T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

MAKİNE ÖĞRENMESİ BAZLI ARAMA MOTORU

Yüksek Lisans Tezi

Tezi Hazırlayan:

Mehmet Fuat RIHTIM

İstanbul, 2018

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI
BİLGİSAYAR MÜHENDİSLİĞİ BİLİM DALI

MAKİNE ÖĞRENMESİ BAZLI ARAMA MOTORU

Yüksek Lisans Tezi

Tezi Hazırlayan:

Mehmet Fuat RIHTIM

Öğrenci No:

150820016

Danışman:

Yrd. Doç. Dr. Atınç Yılmaz

İstanbul, 2018

YEMİN METNİ

Yüksek Lisans Tezi olarak sunduğum "Makine Öğrenmesi Bazlı Arama Motoru" başlıklı bu çalışmanın, bilimsel ahlak ve geleneklere uygun şekilde tarafımdan yazıldığını, yararlandığım eserlerin tamamının kaynaklarda gösterildiğini ve çalışmamın içinde kullanıldıkları her yerde bunlara atıf yapıldığını belirtir ve bunu onurumla doğrularım. 18/01/2018

Mehmet Fuat RIHTIM

T.C.
BEYKENT ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZ SAVUNMA SINAVI SONUÇ TUTANAĞI

Beykent Üniversitesi
Fen Bilimleri Enstitüsü Müdürlüğü'ne,

Aşağıda tez adı belirtilen yüksek lisans öğrencisi...150820016...no'lu ...Mehmet Fuat Bahtım.....'in 13/02/2018 tarihinde yapılan tez savunma sınavı[1] sonucunda..40.. dakika süreyle sunduğu ve savunduğu tezi hakkında[2] oybirliğiyle, ..BAŞARILI.. kararı verilmiştir.

Bilgilerinize saygılarımızla arz ederiz.

Anabilim Dalı : Bilgisayar Mühendisliği
Programı : Bilgisayar Mühendisliği
Tez Başlığı[3] : Makine Öğrenmesi Bazlı Arama Motoru

Tez Sınav Jürisi	Öğretim Üyesi	İmza
Danışman	: Yrd.Doç.Dr. Atınç YILMAZ	
Üye	: Doç. Dr. Gökhan ŞİLAHTAROĞLU	
Üye	: Yrd.Doç.Dr. Ediz ŞAKOĞLU	

---

[1] Jüri üyeleri söz konusu tezin kendilerine teslim edildiği tarihten itibaren en geç bir ay içinde toplanarak öğrenciyi tez savunma sınavına alır. Belirlenen günde yapılamayan jüri toplantısı, katılanların hazırladığı bir tutanakla enstitü yönetimine bildirilir. Bu durumda jüri en geç onbeş gün içinde toplanarak adayı tez savunma sınavına alır. Tez savunma sınav süresi en az 45 dakikadır. Yüksek lisans tez savunma sınavı, tez çalışmasının sunulması ve bunu izleyen soru-yanıt bölümlerinden oluşur ve dinleyiciye açıktır. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-3)

[2] Tez sınavının tamamlanmasından sonra jüri, tez hakkında "kabul", "düzeltme" veya "red" kararı verir. Jüri başkanı, jüri üyelerince imzalanmış sınav tutanağını, tez sınavını izleyen üç gün içinde ilgili enstitü yönetimine teslim eder. Tezi başarısız bulunan öğrencinin Enstitü ile ilişiği kesilir. Tezi hakkında düzeltme kararı verilen öğrenci en geç üç ay içinde gerekli düzeltmeleri yaparak ve yönetmelikte belirtilen usullere uygun olarak tezini aynı jüri önünde yeniden savunur. Bu savunma sınavında da tezi kabul edilmeyen öğrencinin enstitü ile ilişiği kesilir. (Beykent Lisansüstü eğitim ve Öğretim Yönetmeliği-Madde30-4)

[3] İleride doğabilecek aksaklıkların engellenmesi için tezin başlığının yazılması gerekmektedir.

Adı ve Soyadı : Mehmet Fuat RIHTIM

Danışmanı : Yrd. Doç. Dr. Atınç Yılmaz

Türü ve Tarihi : Yüksek Lisans, 2018

Alanı : Bilgisayar Mühendisliği

Anahtar Kelimeler : Arama Motoru, Yeni Teknoloji, İlişkisel Olmayan Veritabanı, Makine Öğrenimi, Performans

ÖZ

MAKİNE ÖĞRENMESİ BAZLI ARAMA MOTORU

Bu çalışma, bir web uygulaması için daha efektif arama işlemlerinin yapılmasına olanak sağlamak için geliştirilmiştir. Entegre bir biçimde değil, ayrık bir sistem olarak tasarlanmıştır. Bu sayede, uygulama üzerinde yük oluşmayacağı gibi, arama motoru da özelleşmiş bir şekilde hizmet vermesi hedeflenmiştir.

Bu çalışmada, uygulamalara kolayca uyarlanabilecek ve ayrık şekilde çalışan bir tasarım yapılmıştır. Eski teknolojilerle, yeni teknolojinin birleştirilerek kullanılmasına olanak sağlayan bu uygulama, verilerin daha efektif bir biçimde kullanılarak, kullanıcılara performanslı bir kullanıcı deneyimi yaşamalarına olanak sağlamaktadır. Bu yeni teknolojiler, ilişkisel olmayan veritabanlarına duyulan ihtiyaç sonrası, son senelerde ortaya çıkması ve gelişmeye devam etmesiyle birlikte proye dahil edilmiştir. Kullanıcıların uygulama üzerinde yaptıkları işlemlerse, girilen veriler ile birlikte sistemin eğitilmesi konusunda sistemi beslemekte ve kullanıcılara daha doğru sonuçlar verilmesini sağlamaktadır.

Name and Surname : Mehmet Fuat RIHTIM

Supervisor : Yrd. Doç. Dr. Atınç Yılmaz

Degree and Date : Yüksek Lisans, 2018

Major : Bilgisayar Mühendisliği

Key Words : Search Engine, New Technology, Non-Relational Database, Machine Learning, Performance

ABSTRACT

MACHINE LEARNING BASED SEARCH ENGINE

In this work, an application is developed for providing users to search their texts in a different web site applications efficiently. This search engine is not integrated an application, it is designed as distributed. In this way, a web application do not responsible search processes so the search engine is aimed to serve data seperately.

In this work, a design was done with working other system distributedly and adapt to them easily. This application provides an easy and effective usage of data search with union of new and old technology. This is for user friendly application for users. These new technologies are presented after needs for non-relational database and these databases usage are increasing everyday with their improvements. In order to give more successful and meaningful results with the application, some machine learning algorithms are implementing into the application. For this reason, all data is entered by users are using for teaching the system.

İÇİNDEKİLER

ŞEKİLLER DİZİNİ

KISALTMALAR

HTML : Hyper Text Markup Language (Hiper Metin Biçimlendirme Dili)

SEO : Search Engine Optimization (Arama Motoru Optimizasyonu)

NoSQL : Not Only Structured Query Language (Sadece Yapılandırılmış Sorgu Dili Değil)

SQL : Structured Query Language (Yapılandırılmış Sorgu Dili)

MVC : Model View Controller

ADO : ActiveX Data Object (ActiveX Veri Objesi)

JSON : JavaScript Object Notation (JavaScript Nesne Notasyonu)

DBMS : Database Management System (Database Yönetim Sistemi)

SOAP : Simple Object Access Protocol (Basit Nesne Erişim Protokolü)

ORM : Object Relational Mapping (Nesne İlişkisel Haritalama)

DLL : Dynamic Link Library (Dinamik Bağlantı Kütüphanesi)

IOC : Inversion Of Control

1.GİRİŞ

1.1) Arama Motorları

Teknolojinin sadece gelişmekle kalmayıp, bu gelişime bağlı olarak gelişiminin de hızlandığı günümüzde, ihtiyaçlara yönelik yazılımların sayısı hızlı bir biçimde günden güne artmaktadır. Teknolojik gelişmelerin farklılaşarak, her alanda, daha hızlı bir biçimde gelişmesi ve insanların bu gelişimler sonucu donanımlarını artırması ile birlikte, özelleşmiş yazılımların sayısı da artmaktadır. Zaman kavramı ise bu gelişmelerden sonra git gide daha önemli hale gelmektedir.

İnternet ortamında bulunan bazı dev arama motorlarına karşın, uygulama bazında geliştirilen arama motorlarının gelişimine, uygulamaların diğer işlevleri kadar önem verilmemiştir. Ek olarak verilerinde parabolik olarak hızlı bir şekilde artışı, arama fonksiyonlarını çok önemli bir hale getirmiştir.

Sistemlerin gelişmesi sonucu, verilerin artışı da çok büyük oranda artmıştır. Emin Çapa sunumunda da belirttiği üzere, tüm insanlık tarihi kadar verinin artık bir sistemle 5 günde toplanabilmesine olanak sağlayacak kadar fazla veriyi insanoğlu elde edebilmektedir [1]. Bu verilerin toplanması kadar; işlenmesi, anlamlı hale getirilmesi, sunulması, başka araştırmalarda kullanılması ve yeni gelişmelere ışık tutması da önemlidir. The Atlantic'e göre 1994 yılında 3000 internet sitesi vardı ve bu sayı 2014 yılında 1 milyarı aştı [2]. Günümüzde ise 1 milyarlık rakam çok daha ileri gitmiştir.

Verilerin toplanmasının da bu kadar kolaylaştığı bir dönemde, çoğu kişi en kısa sürede ve en kolay şekilde istediği veriye erişmek istemektedir. Bu verinin doğruluğu, anlamlılığı çok önemlidir. Mevcut birçok uygulamada artık bir kutucuğa bir kelime yazıldığında, o uygulamanın içerisinde bulunan birçok sonuç dönülmekte ve kullanıcıların işleri kolaylaştırılmaya çalışılmaktadır. Son kullanıcıların bu durumla ilgili şikayetleri ise genelde sonuçların doğru gelmediği, işlevsel olmadığı ya da sistemlerin yavaşlığıyla ilgili olarak süreklilik arz etmektedir.

1.2) Arama Motorları ve Çalışma Mantığı

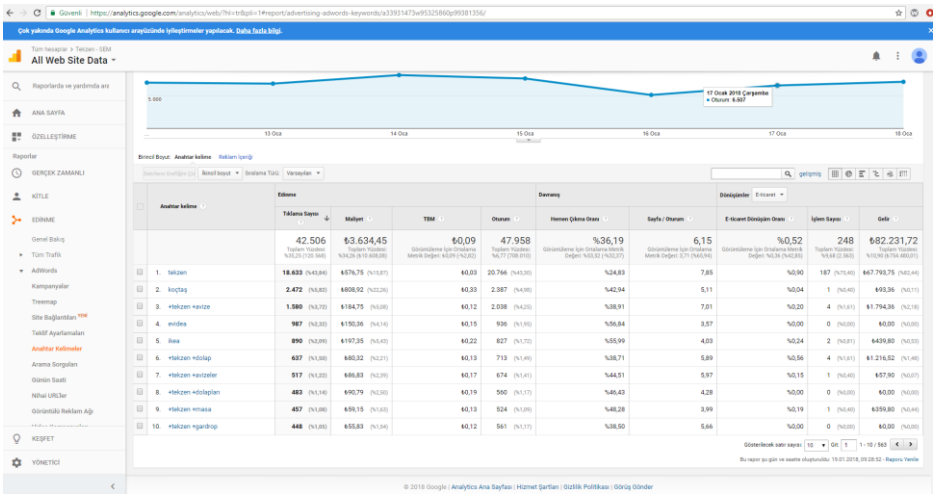
Genelde internet ortamında, bilinen arama motorlarının indeksleme yöntemleri uygulamalara uyarlansa da, uygulamaların kendi içindeki algoritmalar farklı çalışmaktadır ve bunlar kullanıcılara yeterli sonuç vermemektedir. Örnek vermek gerekirse; bir e-ticaret sitesinde bulunan bir ürünü Google'dan arattığınızda bulabilmeniz mümkünken, site

içerisindeki arama algoritmaları yeterli değilse; site içerisinden ürüne ulaşmanız daha zor olacaktır.

Herhangi bir kelimeyi uygulama içerisinde aramanın doğru olması kadar, hızlı sonuç vermesi de önemlidir. Bir veritabanı tablosunda bulunan milyonlarca kayıt içerisinden size en doğru sonuç, uzun bir süre sonra geliyorsa, bu arama algoritması efektif değildir. Bununla ilgili performans çalışmaları da, uygulama bazında yeteri kadar yapılmamaktadır.

Arama motorları, web crawling yöntemi yani internet emeklemesi ile internet ortamındaki siteleri gezer ve site üzerindeki tüm bilgileri toplar. Toplanan bilgilerin arama motorlarında indekslenmemesi için yazılımlar üzerinde tutulan özel dosyalar vardır ve bu dosyalara indekslenmemesi gereken sayfaların ismini yazarak, arama robotlarının indeksleme yapmasına engel olabilmekteyiz. Bu dosya robot.txt şeklinde adlandırılmaktadır ya da HTML içerisinde no follow etiketi ile bunu sağlayabiliriz. Veriler sitemizden alındıktan sonra arama motorlarının kendi filtrelerinden geçmektedir. Bunlar güvenlik için gerekli olup, gereksiz verilerden sistemin etkilenmemesi, zararlı sitelerin saptanması ve kullanıcıya sunulacak verinin anlamlı halde işlenmesi amaçlanmıştır.

Google'ın çalışma mantığında SEO'nun büyük bir önemi vardır. SEO, arama motoru nesnesi anlamına gelmektedir. SEO, hem internet sitesi sayfalarının sayfalarının arama motorlarının anlayabileceği şekilde yazılmasına ihtiyaç duyar. Buna uygun yazılmış internet sitelerinden anahtar kelimelerin alınması amaçlanmıştır. Bu anahtar kelimeler, sitelerin aramalar sonucunda, sonuçlar sırasında daha üstte gelmesini sağlar. Arama sisteminde, sitelerin aranmasıyla ilgili verimli sonuçlar alınması, bulunabilirliğin artması gibi sebepler için günümüzde bir çok SEO uzmanı bulunmakta ve sitelerin bu yönde optimizasyonlarına destek verdikleri gibi, Google'ın kendi sisteminde de gerekli işlemleri yapmaktadırlar. Şekil 1'de Google'ın yönetim paneli verilmiştir.



Şekil 1 – Google'ın anahtar kelime ve yönetim paneli

1.3) Tezin Amacı ve Literatüre Katkısı

Bu çalışma, uygulama bazında ElasticSearch, Kibana gibi yeni teknolojilerin, hedef veritabanları ile efektif bir şekilde, daha hızlı ve doğru bir biçimde istenilen verilere ulaşabilmesine olanak sağlayan bir servis şeklinde hazırlanacaktır. Makine öğrenimi de buna dahil edilmeye çalışılarak, kullanıcıların hedeflerine daha hızlı ve doğru bir biçimde ulaşması amaçlanmaktadır.

Literatür incelendiğinde, daha önce arama motoru ile ilgili araştırmalar yapılsa da, geniş ölçekli bir uygulamaya ya rastlanmamaktadır ya da eski teknolojiler kullanıldığı için yetersiz kalınmıştır ve başarısız sonuçlar alınmıştır. Bu projenin ilk önemli katkısı, yeni teknolojilerin kullanılmasıdır. Mevcut algoritmalar, eski teknolojilere yönelik olup, ne kadar efektif çalışırsa çalışsın, yeni teknolojilerin sağladığı bazı avantajları içermediğinden ötürü yeteri kadar verimli çalışmamaktadırlar.

Literatürdeki bu eksiklik düşünüldüğünde, uygulamanın modüler bir yapıda oluşu da çok önemlidir. Bir projenin alt yapısını değiştirmek, hem maddi anlamda, hem manevi anlamda büyük bir maliyet getirmektedir. Bu yüzden mevcut işleyişi bozmadan, yeni teknolojilerin ya da algoritmaların uygulanması, projenin tasarım deseninin buna göre seçilmesi ve uyarlanması, projenin sadece sürdürülebilirliği için değil, geleceği için de önemlidir. Teknoloji geliştikçe, yeniliğe ayak uydurmak gerekliliği her sektörde olduğu gibi, yazılım sektöründe de birinci önceliktir. Bu yüzden projenin dikey olarak değil, yatay olarak geliştirilmesi; projenin değişimi ve gelişimi için daha doğru ve faydalı olacaktır.

Ayrıca kullanılacak teknolojilerin dökümantasyonları ile ilgili ön araştırmalar yapılmış olup, yeteri kadar uygulama yazılmamış olması da; hem arama motorunun yeni teknolojilerle kullanılması, hem de yeni teknolojilerin dökümantasyonunun yapılarak, diğer türlü yazılımların yapılmasına da öncülük etmesi de ana hedeflerden olmuştur.

1.4) Tezin Ana Hatları

Uygulama, son kullanıcı deneyimi açısından basit bir yapıya sahip olup, alt yapı olarak güçlü olması hedeflenmiştir. Kullanıcı herhangi bir kelime ya da bir cümle arattığında, buradaki kelimelerin işlenerek, sistemde bulunan sonuçlardan en uygun olanını son kullanıcıya listelemek amaçlanmıştır. Bu sistem uygulanırken, veritabanındaki veriler için bir puanlama sistemi oluşturulmuştur. Ayrıca doküman bazlı bir NoSQL yapısı kullanılacağı için, verilerin genişlemesi ve daraltılması da kolay bir biçimde mümkün olup, hız konusunda ilişkisel veritabanları gibi bir ağırlık yaşanmayacaktır.

2. BENZER ÇALIŞMALAR

Literatürde arama motorları ile ilgili yapılmış birçok araştırma, çalışma, uygulama geliştirme örnekleri bulunmaktadır. Bu çalışmalardan ve uygulama geliştirme örneklerinden bazıları bu bölümde özetlenmiş olup, yapılacak olarak uygulamanın daha iyi bir ürün olarak ortaya çıkması açısından bu literatür taraması bu çalışmaya direkt olarak etki etmiştir.

İrem Soydal Web Arama Motorları'nda Performans Değerlendirmesi Yüksek Lisans Tezi'nde yaptığı çalışmada; arama motorlarının algoritma performanslarını değerlendirmiş ve baz aldığı arama motorlarının karşılaştırmalarını yapmıştır [3]. Bu çalışmada arama motorlarının başarılılık oranı hesaba katılmamıştır. Seçilen arama motorları Alta Vista, Excite, Hotbot, Infoseek ve Northern Light olup, bu arama motorlarına sorular sorulmuştur. Soruların formülasyonu, sonuç ilgililiği gibi konularda incelemeler yapılmış ve performans değerlendirmesi yapılmıştır. Bu değerlendirmede, kesin isabet ve erişim isabeti gibi iki hesaplama yapılmış olup, bu sonuçlar değerlendirilmiştir. Buna göre grafiksel ve istatistiksel sonuçlar verilmiş olup, nitel ve nicel yorumlar yapılmıştır.

Can Odabaşıoğlu İnternet Arama Motorları Analizi Yüksek Lisans Tezi'nde yaptığı çalışmada; yaygın olmayan kelimelerin sırasıyla Google, MSN Ve Yahoo arama motorlarında sorgulamasını yapmıştır ve bunlarla ilgili analiz çalışması sunmuştur [4]. Bu doğrultuda 3 arama motorunda da aratılan sözcükler için çıkan ilk 10 sonuçlar incelenmiş ve doğru, hatalı, ilişkili gibi analizlerle sonuç puanlandırılmaları yapılarak, karşılaştırma yapılması sağlanmıştır. Bu çalışmada, arama motorlarının ne kadar gelişmiş olsa da, içeriklerinin ne kadar fazla olduğu bilinse de, yol alınması gereken birçok nokta olduğuna dikkat çekmek istenmiştir.

Buket Büyük İnternet Arama Motoru Kullanıcı Verilerinin Analizinde Simülasyon ve Olasılıksal Yöntemlerin Uygulanması Yüksek Lisans Tezi'nde yaptığı çalışmada; kullanıcıların oturum bazlı olasılıklarını hesaplamaktadır [5]. Yani bir kullanıcı, internet üzerinde arama motoruna girdiğinde, diğer kullanıcıların aramalarından bağımsız olarak çalışan; aratılan konuları tespit ederek, sonraki aramalarda bu konulara bağımlı olarak sonuç getiren bir yapı anlatılmıştır. Bu yapılırken, Şartlı Olasılık Yaklaşımı ve Monte Carlo Simülasyonu kullanılmıştır. Bu yaklaşımlar, Excite ve FAST arama motorlarından alınan veriler ile birlikte kullanılmıştır. Bu yaklaşımlarda, verilere yapılan her sorgulama için konu değişikliği var ve konu değişikliği yok şeklinde atamalar yapılmaktadır. En sonunda performans ölçümleri yapılmış ve diğer uygulamalar ile olan tutarlılıklar karşılaştırılmış; diğer uygulamalara göre daha iyi tutarlı sonuçlar alınmıştır.

Görkem Sezgin Arama Motorlarının Davranışlarının Çözümlenmesi ve Web Sayfalarına Tasarım Aşamasında Yansıtılması Yüksek Lisans Tezi'nde yaptığı çalışmada; internet ortamında arama motorlarının gelişiminden, arama motoru ile ilgili kavramlardan ve web sitelerinin bulunabilirliğini artırmak için neler yapılabileceği ile ilgili konulardan

bahsedilmiştir [6]. Buna istinaden Google'ın siteleri nasıl dizinlediği ve arama işlemlerinin nasıl gerçekleştiğine değinilmiştir. Yapılan bu araştırmada, Google'ın kurallarına göre aramalarda daha iyi sonuç alınacağına ilişkin bir arama motoru optimizasyonu çalışması yapılmıştır. Burada referans bağlantılar, meta etiketleri, sayfa rütbeleri, dizinleme gibi konular ele alınmıştır.

Can Razbonyalı Dikey Arama Motorlarının İncelenmesi ve Bir Dikey Arama Motoru Uygulaması Yüksek Lisans Tezi'nde yaptığı çalışmada; internetteki yatay arama motorlarına karşın, sadece arama motoru kapsamına giren konulara özel verileri dizinleyerek geliştirilen dikey arama motorlarının araştırılması ve bir uygulamanın geliştirilmesi üzerine yapılmıştır [7]. Geliştirilen bu dikey arama uygulamasında makine öğrenmesinden faydalanılmıştır. Arama motorları, dizin tabanlı, hibrit arama motoru, örümcek arama motoru gibi bölümlerde incelenmiş, çalışma yapıları ve mantıkları açıklanmıştır. Uygulama sadece web uygulamasına yöneliktir. Geliştirilen uygulamada WEKA'dan, yani hazır bir makine öğrenme yazılım paketinden yararlanılmıştır.

Furkan Gözükara Fiyat Karşılaştırmalı Ürün Arama Motoru Geliştirme Yüksek Lisans Tezi'nde yaptığı çalışmada; e-ticaret sektörünün son yıllarda gelişmesi ile birlikte kullanıcıların ihtiyaçlarını gidermesi için web üzerinde e-ticaret sitelerinde ürün aramaları durumu ortaya çıkmıştır [8]. Farklı web sitelerinden aynı ürün gruplanması işlemi çok problemli bir işlemdir ve bu çalışma bu problemin çözümünü sunmak amacıyla hazırlanmıştır. Arama yapılırken kullanıcılar ürün tanımlamasının dışında aramalar yapabilmektedirler ve bu çalışmanın tanımlarına göre de bu bir gürültü oluşturmaktadır. Bu gürültüler, kümelenmede sorunlar çıkarmaktadır. Genelde daha geniş ölçekli çözümler sunulmasına karşın bu çalışma daha özelleştirilmiş bir uygulama ve çözüm sunmaktadır. Ayrıca bu çalışmada gruplamalardaki hata oranlarının hesaplamaları ve performans ölçümleri de sunulmuştur. Yeni bir kümeleme algoritması geliştirilmiş olunup, %90'ın üzerinde bir başarı sağlanmıştır.

Google; Google Translate uygulamasıyla birlikte kullanıcılara sunduğu Google Translate Makine Öğrenimi yeniliğini devreye alındığını duyurdu. Google, çevirilerde kullanıcıların girdiği kelimeleri sözcük sözcük çeviriyor ya da öneriler sunarak, cümleleri daha doğru hale getiriyordu. Sinirsel Makine Çevirisi özelliği ile birçok dili makine öğrenmesi aracılığıyla çevirebilmesi için kullanıcılarına sundular. Bunu yapmalarının sebebi, 100'den fazla dil desteği veriliyorken, Türkçe gibi bir çok dilde doğru sonuçlar alınamıyordu. Bu yüzden sinirsel makine çevirisi konusunda bir çalışma yapılmıştır. Bununla birlikte performanslarının %30-35 daha fazla artacağını belirten Google, bunu bir API olarak kullanıcıların hizmetine de sunmuştur. Şimdilik test amacıyla belirli dillerde bu geliştirmeyi kullanan Google, ilerleyen günlerde başarılı olmaları taktirde tüm dillerde uygulamak istediklerini açıklamışlardır.

Fethi Burak Sazoğlu Efficient Result Caching Mechanisms In Search Engines Yüksek Lisans Tezi'nde yaptığı çalışmada; arama motorlarının verimini artırmak için önbellek mekanizmasının kullanılmasına vurgu yapmış ve yapılması gerekenleri açıklamıştır [9].

Yapılan sorguların gecikme süreleri, sorguların doğruluğu, güncelliği gibi parametreleri baz alarak, arama motorunun performansı üzerinde durulmuştur. Performans artırma çalışmalarında yakın zamanlı sorguların önbellekte tutularak, aynı kullanıcıya ya da farklı kullanıcılara, aynı sorgunun yapılması halinde tekrar hesaplama yapılmadan yanıt dönülmesi esas alınmıştır. Bu da eldeki kaynaklardaki yükü azaltmaktadır. Frekans, sorgu zamanı, maliyeti gibi istatistiksel bilgiler göz önüne alınmıştır.

İbrahim Bahattin Vidinli Eğitim Arama Motorunda Sorgu Önerme Doktora Tezi'nde yaptığı çalışmada; internet ortamındaki devasa veriye karşın arama motorlarının sorgu önerme işlevlerinin daha fazla nasıl geliştireceğine yönelik öneriler sunmuş, bazı örneklemeler yapmış, test ve sonuçlarıyla birlikte istatistiksel değerlendirmeler yapmıştır [10]. Bu çalışmada öncelikle önerme probleminin indirgenmesi yapılmış ve sonrasında sorgu önermesine yönelik bir mimari geliştirilmiştir. Bu mimari yapılırken bir ağaç yapısı kullanılmıştır ve skorlama yöntemi ile arama sonuçları sunulmuştur. Sorgu önerme algoritmalarının birleştirilmesi ve geliştirilmesi üzerine kurulmuş bir tezdir ve geliştirilme ile literatürde ilerleme kaydettiği görülmüştür.

İncelenen bu çalışmalar sonucunda, geliştirilecek olan yeni arama algoritmasının, geçmişte geliştirilmiş algoritmalardan daha akıllı olacaktır. Araştırmalarda genellikle performansa doğruluğa önem veriliyorken, performans konusunda zayıf kalındığı saptanmıştır. Bu doğrultuda, geliştirilecek olan algoritma performansta da çok daha iyi olacaktır. Yapılan çalışmalar genelde web üzerinde ve bilinen arama motorları ile yapılıyorken, yapılacak olan çalışmanın masaüstü uygulamalar, mobil uygulamalar ve web uygulamaları için birer servis hizmeti veriyor olması hedeflenmektedir. Ayrıca yapılan araştırmalarda genellikle veritabanları testler için kullanılan tek tip veritabanlarıyken, yapılacak olan uygulamada bilinen en önemli veritabanları için destek verilmeye çalışılacaktır. Genelde geliştirilen yazılımlarda arama motorlarına uyarlanma gözlemlenmiştir. Yapılacak ve geliştirilecek olan uygulama ise birçok veritabanına uyumlu olurken, uygulama bazında da servis hizmeti vereceği için, uyarlanan değil, tak-çalıştır modeline sahip olacaktır.

Yapılan çalışmalarında birçoğunda gözlemlenen durum ise, son senelerde kullanıcılara sunulan teknolojilerden faydalanılmaması olmuştur. Yapılacak olan uygulamada ise özellikle son versiyon teknolojiler kullanılmaya çalışılacaktır.

3. KULLANILAN YÖNTEM VE TEKNOLOJİLER

Yapılacak araştırma, piyasada bilinen 300'ün üzerinde bulunan veritabanı düşünülerek yapılmıştır. Veritabanları çeşit olarak ilişkisel veritabanları, key-value depolama veritabanları, arama motoru veritabanları, doküman saklama veritabanları, graph veritabanları gibi birçok veritabanı modelleme bölümüne ayrılmaktadır.

Örneklemek gerekirse, ilişkisel veritabanlarında Microsoft SQL Server, doküman saklama veritabanı kısmında MongoDB, key-value modelinde Redis, arama motoru modelinde Elasticsearch, graph veritabanı modelinde Neo4J gibi örnekler bulunmaktadır. Farklı tiplerde veritabanları için bir çok örnek internetteki kaynaklarda yer almaktadır [11].

Veritabanı modellerinin yanı sıra, algoritmanın doğru kurulması ve veritabanı modellemelerine uyarlanması başarılı olduğu taktirde, makine öğrenmesi metodolojiler de çalışma içerisinde kullanılacaktır. Makine öğrenmesinin bu çalışmadaki en büyük faydası sonuçların kullanıcılara doğru bir biçimde sunulması anlamında olacaktır. Kullanıcıların arama alışkanlıklarının getirdiği verileri bir yerde saklayarak, arama algoritmasını her seferinde çalıştırmadan kullanıcıya en doğru sonucu verecek bir yapı geliştirilmesi mümkündür. Bu çalışmadaki en son hedef makine öğrenmesinin uyarlanması olacaktır. Bu kısımda hızdan çok, doğru sonuç alınması hedeflenmektedir.

Elasticsearch çalışması, MVC projesi ile arama motoruna entegre edilecek olup, makine öğrenmesi için özelleşmiş bir model kullanılacaktır. Matlab ile yapılması düşünülen entegre sistem, performans olarak verimli çalışmayacağı için, modellemenin çalışan sistem üzerinde oluşturulması ve stabilizasyonun sağlanması ihtiyacını doğurmuştur. Modelleme her ne kadar aynı sistem üzerinde olursa olsun, yapılacak işlemlerden ötürü yaşanacak süre kaybını, daha hızlı ve daha stabil sistemlere uyarlamak mümkündür.

Bir proje yapılırken tasarım deseninin önemi de çok büyüktür. .NET için kullanılan Repository Pattern, Dapper, ADO.NET gibi tasarım desenlerinden Repository Pattern seçilmiştir. Bundaki amaç, yazılan kodların daha okunabilir, nesneye yönelik programlamanın daha doğru uyarlandığı ve kullanıcının sisteme bağlantı yönetiminin daha kolay, daha doğru bir biçimde yapılmasıdır.

3.1) Naive Bayes

Naive Bayes yöntemi bir çok özelliğin bulunduğu durumlarda, hedef sonucu bulmamıza yardımcı olan bir metoddur. Basit bir formülü olmasına karşın, başarılı sonuçlar vermekte olup, bu yöntemle tüm ihtimallerin olasıkları bulunabilmektedir. Denklem 1'de Naive Bayes formülü verilmiştir.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)} \tag{1}$$

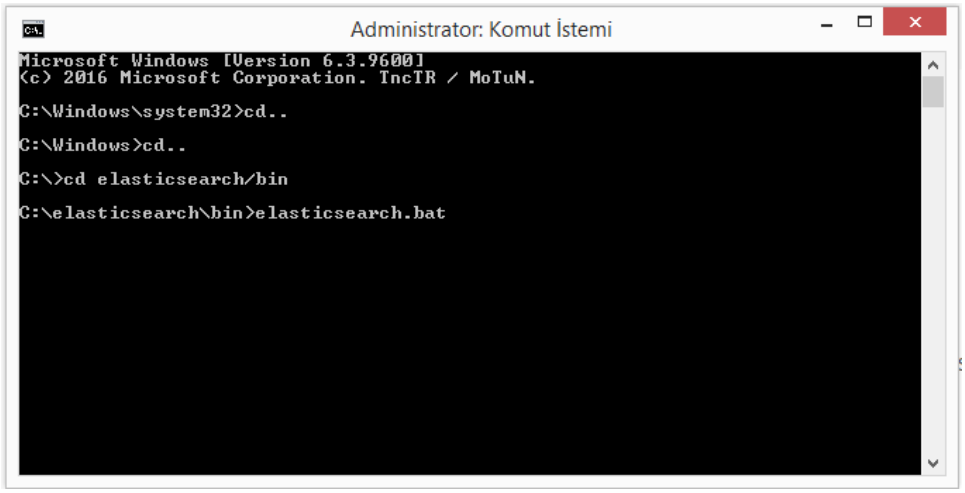
Yukarıdaki formüle göre P(A|B), B'nin gerçekleştiği durumlarda, A olayının olma olasılığıdır. P(B|A), A'nın gerçekleştiği durumlarda, B olayının olma olasılığıdır. P(A), A olayının olasılığı, P(B) ise B olayının olasılığıdır.

Bu tezde A ve B olayları için, sistemdeki veriler ve kullanıcıların girdiği veriler modellemeye çalışılmıştır. Buna karşın sistemin yavaşlaması, yapı daha basit olduğu için sınıflandırılmanın doğru bir biçimde yapılamaması, modelin başarısız olmasına sebep olmuştur. Ayrıca Karar Ağaçları ile ilgili de bir model oluşturulup oturtulmaya çalışılsa da, yapının içerisindeki veri, bu modellemeler uygun olmadığı için başarı sağlanamamıştır. Bundan ötürü, sisteme öğretilen verileri kullanıcıya doğru bir biçimde sunmak için, farklı bir modellemeye ihtiyaç duyulmuştur.

3.2) Elasticsearch

Elasticsearch, büyük verilerle çalışan uygulamalar için veri arama ya da sorgulama gibi işlemlerde kullanılan bir arama veritabanıdır. Dünya üzerinde bir çok firma, Elasticsearch'ü farklı alanlarda kullanmaktadır [12]. Elasticsearch, Lucene ile kurgulanmış bir arama motorudur. Java altyapısını kullandığı için, kurulumu esnasında Java'nın kurulumuna ihtiyaç duymaktadır.

Elasticsearch'ün kurulumu için Elasticsearch'ün websitesindeki [13] download kısmı kullanılarak setup dosyaları indirilebilir. İndirilen dosyalara, Şekil 2'deki gibi komut istemi ekranı yönetici olarak çalıştırılarak erişilir ve Elasticsearch aktifleştirilir.

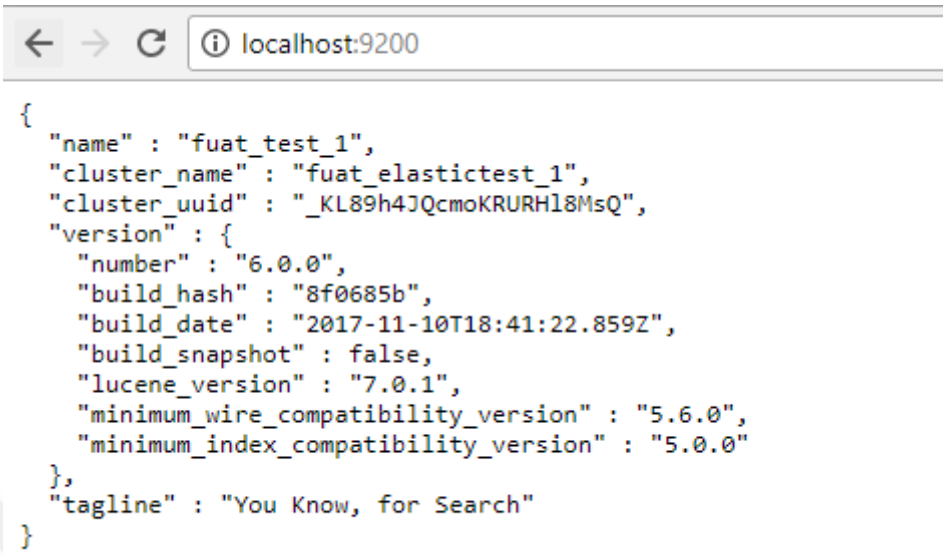


Şekil 2 – Elasticsearch'ü aktifleştirme

Komut istemci ekranını kapatmamamız gerekmektedir. Bu ekran, local portlardan birinde Elasticsearch veritabanının çalışmasını sağlar. Elasticsearch aktifleştirildikten sonra Şekil 3'teki gibi bir ekran karşımıza çıkar.

Şekil 3 – Elasticsearch'ün başlatılması

Elasticsearch veritabanına default olarak http://localhost:9200/ ile erişebiliriz. Başlangıçta karşımıza Şekil 4'teki gibi bir ekran çıkacaktır.



Şekil 4 – Elasticsearch veritabanı anasayfası

Bu veritabanında veriler JSON olarak tutulduğu için, yüklediğimiz veriler anlamlı olarak gözükmeyebilir. Verileri tutabileceğimiz veritabanı Elasticsearch olsa da, arayüz olarak Kibana kullanılmıştır. Kibana, verileri kolayca düzenleyebileceğimiz, görselleştireceğimiz ve bir çok işlem yapabildiğimiz, Java tabanlı, Elasticsearch'ün bir uygulamasıdır. Elasticsearch'ü başlattığımız gibi bu uygulamayı da aynı şekilde (ayrı bir komut istemcisinde) başlatacağız ve karşımıza Şekil 4'te görülebilecek ekran görüntüsü gelecektir.

Kibana'nın ekranlarında da göreceğimiz üzere, Microsoft SQL yazım biçimine göre Elasticsearch'te bazı farklı terimler bulunmaktadır. Bunların bazılarının açıklamaları aşağıdadır.

Index: Elasticsearch, her kayıt JSON dökümanları ile saklanmaktadır. Index'ler, Elasticsearch içerisinde bulunan ayrı veritabanları gibidir. JSON dökümanları, Index'ler içerisinde bulunan Type'larda saklanan verilerdir.

JSON: Bir veri tipidir. XML'den daha az yer kaplar. Bu yüzden yeni nesil yazılımlarda çokça tercih edilir.

Type: Veritabanlarındaki tablolar gibi düşünülebilir. Index'lerin içerisinde bulunurlar ve JSON dökümanları, bu Type'lar içerisine yazılmaktadır.

Mapping: İlişkisel veritabanlarında bulunan schema'nın karşılığıdır. Data aktarılırken tipler Mapping sayesinde otomatik olarak oluşturulur. Mappingi kendimiz tamamlamamız halinde, varsayılan Mappingi devre dışı bırakabiliriz.

Document: İlişkisel veritabanlarındaki her bir satıra Elasticsearch içerisinde Document denmektedir.

Field: İlişkisel veritabanındaki sütunlar, Elasticsearch'teki Fieldlar olarak karşımıza çıkar.

Elasticsearch'ün aramayı hızlı yapmasının sebebi, Apache Lucene altyapısını kullanmasıdır. Bu altyapıya göre, her bir kelime, her satırda (Document) indekslenir. Daha sonra herhangi bir kelime aratıldığında, arama yapılan yer tüm veritabanı yerine bu indeksler olmaktadır. Bu yüzden hız çok yüksektir. Bu durum Şekil 5'te görülmektedir.

1	The old night keeper keeps the keep in the town
2	In the big old house in the big old gown.
3	The house in the town had the big old keep
4	Where the old night keeper never did sleep.
5	The night keeper keeps the keep in the night
6	And keeps in the dark and sleeps in the light.

Term	Documents
and	<6>
big	<2> <3>
dark	<6>
did	<4>
gown	<2>
had	<3>
house	<2> <3>
in	<1> <2> <3> <5> <6>
keep	<1> <3> <5>
keeper	<1> <4> <5>
keeps	<1> <5> <6>
light	<6>
never	<4>
night	<1> <4> <5>
old	<1> <2> <3> <4>
sleep	<4>
sleeps	<6>
the	<1> <2> <3> <4> <5> <6>
town	<1> <3>
where	<4>

Şekil 5 – Elasticsearch'te indeksleme [14]

CRUD (Oluşturma, güncelleme, silme) işlemlerini Kibana üzerinden yapacağımız gibi, HttpGet ile üçüncü parti yazılımlar ile de yapabiliriz. Bunlardan bazıları Postman, Soap UI, Telerik Fiddler gibi uygulamalardır.

Visual Studio tarafında kullanılabilmesi için NEST kütüphanesine ihtiyaç bulunmaktadır. Nuget Package Manager Console aracılığı ile bu kütüphaneyi indirebilir ve Elasticsearch için kullanacağımız tüm sınıflara, indirilen DLL'ler aracılığı ile erişebiliriz.

Term: Filtreleme yapmak istediğimizde .Net üzerinde kullanılan metoddur.

Filter: Sorguları daha fazla daraltmak için kullanılan metoddur.

Hits: Elasticsearch'ten alınan cevabın içerisinde bulunur. Hits içerisinde sorgumuz sonucunda dönen veriler bulunur.

Bu projede, örnek veri olarak Tekzen A.Ş. firmasının verilerinden bir kısmı canlı ortamdan alınarak Elasticsearch veritabanına aktarılmıştır ve kullanılacaktır. Veriler aktarılırken, .Net üzerinde bir konsol uygulaması yazılmış olup, veriler Entity Framework aracılığıyla Microsoft SQL'den çekilmiş ve Elasticsearch'e JSON olarak aktarılmıştır.

3.2.1) NoSQL

NoSQL 2000'li yıllarda, ilişkisel veritabanlarına alternatif olarak ortaya çıkarılan bir kavramdır. NoSQL genellikle "SQL değil" olarak anılsa da, asıl anlamı "Not Only SQL" yani "Sadece SQL değil" şeklindedir.

NoSQL mantığı, ilişkisel olmayan bir biçimde yatay olarak genişlemeye yarar. İlişkisel veritabanlarında yaşanan maliyet sorunlarından ötürü (hız, gereksiz data trafiği vs.) bu ihtiyaç ortaya çıkmış olup, JSON ve XML formatları gibi farklı veri tipleriyle saklama işlemi yapılmaktadır. NoSQL mantığında tablo ve sütun olmaması, maliyeti minimuma indirmekte ve bu yüzden hız konusunda büyük avantajlar sağlanmaktadır.

Yatay büyüme, yeni bir sunucu alınarak, işlem yükünü bölme ile gerçekleştirilmektedir. Dikey büyüme ise, kaynakların yetersiz kalması durumunda sisteme ek donanım alma veya sistemi yenileme şeklinde olmaktadır.

Şekil 6'da NoSQL ve SQL'in farkları açıklanarak, avantaj ve dezavantajları listelenmiştir.

Kriter	NoSQL	SQL
Tip	Anahtar-değer, döküman veritabanı gibi birden çok tipi vardır.	Tek tiptir.
Örnek	MongoDb, Cassandra, Elasticsearch, Neo4j	MySQL, Postgre, Microsoft SQL Server, Oracle

Veri Saklama Modeli	Veritabanı tipine göre değişir. Mesela anahtar-değer tipi veritabanları SQL'e çok benzemesine karşın, sadece iki sütuna sahiptirler. Kimi zaman karmaşık sorgular gerektiğinde, value sütununda BLOB'lar şeklinde bu değerler çekilebilir. Bu tip veritabanlarında her bir kayda "document" adı verilir ve bunlar JSON ya da XML olarak saklanır.	Her bir kayıt, satırlar halinde tablolara yazılmaktadır ve her sütunla ilgili bir yer bu satırlara ayrılmıştır. İlişkili kayıtlar, farklı tablolarda saklanabilir ve bunlar bir araya geldiğinde karmaşık sorgular ortaya çıkar. Bu karmaşık sorgular sonucu, gereksiz bir çok veri bir araya getirilir ve sonra aralarından işimize yarayanlar çıkarılır.
Şema	Veri doğrulama kurallarıyla birlikte dinamik tiplere sahiptirler. SQL'den farklı olarak canlı ortamda yeni alanlar eklenebilir.	Yapı ve veri tipleri önceden belirlenmiştir. Yeni bir tip eklenebilir ama veritabanı çevrimdışı olmalıdır.
Ölçeklendirme	Yatay olarak genişler. Yer artırmak istendiğinde, veriler yeni kaynaklara otomatik olarak dağılır.	Dikey olarak genişler. Veri arttıkça sunucu daha güçlü bir hale gelmelidir. Ayrı SQL veritabanları kurulabilir ama JOIN'ler vs. genellikle gereklidir.
Geliştirme Modeli	Açık kaynak kodludur.	Postgre, MySql gibi veritabanları açık kaynak kodludur. Oracle veritabanı gibi veritabanları kapalı kaynak kodludur.
Veri Dili	API'lerin iletişimi aracılığıyla yapılır.	Ekleme, silme, güncelleme işlemleri için kullanılan özelleşmiş dilleri vardır.

Şekil 6 – NoSQL ve SQL Karşılaştırması [15]

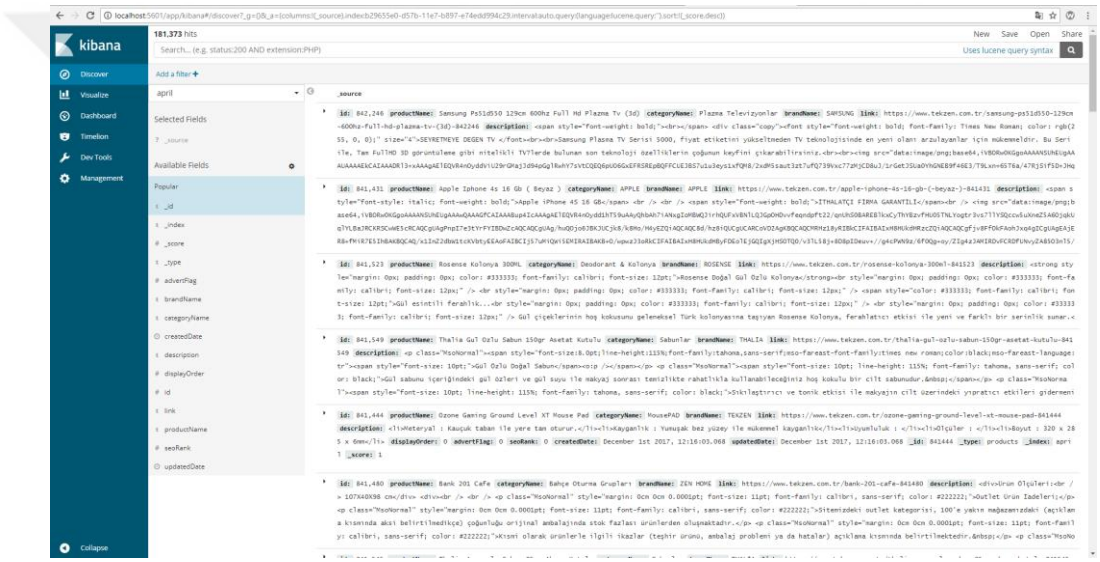
NoSQL veritabanı olarak birden fazla çeşit vardır. Bunlardan bazıları “Document Store”, “Key-Value Store”, “Search Engine”, “Graph DBMS” şeklindedir. Bu veritabanları, kullanım amaçlarına göre farklılık göstermekte ve farklı performanslar vermektedir. Şu anda ilişkisel olmayan veritabanlarından bilinen ve en çok kullanılanı MongoDB’dir.

NoSQL’in bir çok avantajı olmasına karşın, yine de tam anlamıyla tüm sistemin üzerine kurulacağı yapılar değillerdir. İlişkisel veritabanlarında olduğu gibi transaction kavramı olmadığı için, veri kaybı olabilmektedir. Bu yüzden önemli veriler SQL üzerinde yapılandırılıyorken, daha az öneme sahip veriler NoSQL veritabanları üzerinde işlenmektedir. Bu dezavantaj, NoSQL’in “Sql Değil” ismi yerine “Sadece SQL Değil” ismiyle anılmasını daha mantıklı kılmaktadır.

3.3) Kibana

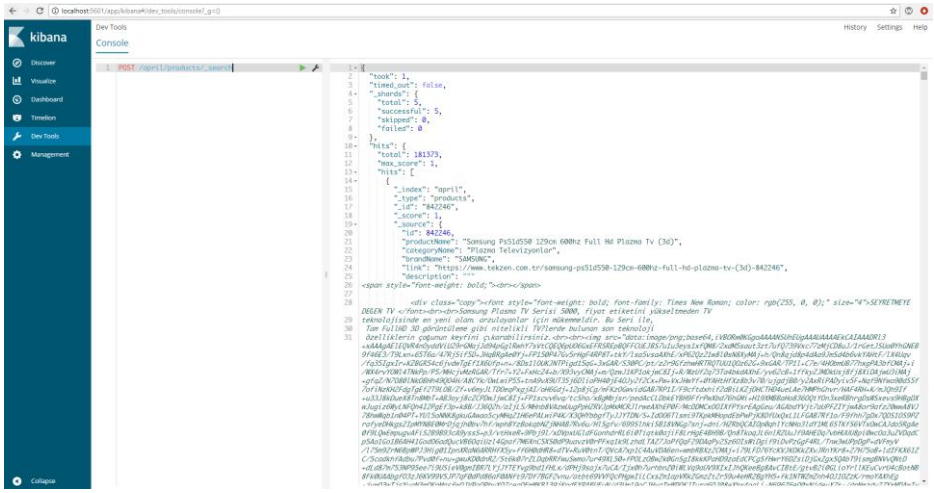
Kibana, veri gösterimini sağlayan bir uygulamadır. Web tabanlı olarak çalışmaktadır. Kibana'nın sistemde çalışabilmesi için sistemde Java yüklü olmalıdır. Elasticsearch gibi, Kibana'nın dosyalarını da sitelerinden indirdikten sonra komut satırından .bat uzantılı dosyalarını çalıştırabilir ve varsayılan portları ile bu iki sisteme erişim sağlayabiliriz.

Elasticsearch için varsayılan port 9200 olup, http://localhost:9200/ adresi ile Elasticsearch veritabanındaki verilere erişim sağlayabiliriz. Kibana için varsayılan port 5601'dir ve http://localhost:5601/app/kibana linki üzerinden sisteme erişilebilmektedir. Şekil 7'de Kibana'nın başlangıç ekranı görülmektedir.



Şekil 7 – Kibana başlangıç ekranı

Kibana ile klasik bir Microsoft SQL Server veritabanı sorgusu yapamayız. Bu uygulama web tabanlı olduğu için, verileri alırken, işlerken ya da yeni veri kaydetmeye çalışırken, http işlemleri ile bunları gerçekleştiririz. Bu http işlemlerinden bazıları POST; GET; PUT gibi işlemlerdir. Şekil 8'de Kibana üzerinde yapılmış bir Elasticsearch veritabanı sorgusunu bulabilirsiniz.



Şekil 8 – Kibana üzerinde yapılmış get sorgusu

Yukarıda yapılan sorgu, bir POST sorgusudur. Bu sorgu ile, veritabanındaki verilerin tümü çekilmektedir. Sorgu “POST /april/products/_search” şeklindedir.

Yaptığımız bu sorguları 9200 portu üzerinde de yaparak, veritabanındaki verilere erişebiliriz. Bununla ilgili aşağıda bulunan Şekil 9’da örnekleri görülmektedir.

Şekil 9 – Elasticsearch sorgu örneği

Yukarıdaki sorgu link alanına yazılmış olup, “http://localhost:9200/april/products/_search” şeklinde bir sorgu yazılmıştır. Bu gibi işlemler, http işlemlerini yapan Postman, SOAP gibi programlar ile de kolayca yapılabilmektedir.

3.4) MVC

MVC’nin açılımı Model-View-Controller’dır. Çalışma mantığı ise; Controller ile işlemleri yaparak, Model katmanı ile veriyi taşımak ve View katmanında ise bu verileri kullanıcıya sunmak ya da View katmanında yapılan işlemler sonucu, Model ile verileri Controller’a ileterek işlem yapmaktır.

MVC, daha güvenilir, daha basit, daha hızlı bir yazılım mimarisi olarak; WebForms'un eksikleri de düşünülerek dizayn edilmiştir. MVC incelendiğinde, Model ve Controller birer class olmasına karşın, görevleri farklıdır ve bir tasarım deseninin ayrı elemanları olarak düşünülmelidir. WebForms incelendiğinde, Views tarafında aspx yerine cshtml yapısına geçilerek, Razor'un kullanılması sağlanmıştır. Yani uygulamanın HTML tarafında C# kodu yazılmasına olanak sağlanmıştır. Bu sayede ön yüzde bağımsız bir biçimde çok fazla işlem yapılmasına olanak sağlanmıştır.

Visual Studio üzerinde bir MVC projesi açıldığında bir proje kütüphanesi içerisinde Model-View ve Controller klasörleri ile bize gelmektedir. Projenin amacına yönelik olarak yeni proje kütüphaneleri açılarak düzen sağlanmaya çalışılır. Bu projede toplam 5 katman bulunmaktadır. Core kısmı, Microsoft SQL Server'da bulunan veritabanımızdaki tablolar ve içerisinde bulunan alanları içerir. Data kısmı, bu veriler arasındaki ilişkileri kurar ve Repository Pattern sınıflarını içerir. Service kısmında ise, verinin veritabanından alınarak işlenmesi ve sistemin içerisinde aktarılması hedeflenir. Web kısmında, kullanıcının arayüzde yapacağı tüm işlemler yapılır ve diğer katmanlarda yapılacak işler yönetilir. Framework katmanında ise, Web katmanındaki sınıflarda sürekli olarak kullanılan ya da uygulamanın temelini oluşturacak sınıflar bulunmaktadır.

3.5) Entity Framework

Entity Framework, en yaygın olarak kullanılan ORM araçlarından birisidir. ORM'in açılımı Object Related Mapping, yani Nesneleri İlişkisel Haritalama anlamına gelir. Veritabanı ve kodumuz arasındaki bağlantı Entity Framework ile yapılır. Bu sayede veritabanında kayıt güncelleme, kayıt alma, kayıt ekleme gibi işlemlerimizi de gerçekleştirebiliriz.

ORM araçları arasında ADO.NET, Nhibernate, Dapper gibi yapılar vardır. Bu projede kullanılan yapı Entity Framework olduğu için Entity Framework açıklaması ile ilerlenecektir.

Entity Framework sayesinde Object Oriented Programming düzenine, yani Nesne Yönelimli Programlama'yı da doğru bir biçimde yapabiliriz.

Entity Framework'ün Model First, Database First ve Code First şeklinde 3 farklı uygulanma yöntemi vardır. Model First yönteminde, projenin Model dosyasına .edmx dosyası eklenir ve veritabanı daha önce bir script ile oluşturulmuş olmak zorundadır. Database First yönteminde ise veritabanı model dosyası ile projeye eklenir ve entity sınıfları Entity Framework tarafından otomatik olarak oluşturulur. Code First mantığında ise tüm sınıflar yazılımcılar tarafından oluşturulur. Programlanma tipine göre, eğer sistemde veritabanı yoksa, Code First yapısı ile veritabanı da otomatik olarak proje başlayınca oluşturulacaktır.

3.6) Repository Pattern

MVC bir yazılım mimarisi olmasına karşın, MVC içerisinde daha düzenli ve az kod yazabilmek, Nesneye Yönelik Programlama mantığına uymak için bazı tasarım desenleri kullanılır. Bunlardan birisi de Repository Pattern'dir.

Örnek olarak, yaptığımız projenin büyük bir veritabanı mimarisine sahip olduğunu düşünülürse, her tablo için kayıt ekleme, kayıt silme ve kayıt güncelleme işlemlerinin yapılacağı birer sınıfın yazılması gereklidir. Burada tablo isimleri haricinde sürekli olarak birbirini takip eden kodlar ortaya çıkacaktır ve bu Nesneye Yönelik Programlama prensibine uygun değildir. Her sınıf için kayıt işlemlerini yapacağımız sınıflar yazmak yerine, bunları Repository Pattern'in sağladığı bir interface (arayüz) aracılığı ile tekilleştirilmesi ile projede kullanılacaktır.

3.7) Singleton

Singleton yapısı, tasarım deseni olarak kabul edilen bir yapıdır. Singleton'ın amacı, çok kullanılan bir sınıfı ya da örneklemesinin sürekli olarak yeniden oluşturulmamasını ve var olan örneklemenin kullanılmasını amaçlar. Bu tanımı örneklemek gerekirse; bir kullanıcı web sitesine giriş yaptığında, sayfaya basılacak veriler için veritabanından kayıtların çekilmesi gerekir. Bunun için bir bağlantı kurulur. Kullanıcı, aynı site içerisinde login olmak isterse, veritabanı ile yeniden bir bağlantı kurma zorunluluğu doğacaktır. Her seferinde yeni bir bağlantı kurmak yerine, Singleton sınıfı ile var olan bağlantıyı kullanabiliriz.

Bu sayede sunucu kaynaklarını daha az kullanacağımız gibi, uygulamalarımızı daha performanslı ve hatasız hale getirmiş olmaktayız. April Search projesinde de Singleton sınıfı kullanılmaktadır.

3.8) Autofac

Autofac'in kullanımından önce Dependency Injection ve IOC kavramlarının bilinmesi gerekmektedir. Dependency Injection, bağımlı sınıfların dışarıdan eklenmesi anlamına gelir. IOC Container ise sınıfların yönetilmesi işlemidir.

Autofac bir IOC Container çeşididir. Projenin, Service katmanında yapılan işlemleri Controller'da kullanabilmek için, Service katmanında yazılmış sınıfların birer interface (arayüz) aracılığıyla Controller'a eklenmesi işlemidir. Bu işlem bir DLL aracılığıyla yapılmaktadır.

Bu projede, Controller sınıfımızda yaptığımız alınacak ya da verilecek verilerle ilgili işlemlerle alakalı olarak, service katmanında bulunan metodlara erişimi arayüzler üzerinden sağlarız. Direk metodlara erişmek yerine bu arayüzlerin kullanılması, ileride arayüzler arasındaki geçişlerle ilgili bizlere kolaylık sağlayacaktır.

3.9) Microsoft SQL Server

Microsoft SQL Server günümüzdeki ilişkisel veritabanları arasında en çok kullanılan veritabanıdır. Temelde verileri saklamaya yarayan bu uygulama, yazılacak T-SQL yani verileri işlemeye ve sunmaya yönelik sorgularla verileri kullanıcıya anlamlı şekilde sunabilir.

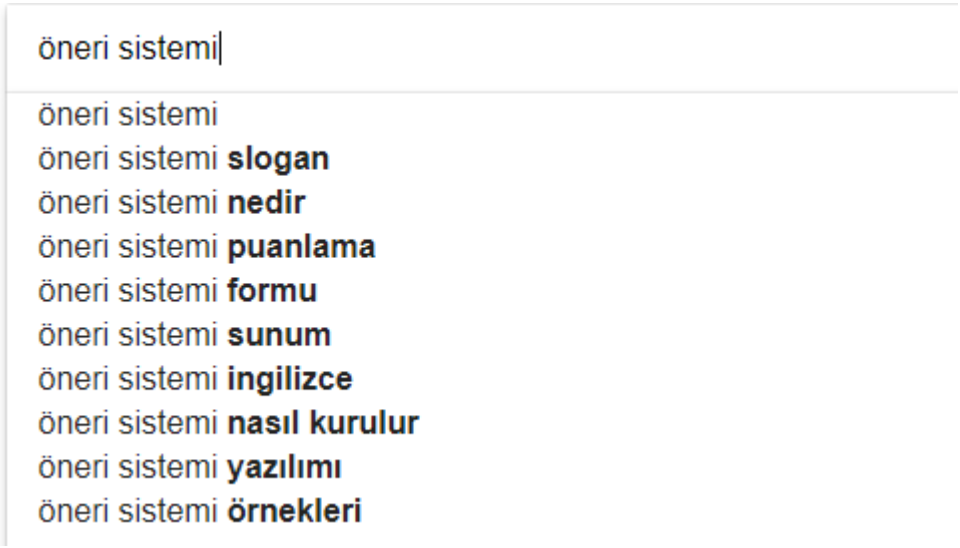
SQL'in açılımı ise Structered Query Language olup, Yapılandırılmış Sorgulama Dili anlamına gelmektedir. Veritabanı içerisinde depolanan verileri yönetmek için kullanılan dildir.

Elasticsearch veritabanı her ne kadar hızlı ve bu proje için etkili bir veritabanı olsa da, kalıcı ve önemli veriler için Microsoft SQL Server kullanmaya devam etmekteyiz. Uygulamanın konfigürasyon ayarları gibi ayarları tutabilmek, verileri yorumlarken, NoSQL veritabanlarında oluşabilecek kayıplardan ötürü yanlış sonuçlar getirmemek adına, eğitim Microsoft SQL Server üzerinde yapılmıştır.

3.10) Fuzzy Search

Fuzzy Search, bulanık arama anlamına gelmektedir. Uygulama içerisinde bulanık arama farklı şekillerde yapılmaktadır. Diğer arama motorları araştırıldığında, arama esnasında kelimelerin bazı kısımları farklı girilmesine karşın, öneri sistemi bir öneri sunsa da, kullanıcıya sonuç dönüldüğü görülmüştür. Bu durum da, öneri sistemi ve arama sisteminin bütünleşik değil, asenkron olarak, ortak noktaları olmasına rağmen farklı yöntemlerle çalıştığını göstermektedir.

Uygulama içerisinde kelime araması yapıldığında, kelimeleri bazı formatlara çevirerek, en yakın kelimelerle kullanıcıya sonuçları dönmektedir. Öneri sistemi içinse, daha önce sistemde bulunan kayıtlar ve aratılan kelimeler yorumlatılarak kullanıcıya sunulmaktadır. Şekil 10'da öneri sistemi örneği bulunmaktadır.



Şekil 10 – Öneri sistemi örneği

Bulanık arama için bir çok açık kaynak kod bulunmaktadır. Bunlardan bazıları Levenshtein Distance, Damerau-Levenshtein Distance, Needleman-Wunsch Algoritması'dır [16]. Bunlar sisteme uyarlanmış olup, performans olarak son kullanıcı deneyimini olumsuz etkilediği ve daha basit bir şekilde modelleme mümkün olduğu için, özelleşmiş bir algoritmaya gidilmiştir.

Elasticsearch'ün bulanık arama ile ilgili eklentileri de bulunmaktadır. Daha basit bir şekilde bulanık aramayı Elasticsearch ile yapmak mümkünken, bunu projeye uyarlamadık. Bunun sebebi, sistem üzerinde bir model kurmak ve bu modele göre sistemi eğitmek, sonrasında ise sistemin ezber yapmasının önüne geçerek, kullanıcıya anlamlı sonuçlar sunmak için Elasticsearch'ün sunduğu özellik kullanılmamıştır. Eğer bu özellik kullanılmış olsa, Elasticsearch üzerinde sakladığımız veriler üzerinden ezberlenmiş sonuçlar dönülmesi muhtemel olacaktı.

3.11) Levenshtein Distance Algoritması

Levenshtein Distance Algoritması, bu projede öneri sunabilmek için kullanılan algoritmadır. Eksik ya da bulunamayan kelimeler için çalışan Levenshtein Distance Algoritması, temel olarak iki kelimenin birbirine benzerliğini ölçümler ve rakamsal olarak bir sonuç verir. Bu karşılaştırmanın yapılabilmesi için, son kullanıcının girdiği kelimeyle karşılaştırılabilecek bir kelime ya da kelime grubu gerekmektedir. Bu işlem ilk etapta, var olan verileri, kelimelerine parçalayarak yapılmıştır. Arama işlemleri yapıldıkça bu kayıtlar tutulmakta olup, bu kayıtlara göre öneri kelimeleri çıkarılmaktadır [17].

4. UYGULAMA

Bu tezin amacı, uygulama bazlı bir arama motoru olduğu için, uygulama öncesinde bazı hazırlıklar yapılmıştır. Veriler, Tekzen E-Ticaret sitesinden, arama motoruna uygun olarak hazırlanarak alınmıştır. Microsoft SQL Server tarafında herhangi bir veri aktarımı yapılmamış olup, küçük ölçekli bir uygulama aracılığıyla Microsoft SQL Server'daki veriler, JSON formatına dönüştürülerek Elasticsearch'e kaydedilmeye uygun bir hale getirilmiştir ve Elasticsearch'e kaydı, bu veri formatlama işleminden sonra yapılmıştır. Bu veri aktarımından sonra uygulamaya başlanmıştır.

Bu uygulamanın arama algoritmasındaki en önemli kısım, aratılacak alanlara göre puanlama sisteminin kolayca yapılabilmesi, NoSQL yapısı ile hızlı çalışabilmesidir.

Veriler Elasticsearch'e aktarıldıktan sonra, öneri sisteminin yapılabilmesi için veriler tekrar Microsoft SQL Server için hazırlanmış olup (Bu kısımda veriler ayrıştırılarak, kullanıcının arama yaptığında ona öneri sunabileceği hale getirilmitşir), Microsoft SQL Server'a kayıt işlemleri gerçekleşmiştir. Şekil 11'de Elasticsearch veritabanına yapılan bağlantının C# üzerinde yazılımı ve Şekil 12'de ise verilerin JSON formatına çevrilerek veritabanı üzerine yazımı gösterilmiştir.

```
class ConnectionToES
{
    1 reference
    public static ElasticClient EsClient()
    {
        ConnectionSettings connectionSettings;
        ElasticClient elasticClient;
        StaticConnectionPool connectionPool;

        //Multiple node for fail over (cluster addresses)
        var nodes = new Uri[]
        {
            new Uri("http://localhost:9200/"),
        };

        connectionPool = new StaticConnectionPool(nodes);
        connectionSettings = new ConnectionSettings(connectionPool);
        elasticClient = new ElasticClient(connectionSettings);

        return elasticClient;
    }
}
```

Şekil 11 – Elasticsearch'e .NET üzerinden yapılan bağlantı

```
public static bool insertDocument(dynamic item, int itemId)
{
    bool status;

    var myJson = new
    {
        id = item.Id,
        productName = item.ProductName,
        categoryName = item.CategoryName,
        brandName = item.BrandName,
        link = item.Link,
        description = item.Description,
        displayOrder = 0,
        advertFlag = 0,
        seoRank = 0,
        createdDate = DateTime.Now,
        updatedDate = DateTime.Now
    };

    var response = ConnectionToES.EsClient().Index(myJson, i => i
        .Index("april")
        .Type("products")
        .Id(itemId)
        .Refresh(Elasticsearch.Net.Refresh.True));

    if (response.IsValid)
    {
        status = true;
    }
    else
    {
        status = false;
    }

    return status;
}
```

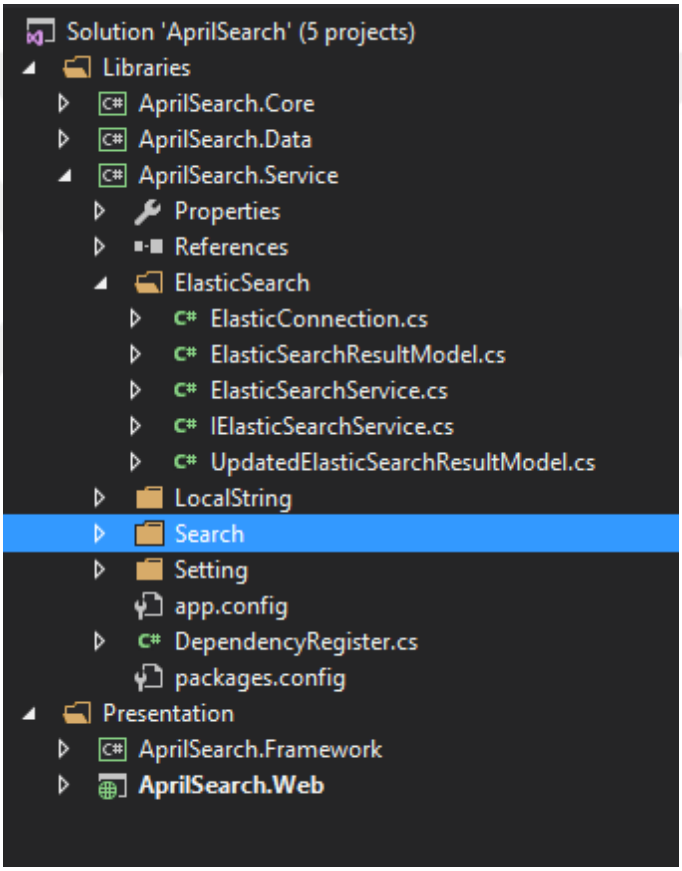
Şekil 12 – JSON çevrimi ve Elasticsearch'e yapılan aktarım

Uygulama, C# dili kullanılarak .NET üzerinde yazılmıştır. Microsoft SQL Server ve Elasticsearch olmak üzere iki tane veritabanı kullanılmıştır. Microsoft SQL veritabanı, uygulamanın ayarlarını, kalıcı ve önemli verileri tutmak için kullanılmaktadır. Elasticsearch ise gelişmiş indeksleme yapısıyla birlikte, arama işlemlerinde hızlı çözümler sunduğu için kullanılmaktadır.

.NET üzerinde yazılan kısım MVC projesi ve Entity Framework kullanılarak geliştirilmiştir. Uygulama yazılırken Repository Pattern kullanılmıştır. Bu pattern içerisinde Elasticsearch, servis katmanında projeye dahil edilmiştir. Kayıtların alınması, veritabanına kaydı, güncellenmesi bu katmanda gerçekleşmektedir. Aşağıda bu katman ile ilgili bir ekran görüntüsü bulunmaktadır.

Servis katmanının amacı, proje içerisinde yapılacak veritabanı ve veri işleme işlemlerini, bu birim üzerinden ilerletmekir. Ayrıca Elasticsearch gibi farklı sistemlerin uygulama içerisinde kullanımı da, bu katmandaki yapıyla daha kolay hale gelmekte olup, entegrasyon ve veri alış verişini kolaylaştırmıştır.

Uygulamanın Core kısmında, Microsoft SQL veritabanında tanımlı entityler ve Singleton gibi sabit sınıflar bulunmaktadır. Data kısmında, veritabanı ile yapılan iletişim yönetilmektedir. Service kısmı ise uygulama içerisinde Core ve Data kısmında bulunan sınıflarla ilgili Presentation'da yapılacak işlemlerin yönetildiği kısımdır. Burada Autofac kullanılmakta olup, bu yöntemle arayüzle ile ilgili işlemler Presentation kısmına aktarılmaktadır. Şekil 13'te proje yapısının genel görünümü verilmiştir.



Şekil 13 – Proje yapısının görünümü

Uygulamanın Presentation kısmında bulunan Framework bölümünde paging, action filters ve helper gibi işlemler yönetilmektedir. Bunların Web bölümünde değil de, ayrı bir bölümde yapılmasının sebebi, projenin geneline etki edecek bir class library sınıfı olarak hazırlanmasıdır. Eğer Presentation kısmına başka bir proje ekleyecek olsak, Framework bölümünde bulunan bu işlemleri, açacağımız yeni projede de kolayca kullanabiliriz. Bu şekilde karmaşık kod yapısının önüne geçerek, sağlıklı bir pattern oluşturulmuştur.

MVC kullanılmasının ve bu tasarım deseninin seçilmesinin sebebi, Elasticsearch dışında yeni bir teknolojinin de bu projeye dahil edilmesinin önünü açarak, projenin gelişimini kolaylaştırmaktır.

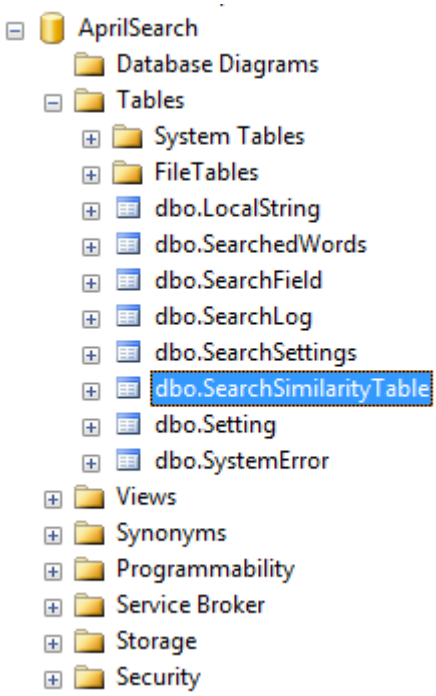
Projenin Controller kısmı incelendiğinde tek bir Controller sınıfı kullanılmıştır. Bunun sebebi, projede sadece arama ekranın görüntülenmesi ve arama sonucunda listeleme işlemi yapılmaktadır. Bunları sağlamak için tek bir Controller sınıfı yeterlidir. Kullanıcı bir kelime arattığında, Controller sınıfına istek geldikten sonra geriye kalan tüm işlemler diğer katman ya da başka sınıflara paylaştırılarak yapılmaktadır. Otomatik tamamlama, puanlama gibi işlemler, Helper sınıfları aracılığıyla yapılmaktadır.

Puanlama işlemi, uygulama içerisinde hangi alanın ne kadar efektif olarak aranması gerektiği ve içerisinde bulundurduğu verinin önemini belirtmektedir. Bu sistem uyarlanırken, Elasticsearch veritabanında bulunan ve verilerin bulunmasını istediğimiz alanları puanlama sistemine dahil etmekteyiz. Uygulama bazlı bir arama motoru yaptığımız için bu alanlar sabit şekilde ayarlanmıştır. Projenin geliştirilmesi durumunda ya da sonraki safhalarında daha dinamik hale getirilecek bir yapı yapılarak, veritabanından yönetim ya da XML dosyalarından yönetim sağlanabilir. Şekil 14'te puanlama sisteminin bir bölümünün örneği bulunmaktadır.

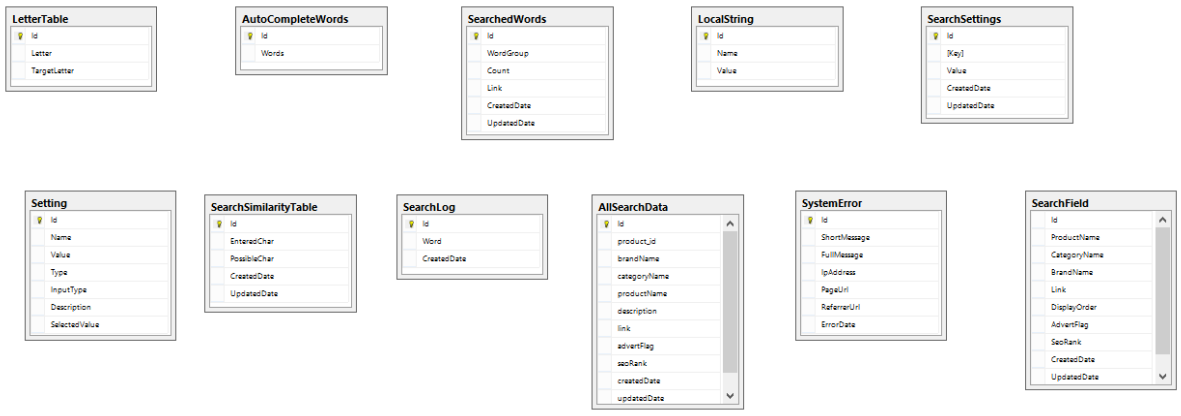
```
case "brandName":
    wordRank = 4;
    break;
case "categoryName":
    wordRank = 3;
    break;
case "productName":
    wordRank = 2;
    break;
case "description":
    wordRank = 1;
    break;
```

Şekil 14 – Puanlama örneği

Microsoft SQL Server'da bulunan tablolardan Setting kısmında uygulamanın ayarları bulunmaktadır. SearchLog tablosunda, daha önce yapılmış aramalar bulunmaktadır. LocalString kısmında, arama motorunda gösterilecek kelimelerin girilmesi birer anahtara bağlanmıştır. Bu anahtarlar HTML sayfalarına girilerek, veritabanında bunun karşılığını almaya çalışırlar. Eğer bulunamazsa, bu anahtarların kendilerinin gösterimi sağlanır. Diğer tablolarda ise makine öğrenimi ile ilgili işlemlerde kullanılacak veriler saklanmaktadır. Veritabanında bulunan tablolardan bazıları Şekil 15'teki gibidir. Şekil 16'da da veritabanı tablolarının içerdiği sütunlar bulunmaktadır.

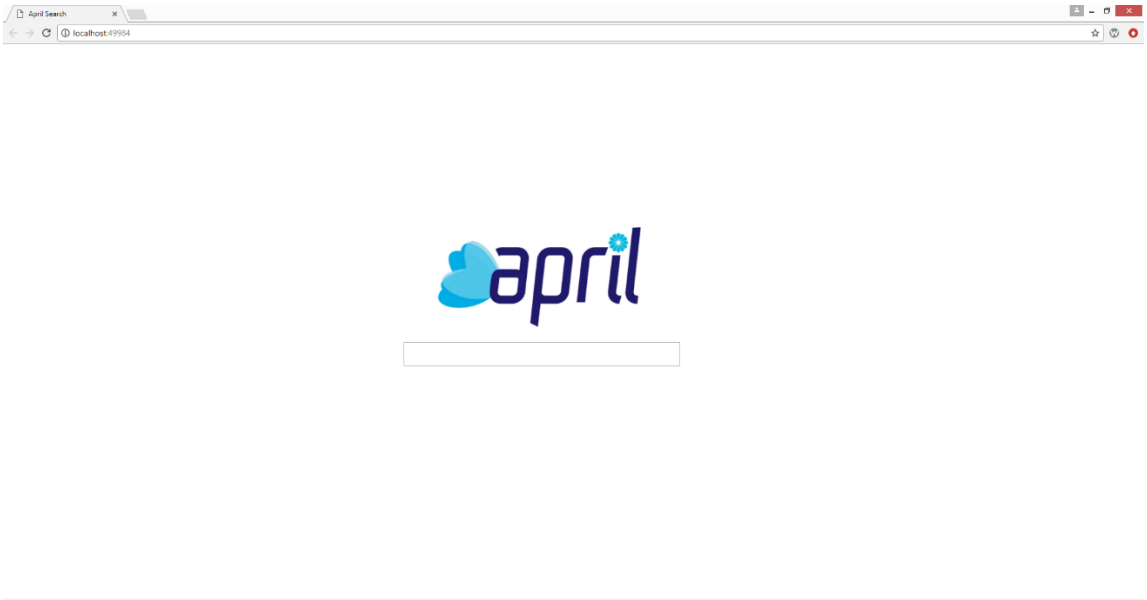


Şekil 15 – Veritabanı tabloları

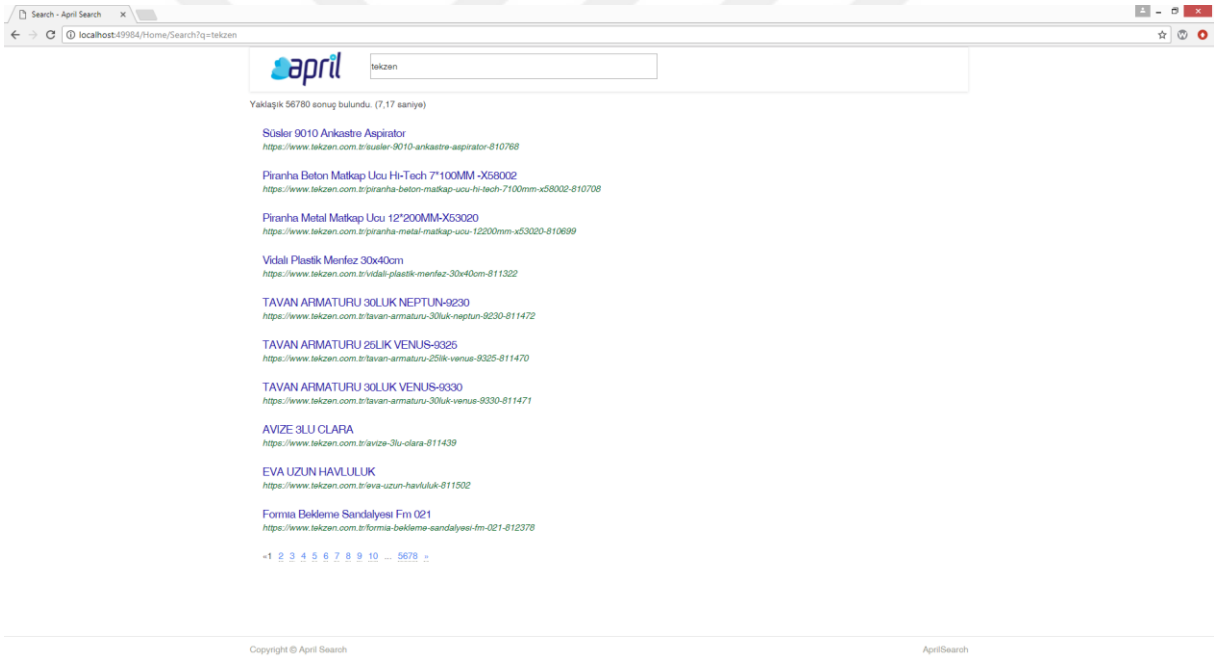


Şekil 16 – Veritabanı tabloları ve sütunları

Ön yüz çok basit bir şekilde aranacak kelimeyi kullanıcıdan alıp, sonuçları sayfalama yapısıyla birlikte kullanıcıya göstererek, basit bir tarasım ile kullanıcının arama amacına uygun şekilde yapılmıştır. Arama ekranı ve sonuç ekranının örnek görüntüleri Şekil 17 ve Şekil 18'deki gibidir.



Şekil 17 – Arama sayfası



Şekil 18 – Arama sonuç sayfası

Arama sonuç sayfasında detayların kullanıcıya gösterilmemesinin sebebi, bu uygulamanın bir e-ticaret uygulamasından alınan verilerle oluşturulmuş olması ve uygulama bazlı bir arama motoru oluşundan kaynaklanmaktadır. Daha fazla detay verilmek istenirse, Web katmanındaki Models klasöründe bulunan HomeModel kısmına sadece bir özellik (field) eklenerek, ön yüzde bu modelin özelliğinin çağrılması ile basitçe gösterilebilir.

Uygulamada bir ranking yani puanlama sistemi kullanılmıştır. Bu puanlama sistemi, uygulama bazlı bir arama motoru olduğu için, Elasticsearch üzerinde bulunan verilerin

alanlarına istinaden düzenlenmiş olup, verinin önem derecesine göre bir puanlama yapılmaktadır. Bu alanlar, marka ismi, ürün ismi, açıklama gibi alanlardır.

Puanlamaya ek olarak, arama motorlarında SEO şeklinde bir kavram bulunmaktadır. Açılımı Search Engine Optimization yani Arama Motoru Optimizasyonu'dur. Bu mantığa göre, sisteme bazı kelimeler yani literatüre göre taglar tanımlarız. Buradaki amaç, link ve içeriğinde gelen tüm verileri taramak yerine; SEO tanımlamaları yapılmış sonuçları daha önce getirerek (seo verilerindeki indekslemeye göre arama sonuçları, seo keliemeleri arasında daha hızlı gelecektir), hem arama motorunun daha doğru sonuçlar vermesini sağlamak, hem de makine öğrenmesi kısmında sistemin daha anlamlı bir biçimde eğitilmesi anlamına gelmektedir. Bunlar düşünülerek, puanlama kısmında SEO'ya yönelik bir alan verilmiş olup, sonuçların daha doğru bir biçimde dönmesi sağlanmıştır.

Bulanık arama konusunda Elasticsearch'ün sunduğu imkanlar olmasına karşın uygulama içerisinde öneri sisteminde farklı bir algoritma kullanılmıştır. Eldeki datalar ayrıştırılarak, aranan kelimeye en yakın sonuç çıkarılarak öneri sunulur. Bu yapılırken, özelleştirilmiş bir benzerlik algoritması kullanılmıştır. Elasticsearch'ün benzerlik algoritmasının kullanılması, uygulamanın sonraki dönemlerde geliştirilceği düşünülürse, kısıtlayıcı bir kıstas olup; daha anlamlı sonuçlar bulunmasında makine öğreniminin verimini düşürecektir.

Otomatik tamamlama için, Elasticsearch veritabanına alınmış veriler, kelimelerine ayrıştırılarak ve tekilleştirilerek, kayıtlarına bağlı olarak Microsoft SQL Server'a tekrar aktarılmıştır. Burada kelimeler indekslenerek, otomatik tamamlama için en iyi 5 sonuç kullanıcıya gösterilmiştir. Burada, arama sayıları da önemlidir. Daha önce aratılmış sonuçlar üzerinden sitelere gidilirken, bu kayıtlar sistemde tutulur ve otomatik tamamlama için en doğru sonuçlar verilmeye çalışılmaktadır.

Otomatik tamamlama için bir kıstas koyulması, sistemin performansını artırabilmektedir. Arama motorunda bu projede bir hesap mantığı olmadığı için, kullanıcıların geçmiş aramaları kişi bazında tutulup, kullanıcılara yine kişi bazında sunulmamaktadır. Bundan ötürü tek harfle arandığında otomatik aramada sonuç çıkması anlamsız olacaktır. Bundan ötürü 2 ya da 3 harften sonra otomatik tamamlamanın yapılması daha anlamlı sonuçlar alınmasını sağlayacaktır. Projenin alt yapısına hesap mantığı da eklenir, kullanıcılar çerezle tanınmaya başlanırsa, yapılan arama kayıtlarına kullanıcı bilgileri de eklenebileceğinden, kullanıcıya özel sonuçlar dönülebilir.

4.1) Uygulamadaki Farklılıklar

Google'ın arama algoritmaları sadece metin arama üzerine değildir. Google'ın açıkladığı algoritmalar aşağıdaki gibi sıralanmıştır [18]:

- Yanıtlar
- Mobil
- Otomatik Tamamlama
- Haberler
- Yazım
- Kitaplar
- Sorguyu Anlama
- Eş Anlamlılar
- Güncellik
- Güvenli Arama
- Evrensel Arama
- Google Anında Arama
- Arama Yöntemleri
- Kullanıcı Bağlamı
- Görseller
- Site ve Sayfa Kalitesi
- Videolar
- Dizine Ekleme
- Bilgi Grafiği

Bu algoritmalar, Google'ın devasa boyuttaki verilerini anlamlı kılmak için günden güne geliştirilmektedir. Metin arama algoritmalarını incelediğimizde, April Search kendi öğrenme ve yorumlama algoritmalarında daha başarılıdır. Google, verilerini toplarken, web sayfalarının bazı formatlarda hazırlanması gerektiğini, yoksa bu sayfaların indekslenmeyeceğini belirtir. Hatta bazı servisleri içinse ücret talep etmektedir. Bunlar, aramada üst sıralara çıkılması için zaman zaman manuel işlemlerin yapılması anlamına gelmektedir (Google'ın alışveriş modülü gibi). April Search'te ise indekslemeler, sayfaların içeriklerine göre yapılmaktadır.

April Search'ün uygulama bazlı bir arama motoru olduğunu düşünürsek ve Google'ın algoritmasından daha iyi bir ayrıştırmaya gittiğini de hesaba katarsak; herhangi bir uygulama için NoSQL yapısıyla en doğru yanıtları verebilen ve en performanslı şekilde çalışan arama motoru olacaktır.

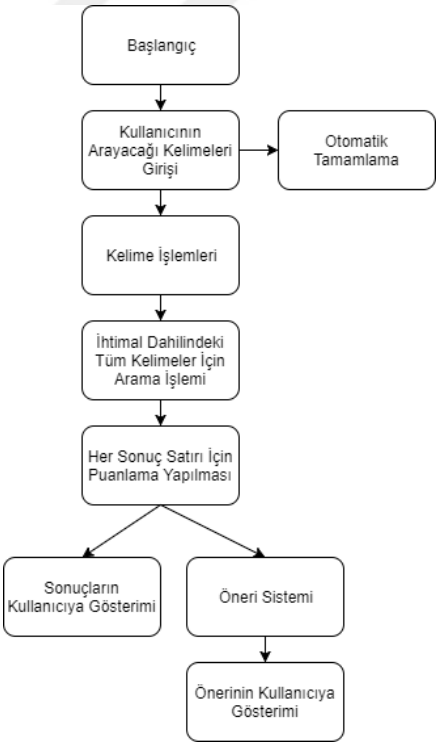
PageRank, sayfaların içerdiği kelimelere ve sayfalarda öne çıkan kelimelere bakılarak hesaplanmaktadır. Burada Google'ın bir çok kriteri vardır ve tüm sayfaları indeksleyememektedir. April Search'te ise (aksi belirtilmektedikçe) veritabanındaki tüm veriler değerlendirilerek kullanıcılara sonuç sunulmaktadır.

Arama motorlarında genellikle arama işlemi bir bütün olarak ele alınmamakta olup, tam metin arama algoritmaları ve eş anlamlı kelimelerin aratılarak asıl sonuçların bulunması gibi senaryolar birbiriyle çakışmakta ve verimli sonuçlar alınamamaktadır. Son dönemlerde makine öğrenmesi ile ilgili geliştirmeler de bu işin içine dahil olduktan sonra, makine öğrenmesindeki en büyük tehlike olan "ezber" terimi bir çok arama için gerçekleşmeye başlamış ve manuel çözümler getirilmiştir.

April Search'ün bir diğer farklılığı, herhangi bir projeye ayrık olarak entegre edilebilir olmasıdır. April Search'ün verileri, SQL'den Elasticsearch'e bir konsol uygulaması aracılığıyla aktarılmıştır. Bu konsol uygulaması, ADO.NET Entity Framework aracılığıyla verileri çekerek, uygun formata getirilerek Elastisearch'e aktarımı sağlanmıştır. Herhangi bir uygulama için de bu veriler, yine aynı yöntem ile belirli periyotlarla aktarılabilir ve April Search bu verilerle, uygulamalara özel olarak ayrık bir biçimde çalışabilir. Algoritmayla herhangi bir oynama yapmadan, arayüzde de değişiklik yapılarak, herhangi bir web sayfasının formatına çevrilebilir ve bu esnekliği sayesinde daha verimli arama yapılabilirken, kullanıcı deneyimi konusunda da herhangi bir problem yaşanmamış olacaktır.

4.2) Uygulamanın Algoritması

Uygulamanın akış şeması Şekil 19'da gösterilmektedir.



Şekil 19 – Akış şeması

Akışta da görüleceği üzere, kullanıcının arayacağı kelimeleri girişi esnasında işlemler başlamaktadır. Kullanıcı kelime girerken, otomatik tamamlama fonksiyonları çalışmaktadır. Otomatik tamamlama, SQL'de tutulan ve daha önce aratılmış verilerle çalışmaktadır. Bu veriler, daha önce puanlaması yapılarak kullanıcıya sunulmuş arama sonuçlarından alınarak tekrar kullanıcıya sunulmaktadır. Anlamlı kelimelerin sunulabilmesi için, en az 3 karakter girişinin yapılması gerekmektedir. 3 karakterden az girişler için sınırlı sayıda kelime olması, sunulan sonuçların anlamlı olma ihtimalini azaltacağı için bu sınır konulmuştur. Uygulama üzerinde bu fonksiyonu çalıştırabilmek için ilk veriler, Elasticsearch'e alınan verilerin kelimelerine parçalanarak, bu kelimelerin tekilleştirilmesi sonucu SQL veritabanına atılmıştır. Ajax yöntemi ile anlık olarak her harf değişimi sonrası bu tablodan veriler çekilmektedir.

Burada da ufak çaplı bir arama işlemi yapılmaktadır ve ortaya bir soru daha çıkmaktadır. Bu soru, "Neden arama işlemlerini Elasticsearch'te yaparken, kelime öneri işlemleri SQL'de yapılmaktadır?" sorusudur. Elasticsearch'te, büyük çaplı veriler JSON formatında tutulurken, tek bir alan için bu formatı kullanma gereksinimi bulunmamaktadır. SQL'de veriler, bir sütunluk bir tabloda tutulmakta olup, bu tablonun indekslenmesi sonucu verimli bir sonuç alınabilmektedir. Bu işlemin Elasticsearch üzerinde de yapılmasında bir sakınca bulunmamaktadır. Otomatik tamamlama, asenkron olarak arama işleminden ya da kelime yazma işleminden bağımsız çalıştığı için, sistemin performansını ya da çalışmasını engelleyecek bir risk içermemektedir.

Kelime işlemleri adımında, kelimelerin ihtimalleri çıkarılmaktadır. Burada yanlış yazım ihtimallerine karşın ve dillerdeki özel harflere karşın bir kelimenin tüm ihtimalleri bir liste olarak ortaya çıkarılmaktadır. Bu adım gerçekleştirilirken kullanılan algoritma, bubble sort yani sıralama algoritmasına benzer olan ama özgün bir algoritmadır. Önce aranan yazı bir cümle ya da kelime olup olmadığı ayrıştırması yapılmaktadır. Sonrasında ise ayrıştırılan ya da tek başına aratılan kelimenin yazım yanlışı ya da çekim eki alma ihtimallerine göre harflerinin yeri değiştirilerek, aranacak kelimeler listesine alınmaktadır. Sonrasında ise dillere özel harf ve karakter değişiklikleri yapılarak, bunlar da aranacak kelimeler listesine eklenir ve tüm ihtimaller ortaya çıkarılmış olur.

Tüm kelime ihtimalleri ortaya çıkarıldıktan sonra bu arama sonuçları Elasticsearch üzerinde aratılır. Aratılan sonuçlar bir listeye toplanır ve bu liste üzerinde bir puanlama işlemi yapılır. Bu puanlama işlemi, aratılan kelimelerin hangi sonuçla daha çok bağlantılı olduğunu göstermektedir. Puanlama işlemi kelimelere göre yapılmaktadır. Eğer bir cümle ya da kelime grubu aratılmışsa, bulunan sonuçlar içerisinde iki defa puanlama yapılacağından, bağlantılı sonuçlar otomatik olarak daha yüksek puan alacağından, bu sonuçlar daha ön plana çıkacaktır ve kullanıcının istediği veriler ön planda bulunabilecektir. Puanlama işlemi de yapıldıktan sonra sonuçlar kullanıcıya sunulur.

Ayrıca April Search içerisinde bir öneri sistemi bulunmaktadır. Bu öneri sistemi, aratılan kelimenin doğru sonuç getirmemesi durumunda, kullanıcıya "Bunu mu demek

istediniz?” sorusunu sorarak, en yakın kelimeyi kullanıcıya sunmaktadır. Bu işlemi yaparken Levenshtein Distance algoritması kullanılmaktadır. Otomatik tamamlamanın da kullandığı tablodan çekilen otomatik tamamlama kelimeleri ile girilen yazı, Levenshtein Algoritması’na göre ölçümlenir. Bu ölçümleme yapılırken, her bir kelime için bir oran hesaplaması yapılır. Hesaplama öncesinde verilmiş sabit bir oranın üzerinde kalan oranlar, ihtimaller listesine eklenir. Diğer sonuçlar ise elenir. İhtimaller listesinde ise oranı en yüksek olan kelimeler kullanıcıya sunulur. Bu algoritma ile ilgili kod parçacığı aşağıda bulunan şekil 20’de gösterilmiştir.

```
int[,] d = new int[enteredWord.Length + 1, targetWord.Length + 1];
int i, j, cost;
char[] str1 = enteredWord.ToCharArray();
char[] str2 = targetWord.ToCharArray();

for (i = 0; i <= str1.Length; i++)
{
    d[i, 0] = i;
}
for (j = 0; j <= str2.Length; j++)
{
    d[0, j] = j;
}
for (i = 1; i <= str1.Length; i++)
{
    for (j = 1; j <= str2.Length; j++)
    {
        if (str1[i - 1] == str2[j - 1])
            cost = 0;
        else
            cost = 1;

        d[i, j] = Math.Min(d[i - 1, j] + 1, Math.Min(d[i, j - 1] + 1, d[i - 1, j - 1] + cost));

        if ((i > 1) && (j > 1) && (str1[i - 1] == str2[j - 2]) && (str1[i - 2] == str2[j - 1]))
        {
            d[i, j] = Math.Min(d[i, j], d[i - 2, j - 2] + cost);
        }
    }
}

return d[str1.Length, str2.Length];
```

Şekil 20 – Levenshtein Distance algoritması

4.3) Puanlama Algoritması

Makine öğrenmesi, bir problemi eldeki veriye göre modelleyen algoritmalardır. Günümüzde, daha doğru sonuçlar alınabilmesi için eldeki verileri en iyi şekilde işlemek ve kullanıcıya sunmak esastır. Bundan ötürü makine öğrenmesi algoritmalarından birini kullanmak ya da kendi algoritmamızı geliştirmek kaçınılmazdır. Makine öğrenimi algoritmalarını oluştururken, performans ile ilgili sorunlar da göz ardı edilmemelidir.

Makine öğrenmesinde yapay sinir ağları ve bulanık mantık algoritmaları gibi algoritmalar, performans anlamında sistemleri çok yormakta ve sonuç hazırlama da çok yavaş kalmaktadırlar. Bayes Teoremi, karar ağaçları, benzerlik-uzaklık algoritmaları ise daha hızlı

sonuçlar vermektedir. Bu algoritmaların uyarlanması esnasında da doğru modeli oluşturmaya çalışırken, tam anlamıyla doğru sonuç alınamayacağı ya da performans alınamayacağı, testler aşamasında görüldüğünden, puanlama algoritması geliştirilmiştir. Bu puanlama algoritması, verilerin içerisinde aranacak alanlardan, hangi veriye ne kadar puan verileceğine kadar bazı kısıtlar ve ayarlar bulundurmakta; buna göre hesaplamalar yapmaktadır. Bu modelleme ile performanstan çok önemli bir derecede kazanımda bulunduğumuz gibi, sonuç doğruluğunu konusunda da verimli bir sistem oluşturmuş olduk. Yapı olarak kısmen karar ağaçlarına benzese de, karar ağaçlarından farklılaşmış bir algoritma yapılmıştır.

5. SONUÇ

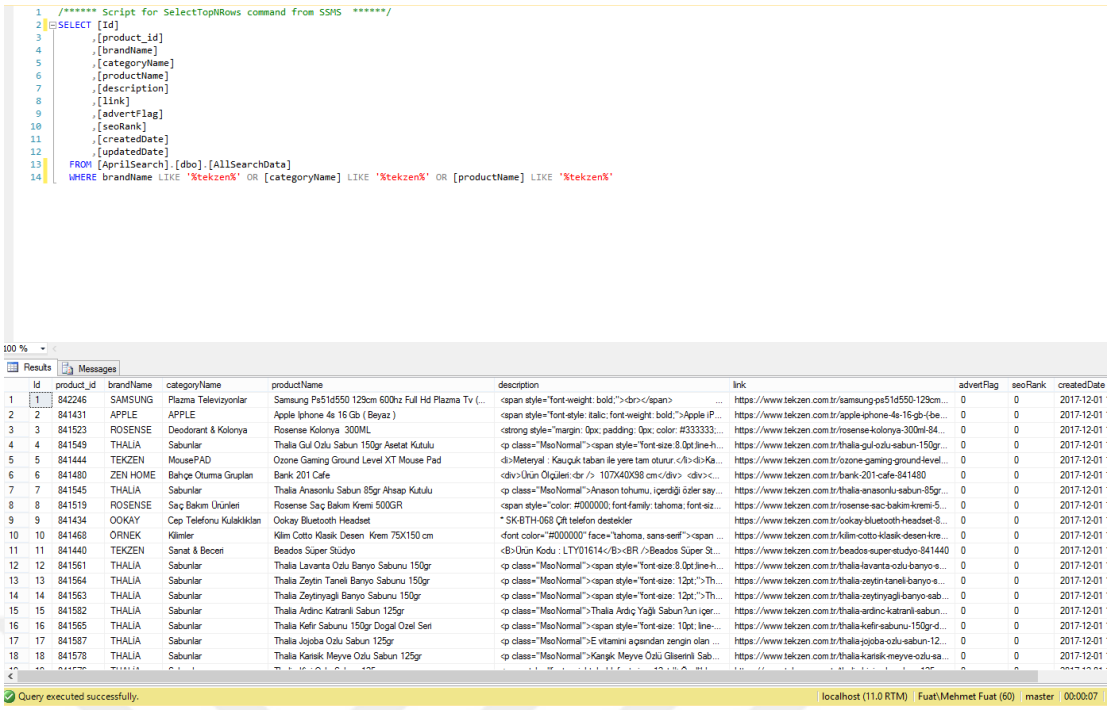
Bu tez çalışmasının ana hedefi, arama fonksiyonlarının daha efektif bir biçimde yapılmasıdır. Uygulama, katmanlı mimariye ve esnekliğe sahip olduğundan ötürü; bir web arama motoru olarak hazırlansa dahi, uygulamalara da kolayca entegre edilerek kullanılabilecek şekilde dizayn edilmiştir. Tekzen E-Ticaret projesindeki arama motorunun yetersizliğinden esinlenerek yola çıkılmış bu proje; herhangi bir web projesine asenkron olarak çalışacak şekilde, ufak değişiklikler yapılarak kolayca entegre edilebilir.

Uygulamanın performansını incelediğimizde ise, bir çok puanlama ve aranan kelimelerin işlenmesi ile ilgili çeşitli işlemler yapılmasına karşın, verilerin kullanıcıya 7 saniyede verilmesi çok önemli bir noktadır. Kişisel bilgisayarda alınmış bu sonuç, güçlü bir sunucuda ve uygulamanın daha da geliştirilmesi ile yerli bir uygulama olarak dünyanın en iyi arama motoru olan Google'a rakip olabilecektir. Şekil 21'de arama işleminin süresi ile ilgili görüntü bulunabilir.

Şekil 21 – Arama işleminin süresi

Sonuç süresi incelendiğinde, yeni teknolojilerin kullanılmasının ne kadar önemli olduğu da ortaya çıkmıştır. İlişkisel veritabanı olan Microsoft SQL Server'da bu arama yapıldığında, veriler işlenmeden, sadece çekilmeye çalışıldığında dahi çok daha fazla sürecektir. Son dönemlerde büyük veri ile ortaya çıkan NoSQL yapılarının ve bu ihtiyaca yönelik doküman tabanlı veritabanlarının önemi bu tip uygulamalarda daha iyi anlaşılmaktadır. NoSQL yapılarında önbellekle ilgili de çalışmalar yapılmış olup; bu uygulamanın sonraki safhalarında daha hızlanması amacıyla, yeni tipte önbellek işlemleri de kullanılabilecek altyapı hazırlanmıştır.

Şekil 22'de, Microsoft SQL Server'da "tekzen" kelimesi aratıldığında geçen süre bulunmaktadır. April Search'ün puan algoritması ve "Bunu mu demek istemiştiniz?" işlemlerini yaptığını da hesaba katarsak, Microsoft SQL Server'dan verilerin çekilmesi durumunda bu işlemin 3 saniyenin üzerine çıkması ön görülmektedir.



```
/****** Script for SelectTopNRows command from SSMS  ******/
SELECT [Id]
      ,[product_id]
      ,[brandName]
      ,[categoryName]
      ,[productName]
      ,[description]
      ,[link]
      ,[advertFlag]
      ,[seoRank]
      ,[createdDate]
      ,[updatedDate]
  FROM [AprilSearch].[dbo].[AllSearchData]
  WHERE brandName LIKE '%tekzen%' OR [categoryName] LIKE '%tekzen%' OR [productName] LIKE '%tekzen%'
```

100 %

Results | Messages

	Id	product_id	brandName	categoryName	productName	description	link	advertFlag	seoRank	createdDate
1	1	842246	SAMSUNG	Plazma Televizyonlar	Samsung Ps51d550 129cm 600hz Full Hd Plazma Tv (...	 ...	https://www.tekzen.com.tr/samsung-ps51d550-129cm...	0	0	2017-12-01
2	2	841431	APPLE	APPLE	Apple Iphone 4s 16 Gb (Beyaz)	Apple iP...	https://www.tekzen.com.tr/apple-iphone-4s-16-gb-(be...	0	0	2017-12-01
3	3	841523	ROSENSE	Deodorant & Kolonya	Rosense Kolonya 300ML	<strong style="margin: 0px; padding: 0px; color: #333333;...	https://www.tekzen.com.tr/rosense-kolonya-300ml-84...	0	0	2017-12-01
4	4	841549	THALİA	Sabunlar	Thalia Gul Ozlu Sabun 150gr Asetat Kutulu	<p class="MsoNormal"><span style="font-size:8.0pt;line-h...	https://www.tekzen.com.tr/thalia-gul-ozlu-sabun-150gr...	0	0	2017-12-01
5	5	841444	TEKZEN	MousePAD	Ozone Gaming Ground Level XT Mouse Pad	Meteryal : Kauçuk taban ile yere tam oturur. Ka...	https://www.tekzen.com.tr/ozone-gaming-ground-level...	0	0	2017-12-01
6	6	841480	ZEN HOME	Bahçe Oturma Grupları	Bank 201 Cafe	Ürün Ölçüleri : 107X40X98 cm <...	https://www.tekzen.com.tr/bank-201-cafe-841480	0	0	2017-12-01
7	7	841545	THALİA	Sabunlar	Thalia Anasonlu Sabun 85gr Ahsap Kutulu	<p class="MsoNormal">Anason tohumu, içerdiği özler say...	https://www.tekzen.com.tr/thalia-anasonlu-sabun-85gr...	0	0	2017-12-01
8	8	841519	ROSENSE	Saç Bakım Ürünleri	Rosense Saç Bakım Kremi 500GR	<span style="color: #000000; font-family: tahoma; font-siz...	https://www.tekzen.com.tr/rosense-sac-bakim-kremi-5...	0	0	2017-12-01
9	9	841434	OOKAY	Cep Telefonu Kulaklıkları	Ookay Bluetooth Headset	* SK-BTH-068 Çift telefon destekler	https://www.tekzen.com.tr/ookay-bluetooth-headset-8...	0	0	2017-12-01
10	10	841468	ÖRNEK	Kilimler	Kilim Cotto Klasik Desen Krem 75X150 cm	<font color="#000000" face="tahoma, sans-serif"><span ...	https://www.tekzen.com.tr/kilim-cotto-klasik-desen-kre...	0	0	2017-12-01
11	11	841440	TEKZEN	Sanat & Beceri	Beados Süper Stüdyo	<B>Ürün Kodu : LTY01614</B> Beados Süper St...	https://www.tekzen.com.tr/beados-super-studyo-841440	0	0	2017-12-01
12	12	841561	THALİA	Sabunlar	Thalia Lavanta Ozlu Banyo Sabunu 150gr	<p class="MsoNormal"><span style="font-size:8.0pt;line-h...	https://www.tekzen.com.tr/thalia-lavanta-ozlu-banyo-s...	0	0	2017-12-01
13	13	841564	THALİA	Sabunlar	Thalia Zeytin Taneli Banyo Sabunu 150gr	<p class="MsoNormal">Th...	https://www.tekzen.com.tr/thalia-zeytin-taneli-banyo-s...	0	0	2017-12-01
14	14	841563	THALİA	Sabunlar	Thalia Zeytinyagli Banyo Sabunu 150gr	<p class="MsoNormal">Th...	https://www.tekzen.com.tr/thalia-zeytinyagli-banyo-sab...	0	0	2017-12-01
15	15	841582	THALİA	Sabunlar	Thalia Ardınc Katranli Sabun 125gr	<p class="MsoNormal">Thalia Ardıç Yağlı Sabun?un içer...	https://www.tekzen.com.tr/thalia-ardinc-katranli-sabun...	0	0	2017-12-01
16	16	841565	THALİA	Sabunlar	Thalia Kefir Sabunu 150gr Dogal Ozel Seri	<p class="MsoNormal"><span style="font-size: 10pt; line-...	https://www.tekzen.com.tr/thalia-kefir-sabunu-150gr-d...	0	0	2017-12-01
17	17	841587	THALİA	Sabunlar	Thalia Jojoba Ozlu Sabun 125gr	<p class="MsoNormal">E vitamini açısından zengin olan ...	https://www.tekzen.com.tr/thalia-jojoba-ozlu-sabun-12...	0	0	2017-12-01
18	18	841578	THALİA	Sabunlar	Thalia Karisik Meyve Ozlu Sabun 125gr	<p class="MsoNormal">Karışık Meyve Özlü Gliserinli Sab...	https://www.tekzen.com.tr/thalia-karisik-meyve-ozlu-sa...	0	0	2017-12-01

Query executed successfully. | localhost (11.0 RTM) | Fuat\Mehmet Fuat (60) | master | 00:00:07

Şekil 22 – Microsoft SQL Server üzerinde arama yapılması

Microsoft SQL Server'da bu sorgunun daha uzun süreceğini şu örnekten de anlayabiliriz. LIKE komutu ile bir alan içerisinde herhangi bir harf ya da karakter girdiğinizde, bu alandaki tüm karakterler taranmaktadır. Eşittir ile direk sonuca erişebilmek daha hızlı iken, bir arama motorunda her zaman var olan veriye eşit bir arama mümkün olmayabilir. Sistemde 181373 adet kayıt bulunmaktadır. Her bir dökümanın ise 10 alanı bulunmaktadır. Aramalar ise bu alanlardan sadece 5'inde yapılmaktadır. Bu alanlar ürün markası, kategori adı, ürün adı, seo etiketi ve açıklama alanlarıdır. Açıklama alanı, bu alanlar arasında en çok veri içeren alandır. Kimi zaman bir sayfayı dolduracak kadar metin içermektedir. Bunun içerisinde, "içerir" şeklinde bir sorgu yazdığımızda, 7 saniye gibi bir sürede 56780 kaydı getirmesi, NoSQL yapıların arama motorlarında kullanılması zorunluluğunu bizlere göstermektedir.

Bu arama motorunda kurulan yapıyı Microsoft SQL Server üzerinde düşünürsek, bu 5 alanın da farklı tablolarda tutulması gerektiğini görürürüz. Tekzen E-Ticaret veritabanından bu verileri çekerken yaklaşık 3 dakika süren bir sonuç beklenilmekteydi. Bu, verinin büyüklüğünün yanı sıra, veritabanı aramasından da kaynaklanmaktadır.

Literatürdeki diğer çalışmaların birçoğu; ya mevcut arama motorlarının analizinden oluşmaktadır ya da çok basit yapılara sahip olduklarından ötürü, efektif çalışmamaktadırlar. Yapılan bu uygulamada, kullanıcıya en hızlı sonuç, en doğru şekilde getirilmektedir.

Ayrıca son teknolojilerin kullanılacak olması, projenin sadece araştırılmasından daha çok, gerçek hayata uyarlanabilir oluşunu desteklediği gibi, bu uygulamayı örnek alacak yeni uygulamalar geliştirecek kişilere de geniş bakış açısı katacaktır. Yapılmış olan bu uygulamanın

diğer fazları ve genişletilebilir olduğu düşünüldüğünde; yeni çıkacak teknolojilerin entegrasyonu, var olan algoritma ya da entegrasyonların gelecekte daha iyileri ile değişimi de kolay olacaktır.

Ek olarak, uygulama bazlı bir arama motoru olduğu için, puanlama işleminin yapıldığı katmana kolayca yeni puanlama türleri ve şekilleri eklenerek, arama sonuçları ile ilgili, entegre edilecek uygulamalar ya da veritabanlarına uygun çalışma sağlanabilmektedir.

Tüm bunlara ek olarak, veritabanı üzerlerinde daha fazla optimizasyonlar da yapılabilmektedir. Veritabanı yöneticileri genelde tablolar üzerindeki hareketleri inceleyerek, indeksleri yeniden düzenlemek ya da indeks belirlemek gibi, sunucu yapısında değişiklikler yapmak gibi işlemler yapabilmektedirler. Örneklemek gerekirse Microsoft SQL Server'da partionaning, yani veriyi bölme yapısı kullanılabiliyorken; Elasticsearch içinse shard ve replica gibi sistemler kullanılmaktadır.

Literatür çalışmasında da görüleceği üzere bu tip bir çalışma ülke içerisinde daha önce yapılmamıştır veya başarısız olmuştur. Tübitak'ın şirketlere yenilik getiren, ülkeye yenilik getiren ve uluslararası yeniliklere göre verdiği proje destekler göz önüne alındığında, yapılacak bu projenin spesifik bir proje olmamasına ve ülkeye yenilik getirecek ve vizyon katacak bir proje olması amaçlanmıştır. Son dönemde büyük şirketlerin çıkardığı ve altyapısında diğer global şirketlerin arama motorlarını kullanan uygulamaların yerine, bu tip yeni teknolojileri kullanan ve üzerine gidildiğinde gelecekte global olacak bir projenin oluşumu; yeni istihdam kapıları açacağı gibi, ülkenin teknoloji anlamında tanıtımında da büyük rol oynayacaktır.

Arama motorlarından Google, Bing ve Yandex'e "Tekzen" kelimesi yazıldığında Şekil 23'teki gibi bir tablo ortaya çıkmıştır.

	Google	Yandex	Bing	April
Sonuç Sayıları	1.510.000	-	513.000	56780 / 181373
Süre	0.50 sn	-	-	7.17 sn
SEO	Var	Yok	Var	Var
Makine Öğrenimi	Var	Var	-	Var

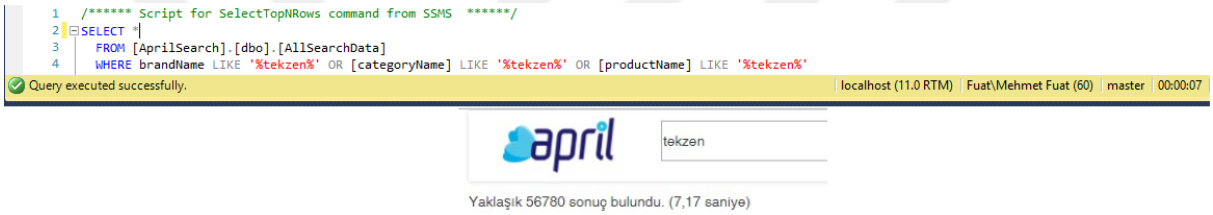
Şekil 23 – Arama motorlarının sonuçları

Google ve diğer arama motorlarına bakıldığında, bilgi kirliliğinin fazla olduğu ve ilk sayfada gelen sonuçların birbirinden farklı olduğu gözlemlenmektedir. Buna istinaden daha iyi filtrelemeler ile, daha doğru sonuçları kullanıcıya göstermek için, daha iyi bir algoritmaya

ihtiyaç duyulmaktadır. Veriler, doğru yorumlandığı taktirde, teknoloji daha da iyileştirilerek, süre düşürülebilecektir.

April Search algoritmasını farklı kılan etkenler üzerinde konuşmak gerekirse; günümüzde bir çok kılan NoSQL veritabanları olsa bile, doğru alanlarda doğru veritabanlarının kullanılmadığı görülmektedir. Bundan ötürü MongoDB NoSQL veritabanının (dbengines) kullanım oranının şu anda zirvede olduğunu görebiliriz. Bu veritabanlarının yapılarını incelediğimizde, hepsinin kullanım alanının özel olduğunu ve daha fazla performansın alınmasının mümkün olduğu da görülebilir. Redis, genellikle önbellekte tutulacak işlemler için kullanılan bir veritabanıyken, Elasticsearch ise indekslemelerinden ötürü arama işlemlerinde özelleşmiştir.

Kullanımı en yaygın olan Microsoft SQL Server ile Elasticsearch arasındaki fark ise Şekil 24'te gösterilmektedir.



Şekil 24 – Süre karşılaştırması

Verilerin hızlı gelmesi ne kadar önemli ise doğru gelmesi de o kadar önemlidir. Hızlı çalışan bir sistemde, son kullanıcılar aradığı sonuçlara ulaşamıyorsa, sistemin hızlı çalışmasının bir önemi bulunmamaktadır. Bu yüzden verinin anlamlandırılması da çok önemli bir hususdur. Ortaya çıkan ilk arama motorları, veri biriktirmeye yönelmekteyken, Google'ın çalışmaları verinin anlamlandırılması ve en çok ziyaret edilen sitelerin tespiti üzerineydi. Var olan veri toplama yöntemlerinin aynısını ya da bir seviye daha üstünü uyarlamak yerine daha farklı bir açıdan yaklaşılması, şu anda Google'ın dünyanın en büyük arama motoru olmasının temellerini atmıştır.

Genellikle arama motorları veri toplamak için botlar üretirken, algoritmasının anlamlı sonuçlar getirmesi sonrası büyüyen Google, şu anda arama sonuçlarında ilk sıralarda çıkmak isteyen şirket ve kişilerden ücret almaktadır. Kendi veri toplama botlarını da devrede tutan Google, internet sitelerinin ara yüzlerini tarayan botlarının, siteleri daha iyi anlayabilmesi için bir çok kıstas koydurtmuştur.

Bir arama motorunu yaparken üstünde durulacak hususlar şu şekildedir;

- Yeni teknolojilerin kullanılması

- Bu projenin katmanlı mimari de hazırlanmasına dikkat edilmesi
- Birden fazla veritabanının kullanılarak, en efektif şekilde çalışmasının sağlanması
- Projede özelleşmiş makine öğreniminin kullanımı
- Projenin, ayrık olarak çalışabilecek ve hızlı bir biçimde bir web uygulaması için uyarlanabilir olmasının sağlanmasıdır.

Bu hususlar, yeni teknoloji ve yapılara göre değişebilir ya da geliştirilebilirdir. Burada önemli olan ihtiyaçları belirlemek ve bunlarla ilgili çözümler üretmektir.

Son dönemlerde ülkemizde kullanıma sunulmuş olan arama motorlarından Yaani incelendiğinde, diğer arama motorlarının API'lerini kullandığı ortaya çıkmıştır [19][20]. Bir arama motorunun olumlu bir biçimde çalışması için, öncelikle internet ortamından olabildiğince veri toplanmalıdır. Sonrasında ise bu verilerin işlenmesi için gerekli çalışmalar yapılmalıdır. Bir başka arama motorunun sunduğu API'leri kullanmak, var olanı kullanmak anlamına geldiği gibi, ülkemizde yeni teknolojilerin kullanımını sağlamak yerine, kopyalama yoluna gidilmesine sebep olur.

Bu proje de gerçek hayatta ilerletilecek olup, hedeflenen seviyeye getirildikten sonra Uygulama Bazlı Arama Motoru olmaktan çıkıp, ülkemizdeki en büyük arama motoru haline getirilmesi hedeflenmektedir. Bunun için puanlama algoritması geliştirilecek ve daha efektif bir hale getirilecektir. Kelime ihtimalleri çıkarılırken ise eş anlamlı kelimelerin de ihtimaller dahiline eklenmesi sağlanacaktır. Bu algoritmalar geliştirilirken en önemli unsur, verilerdir. Öğrenme ve öğrenilen veriyi yorumlama modellemesi oluşturulurken bir çok kıstas göz önünde bulundurulmaktadır. Bunlardan bazıları verinin boyutu ve içeriği gibi kıstaslardır. Bu kıstaslara göre yeniden performans değerlendirmesi yapılarak, algoritmanın geliştirilmesi sağlanacaktır.

6.KAYNAKLAR

1. TEDx İstanbul Konferansı, 20.12.2017, https://www.youtube.com/watch?v=AXK9ayTVq6c

2. Websitesi Sayıları, 20.12.2017, https://www.theatlantic.com/technology/archive/2015/09/how-many-websites-are-there/408151/

3. Soydal, İ. (2000), *Web Arama Motorlarında Performans Değerlendirmesi*, Yayınlanmış Yüksek Lisans Tezi, Hacettepe Üniversitesi

4. Odabaşıoğlu, C. (2009), *İnternet Arama Motorları Analizi*, Yayınlanmış Yüksek Lisans Tezi, Haliç Üniversitesi

5. Büyük, B. (2009), *İnternet Arama Motoru Kullanıcı Verilerinin Analizinde Simülasyon ve Olasılıksal Yöntemlerin Uygulanması*, Yayınlanmış Yüksek Lisans Tezi, Uludağ Üniversitesi

6. Sezgin, G. (2009), *Arama Motorlarının Davranışlarının Çözümlenmesi ve Web Sayfalarına Tasarım Aşamasında Yansıtılması*, Yayınlanmış Yüksek Lisans Tezi, Beykent Üniversitesi

7. Razbonyalı, C. (2011), *Dikey Arama Motorlarının İncelenmesi ve Bir Dikey Arama Motoru Uygulaması*, Yayınlanmış Yüksek Lisans Tezi, Trakya Üniversitesi

8. Gözükara, F. (2012), *Fiyat Karşılaştırmalı Ürün Arama Motoru Geliştirme*, Yayınlanmış Yüksek Lisans Tezi, Mersin Üniversitesi

9. Sazoğlu, F.B. (2014), *Efficient Result Caching Mechanisms In Search Engines*, Yayınlanmış Yüksek Lisans Tezi, Bilkent University

10. Vidinli, İ.B. (2016), *Eğitim Arama Motorunda Sorgu Önerme*, Yayınlanmış Doktora Tezi, Turgut Özal Üniversitesi

11. Database Ranks, 16.11.2017, http://db-engines.com/en/ranking

12. Elasticsearch References, 16.11.2017, https://www.elastic.co/use-cases

13. Elasticsearch Website, 16.11.2017, https://www.elastic.co

14. Elasticsearch Index, 16.11.2017, http://www.borakasmer.com/elasticsearch-nedir/

15. NoSQL Avantaj ve Dezavantajları, 25.12.2017, https://kodcu.com/2014/03/nosql-nedir-avantajlari-ve-dezavantajlari-hakkinda-bilgi/

16. Levenshtein Distance Algoritması, 25.12.2017 http://www.buraksenyurt.com/post/Levenshtein-Distance-Algoritmasi

17. Elasticsearch Crud .Net Provider, 19.11.2017, https://damienbod.com/2014/09/22/elasticsearch-crud-net-provider/

18. Google Algoritmaları, 25.12.2017, https://www.google.com/intl/tr/insidesearch/howsearchworks/algorithms.html

19. Fuzzy Search In Elasticsearch, 04.12.2017, https://www.elastic.co/blog/found-fuzzy-search

20. Fuzzy Search Algoritmaları, 05.12.2017, https://stackoverflow.com/questions/32337135/fuzzy-search-algorithm-approximate-string-matching-algorithm

21. Gao, J. (2015), Deep Learning for Web Search and Natural Language Processing, Shanghai, Çin

22. Deep Learning for Java, 13.06.2017, https://deeplearning4j.org/

23. Elastic Search Tutorial, 16.11.2017, https://lnubla.wordpress.com/2017/10/27/an-elasticsearch-tutorial-for-net-developers/

24. Retrieving All Records With Nest, 16.11.2017, https://stackoverflow.com/questions/37780803/elasticsearch-search-query-to-retrieve-all-records-nest

25. Elasticsearch Mimari Özellikleri, 25.11.2017, http://www.buraktungut.com/elasticsearch-serisi-02-mimari-ozellikleri-sharding-failovering-ve-scaling

26. Deep Learning, 13.06.2017, http://www.derinogrenme.com/

27. Introducing FBLearner Flow, 13.06.2017, https://code.facebook.com/posts/1072626246134461/introducing-fblearner-flow-facebook-s-ai-backbone/

28. MongoDb NoSQL Explanation, 25.12.2017, https://www.mongodb.com/nosql-explained

29. Yaani İncelemesi, 23.12.2017, http://www.webtekno.com/geliyoo-dan-yeni-yerli-arama-motoru-yaani-ye-kullandiklari-veriler-yandex-ve-google-dan-h35697.html

30. Yaani İncelemesi, 23.12.2017, https://www.teakolik.com/yersiz-arama-motoru/

ÖZGEÇMİŞ

14 Mayıs 1990 tarihinde, İstanbul ilinin Fatih ilçesinde doğdum. İlköğretimi Fatih ilçesinde bulunan Özel Asfa Mustafa Enver İlköğretim Okulu'nda tamamladım. Sonrasında Sakıp Sabancı Anadolu Lisesi'ni kazanmama karşın Üsküdar'da bulunan Özel Asfa Ahmet Mithat Lisesi'nde Matematik-Fen bölümünde okuyarak liseyi bitirdim. Üniversitede ise Doğuş Üniversitesi Bilgisayar Mühendisliği Bölümü'nde eğitim aldım. 4.sınıfta Polonya'nın Politechnika Lubelska Üniversitesi'nde 1 sene boyunca eğitim gördükten sonra, Doğuş Üniversitesi'nde 1 dönem boyunca tezimi hazırladım. Nisan 2014'de Doğuş Üniversitesi'nden mezun oldum. 2015 yılında Beykent Üniversitesi, Fen Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladım.

2014 yılında mezun olduktan sonra Aristo A.Ş.'de Yazılım Geliştirme Uzmanı olarak göreve başladım. Bu firmada .Net üzerinde sadakat uygulamalarının yapımında projeler geliştirdim. 2.5 senelik çalışma sürecimden sonra Tekzen A.Ş.'nin E-Ticaret bölümünde, Şubat 2017'de,Yazılım Geliştirme uzmanı olarak göreve başladım ve kadronun yenilenmesiyle birlikte bir müddet Takım Liderliği yaptım. 2017'nin Aralık ayından itibaren Innova Bilişim Çözümleri A.Ş.'de Uygulama Geliştirme Uzmanı olarak çalışmaktayım.

Hobilerim arasında spor yapmak, teknolojik gelişmeleri takip etmek ve teknolojiyle ilgili konularda eğitimler almak vardır.

Yabancı dil olarak iyi derecede İngilizcem olup, giriş seviyesinde Lehçe ve Almanca bilgim vardır. Askerliğimi tamamlamadım ve bekarım.

Mehmet Fuat Rıhtım

ÖZGEÇMİŞ

14 Mayıs 1990 tarihinde, İstanbul ilinin Fatih ilçesinde doğdum. İlköğretimi Fatih ilçesinde bulunan Özel Asfa Mustafa Enver İlköğretim Okulu'nda tamamladım. Sonrasında Sakıp Sabancı Anadolu Lisesi'ni kazanmama karşın Üsküdar'da bulunan Özel Asfa Ahmet Mithat Lisesi'nde Matematik-Fen bölümünde okuyarak liseyi bitirdim. Üniversitede ise Doğuş Üniversitesi Bilgisayar Mühendisliği Bölümü'nde eğitim aldım. 4.sınıfta Polonya'nın Politechnika Lubelska Üniversitesi'nde 1 sene boyunca eğitim gördükten sonra, Doğuş Üniversitesi'nde 1 dönem boyunca tezimi hazırladım. Nisan 2014'de Doğuş Üniversitesi'nden mezun oldum. 2015 yılında Beykent Üniversitesi, Fen Bilimleri Fakültesi, Bilgisayar Mühendisliği Bölümü'nde yüksek lisans eğitimine başladım.

2014 yılında mezun olduktan sonra Aristo A.Ş.'de Yazılım Geliştirme Uzmanı olarak göreve başladım. Bu firmada .Net üzerinde sadakat uygulamalarının yapımında projeler geliştirdim. 2.5 senelik çalışma sürecimden sonra Tekzen A.Ş.'nin E-Ticaret bölümünde, Şubat 2017'de,Yazılım Geliştirme uzmanı olarak göreve başladım ve kadronun yenilenmesiyle birlikte bir müddet Takım Liderliği yaptım. 2017'nin Aralık ayından itibaren Innova Bilişim Çözümleri A.Ş.'de Uygulama Geliştirme Uzmanı olarak çalışmaktayım.

Hobilerim arasında spor yapmak, teknolojik gelişmeleri takip etmek ve teknolojiyle ilgili konularda eğitimler almak vardır.

Yabancı dil olarak iyi derecede İngilizcem olup, giriş seviyesinde Lehçe ve Almanca bilgim vardır. Askerliğimi tamamlamadım ve bekarım.

Mehmet Fuat Rıhtım