Sosyal Bilgile     9.869   0.07  4.86    7.81  10.05   8.434  4.322   6.93    2.5    2.5   0.13   23.8  10.4    4
 5)  10.412 "islamiyetin doğuşu ve yayılışı"  Sosyal Bilgile     9.691   1.55  4.81    7.36   7.83   2.807  4.524   7.94    8.0    8.0   0.17   12.8  10.4    2
 6)  10.293 "ipek yolu"                       Sosyal Bilgile     9.916   0.00  6.95    9.39   9.42   6.560  6.392   7.61    3.5    3.5   0.18   17.0   8.7    3
 7)   9.677 "islamiyet"                                         9.733   0.00  3.17    5.55   6.36   2.585  2.585   7.67    4.0    4.0   0.19   18.9  10.2    3
 8)   9.342 "islamiyetin doğuşu"              Sosyal Bilgile     9.687   0.00  3.81    6.36   7.83   2.987  3.459   8.53    5.5    5.5   0.17   14.4   9.9    2
 9)   8.948 "ilk müslüman türk devletleri"    Sosyal Bilgile     9.908   0.00  3.70    7.26   9.99   9.372  2.000   8.44    2.5    2.5   0.08   16.9   8.5    3
10)   8.227 "dandanakan savaşı"               Sosyal Bilgile     9.687   0.00  3.32    6.02   9.35   6.461  3.000   7.34    4.0    4.0   0.15   16.7   7.8    3
11)   7.784 "talas savaşı"                    Sosyal Bilgile     9.709   1.00  4.09    6.79   9.31   5.977  3.907   8.43    4.0    4.0   0.15   11.8   6.6    2
12)   7.517 "kök türkler"                     Sosyal Bilgile     9.893   0.00  3.81    4.75   8.91   5.494  3.322   8.42   10.5    5.2   0.52   14.5   6.0    2
13)   7.371 "uhud savaşı"                                       9.790   1.95  2.32    5.55   6.54   3.000  2.000   7.17    2.5    2.5   0.11    9.8   6.1    2
14)   6.880 "malazgirt savaşı"                                  9.945   0.00  6.17    8.45  10.59   6.428  5.459   8.52    3.5    3.5   0.21    9.7   5.0    2
15)   6.374 "dört halife dönemi"                                9.786   0.20  4.32    6.99   9.06   4.768  4.087   7.68   13.2    6.6   0.16    8.6   4.3    1
16)   6.225 "islamiyetin yayılışı"                              9.751   0.25  3.00    5.32   5.83   1.700  2.000   8.16    2.5    2.5   0.20   10.2   6.4    2
17)   5.933 "türk islam devletleri"                             9.861   0.00  2.81    5.67   8.92   9.391  2.585   8.35    3.0    3.0   0.14   10.4   5.5    2
18)   5.889 "savaş"                                             9.943   0.00  6.17    8.90  11.26   6.958  5.358   7.62    2.5    2.5   0.15    8.5   4.0    2
19)   5.752 "ilk türk devleti"                Sosyal Bilgile     9.925   0.00  3.32    4.32  10.12   9.822  2.322   7.00   16.2    5.4   0.50   13.2   3.9    2
20)   5.298 "harita"                                            9.975   0.00  9.44   12.51   8.91   4.328  8.771   7.47   20.8    5.2   0.12    6.7   1.7    1
21)   5.152 "hicret nedir"                    Sosyal Bilgile     9.702   0.00  3.00    3.81   8.38   4.954  0.000   9.23   13.0    6.5   0.57    8.5   4.2    1
22)   4.636 "ilk türk devletleri"             Sosyal Bilgile     9.919   0.00  5.17    6.67  10.43   9.918  4.000   7.60   16.2    5.4   0.35    7.2   2.4    1
23)   4.568 "dört halife"                                       9.742   0.00  3.17    5.55   7.97   3.205  3.170   8.18   12.0    6.0   0.19    8.0   4.0    1
24)   4.504 "selçuklu devleti"                                  9.950   0.00  5.61    8.11   9.32   6.643  5.170   7.51   16.5    5.5   0.18    7.3   2.4    1
25)   4.415 "hak ve sorumluluklarımız"        Sosyal Bilgile     9.972   0.00  4.32    6.77   9.82   6.666  3.322   6.73   10.2    5.1   0.18    7.1   3.6    1
26)   4.347 "savaşlar"                                          9.925   0.00  3.91    7.61   8.89   4.671  3.170   7.45   10.8    5.4   0.08    7.4   3.7    1
27)   4.211 "türk islam bilginleri"           Sosyal Bilgile     9.920   0.00  4.64    6.60   9.64   9.494  3.907   7.02   16.3    5.4   0.26    7.3   2.4    1
28)   4.146 "hz muhammed"                     Sosyal Bilgile     9.729   0.33  2.81    5.61   4.00   1.363  2.322   7.46   12.2    6.1   0.14    8.1   4.1    1
29)   4.084 "büyük selçuklu devleti"                            9.957   0.00  6.57    8.77  10.03   6.937  5.781   7.09   14.9    3.7   0.22    6.0   1.5    1
30)   4.006 "türk tarihine yolculuk"                            9.969   0.00  5.98    8.48  10.39   8.210  4.858   7.78   15.5    3.9   0.18    6.1   1.5    1
-------------------------------------------------------------------------------------------------
Cache degerlerini diske kaydediyor.. Frekanslar değişmemiş, kaydetmiyor..
Bitti
(s)tandart,(s2)standart2, y1:yeni1,y2,y3,y4, (h)ittingtime, (d)welltime, (c)ustom değerler, (r)esulta göre, (session) proximity,tik/sorgu (oran)i, (q)uit/return
Hangi metoda göre sıralasın?
```

Şekil 10: Geliştirilen Programın Örnek Çıktısı

```
12:04:16 ht4 bitti../301
12:04:16 alg1_3_2 bitti
12:04:16 sort_combined3.2: adaylar sayısı: 288
Eleme öncesi aday uzunluk1: 288
Eleme öncesi aday uzunluk: 288 2 3
Eleme sonrası aday uzunluk*: 80
Kullanılan metodlar ve katsayıları: (Toplam 13 adet metod, top değeri:25.0)
Metod: hitting time skorları                    katsayisi: -3    toplama orani: -0.12
Metod: session proximity                        katsayisi: 2     toplama orani: 0.08
Metod: tık sayıları (logaritmik)                katsayisi: 2     toplama orani: 0.08
Metod: sorgu sayilari (logaritmik)              katsayisi: 1     toplama orani: 0.04
Metod: sınıf benzerlikleri (logaritmik)         katsayisi: 0.5   toplama orani: 0.02
Metod: result sayıları (logaritmik)             katsayisi: 0.5   toplama orani: 0.02
Metod: user sayıları (logaritmik)               katsayisi: 1     toplama orani: 0.04
Metod: Dwell time (logaritmik)                  katsayisi: 2     toplama orani: 0.08
Metod: Tık/Sorgu oranı                          katsayisi: 1     toplama orani: 0.04
Metod: yeni3 = multiple (path(j)*2^-j)/len      katsayisi: 4     toplama orani: 0.16
Metod: yeni4 = multiple (path(j)*2^-j)/len^2    katsayisi: 8     toplama orani: 0.32
Orjinal (Önermesi hazırlanacak) Sorgu: "erzurum kongresi"  Dersi: "Inkılap"
Sno  AnaSkor  Sorgu Önerme                          Dersi            Hitting SessProx Tik# Sorgu# SinifBen Result# User# dwellT  Yeni1 Yeni2 Tik/Sor Yeni3 Y4  #Path
 1)  15.247 "maarif kongresi"                       İnkılap            8.055   0.25  5.55    8.89   9.91   4.811  4.907   8.63   36.5   36.5   0.10  113.5   57.6   12
 2)  13.449 "sivas kongresi"                        İnkılap            8.312  16.91  6.09    7.08   9.86   4.797  5.585   9.41   36.0   36.0   0.50   66.7   38.1   11
 3)  10.138 "milli mücadele"                                           9.445   0.00  5.81    7.46  10.63   6.594  4.755   7.58   36.0   36.0   0.32   58.9   37.7    8
 4)   9.648 "misak-ı milli kararları"               İnkılap            9.099   0.21  5.43    6.67  11.12   7.681  4.858   8.61   36.0   36.0   0.42   39.3   36.2    2
 5)   8.956 "milli uyanış"                          İnkılap            8.522   0.00  3.81    6.83  10.81   7.492  3.459   7.83   36.5   36.5   0.12   46.0   37.2    4
 6)   5.198 "inkılap"                                                   9.460   0.00  6.41    9.93  13.33   8.819  5.248   8.17   15.8    3.9   0.09   30.8    5.3    5
 7)   4.927 "ya istiklal ya ölüm"                   İnkılap            9.670   0.00  7.03    9.93  12.11   6.959  5.977   8.29   19.8    6.6   0.13   18.4    5.2    2
 8)   4.836 "amasya genelgesi"                      İnkılap            9.501   9.61  5.55    7.53   8.54   3.358  5.170   9.28    8.2    0.8   0.25   10.4    0.8    3
 9)   4.812 "havza genelgesi"                       İnkılap            9.078   6.08  5.17    6.09   7.76   2.650  4.755   8.73    8.2    0.8   0.53   22.0    1.6    7
10)   4.249 "maarif kongresi nedir"                                     8.619   0.00  4.32    8.35  10.63   5.952  3.907   6.54   20.5   10.2   0.06   22.0    9.9    2
11)   4.158 "doğu cephesi"                          İnkılap            9.865   0.10  6.17    7.86  11.20   6.613  5.322   8.82   11.3    1.9   0.31   16.7    2.3    3
12)   4.151 "savaş"                                                     9.824   0.00  6.17    8.90  11.58   6.958  5.358   7.62   20.2    6.7   0.15   15.0    5.0    1
13)   3.901 "mustafa kemal hayatı"                  İnkılap            9.789   0.00  2.32    3.58  11.26   8.280  1.000   8.37   15.8    3.9   0.42   47.9    7.6    9
14)   3.858 "çanakkale cephesi"                     İnkılap            9.873   0.40  6.11    7.64  10.41   5.027  5.322   9.01   11.9    2.0   0.35   13.2    1.6    3
15)   3.601 "kurtuluş savaşı"                                           9.820   0.00  7.24   10.32  12.23   6.520  6.109   7.68   11.9    0.8   0.12    3.1    0.2    1
16)   3.567 "bir kahraman doğuyor"                  İnkılap            9.838   0.00  4.64    7.00  10.68   5.521  4.087   7.67   11.5    1.9   0.20   29.0    3.4    7
17)   3.475 "e okul"                                                    9.485   0.00  4.09    6.71  10.57   9.905  3.585   6.96   20.3    6.8   0.16   21.0    5.7    2
18)   3.467 "mudanya ateşkes anlaşması"             İnkılap            9.872   0.00  6.46    8.11   9.95   4.902  6.170   8.82   10.2    0.7   0.32    3.3    0.2    1
19)   3.315 "izmir iktisat kongresi"                İnkılap            8.273   0.00  4.86    6.51   9.76   5.493  4.170   8.00   10.7    0.6   0.32   13.3    0.8    5
20)   3.311 "amasya görüşmeleri"                    İnkılap            9.643   3.02  5.29    6.21   9.60   4.528  4.459   8.82   11.5    0.8   0.53    3.1    0.2    1
21)   3.307 "sakarya meydan muharebesi"             İnkılap            9.893   0.00  6.30    7.56   9.86   4.780  5.615   8.49   10.2    0.7   0.42    3.3    0.2    1
22)   3.223 "atatürkün hayatı"                                          9.946   0.00  5.09    6.94  11.80   6.283  4.087   7.90   13.0    2.6   0.28   15.7    2.1    4
23)   3.142 "istiklal milletimindir"                İnkılap            9.666   0.00  3.17    5.61  10.86   6.367  2.000   7.03   19.5    9.8   0.18   19.0    9.5    1
24)   3.135 "su döngüsü"                                                9.970   0.00  6.39    9.39  10.38   6.641  5.728   8.02    8.5    0.4   0.12    2.1    0.1    1
25)   3.061 "kütahya eskişehir savaşları"           İnkılap            9.763   0.24  5.49    7.35   9.65   4.621  4.907   9.19    9.9    0.6   0.28    4.7    0.2    2
26)   3.034 "trablusgarp savaşı"                                        9.818   0.14  5.58    7.77  10.38   6.641  4.755   7.62   11.3    1.9   0.22    7.6    1.3    1
27)   3.029 "çağdaş türkiye yolunda adımlar"        İnkılap            9.880   0.00  5.09    7.43  12.42   7.884  4.459   8.90   10.4    0.6   0.20    6.8    0.5    2
28)   3.024 "sıralı çalışmalar"                                         9.971   0.00  6.48    9.36  10.60   4.197  5.644   7.91    8.5    0.4   0.14    2.1    0.1    1
29)   2.819 "doğu cephesi şehirleri"                                    9.869   0.00  3.58    5.17  11.22   6.530  2.585   8.61   11.2    1.9   0.33   19.2    2.8    3
30)   2.773 "ingiliz sömürgeciliği"                                     9.710   0.00  4.86   10.43   9.65   4.222  4.392   7.80   10.9    1.6   0.02    9.4    1.1    2
-------------------------------------------------------------------------------------------------------------------------------------------------------------------
Cache degerlerini diske kaydediyor.. Frekansları kaydedecekti, ama inter_mode dolayısıyla kaydetmedi..
Bitti
(s)tandart,(s2)standart2, y1:yeni1,y2,y3,y4, (h)ittingtime, (d)welltime, (c)ustom değerler, (r)esulta göre, (session) proximity,tik/sorgu (oran)i, (q)uit/return
Hangi metoda göre sıralasın?
```

Şekil 11: Geliştirilen Programın Örnek Çıktısı

Söz konusu şekillerde iki sorgunun çıktıları görülmektedir. Çıktıların sütunlarında en solda sıra nosu, sonraki sütunda o anda sıralama yapılan metod veya metodlar neyse onun skorları, sorgu önermesi, sorgunun dersi ve o sorgunun tüm metodlardaki skor değerleri bulunmaktadır. Yazılan bu skor değerleri sayesinde varsa hatalar görülebilmekte, yeni düzeltmeler ve geliştirmeler tasarlanabilmektedir. Skor değerlerinin birçoğunun logaritmik ölçekte yazıldığı unutulmamalıdır. Çıktının en son sütununda orijinal sorgudan önerilen sorguya kaç tane yol *(path)* olduğu görülmektedir ki, bu yol frekansları algoritmalarında kullanılmaktadır.

Gerçekleştirilen yazılımla, aşağıda tekrar hangi algoritma ile sıralama yapılacağı belirtilerek, hızlı bir şekilde aynı sorgunun farklı algoritmalarla önermeleri alınabilmektedir. Bu sayede, hangi algoritmaların daha verimli çalıştığı tesbit edilmeye çalışılmıştır. Hibrid algoritmanın katsayıları da buradaki v.b. denemeler sayesinde yaklaşık olarak belirlenmiştir.

Normalde kullanıcılara genellikle 10 sorgu önermesi gösterilmekle beraber, geliştirme aşamasında hataları görebilmek ve muhtemel daha doğru önermeleri görebilmek adına 30 sorgu önermesi gösterilmiştir. Sonraki bölümdeki testlerimizde kullanıcılara sormak için 10 sorgu önermesi kullanılmıştır.

3.2.1. Web Servis

Geliştirilen algoritma ve konsol yazılımlarına ek olarak, aynı sorgu önerme algoritmalarının başka yazılım ve arama motorlarından çağrılabilmesini kolaylaştırmak amacıyla bir web servis hazırlanmıştır. Web servis ile, http://ipadresi/qs/fiiller şeklinde bir sorgu çağrılabilmekte ve sorgu sonuçları web servisi aracılığıyla alınabilerek istenen programda ya da mobil uygulamada kullanılabilmektedir.

3.3. DENEY PROSEDÜRÜ

Test-deney aşamaları şu şekilde sırayla özetlenebilir:

1) Sorgu kayıtlarının yapısal veritabanına (verideposuna) kaydedilmesi/aktarılması,
2) Oturum bilgilerinin bulunması/oluşturulması,
3) Rastgele olarak, sık kullanılan, nadir kullanılan ve orta kullanıma sahip sorguların seçilmesi,
4) Hibrid sorgular için Cross-Validation tekniği ve manüel yöntem ile parametre belirlenmesi (Bölüm 3.3.5),
5) Seçilen rastgele sorgular için en az sekiz farklı sorgu önerme algoritması ile sorgu önermelerinin üretilmesi ve bunların verideposuna kaydedilmesi,
6) Üretilerek verideposuna kaydedilen sorgu önerme'lerin değerlendiriciler/anketörler vasıtasıyla elle değerlendirilmesi,
7) Algoritmalara ve sorgu tiplerine (sık, orta, nadir kullanılan sorgular) göre, değişik metriklerle başarım oranlarının hesaplanması
8) İstatistiksel belirginlik testlerinin yapılması

Müteakip bölümlerde bu aşamalar daha detaylı anlatılmıştır.

3.3.1. Verilerin Ön İşlemesi ve Veritabanına Kaydedilmesi

Sorgu önerme çalışmalarında kullanılan sorgu kayıtları öncelikle, kolay kullanım ve analiz amacıyla mysql gibi "ilişkisel" *(relational)* bir yapıya dönüştürülmüş, daha sonra bu verilerin analiz edilmesiyle "Sorgu Önerme" algoritmaları geliştirilmiş ve denenmiştir. İlerleyen bölümlerde yapılan bu çalışmalarla ilgili detaylı bilgiler verilmiştir.

Bir eğitim arama motorunun temin edilen arama ve tıklama kayıtları

incelenmiş, bu kayıtlar orijinal tasarımında mongodb veritabanında ilişkisel olmayan büyük veri *(non-relational bigtable)* mimarisinde tutulmakta iken, mysql uyumlu bir veritabanı formatına *(mariadb, relational SQL)* aktarılmış, bu sayede çeşitli istatistiklerin daha rahat çıkarılması, arama kayıtlarının daha rahat işlenmesi amaçlanmıştır.

Eğitim arama motoru kayıtlarında daha çok sorgu-tıklama kayıtları kullanılmış, bunlar üzerinde işlem yapılmıştır.

3.3.2. Sorgu Oturumlarının Oluşturulması

Geliştirilen SÖ algoritmalarında sorgu oturumu bilgisi kullanıldığından dolayı oturum bilgisinin oluşturulması gerekmektedir. Sorgu veritabanı analiz edilerek, kullanıcı oturumları belirlenmiştir. Literatürde genel kabul gördüğü üzere (Torres & Weber, 2011), art arda 30 dakika içinde gelen sorgular aynı oturumun birer parçası gibi düşünülmüştür. İki sorgu arasındaki zaman farkı 30 dakikadan daha çok ise, o zaman farklı oturumların üyesi gibi düşünülmüşlerdir. Oturum özelliklerini kullanan önerilen algoritmalar (oturum sayısı ve oturum yakınlığı) kullanıcıların girdiği ilk sorgu ile aday sorguların aynı oturum içinde yer almalarını gerektirdiğinden dolayı sadece en az iki sorguya sahip olan oturumlar göz önüne alınmıştır. Yapılan çalışmada, en az iki sorguya sahip toplam 150,214 adet oturum bilgisi bulunduğu gözlenmiştir.

3.3.3. Sorguların Seçilmesi ve Sorgu Önermenin Simülasyonu

Üretilecek sorgu önermelerin değerlendirilmesi ve kalitesinin ölçülmesi gerçek kişiler tarafından elle yapılması gerekmektedir. Bu şekilde elle SÖ'lerinin değerlendirilmesi zaman alıcı bir iş olduğundan dolayı, tüm sorgular üzerinde uygulanması zordur. Bu sebeple, sadece belirli sayıda sorgular seçilerek sorgu önermeleri üretilmiş ve bunların değerlendirmeleri gerçek kişiler tarafından

yapılmıştır.

Sık kullanılan sorgulardan 20 tane, nadir kullanılanlardan 20 tane ve bunların arasında olanlardan 20'şer tane olmak üzere toplam 60 sorgu, sorgu kayıtları veritabanından bir yazılım parçacığı ile rastgele seçilmiştir. Burada amaç, değişken frekanslı (kullanım sıklığı olan) sorguların başarımlarının ölçülmesidir.

Sorgular seçilirken sık, nadir ve bunların arasında olması ile ilgili kriterler Tablo 6'da gösterilmiştir.

Tablo 6: Sorgu Seçimi ile İlgili Kriterler

Sorgu Tipi	**En az tıklama sayısı**	**En çok tıklama sayısı**
Nadir sorgular *(Tail)*	5	20
Orta kullanımlı sorgular *(Torso)*	21	500
Sık kullanılan sorgular *(Head)*	501	∞

5 tıklamadan az tık çeken sorgular dahil edilmemiştir, zira bunların çoğunlukla hatalı sorgular içerdiği gözlenmiştir.

Rastgele seçilen bu sorgular kullanılarak her bir sorgu için sorgu önermeleri (bir sonraki Bölüm 3.3.4'de özetlenen) farklı algoritmalarla hesaplanarak yine veritabanına kaydedilmiştir. Her bir sorgu önerme simülasyonu için 10 tane sorgu önerme üretilmiştir.

3.3.4. Gerçeklenip Test Edilen Algoritmalar

Bu tez çalışmasında, sorgu önerme ve skorlama yapan 10 farklı algoritma (yol frekansları algoritmalarının farklı versiyonları göz önüne alınırsa 13) önerilmiş/kullanılmış, bu algoritmaların birleştirilerek olarak kullanılabilmesi için de 4 farklı birleştirme yöntemi ile hibrid yöntemler önerilmiştir. Bu sorgu önerme algoritmalarının ya da birleştirme yöntemlerinin herhangi bir kombinasyonu kullanılabilir. Bu algoritmalardan aşağıdaki 8 tanesi deney çalışmalarımızda

kullanılmış, performans sonuçları hesaplanmış ve sonuçları değerlendirilmiştir:

1. Hibrid olmayan algoritmalar:
    a) DFS grafik gezinme metodunu kullanan Hitting Time algoritması (HT-DFS), (Mei et al., 2008) makalesinde orijinali belirtilen algoritmanın çok benzeridir. Orijinalinde olmamasına rağmen bu algoritmaya genel kontroller uygulanmıştır (Bölüm 2.3.1),
    b) BFS grafik gezinme metodunu kullanan Hitting Time algoritması (HT-BFS),
    c) Yol Frekansları 3. Algoritması (PF3) (Bölüm 2.3.3.2),
    d) Yol Frekansları 4. Algoritması (PF4) (Bölüm 2.3.3.2),
2. Hibrid Algoritmalar:
    a) Hybrid-1-BFS (skor temelli birleştirme kullanır),
    b) Hybrid-2-Borda Count,
    c) Hybrid-3-Weighted Borda Count,
    d) Hybrid-4-Weighted Voting,

Hitting Time (Mei et al., 2008) makalesinde belirtilen algoritma olup, diğerleri bu tez çalışmasında önerilmiştir. Hitting Time için orijinal algoritmada grafik gezinme metodu olarak DFS kullanılmakla beraber, BFS grafik gezinme metodunun kullanıldığı algoritma da denenmiş, BFS grafik gezinme metodunun DFS'e göre daha başarılı sonuçlar ürettiği gösterilmiştir. Tüm algoritmalar için anlamsız ve hatalı sorguların elenebilmesi için genel kontroller (Bölüm 2.2.2) çalıştırılmıştır. Genel kontroller, Hitting Time algoritmasına da uygulanmış olup, bu sebeple orijinal algoritmasından biraz farklıdır.

Testi yapılan her bir algoritma için (20 nadir, 20 orta, 20 sık kullanılan olmak üzere) 60 sorgudan oluşan sorgu örneklem kümemizdeki her bir sorgu için top-10 sorgu önerme üretilmiştir. Yani aynı 60 sorguluk setin, her bir algoritma için ayrı ayrı

sorgu önermeleri üretilmiştir. Her sorgu önermesi için ilgililik skorları değerlendiriciler tarafından belirlendikten sonra, her algoritma ve sorgu için başarım oranları ölçülmüştür. Sorgu önermelerinin değerlendirmeleri, yaşları 30-40 arasında değişen ve arama motorlarına aşina olan ikisi bilgisayar bilimleri doktora öğrencisi, birisi lise öğretmeni, birisi de üniversite öğretim üyesi olmak üzere dört kişi tarafından yapılmıştır.

3.3.5. Hibrid Metodlar için Parametre Belirleme

Borda Count haricindeki hibrid algoritmalar, alt metodların birleştirilmesinde kullanılan katsayıların (parametrelerin) belirlenmesini gerektirmektedir. Parametre belirleme için aşağıdaki yöntemler kullanılmıştır.

3.3.5.1. Cross-Validation Tekniği ile Parametre Belirleme

Öncelikle parametre belirleme için "Cross-Validation" tekniği kullanılmıştır. Cross-Validation uygulamak için bu deney setine 60 sorgu alınmış, her biri 15 sorgudan oluşan 4 tane küme oluşturulmuştur. Her bir küme için, sorguların başarımları, hibrid metodlarda kullanılan (ama kendisi hibrid olmayan) tüm alt metodlar (Bölüm 2.3) için hesaplanmış, bu başarımlar o alt metodların hibrid metoddaki katsayısı olarak kullanılmıştır. Bir küme için belirlenen katsayılarla, kalan 3 kümenin hibrid metoda göre başarımı hesaplanmıştır. Bu işlem tüm kümeler için yapıldıktan sonra, bulunan hibrid başarımlarının ortalaması alınmıştır.

Konunun daha net anlaşılabilmesi için Cross-Validation tekniğindeki kümelere bölme işlemi Tablo 7'de gösterilmiş olup, yapılan işlem adım adım şu şekildedir:

Deney seti: 60 sorgu,

Deney setinin kümelere ayrılması: her biri 15 sorgudan oluşan set_a, set_b, set_c, set_d kümeleri,

Bir küme için, örneğin set_a için uygulanan aşamalar:

1. set_a kümesi için, alt metodlar kullanılarak sorgu önermelerinin üretilmesi,
2. Tüm alt metodlar için set_a'daki sorgu önermelerin değerlendirici insanlar tarafından değerlendirilmesi, skorlanması,
3. Alt metodlar için ortalama ilgililik değerlerinin hesaplanması,
4. Alt metodların ortalama ilgililiğinin, o metod için katsayı olarak kullanılması,
5. set_b+set_c+set_d kümeleri için bir önceki adımda belirlenen katsayıları kullanan hibrid metodlarla sorgu önermelerin üretilmesi,
6. set_b+set_c+set_d kümeleri için hibrid metodlarla üretilen sorgu önermelerin değerlendirici insanlar tarafından değerlendirilmesi, skorlanması,
7. Hibrid metodlar için ortalama performansın hesaplanması,

Burada anlatılan adımlar her bir küme (set_a, set_b, set_c, set_d) için ayrı ayrı tekrar edilir, elde edilen hibrid sonuçların ortalaması alınır. Bu şekilde, hibrid metodlar için optimum parametre belirlenmeye çalışılmıştır. Ancak, bu yöntemle bulunan hibrid başarım oranları, bir alt metod olan Yol Frekansları-3 metodunun başarımını geçememiştir. Bu sebeple, kullanılan hibrid metodların daha başarılı olabilmesi için bir sonraki Bölüm 3.3.5.2'de anlatılan yöntem uygulanmıştır.

Tablo 7: Cross-Validation Tekniğinin Uygulanması İçin Verinin Bölünmesi

Alt metodların “ortalama ilgililiğinin” ölçülebilmesi için sorgu önermeleri üretilecek küme (eğitim verisi *(training data)*)	Belirlenen parametrelerle hibrid metod performansları belirlenecek kümeler
set_a (15 sorgu)	set_b, set_c, set_d (45 sorgu)
set_b (15 sorgu)	set_a, set_c, set_d (45 sorgu)
set_c (15 sorgu)	set_a, set_b, set_d (45 sorgu)
set_d (15 sorgu)	set_a, set_b, set_c (45 sorgu)

3.3.5.2. Manüel Parametre Belirleme

Cross-validation tekniği ile belirlenen parametrelerin istenen başarıyı yakalayamaması sebebiyle, testler sırasındaki tecrübelere ve gözlemlere dayanarak, elle ve deneme yanılma yöntemiyle parametre belirleme yapılmıştır. Test edilen 13 farklı alt algoritma (ve buna bağlı olarak belirlenmesi gereken 13 parametre) olduğundan dolayı, parametrelerin tüm kombinasyonlarının denenmesi zaman alıcı olacaktır. Bu sebeple, şu şekilde bir yöntem uygulanmıştır:

Cross-validation testinden elde edilen, 13 farklı alt algoritmanın başarım oranları kullanılarak, öncelikle en başarılı 4 algoritmanın (Yol Frekansları-3, Hitting Time, Oturum Yakınlığı, Tık Sayısı) farklı kombinasyonları makul bir aralık arasında denenmiş, Hybrid-1, Hybrid-3 Weighted Borda ve Hybrid-4 Weighted Voting algoritmalarının her biri için en iyi parametreler (alt metod katsayıları) belirlenmiştir (Hybrid2 Borda Count algoritmasının katsayısı olmadığından, gerçeklemede sabit olarak 1 alınmıştır).

Hibrid metodlarda daha yüksek başarım oranlarının yakalanması için başka parametre belirleme yöntemlerinin kullanılması ya da geliştirilmesi mümkün olmakla birlikte, bu çalışmanın kapsamı haricinde görüldüğünden ileride yapılabilecek bir çalışma olarak bırakılmıştır. Bu tez çalışmasında belirlenen katsayılar en iyi/optimum olmayabilir. Ancak bu halleriyle bile (Bölüm 3.4'de görüldüğü üzere) temel alınan ve karşılaştırılan yönteme (HT) göre çok daha başarılı sonuçlar elde edilmiştir.

3.3.6. Sorgu Önermelerin Değerlendirilmesi

Rastgele sorgulardan farklı algoritmalarla üretilen ve veritabanına kaydedilen sorgu önermelerinin değerlendirilebilmesi için basit ama etkili bir web arayüzü hazırlanmıştır. Değerlendiriciler bir kullanıcıadı ile sisteme giriş yapmış ve her bir sorgu önermesi için “Çok İlgili”, “İlgili”, “Kötü/Az İlgili”, “Tamamen İlgisiz” seçeneklerinden birini seçmiştir. Web arayüzünün bir ekran görüntüsü Şekil 12'de gösterilmiştir.

Değerlendirmelerin web arayüzünde ve kaydedilen veritabanında 0-3 arasında (3: Çok ilgili) değerleri oluşmuştur. Her bir sorgunun farklı dört algoritma tarafından hesaplanan farklı önermeleri, farklı değerlendiriciler tarafından puanlanmıştır. Her bir algoritma ve sorgu her bir değerlendirici tarafından bir defa puanlanmıştır. Değerlendirme arayüzünün sonuçları puan, algoritma ve sorgunun tipi (nadir, çok kullanılan, orta) ile beraber kaydedilmiştir.

Sorgu/Arama: **atom nedir**

No	Sorgu Önermeleri	Puanlama
1	atom bilim insanları	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
2	ingilizce konu anlatımı sesli	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
3	atom taşı	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
4	fen ve teknoloji atom	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
5	tarih boyunca atom kavramı	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
6	atomun keşfi	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
7	atom dan küçük cisim	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
8	bohr atom modeli	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
9	atom fikirleri	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders
10	güneş enerjisinin ısıya dönüşümü	○ Çok iyi ○ İyi ○ Kötü ○ Çok kötü, ilgisiz ders

Kaydet / İleri ->>

Şekil 12: Sorgu Önerme Değerlendirme/Anket Arayüzü

Değerlendiriciler sorgu önermelerini puanlarken, önerme üretilirken kullanılan algoritmadan habersizdirler. Bu sayede, değerlendiriciler tüm önermelere karşı objektif olmak durumunda olmuşlardır.

Değerlendirmede kullanılan ilgililik skorları/puanlama Tablo 8'de gösterilmiştir. Değerlendiriciler, dersi tamamen farklı olan önermeler için 0 puan kullanmış, eğer dersi aynı ise en az 1 puan kullanmıştır. Değerlendiricilerin verdiği puanlarla ilgili kappa mutabakatları Bölüm 3.4.1'de anlatılmıştır.

Tablo 8: Değerlendirmede Kullanılan Puanlamalar

Skor/Puan	**Anlamı**
0	Çok kötü, tamamen ilgisiz, ders harici
1	Kötü, biraz ilgili
2	İyi, ilgili
3	Çok iyi, çok ilgili

3.4. BULGULAR VE DENEY SONUÇLARI

Bu bölümde, gerçekleştirilen deney ve değerlendirme sonuçları sunulmuştur. Öncelikle, dört değerlendirici arasındaki mutabakatlar Cohen's Kappa ölçütü/metriği ile analiz edilmiştir. Daha sonra, geliştirilen sorgu önerme algoritmalarının başarım performansı ortalama başarım ve DCG *(Discounted Cumulative Gain)* olmak üzere iki farklı metrikle ölçülmüştür. En son olarak, elde edilen sonuçlar üzerinde istatistiksel başarım testleri *(statistical significance tests)* uygulanmıştır.

3.4.1. Değerlendirici Mutabakatları

Kappa ağırlıklı istatistiği (Cohen, 1968) bir deney/anket çalışmasında iki değerlendirici arasındaki mutabakatı, değerlendirmede ne kadar mutabık kaldıklarını ölçer. Yapılan deney çalışması sonucu, değerlendiricilerin ikişerli gruplar halinde ağırlıklı Kappa değeri hesaplanmış ve daha sonra bunların ortalaması hesaplanmıştır. Ortalama Cohen Kappa değeri **0.37** olarak hesaplanmış olup bu değer makul bir mutabakat olarak değerlendirilmektedir (Landis & Koch, 1977). Bu hesaplama yönteminde, sıfırın altındaki değerler bir mutabakat olmadığını, sıfırın üstündeki değerler bir miktar mutabakat olduğunu ve 1 değeri de mükemmel mutabakat olarak ideal durumu gösterir. Yapılan bu hesaplama ile, uygulanan deney/anket sonuçlarında makul bir seviyede mutabakat olduğu gösterilmiştir.

Kappa mutabakatının değerlendirilmesi için önerilen ölçüt Tablo 9'da gösterilmiştir (Landis & Koch, 1977). Bu tabloya göre 0.37 olarak ölçülen kappa değeri makul mutabakat aralığının üst sınırına yakındır.

Tablo 9: Kappa Mutabakatının Değerlendirilmesi

Kappa değeri (k)	Mutabakat *(Agreement)*
< 0	Zayıf *(Poor)*
0 – 0.20	Hafif *(Slight)*
0.21 – 0.40	Makul *(Fair)*
0.41 – 0.60	Orta *(Moderate)*
0.61 – 0.80	Yüksek *(Substantial)*
0.81 – 1.00	Mükemmel *(Almost perfect)*

3.4.2. Yeni Algoritmaların Performansları

Deneyi yapılan sorgu önerme algoritmalarının performansları iki farklı metrik (ölçüt) kullanılarak ölçülmüştür. İlk metrik, her bir algoritma için, sorgu önermelerine verilen (0 ile 3 arasındaki) değerlendirici puanlamalarına dayanan ortalama ilgililik *(average relevance)* değerini kullanır. Her bir algoritma için ayrıca, (nadir, orta ve sık kullanılanlar olmak üzere) farklı frekanslardaki başarım oranları da hesaplanmıştır. İkinci metrik olarak, sorgu önermelerinin sıralamalarının da hesaba katıldığı NDCG (Normalized Discounted Cumulative Gain)'e göre başarım oranları ölçülmüştür.

3.4.2.1. Ortalama Başarım Ölçümü

Öncelikle her bir sorguya ait ortalama ilgililik hesaplanmış, daha sonra her bir algoritma için, o algoritmaya ait sorguların ortalaması alınmak suretiyle algoritmanın

ortalama ilgililik değeri hesaplanmıştır. Tablolarda ve grafiklerde isimleri ve kısaltmaları verilen yöntemlerin kısa açıklamaları Tablo 10'da verilmiştir.

Tablo 10: Kısaltma ve Yöntem Açıklamaları

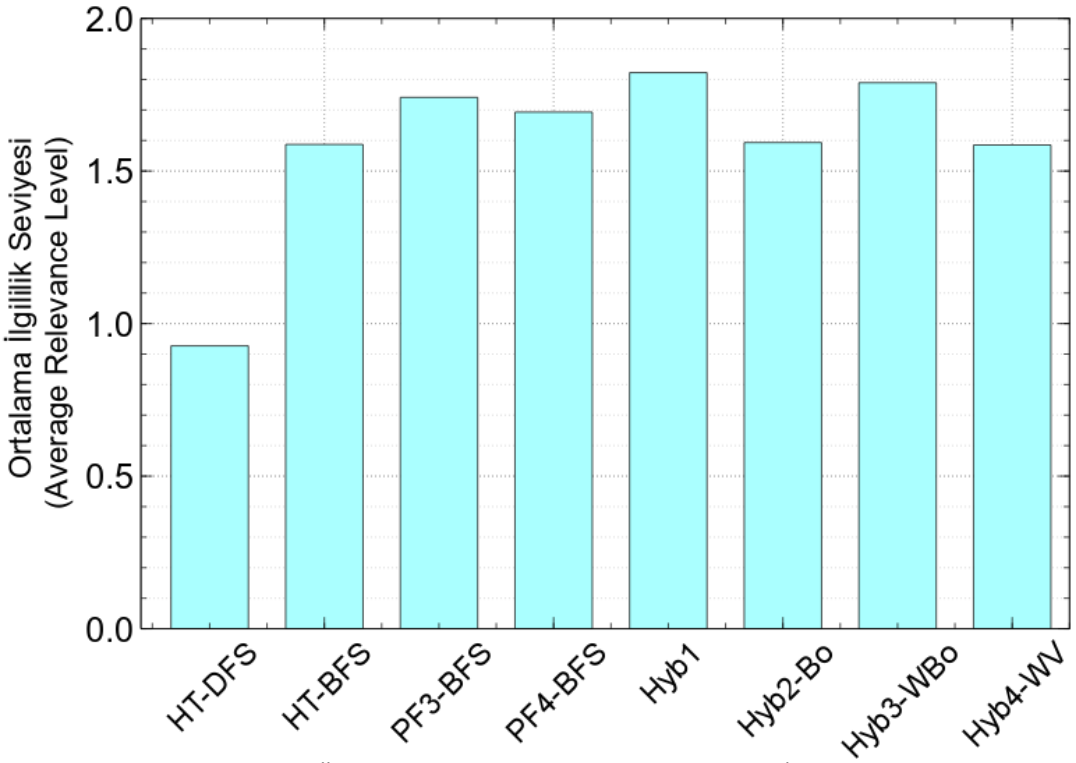
Kısaltma (varsa)	**Uzun İsmi**	**Açıklaması**
HT-BFS	Hitting Time BFS	BFS algoritmasını kullanan Hitting Time
HT-DFS	Hitting Time DFS	DFS algoritmasını kullanan Hitting Time
PF3-BFS	Path Freq-3 BFS	BFS ile Yol Frekansları-3 algoritması
PF4-BFS	Path Freq-4 BFS	BFS ile Yol Frekansları-4 algoritması
Hyb1	Hybrid-1 BFS	BFS ile gezilen, Ağırlıklı skor temelli birleştirme yapılan hibrid algoritma
Hyb2-Bo	Hybrid-2 Borda Count	BFS ile gezilen, Borda Count temelli birleştirme yapılan hibrid algoritma (bu yöntemde katsayı yoktur)
Hyb3-WBo	Hybrid-3 Weighted Borda Count	BFS ile gezilen, Weighted Borda Count temelli birleştirme yapılan hibrid algoritma
Hyb4-WV	Hybrid-4 Weighted Voting	BFS ile gezilen, Weighted Voting temelli birleştirme yapılan hibrid algoritma

Test edilen algoritmalardaki başarımlar ve iyileşme oranları Şekil 13 ve müteakip şekillerde gösterilmiştir. Şekil 13'de görüldüğü üzere, en düşük başarım oranı Hitting Time-DFS metodunda elde edilmiştir. Önerilen Yol Frekansları ve Hibrid algoritmalar en yüksek başarım oranlarını yakalamaktadır. Hibrid olmayan metodlarda Yol Frekansları-3 algoritması en iyi performansı yakalarken, Hibrid-1 algoritması tümü içinde en başarılısıdır.

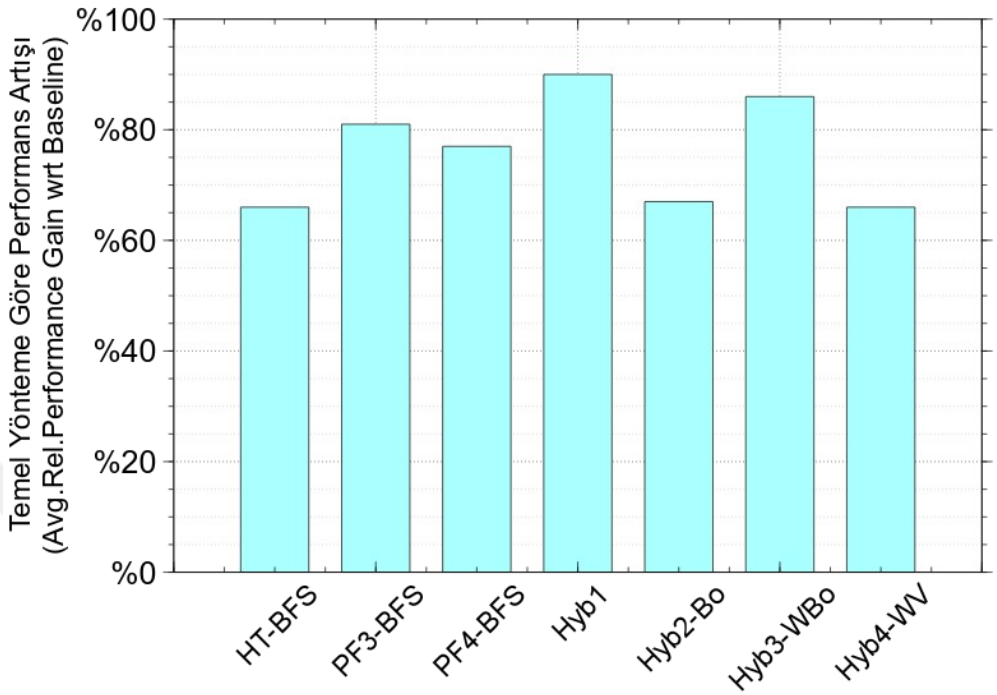
Şekil 14'de, yeni önerilen algoritmaların, Hitting Time-DFS algoritmasına göre başarım artışları gösterilmiştir. Görüldüğü üzere yeni metodlar, önceki Hitting Time-DFS yöntemine göre %66-90 arasında başarım artışı sağlamaktadırlar.

Yapılan ayrı bir testte, yeni önerilen ve gerçekleştirilen algoritmalar tarafından üretilen önermeler ayrıca incelenmiş ve genel kontroller kullanılmasa bile, eski

metodlarda önerilen bazı hatalı/anlamsız sorguların bu yeni yöntemlerde gösterilmediği, elendiği gözlenmiştir. Bu açıdan, genel kontroller (Bölüm 2.2.2) olmaksızın veya zayıf kontrollere rağmen bile yeni yöntemlerde sonuçların anlamlı olduğu gözlenmiştir.



Şekil 13: Sorgu Önerme Algoritmalarının Ortalama İlgililik Skorları



Şekil 14: Yeni Algoritmaların temel Hitting Time DFS'e Göre Ortalama İlgililik Skor Artış Yüzdeleri

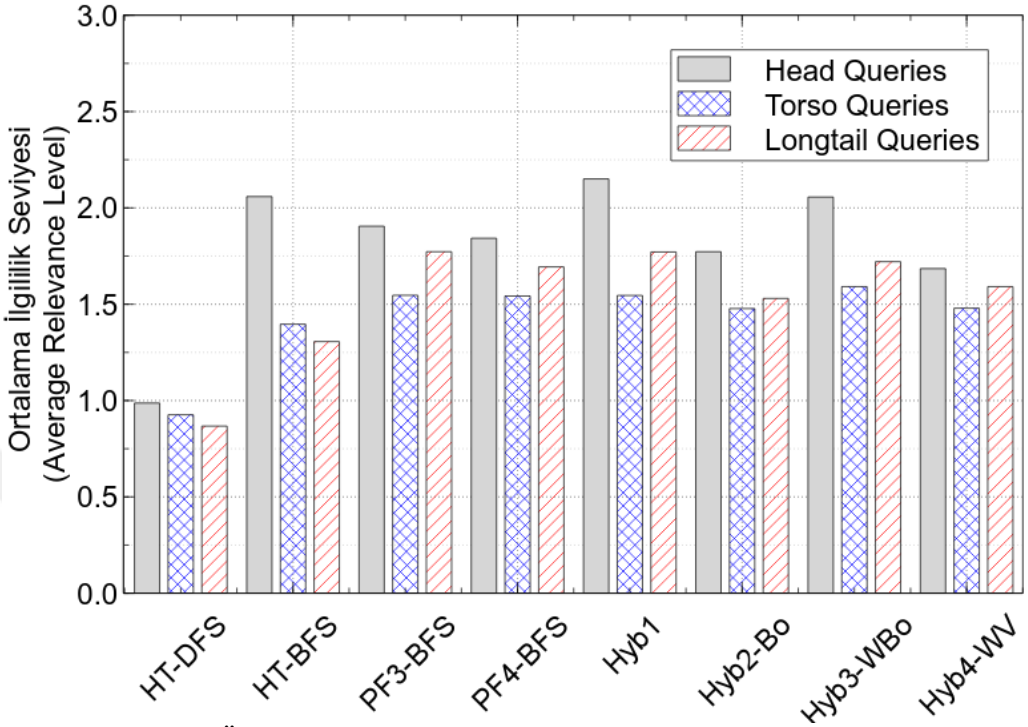
Hibrid-1 algoritması, %90 başarım artışı ile, tüm algoritmalardan üstün başarım sergilemiştir. Skor temelli birleştirme kullanan bu algoritmanın bu başarımı, skor temelli birleştirme yönteminin sıra temelli birleştirmeden daha başarılı olduğunu göstermektedir.

Her bir metodun kendi içerisinde farklı sorgu tipleri (nadir, sık kullanılan, orta sorgular) için performansları Şekil 15'de skor olarak gösterilmekte, Şekil 16'da Hitting Time-DFS'e göre yüzde artışı olarak gösterilmektedir. Tüm algoritmalar için, sık kullanılan sorgularda başarımların o algoritmanın kendi içinde en yüksek olduğu görülmektedir. Bu durum, sık kullanılan sorgular için eldeki veri boyutu orta ve nadir kullanılan sorgulardan daha fazla olduğundan kaynaklanabilir. Tüm algoritmalar ve sorgu tipleri için, yeni algoritmaların daha başarılı olduğu Şekil 15 ve Şekil 16'dan görülmektedir. Hitting Time-BFS algoritmasının, sık kullanılan sorgular için Hibrid-1'e çok yakın başarım gösterdiği görülmektedir. Hitting Time-BFS, orijinal Hitting

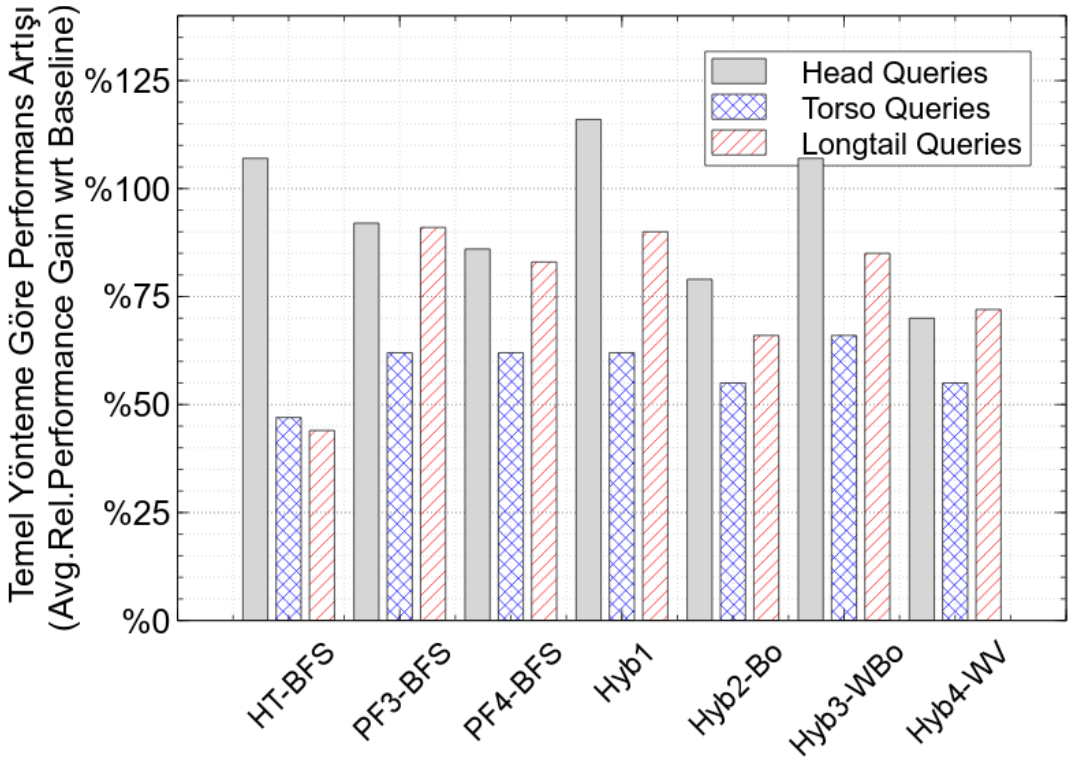
Time algoritmasının hafif değiştirilerek BFS grafik gezinme yönteminin kullanıldığı algoritmadır.

Nadir kullanılan sorgularla orta sorguların başarımlarının her algoritmanın kendi içinde benzer olduğu görülmektedir. Yol frekansları ve hibrid algoritmalar için nadir kullanılan sorguların başarımları orta kullanılanlara göre hafif daha iyidir. Halbuki, Hitting Time DFS ve BFS metodları için nadir kullanılan sorgular en az başarıma sahiptir. Hibrid metodlar için nadir sorguların daha başarılı olması, çerçeve yapıda önerildiği üzere, birden fazla metodun birleştirilmesinin faydalı olduğunu göstermektedir.

Şekil 16'da görüldüğü üzere tamamen yeni geliştirilen algoritmalar, temel alınan algoritmadan belirgin şekilde daha iyi (orta sorgular için %90 civarı, sık kullanılan sorgular için %116 civarı, nadir sorgular için %65 civarı) sonuçlar verdiği, Hitting Time-BFS'e göre bile daha başarılı olduğu görülmektedir. Tüm yeni algoritmalarda, nadir kullanılan sorgular için başarım artışlarının orta kullanılan sorgulardan fazla olduğu ve hatta sık kullanılana yakın olduğu görülmektedir. Yol frekansları metodlarında bu durum belirgin görülmekte olup, bu algoritmanın başarısını göstermektedir.



Şekil 15: Sorgu Önerme Algoritmalarının Sık, Orta ve Nadir (Head, Torso, Tail) Sorgular için Ortalama İlgililik Skorları



Şekil 16: Sık, Orta ve Nadir (Head, Torso, Tail) Sorgular için Yeni Yöntemlerin Ortalama İlgililik Performans Artış Yüzdeleri

Bu başarım karşılaştırmaları yapılırken şu iki husus göz önünde bulundurulmalıdır:

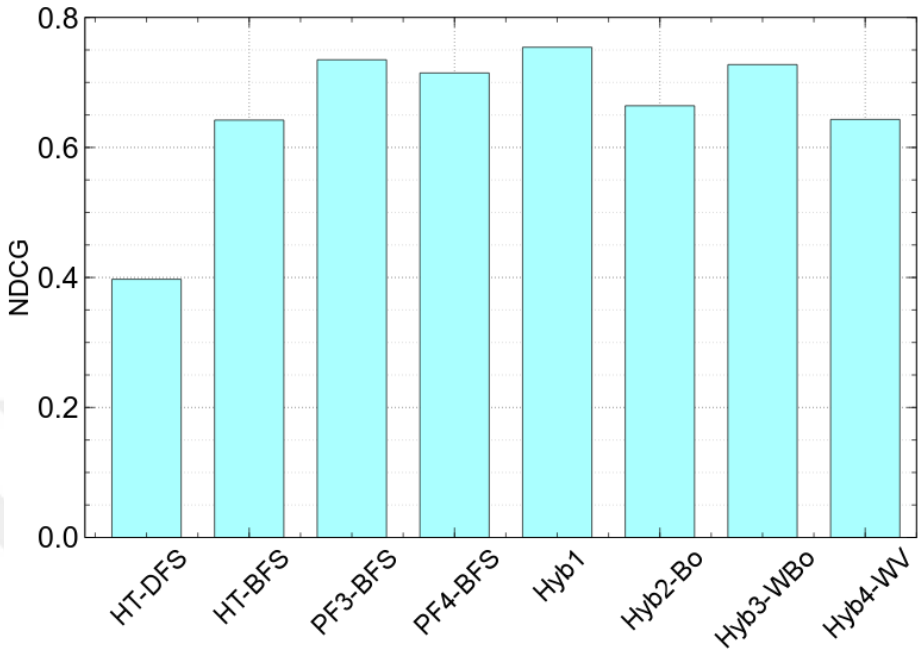
1. Genel kontroller prosedürü, genel başarımı artırmak ve hatalı/anlamsız sorguları elemek adına (temel alınan algoritma dahil) tüm algoritmalara uygulanmıştır. Eğer bu olmaksızın temel alınan orijinal algoritmayı kullanılsaydı, başarım artışı daha da fazla olurdu.

2. Burada yapılan sorgu önermelerin sonuçlarının "ilgililiği" değerlendirilmeye çalışılmıştır. Algoritmaların gerçekleştirilmiş hallerinin "performans değerlendirmesi", yani ne kadar sürede ve hangi kaynaklarla çalıştığına dair bir değerlendirme yapılmamıştır. (Ancak, uygun önbellek mekanizmaları ile bu metod ve yazılımların gerçek zamanlıya yakın olarak çalışacağı düşünülmektedir.)

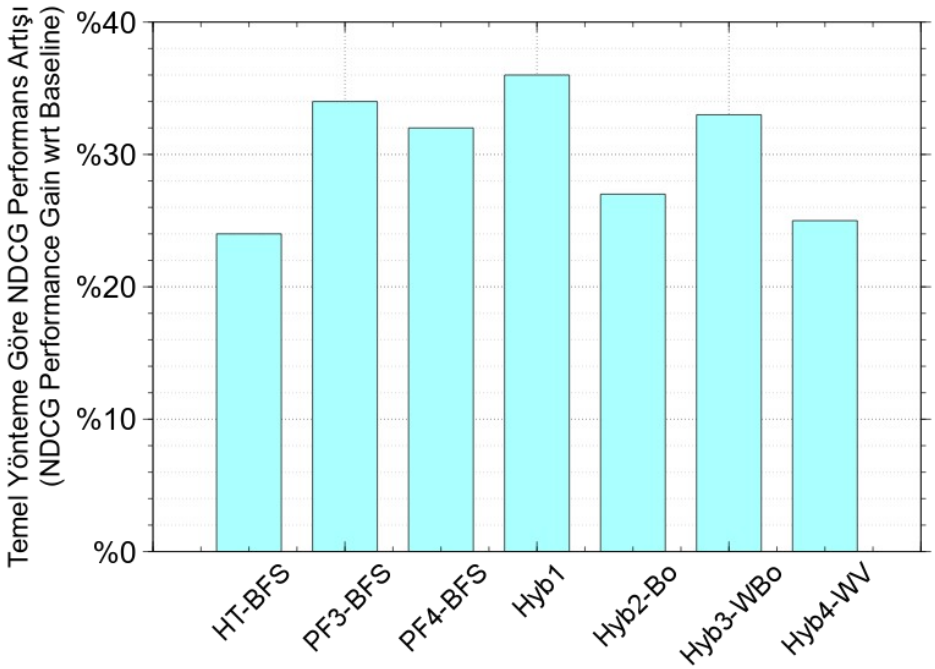
3.4.2.2. NDCG (Normalized Discounted Cumulative Gain) Ölçümü

Sorgu önerme algoritmalarının başarımını/performansını ölçmek üzere diğer bir değerlendirme ölçütü olarak NDCG *(Normalized Discounted Cumulative Gain)* kullanılmıştır. Arama motorları literatüründe bilinen bu ölçüt (metrik) ilgililik seviyesine ek olarak arama sonuçlarındaki sıralamaları da hesaba katar. Şekil 17'da algoritmaların top-10 sorgular için NDCG başarımları gösterilmiştir. NDCG hesaplanırken, öncelikle her bir sorgu ve değerlendirici için NDCG (0-3 arası) hesaplanmış, daha sonra bunların ortalaması alınmıştır.

Tüm yeni metodlar için NDCG metriğine göre de başarımın arttığı ve başarım artışlarının diğer metriğe paralel olarak gittiği açıkça görülmektedir. Ortalama başarım ölçütünde olduğu gibi Hibrid-1 algoritması için başarım en yüksektir. Şekil 18'de ise, yeni algoritmaların NDCG metriğine göre temel alınan *(baseline)* algoritmadan başarım artışını göstermektedir. Hibrid-1 ve Yol Frekansları-3 algoritmaları yüksek başarım artışı sağlarken, Hibrid Borda ve Hibrid Weighted Voting algoritmalarının başarım artışları HT-BFS'e yakındır. Buradan, sıra temelli birleştirme kullanan Hibrid Borda ve Hibrid Weighted Voting metodlarının, skor temelli birleştirme gibi yüksek başarım artışı sağlamadığı çıkarılabilir.



Şekil 17: Algoritmaların NDCG Performansları



Şekil 18: Hitting Time-DFS Temel Metoduna Göre Yeni Algoritmaların Performans Artışı (NDCG Metriğine Göre)

3.4.3. İstatistiksel Testler

Yapılan deneylerin sonuçlarını değerlendirmek ve sağlamasını yapmak için istatistiksel belirginlik testleri yapılmıştır. Deneylerde ölçülen sekiz farklı sorgu önerme algoritması vardır.

Örnek kümesi içindeki 60 sorguya ait değerlendiriciler tarafından belirlenen ilgililik skorları kullanılarak her bir algoritma çifti üzerinde eşli t-testi *(paired t-test)* uygulanmıştır. Bu şekilde, (düşük performanslı olduğu tahmin edilen ya da diğer metriklerle bilinen) bir algoritmadan (yüksek performanslı olduğu tahmin edilen ya da diğer metriklerle bilinen) diğerine geçişte ilerleme olup olmadığı anlaşılacaktır. Buradaki sıfır hipotezimiz *(null hypothesis)* karşılaştırılan iki algoritma arasında istatiksel olarak belirgin bir fark olmamasıdır.

İlerlemenin görülebilmesi için, temel alınan Hitting Time-DFS ile, onun hafif değiştirilmiş hali olan Hitting Time-BFS algoritmasından diğer yeni ve belirgin algoritmalara geçişteki değerler göz önüne alınmıştır. Buna göre, Tablo 11'da altı algoritma çiftine ait hesaplanan istatistiksel test sonuçları verilmiştir. Her bir teste ait NDCG ve ortalama ilgililik *(Average Relevance)* metriklerine göre p-değeri verilmiştir. Ölçülen temel metoda (Hitting Time DFS) göre diğer tüm metodların istatiksel olarak çok belirgin ilerleme kaydettiği görülmektedir. Hitting Time-BFS baz alınırsa NDCG metriğine göre yeni yöntemlerin belirgin ilerleme sağladığı; Ortalama İlgililiğe göre ise Yol Frekansları-3'de belirgin ilerleme olmazken, Hibrid BFS'e geçişte belirgin bir ilerleme sağlandığı görülmüştür. Bu sonuçlar, yeni algoritmalarımızın belirgin şekilde ilerleme kaydettiğini istatiksel olarak da göstermektedir.

Tablo 11: Sorgu Önerme Algoritma Çiftlerinin Eşli t-test Sonuçları

Algoritma 1	Algoritma 2	NDCG metriği için p-değeri	Ortalama ilgililik için p-değeri	NDCG için anlamı	Ortalama ilgililik için anlamı
Hitting Time-DFS	Yol Frekansları-3 BFS	5.58×10^{-13}	3.85×10^{-12}	Çok Belirgin İlerleme	Çok Belirgin İlerleme
Hitting Time-DFS	Yol Frekansları-4 BFS	1.30×10^{-12}	3.07×10^{-13}	Çok Belirgin İlerleme	Çok Belirgin İlerleme
Hitting Time-DFS	Hibrid-1 BFS	8.22×10^{-14}	7.68×10^{-13}	Çok Belirgin İlerleme	Çok Belirgin İlerleme
Hitting Time-DFS	Hibrid-2 Borda Count	4.09×10^{-11}	5.90×10^{-13}	Çok Belirgin İlerleme	Çok Belirgin İlerleme
Hitting Time-DFS	Hibrid-3 Weighted Borda Count	3.84×10^{-12}	1.39×10^{-14}	Çok Belirgin İlerleme	Çok Belirgin İlerleme
Hitting Time-DFS	Hibrid-4 Weighted Voting	1.92×10^{-10}	2.75×10^{-12}	Çok Belirgin İlerleme	Çok Belirgin İlerleme
Hitting Time-BFS	Yol Frekansları-3 BFS	0.007	0.123	Belirgin İlerleme	Belirgin İlerleme yok
Hitting Time-BFS	Hibrid-1 BFS	0.001	0.0381	Belirgin İlerleme	Belirgin İlerleme

3.5. İleride Yapılabilecek Çalışmalar

Bu çalışmanın en önemli ve belirgin yönlerinden birisi, oluşturulan ve önerilen yöntemlerin geliştirilmeye açık ve modüler olmasıdır. Hem yeni algoritmaların geliştirilmesini kolaylaştırıcı bir yöntem önerilmiş, hem de geliştirilecek yeni yöntem ve algoritmaların var olan yöntemlerle ya da birbirleriyle kolay ve pratikçe birleştirilebilmesi için gerekli modüler mimari önerilmiştir. Bu sebeplerle, sorgu önerme alanında ileride yapılabilecek birçok çalışma için bir temel oluşturulmuştur.

Aşağıda belirtilen konularda (ve başka konularda) sorgu önerme ile ilgili yeni çalışmalar yapılabilir:

3.5.1. Yeni Sorgu Özellikleri

Önerilen ve önceden bilinen sorgu önerme algoritmalarının temelinde kullanıcıların girdikleri sorguların çeşitli özellikleri bulunmaktadır. Yeni sorgu özelliklerinin bulunması ya da dizayn edilmesi ile sorgu önermede yeni algoritmalar tasarlanabilir ve yeni başarım oranları yakalanabilir.

3.5.2. Yeni Sorgu Karşılaştırma Yöntemlerinin Bulunması

Bu tez çalışmasında önerilen mimaride, temel yaklaşım sorguların karşılaştırılmasıdır. Yeni karşılaştırma yöntemlerinin bulunması ile ya da bunların daha pratik hale getirilmesi, optimize edilmesi ile sorgu önerme başarımları artırılabilir. Sorguların karşılaştırılmasında çoğunlukla sorgu özellikleri kullanılabilir.

3.5.3. Mimaride Yapılabilecek İyileştirmeler

Sorgu özellikleri ve sorgu karşılaştırmada yapılabilecek iyileştirmelere ek

olarak, mimaride de iyileştirmeler önerilebilir. Mimaride yapılabilecek iyileştirmeler, sorgu önerme performanslarını artırabilecektir. Mimariye eklenebilecek ve önerilecek yeni modüllerle, sorgu önerme performansı artırılabilir. Burada tek tek modül önerisi yapılmamıştır, sadece bir örnek verilmiştir.

3.5.4. İmla Denetimi Modülü

İmla denetim modülü, sorguların ve cevaplarının kalitesini artırabilecektir. Uygun şekilde mimariye eklenecek bir imla denetim modülü ile sorgu önerme performansının artabileceği düşünülmektedir.

3.5.5. Veri Kalitesinin Artırılmasına Yönelik Çalışmalar

Sorgu önerme algoritmalarına girişi yapılan verilerdeki bozukluk ve yanlışlıklar başarımı düşürebilir. Bu sebeple, kullanılan veri de incelenerek yapılabilecek veri kalitesinin artırılmasına yönelik çalışmalar ile, sorgu önerme başarımı da artabilecektir. Mimarinin sıralamadan önceki aşamasında ve sonrasında kontrol aşaması önerilmiştir. Bu kontrol aşamasının, üzerinde çalışılan veri kümesi de gözlemlenerek geliştirilmesi, yeni kontrollerin eklenmesi ile sorgu önerme başarımının artabileceği değerlendirilmektedir.

3.5.6. Performans İyileştirmeleri

Bu tez çalışmasında önerilen yöntemler için performans ölçümleri yapılmamış, kaynaklar açısından optimum olup olmadıkları yönünde bir değerlendirmede bulunulmamıştır. Gerçek zamanlı olarak çalışacak gerçek bir sistemde, burada önerilen yöntemlerin yeniden tasarlanarak optimize edilmesi faydalı olabilecektir.

SONUÇ

Bu tez çalışmasında kullanıcıların arama motorlarına girdiği sorguya alternatif olarak faydalı olabilecek diğer sorguların önerilmesi problemi üzerinde durulmuş, var olan yöntemler incelenmiş ve yeni yöntemler önerilmiştir.

Sorgu önerme probleminin "sorguların karşılaştırılması" problemine indirgenebileceği gösterilmiş, birden çok sorgu önerme algoritmalarını içerebilen modüler bir çerçeve yapı önerilmiş ve gerçeklenmiştir. Önerilen çerçeve yapı ve problem indirgemesi, sorgu önerme probleminin daha yapısal bir şekilde çözülmesini sağlamakta, parçalara bölmekte, bir geliştiricinin sorgu önerme probleminin tamamına bir çözüm getirmese bile, sadece bir basamağına bile katkıda bulunabilmesini sağlamakta, bu konuda algoritma geliştirilmesini kolaylaştırmakta ve pratik hale getirmektedir.

Sorguların eğitimle ilgili ve diğer özelliklerini kullanabilen 13 farklı sorgu önerme algoritması geliştirilmiş, gerçeklenmiş ve önerilen çerçeve yapı içerisinde kullanılmıştır. Var olan ve yeni geliştirilen algoritmalarının başarımlarını ölçmek için gerçek bir eğitim arama motorunun sorgu kayıtları ile deney çalışmaları yapılmış, deney sonuçlarının kalitesi ölçülmüş, yeni önerilen algoritmaların, sorgu tıklama grafiği kullanan yöntemler içinde temel alınan algoritmaya göre %66-90 oranlarında başarım artışı sağladığı gösterilmiş, yapılan ölçümler istatistiksel olarak da (p-test, kappa ile) teyid edilmiştir.

Bundan sonra yapılacak çalışmalarda, sorgu kayıtlarının başka özelliklerinin, kullanıcının demografik ve bağlamsal özelliklerinin, içerik temelli yöntemlerin kullanılması ile yeni sorgu önerme algoritmalarının üretilmesi ve önerilen çerçeve mimariye kolayca eklenmesi ya da kullanılması mümkündür. Bir imla denetim modülünün eklenmesi sorgu önerme kalitesini artırabilir.

KAYNAKLAR

Arora, M., & Duhan, N. (2013). Design of query suggestion system using search logs and query semantics. *International Journal of Application or Innovation in Engineering & Management (IJAIEM)*, *2*(6), 302–313. Retrieved from http://www.ijaiem.org/Volume2Issue6/IJAIEM-2013-06-25-075.pdf

Aslam, J. a, & Montague, M. (2001). Models for Metasearch. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '01* (pp. 276–284). http://doi.org/10.1145/383952.384007

Babuscu, F., & Özcan, R. (2014). How students use search engines for school informational tasks. In *Proceedings of International Conference on e-Education* (pp. 97–107).

Baeza-Yates, R., Hurtado, C., & Mendoza, M. (2005). Query Recommendation Using Query Logs in Search Engines. In *Current Trends in Database Technology - EDBT 2004 Workshops* (Vol. 3268, pp. 395–397). Springer. http://doi.org/10.1016/j.asoc.2007.11.004

Bhatia, S., Majumdar, D., & Mitra, P. (2011). Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval- SIGIR '11* (pp. 795–804). ACM. http://doi.org/10.1145/2009916.2010023

Boldi, P., Bonchi, F., Castillo, C., Donato, D., Gionis, A., & Vigna, S. (2008). The query-flow graph: model and applications (pp. 609–618). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1458163

Boldi, P., Bonchi, F., Castillo, C., Donato, D., & Vigna, S. (2009). Query suggestions using query-flow graphs. In *Proceedings of the 2009 workshop on Web Search Click Data - WSCD '09* (pp. 56–63). ACM. http://doi.org/10.1145/1507509.1507518

Borda, J. C. (1781). Memoire sur les elections au scrutin. *Histoire de l'Academie Royale Des Sciences*, *5*, 657–665.

Bordogna, G., Campi, A., Psaila, G., & Ronchi, S. (2012). Disambiguated query suggestions and personalized content-similarity and novelty ranking of clustered

results to optimize web searches. *Information Processing and Management (IPM)*, *48*(3), 419–437. http://doi.org/10.1016/j.ipm.2011.03.008

Cao, H., Jiang, D., Pei, J., He, Q., Liao, Z., Chen, E., & Li, H. (2008). Context-aware query suggestion by mining click-through and session data. In *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '08* (pp. 875–883). ACM. http://doi.org/10.1145/1401890.1401995

Christopher D. Manning, Prabhakar Raghavan, & Hinrich Schütze. (2009). *An Introduction to Information Retrieval*. Cambridge University Press. Retrieved from http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, *70*(4), 213–220.

Cui, H., Wen, J. R., Nie, J. Y., & Ma, W. Y. (2003). Query expansion by mining user logs. *IEEE Transactions on Knowledge and Data Engineering*, *15*(4), 829–839. http://doi.org/10.1109/TKDE.2003.1209002

Dwork, C., Kumar, R., Naor, M., & Sivakumar, D. (2001). Rank aggregation methods for the Web. In *Proceedings of the 10th international conference on World Wide Web - WWW'01* (pp. 613–622). ACM. http://doi.org/10.1145/371920.372165

Fonseca, B. M., Golgher, P. B., De Moura, E. S., & Ziviani, N. (2003). Using association rules to discover search engines related queries. In *Proceedings - 1st Latin American Web Congress: Empowering our Web, LA-WEB 2003* (pp. 66–71). IEEE. http://doi.org/10.1109/LAWEB.2003.1250284

He, Q., Jiang, D., Liao, Z., Hoi, S. C., Chang, K., Lim, E.-P., & Li, H. (2009). Web query recommendation via sequential query prediction (pp. 1443–1454). IEEE. Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4812545

Huang, C. K., Chien, L. F., & Oyang, Y. J. (2003). Relevant term suggestion in interactive web search based on contextual information in query session logs. *Journal of the American Society for Information Science and Technology (JASIST)*, *54*(7), 638–649. http://doi.org/10.1002/asi.10256

Jansen, B. J., & Spink, A. (2006). How are we searching the World Wide Web? A comparison of nine search engine transaction logs. *Information Processing and Management (IPM)*, *42*(1 SPEC. ISS), 248–263. http://doi.org/10.1016/j.ipm.2004.10.007

Kraft, R., & Zien, J. (2004). Mining Anchor Text for Query Refinement. In *Proceedings of the 13th international conference on World Wide Web - WWW'2004* (pp. 666–674). ACM. http://doi.org/10.1145/988672.988763

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, *33*(1), 159–174. http://doi.org/10.2307/2529310

Lau, Tessa, Horvitz, E. (1999). UM99 User Modeling. *Proceedings of the Seventh International Conference on User Modeling*, *407*, 119–128. http://doi.org/10.1007/978-3-7091-2490-1

Lewis, D. D., & Croft, W. B. (1990). Term clustering of syntactic phrases. In *Proceedings of the 13th international ACM SIGIR conference on research and development in information retrieval - SIGIR '90* (pp. 385–404).

Mei, Q., Zhou, D., & Church, K. (2008). Query suggestion using hitting time. In *Proceeding of the 17th ACM conference on Information and knowledge management - CIKM '08* (pp. 469–478). ACM. http://doi.org/10.1145/1458082.1458145

Meng, L., Huang, R., & Gu, J. (2014). Query Suggestion Based on Theme and Context 1. *International Journal of U-and E-Services, Science and Technology*, *7*(4), 263–276.

Parikh, N., Singh, G., & Sundaresan, N. (2013). Query Suggestion with Large Scale Data (Vol. 31, pp. 493–518). Elsevier.

Renda, M. E., & Straccia, U. (2003). Web Metasearch: Rank vs. Score Based Rank Aggregation Methods. *Proceedings of the 2003 ACM Symposium on Applied Computing - SAC '03*, 841–846. http://doi.org/10.1145/952532.952698

Silvestri, F. (2010). *Mining Query Logs: Turning Search Usage Data into Knowledge*. *Foundations and Trends® in Information Retrieval* (Vol. 4). http://doi.org/10.1561/1500000013

Sinanoglu, O. (1988). The new pictorial structural covariance method for qualitative quantum chemistry. III: Fused polycyclics and their ions. *Journal of Mathematical Chemistry*, *2*(2), 137–154. http://doi.org/10.1007/bf01165925

Torres, D. S., Hiemstra, D., Weber, I., & Serdyukov, P. (2012). Query recommendation for children. In *Proceedings of the 21th ACM international conference on Information and knowledge management - CIKM '12* (pp. 2010–2014). ACM. http://doi.org/10.1145/2396761.2398562

Torres, D. S., & Weber, I. (2011). What and how children search on the web. In *Proceedings of the 20th ACM international conference on Information and knowledge management - CIKM '11* (pp. 393–402). ACM. http://doi.org/10.1145/2063576.2063638

Torres, D. S., Weber, I., & Hiemstra, D. (2014). Analysis of Search and Browsing Behavior of Young Users on the Web. *ACM Transactions on the Web*, *8*(2), 1–54. http://doi.org/10.1145/2555595

Usta, A., Altingovde, I. S., Vidinli, İ. B., Ozcan, R., & Ulusoy, Ö. (2014). How k-12 students search for learning? Analysis of an educational search engine log (pp. 1151–1154). ACM Press. http://doi.org/10.1145/2600428.2609532

Wang, X., & Zhai, C. (2008). Mining term association patterns from search logs for effective query reformulation. In *Proceeding of the 17th ACM conference on Information and knowledge management - CIKM '08* (pp. 479–488). ACM. http://doi.org/10.1145/1458082.1458147

Wen, J., Nie, J., & Zhang, H. (2001). Clustering user queries of a search engine. In *In Tenth ACM International World Wide Web Conference -WWW'01* (pp. 162 – 168). ACM. http://doi.org/10.1145/371920.371974

Yılmazel, Ö. (2011). Guiding students to answers: query recommendation. *Turkish Online Journal of Distance Education*, *12*(3), 85–94.

(Boş sayfa)

(Arka Kapak)