KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YENİ NESİL YAZILIM GELİŞTİRME EĞİLİMLERİNE YÖNELİK UZMAN BİLGİ VE BECERİLERİNİN OLASILIKSAL KONU MODELLEME YORDAMIYLA BELİRLENMESİ

DOKTORA TEZİ

Fatih GÜRCAN

KASIM 2017
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

YENİ NESİL YAZILIM GELİŞTİRME EĞİLİMLERİNE YÖNELİK UZMAN BİLGİ VE BECERİLERİNİN OLASILIKSAL KONU MODELLEME YORDAMIYLA BELİRLENMESİ

Fatih GÜRCAN

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"DOKTOR (BİLGİSAYAR MÜHENDİSLİĞİ)"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 13.11.2017
Tezin Savunma Tarihi : 30.11.2017

Tez Danışmanı : Prof. Dr. Cemal KÖSE

Trabzon 2017

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalında
Fatih GÜRCAN Tarafından Hazırlanan

YENİ NESİL YAZILIM GELİŞTİRME EĞİLİMLERİNE YÖNELİK UZMAN BİLGİ VE BECERİLERİNİN OLASILIKSAL KONU MODELLEME YORDAMIYLA BELİRLENMESİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 14 /11 /2017 gün ve 1727 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
DOKTORA TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Prof. Dr. Cemal KÖSE .....

Üye : Prof. Dr. Resul KARA .....

Üye : Doç. Dr. Mustafa ULUTAŞ .....

Üye : Doç. Dr. Nergiz Ercil ÇAĞILTAY .....

Üye : Yrd. Doç. Dr. Özcan ÖZYURT .....

Prof. Dr. Sadettin KORKMAZ
Enstitü Müdürü

ÖNSÖZ

İçinde bulunduğumuz şu dönemde yaşanan dördüncü sanayi devrimi, büyük ölçüde bilgi teknolojileri odaklı olup yazılım ve bilgi teknolojilerine dayalı endüstriler için çok yeni fırsatlar sunmaktadır. Özellikle web ve mobil tabanlı yazılım uygulamalarında yaşanan hızlı dönüşüm, yazılım geliştirme eğilimlerini ve mimarilerini büyük ölçüde değiştirmiştir. Bu teknolojik dönüşüm doğrultusunda, yazılım geliştiriciler için her geçen gün yeni uzmanlık sahaları açılmaktadır.

Bu kapsamda gerçekleştirilen tez çalışmamda danışmanlığımı üstlenen, her türlü destek ve katkılarıyla çalışmalarıma yön veren çok değerli danışman hocam Sayın Prof. Dr. Cemal KÖSE' ye, yardım ve desteğinden dolayı sonsuz teşekkür ve şükranlarımı sunarım.

Çalışmanın her aşamasında değerli görüş ve desteğini esirgemeyen sayın Doç. Dr. Mustafa ULUTAŞ'a ve Yrd. Doç. Dr. Özcan ÖZYURT'a teşekkürlerimi sunarım.

Tez çalışması sürecinde yardım ve desteklerini esirgemeyen değerli çalışma arkadaşlarım Öğr. Gör. Dr. Yasin KAYA'ya, Arş. Gör. Dr. Mehmet KOKOÇ'a ve Arş. Gör. Fatih ERDOĞDU'ya teşekkürü borç bilirim. Tez çalışmalarımda yardım ve katkılarından dolayı İstatistik ve Bilgisayar Bilimleri bölümündeki değerli arkadaşlarım Yrd. Doç. Dr. Tolga BERBER'e ve Yrd. Doç. Dr. Uğur ŞEVİK'e teşekkürlerimi sunarım.

Öncelikle, beni yetiştirip bu günlere getiren sevgili annem Çiçek GÜRCAN ve babam Hasan Hüseyin GÜRCAN'a, desteğinden dolayı diğer aile fertlerime ve arkadaşlarıma saygı ve sevgilerimi sunarım. Bu vesile ile ilkokuldan bugüne kadar üzerimde emeği olan tüm hocalarıma saygı ve hürmetlerimi sunarım. Ayrıca çalışmaya emeği geçen ismini yazamadığım tüm arkadaşlarıma ve desteğinden dolayı üniversitem, Karadeniz Teknik Üniversitesi'ne teşekkürlerimi sunarım.

Her zaman varlığıyla yanımda olan ve desteğini esirgemeyen çok sevdiğim eşim İpek GÜRCAN'a ve oğlum Efe GÜRCAN'a sevgi ve şükranlarımı sunarım.

Fatih GÜRCAN
Trabzon 2017

TEZ ETİK BEYANNAMESİ

Doktora Tezi olarak sunduğum "Yeni Nesil Yazılım Geliştirme Eğilimlerine Yönelik Uzman Bilgi Ve Becerilerinin Olasılıksal Konu Modelleme Yordamıyla Belirlenmesi" başlıklı bu çalışmayı baştan sona kadar danışmanım Prof. Dr. Cemal KÖSE'nin sorumluluğunda tamamladığımı, verileri/örnekleri kendim topladığımı, deneyleri/analizleri ilgili laboratuvarlarda yaptığımı/yaptırdığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 30/11/2017

Fatih GÜRCAN

İÇİNDEKİLER

Sayfa No

Doktora Tezi

ÖZET

YENİ NESİL YAZILIM GELİŞTİRME EĞİLİMLERİNE YÖNELİK UZMAN BİLGİ VE BECERİLERİNİN OLASILIKSAL KONU MODELLEME YORDAMIYLA BELİRLENMESİ

Fatih GÜRCAN

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Prof. Dr. Cemal KÖSE
2017, 133 Sayfa

Son yıllarda yazılım alanında yaşanan hızlı teknolojik dönüşüm doğrultusunda, yakın gelecekte bu teknolojik dönüşüme ayak uydurabilecek bilgi ve donanıma sahip yazılım geliştirme uzmanlarına yönelik büyük bir talep patlaması öngörülmektedir. Yazılım endüstrisinde yoğun olarak talep edilen yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerinin analizi ve anlaşılması, bu talep ve ihtiyaçların karşılanması açısından stratejik öneme sahiptir. Bu kapsamda, yazılım endüstrisinde ortaya çıkan güncel talep ve eğilimleri belirlemek amacıyla yazılım odaklı iş ilanları üzerinde Gizli Dirichlet Ataması (Latent Dirichlet Allocation) yöntemiyle olasılıksal konu modellemeye dayalı bir anlamsal analiz gerçekleştirilmiştir. Analizin sonucunda elde edilen bulgular, yazılım endüstrisinde yoğun talep gören mesleki rolleri, bu rollere ait temel becerileri, yükselen eğilimleri ve programlama dillerine yönelik talepleri ortaya koymuştur. Bu çalışmada elde edilen bulguların yazılım endüstrisindeki mesleki talep ve eğilimlerin daha iyi anlaşılması açısından katkı sağlayacağı öngörülmektedir.

Anahtar Kelimeler: Olasılıksal konu modelleme, Gizli Dirichlet ataması, Eğilim analizi, Metinsel veri madenciliği, Bilgi çıkarımı, Yazılım geliştirme becerileri, Yazılım endüstrisi, Yazılım geliştirme eğilimleri, Yazılım geliştirme uzmanlık alanları.

PhD. Thesis

SUMMARY

DETERMINATION OF EXPERTISE KNOWLEDGE AND SKILLS RELATED TO NEW GENERATION SOFTWARE DEVELOPMENT TRENDS USING PROBABILISTIC TOPIC MODELING PROCESS

Fatih GÜRCAN

Karadeniz Technical University
The Graduate School of Natural and Applied Sciences
Computer Engineering Graduate Program
Supervisor: Prof. Dr. Cemal KÖSE
2017, 133 Pages

In the recent years, in line with rapid technological transformation in the field of software, a great demand explosion is expected for the software development professionals who have knowledge and equipment that can keep up with this technological transformation in the near future. The analysis and understanding of the expertise knowledge and skills of the next generation software development trends, which are heavily demanded in the software industry, have strategic importance in terms of meeting these demands and needs. In this framework, a semantic analysis based on the probabilistic topic modeling has been performed on software-focused job advertisements using the Latent Dirichlet Allocation method to determine the current demands and trends in the software industry. The findings of the analysis revealed the highly in-demand occupational roles in the software industry, the basic skills of these roles, their rising trends and demands for programming languages. It is envisaged that the findings obtained in this study will contribute to a better understanding of the occupational demands and trends in the software industry.

Key Words: Probabilistic topic modeling, Latent Dirichlet allocation, Trend analysis, Textual data mining, Knowledge extraction, Software development skills, Software industry, Software development trends, Software development expertise areas.

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

BT	:Bilişim Teknolojileri
DF	:Doküman Frekansı
DTM	:Doküman-Terim Matrisi
GAA	:Gizli Anlamsal Analiz
GDA	:Gizli Dirichlet Ataması
MCMC	: Monte Carlo Markov zinciri
OGAA	:Olasılıksal Gizli Anlamsal Analiz
OKM	:Olasılıksal Konu Modelleme
TA	:Terim Ağırlıklandırma
TDF	:Ters Doküman Frekansı
TF	:Terim Frekansı
YGO	:Yazılım Geliştirme Ortamları

1. GENEL BİLGİLER

1.1. Giriş

21. yüzyıl özellikle bilgi ve yazılım teknolojileri alanında önemli gelişmelere tanıklık etmiştir. Bu nedenle 21. yüzyıl otoriteler tarafından bilişim çağı olarak da tanımlanmaktadır. Bilişim Teknolojileri (BT) önderliğindeki yenidünya düzenine ayak uydurabilmek için gelişen BT eğilimlerinin doğru anlaşılması ve yorumlanması oldukça önemlidir [1]. BT alanındaki baş döndürücü gelişmelere paralel olarak günümüzün yazılım teknolojileri de büyük bir dönüşüm ve gelişim yaşamaktadır. Yazılım endüstrisi günümüzün teknolojik ortamlarında, son 15-20 yıl içinde baskın bir rol oynamıştır ve dijital dönüşümünün devam etmesinde halen lokomotif bir güç olmaya devam etmektedir [1,2]. Yazılım endüstrisi bu yönüyle BT alanında yenilikçiliğin önünü açan temel unsurlardandır. Yazılıma dayalı teknolojiler, günümüzde modern ürün ve hizmetlerin çoğunda yer alan en işlevsel bileşenlerdir [1,3].

Günümüzün yazılım geliştirme ortamları, birçok teknik bilgi ve beceriyi bir arada gerekli kılan ve birbirine rakip farklı platformdan oluşur [1,4]. Özellikle web ve mobil tabanlı yazılım uygulamalarında yaşanan hızlı dönüşüm, yazılım geliştirme platformlarını ve bu platformlarda kullanılan yazılım geliştirme mimarilerini büyük ölçüde değiştirmiş ve çeşitlendirmiştir. İçinde bulunduğumuz şu dönemde yaşanan dördüncü sanayi devrimi, büyük ölçüde bilgi teknolojileri odaklı olup yazılım ve bilgi teknolojilerine dayalı endüstriler için oldukça yeni fırsatlar sunmaktadır. Özellikle son dönemde oldukça popüler olan bulut tabanlı ve büyük veri odaklı yazılım geliştirme ortamları endüstrideki yükselen yazılım geliştirme eğilimleri için güncel örneklerdir [5].

Günümüz ekonomisi ve endüstrileri için yazılım sistemlerine dayalı modern ürün ve hizmetler giderek daha etkin hale gelmektedir [1,6]. Bu bakımdan yazılım odaklı endüstrilerdeki gelişen talep ve gereksinimlerin karşılanabilmesi, teknik zorluklarının aşılabilmesi ve yazılım odaklı ürün ve hizmetlerin günümüzün ihtiyaçlarına cevap verebilecek kalite ve işlevsellikte olabilmesi için yazılım geliştirme uzmanlık alanlarının daha etkin ve daha dinamik bir yapıya sahip olması gerekmektedir. Yazılım geliştirme uzmanlık alanlarında bu dinamik yapının sağlanabilmesi için de güncel piyasa taleplerine

duyarlı yeni nesil yazılım geliştirme mimarileri, teknikleri, araç ve yöntem bilimlerine gereksinim duyulmaktadır [1,7].

Bu noktada, yazılım mühendisliğinin ya da yazılım geliştirme uzmanlığının sadece programlama olmadığını vurgulamak gerekir [8]. Kavramsal olarak, yazılım mühendisliği ve programlama, bina tasarımı ile binada kullanılan tuğla döşemesi olarak düşünülebilir. Yazılım mühendisliği binanın inşası ve programlama da bu inşada kullanılan tuğla döşemesi veya inşanın bir parçası olarak görülebilir. Bu bakış açısıyla yazılım mühendisliği, yazılımın geliştirilmesi, çalıştırılması, uygulanması ve bakımında, sistematik, disiplinli ve ölçülebilir bir yaklaşımın uygulanması ve bu yaklaşımların sürekli yenilenmesi, yani mühendisliğin yazılımına uygulanması olarak tanımlanır [1,8].

Bilgi teknolojileri alanındaki yenilikçi araştırma ve uygulamalar, yazılım üretiminde ve yazılım geliştirme süreçlerinde kendisini geliştiren akıllı sistemler, yöntem bilimler, mimariler, mekanizmalar ve programlama dillerini kapsar [1,4,8]. Yazılım geliştirme uzmanlık alanlarına yönelik araştırmalar, kabul edilebilir kalite güvence sistemine sahip (güvenlik, kalite, gizlilik, performans ve tutarlılık gibi) yazılım sistemlerinin etkin ve verimli bir şekilde oluşturulabilmesi ve sürekli gelişen dijital bilgi çağına uygun yazılım çözümlerinin üretilebilmesi için gerekli olan yöntemlerin ve mimarilerin belirlenmesini ve geliştirilmesini amaçlamaktadır. Bu sayede yazılım endüstrisinin diğer endüstrilere ve pazarlara yazılım tabanlı ürünler ve çözümler sunmada daha yetenekli, daha üretken ve rekabetçi olması hedeflenmektedir [1,6,8].

Yazılım geliştirme uzmanlık alanlarına yönelik çözümler, artan karmaşıklık, sürekli değişen endüstriyel talepler ve çok disiplinli olma özelliği nedeniyle temel disiplinlerden ve yazılım endüstrisindeki talep ve eğilimlerden ayrı tasarlanamaz [3,4,6,8]. Diğer mühendislik disiplinlerine benzer şekilde yazılım geliştirme uzmanlığına yönelik yenilikçi araştırma ve uygulamaların geliştirilmesi, teorik ve pratik anlamda endüstri ve akademilerin birlikte uyumlu çabalarını gerektirir [9]. Bu kapsamda yazılım endüstrisinde ihtiyaç duyulan güncel taleplerin, mesleki niteliklerin ve yükselen eğilimlerin kapsamlı şekilde ortaya konulması, belirlenen talep ve eğilimlere ciddi manada çözümler üretilmesi gerekir [4,6,9]. Özellikle günümüzün yeni nesil yazılım geliştirme ortamlarına yönelik mesleki bilgi ve becerilerin belirlenmesi, bu alanda eğitim gören yazılım geliştirme uzman adayı öğrencilerin daha efektif ve piyasa taleplerine uygun olarak yetiştirilebilmesi

açısından stratejik öneme sahiptir [9]. Bu noktada hem öğrencilerin hem de kariyerine bu alanda devam eden yazılım geliştirme uzmanlarının geleceği açısından yazılım teknolojilerindeki yükselen eğilimlerin belirlenmesi, özellikle nitelikli işgücünün yetişmesi anlamında ciddi katkılar sağlayabilir [9,10].

Günümüzde çoğu eğitim kurumunda ve üniversitelerde yazılım geliştirme uzmanlık alanlarında eğitim gören öğrenciler için gerekli olan güncel teknolojileri seçme süreci yazılım endüstrinin talep ve gereksinimleri ile yakından ilişkili değildir. Yazılım endüstrisinde ihtiyaç duyulan mesleki bilgi ve beceriler ile yazılım geliştirme uzmanlarının örgün eğitimden edindikleri bilgi ve beceriler arasındaki uyuşmazlık ve kopukluk ciddi işgücü kaybına neden olur ve bu durum ciddi ekonomik sonuçlar doğurabilir [8-11].

Yazılım endüstrisi, insan kaynaklarının etkin bir şekilde kullanıldığı dinamik, girişimci ve rekabete açık bir çalışma ortamıdır. Bu çalışma ortamında, mesleki beceriler, meslek unvanları, iş tanımları, iş deneyimleri ve gereksinimleri sürekli değişmektedir [9-11]. Yazılım geliştirme uzmanlarının sahip olduğu güncel mesleki nitelikler, gelişmekte olan teknoloji ve değişen iş gücü talep ve eğilimlerine cevap verebilecek düzeyde değildir [11,12]. Günümüzün dinamik işgücü piyasasında, yazılım geliştirme uzmanlarının farklı bilgi ve becerilere eşzamanlı olarak sahip olmaları beklenmektedir [10,11,13]. Bu nedenle yazılım uzmanlarının teknik bilgi ve becerilerini piyasadaki talep ve ihtiyaçlar doğrultusunda her zaman güncel tutmaları sıklıkla vurgulanan bir durumdur [13,14].

Bilişim teknolojilerinde yaşanan hızlı dönüşüm doğrultusunda yazılım geliştirme uzmanlık alanlarında her geçen gün farklı içeriklere sahip yeni iş fırsatları açılmaktadır. Yazılım endüstrisindeki çevrimiçi iş ilanlarının ve kariyer fırsatlarının güncel ve sistematik olarak sunulmasını sağlayan iş platformları (kariyer web siteleri), yazılım uzmanları ve işverenler tarafından yoğun bir kullanım oranına sahiptir. Bu bağlamda, kariyer odaklı web siteleri yazılım uzmanları ve işverenler arasında en aktif ve etkin iletişim araçlarından biridir. Bu yoğun kullanımdan dolayı, bu sitelerde paylaşılan ve depolanan bilgilerin miktarı ve çeşitliliği son zamanlarda büyük bir artış göstermiştir. Bahsedilen kariyer sitelerinde, her gün yazılım odaklı sayısız iş ilanı yayınlanmaktadır. Bu iş ilanları, yazılım endüstrisindeki işgücü piyasasında ortaya çıkan talep ve eğilimlerin belirlenmesinde önemli bir bilgi kaynağı olarak görülebilir.

Bu nedenle yazılım odaklı iş ilanlarının yöntem bilimsel analizi, bu endüstriye yönelik uzmanlık alanlarında ortaya çıkan talep ve eğilimlerin keşfedilmesinde önemli bir bilimsel öngörü ve bakış açısı sağlayabilir. Bu analizden elde edilen çıkarımların yazılım endüstrisindeki çalışanlar, işverenler ve eğitmenler için de ayrıca farklı açılardan önemli katkılar sağlayabileceği öngörülmektedir [9,13,14]. Yazılım geliştirme uzmanlık alanlarında mesleki talep ve eğilimlerin belirlenmesi, yazılım endüstrisiyle ilişkisi olan her kesim için stratejik bir öneme sahiptir. Şöyle ki, yazılım endüstrisinde işverenlerle yazılım uzmanları arasında dinamik bir bilgi akışı vardır. Teknolojik gelişmeler doğrultusunda yazılım endüstrisinde sürekli değişen talep ve eğilimler bu bilgi akışına yön vermektedir. Bu bilgi akış sürecinin detaylarını ve yazılım sektöründeki aktörlere sağladığı katkıları daha net ortaya koymak amacıyla bu süreç 6 temel aşamaya ayrılmış ve Şekil 1'de verilmiştir. Bu aşamalar aşağıda sırasıyla açıklanmıştır[1,9-14]:

1. Bu aşama yazılım endüstrisinin genelini yani endüstride yer alan tüm bileşenleri ifade etmektedir.

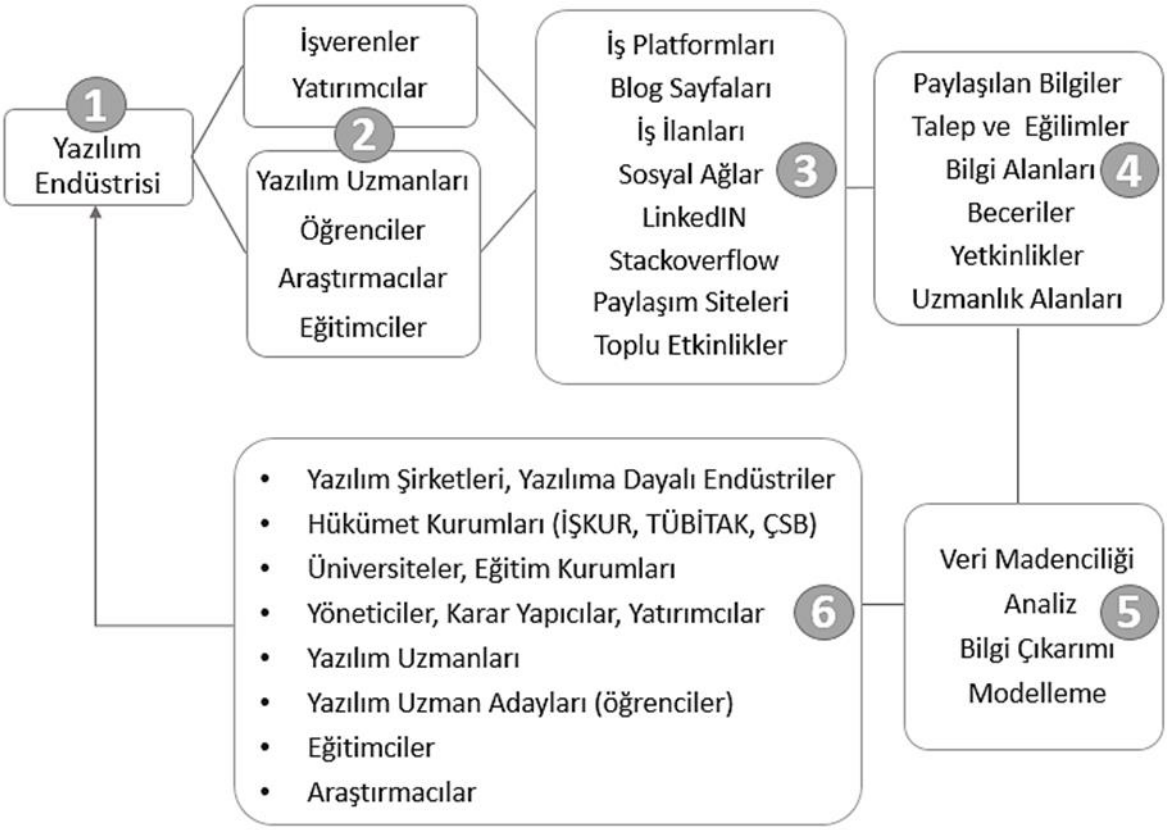
2. Burada bilgiyi paylaşan temel aktörler yer almaktadır. Yazılım endüstrisinde etkileşim ve paylaşımı sağlayan bu bireysel aktörler: İşverenler, yatırımcılar, yazılım uzmanları, öğrenciler, araştırmacılar, eğitimciler, vb. dir. Bu aktörler bilgi paylaşımı ve etkileşim için farklı platformlar kullanırlar [1].

3. Bu aşamada, bilgi paylaşımını saplayan platformlar ifade edilmektedir. Yazılım endüstrisindeki aktörler arasında etkileşimi sağlayan platformlar: İş platformları, kariyer siteleri, iş ilanları, sosyal ağlar, paylaşım siteleri, toplu etkinlik, seminer, konferanslar, vb. şeklinde verilebilir [1].

4. Bu aşamada ise platformlarda paylaşılan bilgilerin türü ve içeriği yer almaktadır. Yazılım endüstrisindeki aktörlerin üzerinde yoğunlaştığı ana başlıklar: eğilimler, talepler, uzmanlık alanları, bilgi, beceri ve yetkinlikler gibi temel konuları içermektedir [1].

5. Paylaşılan bilgilerin analiz edilmesi ve yorumlanması gibi işlemler bu aşamada gerçekleştirilir. Bu şekilde yazılım endüstrideki tüm aktör ve paydaşlara gelişen talep ve eğilimler ile ilgili güncel bilgi sağlanmış olur [1].

6. Bu aşamada, analizlerden elde edilen bulguları kullanan yazılım endüstrisindeki aktör ve paydaşlar ifade edilmektedir. Burada yer alan her aktör bu bulguları kullanarak kendisine farklı yönlerde katkı sağlayabilir [1].



Şekil 1. Yazılım endüstrisinde yer alan kurumlar ve kişiler arasında bilgi akışı

Bu amaç ve kapsam doğrultusunda son yıllarda birçok araştırmacı tarafından yazılım endüstrisinde ihtiyaç duyulan uzman iş gücüne yönelik güncel eğilimlerin, mesleki bilgi ve becerilerin ve uzmanlık alanlarının belirlenmesi amacıyla çevrimiçi iş ilanlarının analizine dayanan farklı bilimsel çalışmalar gerçekleştirilmiştir [1,14-17]. Günümüzün dinamik ve girişimci yazılım endüstrisi içerisinde sürekli gelişen ve değişen eğilimleri, arz ve talepleri daha doğru anlayabilmek için daha fazla araştırmaya ihtiyaç vardır. Özellikle sektöre özgü iş ilanlarının doğal dil işleme, semantik analiz ve istatistiksel modellemeye dayalı analizlerinden elde edilecek farklı çıkarımların bu alandaki bahsi geçen problemlerin çözümüne yönelik çalışmalara katkı sağlayacağı öngörülmektedir [1,17].

1.2. Çalışmanın Amacı ve Katkısı

Gerçekleştirilen bu tez çalışması kapsamında, olasılıksal konu modelleme yordamlarından Gizli Dirichlet Ataması (GDA), (Latent Dirichlet Allocation-LDA) yöntemi alana özgü çevrimiçi iş ilanları üzerinde uygulanarak, yazılım endüstrisinde ortaya çıkan eğilimlerin ve talep edilen teknik bilgi, beceri ve yetkinliklerin tespit edilmesi amaçlanmıştır. Ayrıca yazılım geliştirme uzmanlığının temelini oluşturan programlama dillerine yönelik talep ve eğilimlerin de bu amaç kapsamında analiz edilmesi hedeflenmiştir.

Yukarıda ifade edilen kapsam ve amaç doğrultusunda gerçekleştirilen bu tez çalışmasında uygulanan yöntembilim ve analizlerin sonucunda elde edilen bulgu ve çıkarımların alan literatürüne sağladığı katkılar aşağıda verildiği gibidir:

1. Yazılım endüstrisine yönelik çevrimiçi iş ilanları üzerinde olasılıksal konu modelleme yordamlarından GDA modelinin uygulanması ile bu alandaki kapsamlı çalışmalardan birisi gerçekleştirilmiştir [1]. Önerilen yöntembilim ile GDA modelinin yazılım geliştirme alanındaki güncel eğilimlerin tespit edilmesine yönelik olarak gerçekleştirilen anlamsal metin analizlerindeki etkinliği ortaya konulmuştur [1].

2. Gerçekleştirilen çalışma ile çevrimiçi iş ve istihdam platformlarının yazılım geliştirme alanlarındaki yükselen eğilimlerin tespit edilmesinde, güncel ve uygulanabilir bir veri kaynağı olarak başka araştırmalar için kullanılabilirliği analiz edilmiştir [1].

3. Olasılıksal konu modelleme yaklaşımına dayalı gerçekleştirilen bu anlamsal analiz ile yeni nesil yazılım geliştirme bilgi ve becerilerini en anlamı düzeyde ortaya koyan eğilimler tespit edilmiştir.

4. Konu modelleme analizi ile elde edilen eğilimler üzerinde bir taksonomi oluşturulmuş ve bu eğilimler; teknik alan bilgi ve becerileri, yazılım geliştirme ve programlama becerileri, analitik ve mantıksal beceriler ve teknik olmayan beceriler şeklinde dört ana başlık altında kategorize edilmiştir.

5. Yazılım endüstrisinde en çok talep edilen yazılım geliştirme uzmanlık rolleri ve bu rollerle ilişkilendirilen bilgi, beceri ve yetkinlikler tespit edilmiştir.

6. Programlama dillerine yönelik talepler Türkiye ve Dünya ölçeğinde ayrı ayrı analiz edilmiş ve yazılım endüstrisinde en fazla talep edilen programlama dilleri ve bu programlama dilleri arasındaki birliktelik ilişkileri ortaya konulmuştur [1].

Yazılım geliştirme uzmanlık alanlarındaki güncel eğilimlerin doğru anlaşılması ve yorumlanması açısından yapılan bu tez çalışmasının:

- Yazılım üreten işletmeler için gelecekteki ihtiyaç, yatırım ve yapılanma stratejilerinin belirlemesin de,
- Yazılım geliştirme uzmanları için mevcut piyasa talepleri doğrultusunda kariyer seçimi ve gelişiminde,
- Yazılım uzmanlığı alanında eğitim gören öğrencilerin için piyasanın ihtiyaçlarına cevap verebilecek nitelikte bilgi ve becerilerin edinilmesinde,
- Yazılım alanında eğitim veren akademiler için gelişen talep ve endüstriyel ihtiyaçlara cevap verebilecek kapsamda eğitim müfredatlarının oluşturulmasında, farklı açılardan katkılar sağlayabileceği öngörülmektedir [1].

1.3. Çalışmanın Gerekçesi ve Önemi

Bilişim teknolojilerine dayalı dördüncü sanayi devriminin yaşandığı bu dönemde, yenidünya düzenine ayak uydurabilmek için gelişen BT alanındaki gelişen eğilimlerin doğru anlaşılması ve yorumlanması günümüzün yazılım endüstrisi için oldukça önemlidir [1]. BT alanında yaşanan yenilikçi gelişmeler doğrultusunda yazılım geliştirme ortamlarında kullanılan teknolojilerde de büyük bir dönüşüm ve gelişim yaşanmaktadır. Artık yazılım uygulamaları modern ürün ve hizmetlerin çoğunda etkin bir şekilde yer almaktadır[1,3]. Yazılım sistemlerine dayalı modern ürün ve hizmetlerin giderek daha da yaygın hale gelmesi, yazılım endüstrisini günümüz ekonomisi için büyük bir potansiyel güce dönüştürmüştür. Yaşanan bu gelişmelerin yanında, yapılan çalışmanın gerekçesini ve önemini ortaya koyan diğer etkenler aşağıdaki gibi özetlenebilir:

- Yazılım endüstrisi, insan kaynaklarının etkin bir şekilde kullanıldığı dinamik, girişimci ve rekabete açık bir çalışma alanıdır ve yazılım uzmanları için sayısız kariyer fırsatları sunan yenilikçi bir endüstridir [1].

- Yazılım geliştirme ortamlarında, mesleki beceriler, meslek unvanları, iş tanımları, iş deneyimleri ve gereksinimler sürekli olarak değişmektedir. Bu nedenle yazılım geliştirme uzmanlık alanları çok geniş bir yelpazede mesleki bilgi ve beceriyi (hard and soft skills) gerekli kılmaktadır. Bugünkü durumda yazılım geliştirme alanında talep edilen 25'den fazla uzmanlık rolü (iş ünvanı) yer almaktadır. Bundan dolayı yazılım endüstrisi oldukça kapsamlı ve geniş uygulama alanlarına sahip bir çalışma alanıdır [1].

- Bilişim teknolojilerinde yaşanan hızlı dönüşüm doğrultusunda yazılım geliştirme uzmanlık alanlarında her geçen gün farklı içeriklere sahip yeni iş fırsatları açılmaktadır [1].

- Ülkemizdeki BT odaklı çalışma alanlarının daha çok yazılım geliştirmeye dayalı olması, ülkemizin genç nüfus potansiyeline sahip olması ve diğer endüstrilerin (donanım, nükleer, otomotiv, ağır sanayi) ülkemizde daha geri planda kalması, BT ve yazılım endüstrilerini ülkemizdeki genç nüfus açısından oldukça cazip hale getirmektedir [1].

- İçinde bulunduğumuz dönemde yaşanan dördüncü sanayi devrimi bilgi teknolojileri odaklı olup, yazılım ve bilgi teknolojilerine dayalı endüstriler için çok yeni fırsatlar sunmaktadır. Bilgi teknolojilerinin sunduğu bu fırsatlarla bir sabah aniden hayatınız tamamen değişebilir [1].

- Yazılım ürünleri, BT alanında yenilikçiliğin önünü açan uygulamalardır. Yazılım uygulamaları günümüzde modern ürün ve hizmetlerin çoğunda etkin bir şekilde yer almaktadır. Özellikle web ve mobil tabanlı yazılım uygulamalarında yaşanan hızlı dönüşüm, bunu takip eden büyük veri, sosyal ağlar, bulut teknolojileri, internet nesneleri, gömülü sistemler gibi yenilikçi teknolojiler yazılım endüstrisini günümüz ekonomileri için vazgeçilmez kılan yeni sistemlerdir[1].

- Yapılan piyasa araştırmaları, sektör raporları ve akademik çalışmalar yakın gelecekte yazılım endüstrisinde nitelikli işgücü ihtiyacının zirve yapacağını ortaya koymaktadır [1,16,17].

- Ülkemizdeki üniversiteler açısından bakıldığında, harcanan kaynak ve performansın büyük kısmı eğitim faaliyetlerine (eğitim>araştırma) yöneliktir. Ülkemizdeki üniversitelerde eğitime harcanan bu kaynak ve performans piyasa taleplerine duyarlı bir eğitim modeli içerisinde uygulanabilirse, BT endüstrilerinde talep edilen nitelikli işgücü ihtiyacına cevap verebilecek donanıma sahip uzmanların yetişmesi sağlanabilir.

- Ayrıca üniversiteler ve yazılım endüstrisi arasındaki boşluğun son zamanlarda akademik yayınlarda ve sektörel raporlarda daha sık vurgulanması, yazılım uzmanlığı eğitiminin dinamik, esnek, hızlı güncellenebilir, endüstriyel taleplere cevap verebilecek düzeyde olması gerektiğini daha net bir şekilde ortaya koymaktadır [1].

Yazılım geliştirme ortamlarındaki artan karmaşıklık, sürekli değişen endüstriyel talepler ve bu alanın diğer disiplinlerle yakınlığı nedeniyle, yazılım uzmanlığına yönelik çözümler temel disiplinlerden ve yazılım endüstrisindeki talep ve eğilimlerden ayrı geliştirilemez. Diğer mühendislik disiplinlerine benzer şekilde, yazılım geliştirme uzmanlığı için yenilikçi araştırma ve uygulamaların geliştirilmesi, teorik ve pratik anlamda endüstri ve üniversitelerin birlikte uyumlu çabalarını ve işbirliğini gerektirir. Bu bakımdan yazılım endüstrisinde talep edilen yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerinin analizi ve anlaşılması, bu bilgi ve donanıma sahip yazılım uzmanlarının yetiştirilmesi açısından oldukça önemlidir [1].

1.4. Tezin Kapsamı

Tezin birinci bölümünde yazılım geliştirme süreçleri, yazılım mimarisi, yazılım geliştirme modelleri, yeni nesil yazılım geliştirme eğilimleri, programlama dilleri ve veri madenciliği ile ilgili genel bilgilere yer verilmiştir. Ayrıca, olasılıksal konu modelleme analizlerinde kullanılan temel yaklaşımlardan gizli anlamsal analiz, olasılıksal gizli anlamsal analiz ve bu çalışmada kullanılan yöntem olan GDA hakkında tanım ve bilgilere

yer verilmiştir. Birinci bölümde son olarak, yapılan literatür incelemesi ile bu kapsamda yapılan çalışmalar ele alınmıştır.

İkinci bölümde, yapılan çalışmanın yöntembilimi ve analiz aşamaları verilmiştir. Bu kapsamda analizde kullanılan veri setinin oluşturulması, veri seti üzerinde uygulanan ön işleme adımları, boyut indirgeme ve kelime uzayının tespit edilmesi, doküman-terim matrisinin oluşturulması ve son olarak GDA tabanlı konu modelleme süreçlerinin veri setine uygulanması aşamaları anlatılmıştır.

Üçüncü bölümde, gerçekleştirilen deneysel analizlerin sonucunda elde edilen bulgulara yer verilmiştir. Konu modelleme analizinden elde edilen bulgular, teknik alan bilgi ve becerileri, yazılım geliştirme ve programlama becerileri, analitik ve mantıksal beceriler ve teknik olmayan beceriler şeklinde 4 başlık altında verilmiştir. Ayrıca, yazılım geliştirme uzmanlık rolleri ve sorumlulukları, programlama dillerine yönelik eğilimler ve programlama dilleri arasındaki birliktelik ilişkilerini ortaya koyan bulgular bu bölümde verilmiştir.

Tezin dördüncü bölümünde, bulgularla ilgili tartışma ve yorumlara yer verilmiştir. Beşinci bölümde, çalışmanın sonuçlarından ve son bölüm olan altıncı bölümde gelecekte yapılması planlanan çalışmalardan bahsedilmiştir.

1.5. Yazılım Geliştirme Uzmanlığı

Yazılım geliştirme, bir yazılım ürünü ile sonuçlanan süreçler ve uygulamaların oluşturulması ve sürdürülmesine yönelik fizibilite, tasarım, prototipleme, programlama, raporlama, test etme ve hata düzeltme gibi sıralı işlemler dizisini kapsayan sistematik bir yöntem bilimdir [8].

Yazılım geliştirme uzmanlığı, mühendislik yöntem bilimlerinde kullanılan sistematik yaklaşımların yazılım geliştirme süreçlerine uygulanmasını amaçlayan bir disiplindir. Temel disiplinlerdeki yöntem bilimsel yaklaşımların, yazılım geliştirme sürecindeki temel aşamalara uygulanmasıyla daha etkin, fonksiyonel ve sistematik bir yazılım geliştirme sürecinin elde edilmesi amaçlanmaktadır [1,2]. Bu nedenle yazılım geliştirme uzmanlığı,

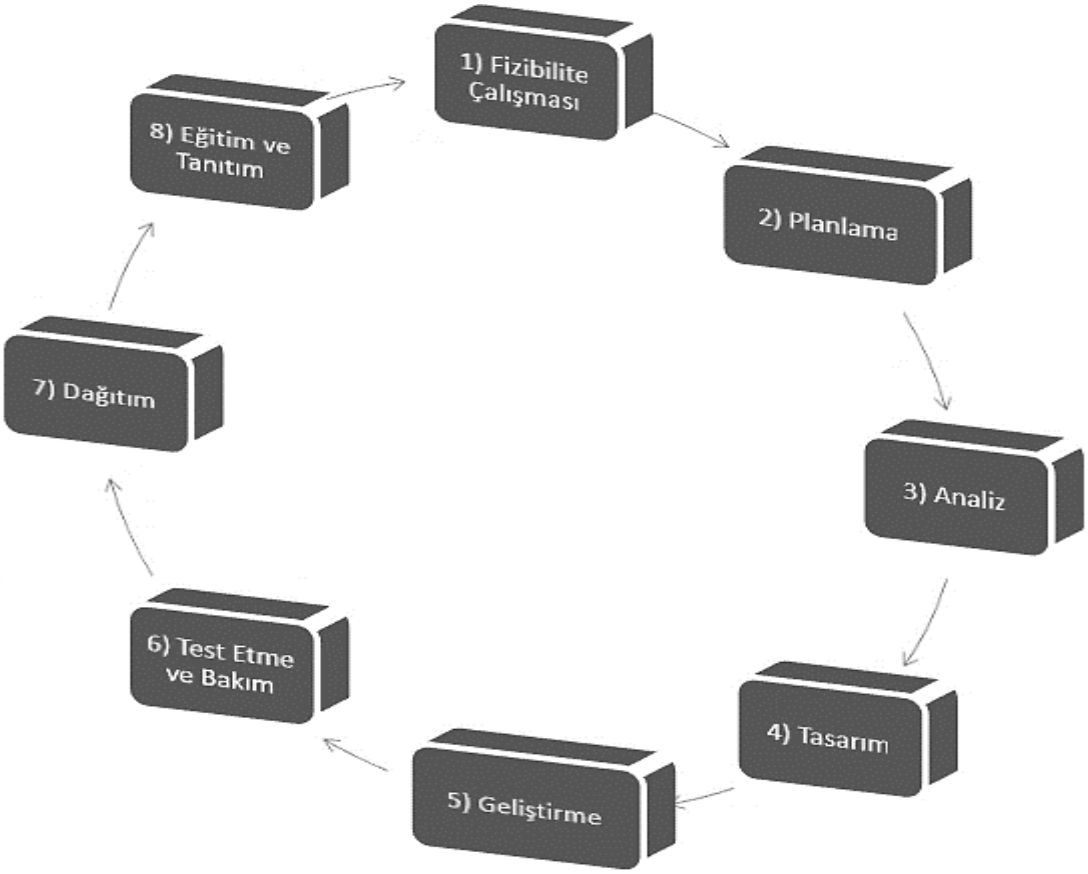
bilgisayar mühendisliği, yazılım mühendisliği, bilgisayar bilimleri, istatistik-bilgisayar gibi temel disiplinleri içeren geniş kapsamlı bir araştırma ve uygulama alanıdır [8].

Son yıllarda, yazılım odaklı sistemlerin günümüzün dijital yaşam döngüsünde ve günlük hayatın hemen her alanında etkin olarak kullanılması, yazılım geliştirme uzmanlık alanlarının daha kapsamlı ve karmaşık bir yapıya bürünmesine neden olmuştur. Bu durum, yazılım odaklı ürün ve hizmetlerin kalitesini artırabilmek için yazılım geliştirme süreçlerinin daha üretken yaklaşımlarla modernize edilmesi gereğini ortaya çıkarmıştır [6,9].

Günümüzün hızla artan yazılım odaklı ürün ve hizmet taleplerini karşılayabilecek işlevselliğe sahip yazılım süreçlerinin geliştirilebilmesi için öncelikle bu süreçlerin doğru anlaşılması gerekir [9-15]. Çünkü yazılım geliştirme süreci oldukça iyi planlanması gereken ve yazılımın başarımında etkin rol oynayan önemli aşamaları içeren karmaşık bir süreçtir.

1.5.1. Yazılım Geliştirme Süreci

Bir yazılım ürününün geliştirilme süreci boyunca yapılan tüm çalışma ve faaliyetlerin ayrı safhalar halinde planlanması ve yürütülmesi genel anlamda "yazılım geliştirme süreci" olarak tanımlanır. Yazılım geliştirme süreci, çoğu kaynakta "yazılım geliştirme yaşam döngüsü" olarak da tanımlanır. Başarılı bir yazılım geliştirme süreci, her aşamanın ayrıntılı bir şekilde planlamasıyla ve süreçlerin etkin bir şekilde yönetilmesiyle mümkündür. Yazılım geliştirme sürecinin her aşaması, farklı disiplinlerde birikime sahip farklı uzman bilgi ve becerilerini gerektirir [8,18]. Yazılım geliştirme sürecinin aşamalara bölünüp yürütülmesi, sürecin daha etkin kontrol ve yönetimini sağlama amacına yöneliktir. Geçmişten günümüze kadar yazılım geliştirme sürecine yönelik çok sayıda farklı model ve yaklaşım önerilmiştir. Değişiklikler göstermekle birlikte, etkin bir yazılım geliştirme sürecinin genel olarak içerdiği aşamalar Şekil 2'de gösterilmiş ve maddeler halinde açıklanmıştır:



Şekil 2. Yazılım geliştirme sürecinin akış şeması

1. Fizibilite Çalışması (Feasibility Study): Fizibilite çalışması yazılım projesine başlamaya karar vermeden önce yapılan bir ön araştırmadır. Bu fizibilite çalışması ile projenin ihtiyaçları ve gereksinimleri belirlenir ve bu ihtiyaçları karşılayabilecek kaynaklar araştırılır. Eğer var olan kaynaklar, ihtiyaç ve gereksinimleri karşılayamayacak düzeyde ise yazılım projesinden vazgeçilebilir [8,18].

2. Planlama (Planning): Yapılan fizibilite çalışmalarının sonucuna bağlı olarak gerçekleştirilecek olan yazılım projesinin yapılabilir bir proje olduğuna karar verilirse planlama aşamasına geçilir ve bundan sonraki aşamaların detaylı planlaması yapılır. Bu aşamada projenin hangi ihtiyaçlara cevap vereceği ve hangi probleme çözüm getireceği belirlenir ve projenin belirlenen sürede ne kadarının tamamlanabileceği belirlenir. Proje üzerinde genel bir planlama yapılarak projenin kaynak, zaman ve işgücü ana bileşenleri üzerinde bir ön değerlendirme yapılır [8,18].

3. Analiz (Analysis): Analiz aşaması, bir problemin çözümü olarak geliştirilecek olan yazılımın tamamlandığında hangi ihtiyaçlara cevap vereceğini, kapsamının neler olacağını ve bu kapsamda sistemin nasıl işleyeceğinin belirlediği, yani problemin ve çözümünün tanımladığı aşamadır. Gerçekleştirilen yazılım, isteneni doğru bir biçimde ve planlanan zamanda yerine getiriyorsa, ancak o zaman başarılı bir yazılımdır. Bu nedenle, problemin ve çözümünün doğru belirlenmesi açısından analiz aşaması oldukça önemlidir [7,8,18].

4. Tasarım (Design): Tasarım aşamasında, projenin gerçekleştirilmesinde hangi teknolojilerin kullanılacağı, yazılımın hangi platformda gerçekleştirileceği, hangi programlama dillerinin ve veri tabanının kullanılacağı ve hangi yardımcı araçlardan yararlanılacağı belirlenir. Bunlar belirlendikten sonra, önceki aşamaların da ışığında, yazılımın kullanıcı ara yüzü tasarlanır. Kullanıcı ara yüzünün hem isteklere cevap verebilmesi, hem de kolay, kullanışlı ve kullanıcı dostu olması gerekir. Tasarım aşamasında, yazılımın hem kullanıcı ara yüzü, hem de yazılımın genel işleyiş ve akış şeması (omurgası) belirlenir. Yazılımın işlevsel ve fonksiyonel olmasının yanında, performans, tutarlılık, güvenlik ve sürdürülebilirlik yönlerinin de dikkate alınarak tasarımın gerçekleştirilmesi, yazılımın kalitesi ve verimliliğine önemli katkılar sağlayacaktır [7,8,18].

5. Geliştirme (Development & Implementation): Tasarımdan aşamasından sonra yazılım projesinin hangi akış şemasına göre hangi teknolojilerle gerçekleştirileceği bellidir. Bu bilgilerin de öncülüğünde geliştirme aşamasında, yazılımın kodlama yani uygulama geliştirme kısmı gerçekleştirilir. Farklı uzmanlar tarafından kodlanan farklı alt bileşenler bu süreçte birleştirilir ve yazılım hem bileşenler düzeyinde hem de birleşik olarak test edilir. İstenilen performans elde edilinceye kadar eksik kısımlar yeniden kodlanır [7,8,18].

6. Test Etme ve Bakım (Testing & Maintenance): Yazılımın kodlama aşaması tamamlandıktan sonra yazılım kullanılabilir hale gelmiş demektir. Bu aşamada yazılımın farklı parametrelerle test edilmesi sağlanır. Yazılım geliştirilirken ilk testler yazılım geliştirme uzmanları tarafından yapılır. Ancak asıl hatalar test ve kontrol ekipleri tarafından belirlenir. Çünkü yazılım geliştiriciler, yazılımı sadece teknik yönlerden test eder ama bir kullanıcının bakış açısı çok daha farklıdır.

Bundan dolayı, gerçekleştirilen yazılım test ekibi ve son kullanıcı grupları tarafından farklı testlerden geçirilir. Test aşaması, yazılım kullanıcılar tarafından kullanıldığı sürece devam eder. Bu süreçte, yazılımda herhangi bir sorun tespit edilirse o sorun giderilerek yazılım güncellenir. Son kullanıcılardan sürekli geri bildirim alındığı için yazılımın aksayan bileşenlerinde, bu geri bildirimler ışığında gerekli güncellemeler gerçekleştirilir. Ya da sadece sorun kaynaklı değil de, ihtiyaçların ve teknolojilerin değişmesi sonucu yazılımda bir güncellemeye gidilebilir. Bu da yazılımın bakım aşaması olarak değerlendirilir [7,8,18].

7. Dağıtım (Deployment): Dağıtım aşaması geliştirilen, test edilen ve kullanıma hazır yazılımın son kullanıcılara dağıtım işlemlerini kapsayan aşamadır. Yazılım ürünü, bir paket ürünü mü olacak veya web üzerinden mi dağıtılacak, bu yöntemlere bu aşamada karar verilir ve yazılımın karakteristiğine göre uygun bir dağıtım şekli belirlenir ve ürün bu yolla kullanıcıya ulaştırılır [7,8,18].

8. Eğitim ve Tanıtım (Training & Promotion): Kullanıma hazır hale gelen yazılımın, son kullanıcılar tarafından daha etkin ve verimli kullanımını sağlayabilmek için kullanıcıların eğitilmesi ve yazılım hakkında bilgilendirilmesi sağlanır. Bu eğitim ve tanıtımlar web, e-mail veya basılı doküman gibi farklı yollarla yapılabilir. Bir yazılımdan verim alınabilmesi, o yazılımın doğru ve bilinçli bir şekilde kullanılması ile mümkündür. Bu nedenle eğitim ve tanıtım süreci, yazılıma ilişkin bilinçli bir son kullanıcı kitlesi oluşturulması açısından da önemli bir aşamadır [8,18].

Günümüzün yazılım geliştirme ortamları dikkate alındığında, yazılım geliştirme süreçlerinin birçok yeni yazılım mimarisi, programlama dilleri ve yazılım teknolojilerini birlikte kullanmayı gerektirdiği görülür. Bu karmaşık ve çoklu platformları içeren yeni nesil yazılım geliştirme mimarilerini kullanarak, etkin yazılım ürünleri ortaya koyabilmek için yukarıda bahsedilen yazılım süreçlerinin titizlikle uygulanması gerekir. Yazılım geliştirme süreçlerini oluşturan aşamalardan bir tanesinin bile eksik uygulanması, yazılım sürecinin ilerleyen aşamalarında telafi edilemeyecek problemlere yol açabilir [8,18].

Özellikle, yazılım sürecinde çözüm aranan problemin belirlendiği analiz aşaması oldukça kritik bir süreçtir. Çünkü hedef net olarak belirlenemezse, yazılımın sonucunda

nasıl bir ürün ortaya çıkacağı kestirilmez ve yazılım projesi başarısızlıkla sonuçlanır. Yukarıda bahsedilen yazılım geliştirme süreçlerinin tamamı, aynı zamanda bir yazılımın yaşam döngüsünü oluşturur ve eksiksiz uygulanması gerekir.[7,8,18].

1.5.2. Yazılım Mimarisi

Yazılım geliştirme süreci oldukça karmaşık ve iyi planlanması gereken bir süreçtir. Bu sürecin planlamasında en önemli yapı taşı yazılım mimarisinin inşa edildiği aşamadır. Yazılım geliştirme sürecinin başlangıcında problemin analizi yapılarak yazılım geliştirme süreci boyunca nelerin yapılması gerektiği ortaya konur. İhtiyaçların nasıl karşılanacağı ve hangi çözümlerin geliştirileceği ise yazılım geliştirme sürecinin tasarım aşamasında belirlenir. Yazılım geliştirme sürecinin tasarım aşamasında geliştirilecek çözümler için belirlenen yöntem ve işleyişler öncelikle üst seviyede planlanır ve diğer aşamalar boyunca detaylandırılır. Yazılım mimarisi, uygulama aşaması başladıktan sonra değiştirilmesi maliyetli olan temel yapısal tercihleri yapmakla ilgilidir. Bir yazılımın mimarisi bir binanın mimarisine oldukça benzeyen bir kavramdır [8,19].

Yazılım mimarisi, performans, güvenlik ve yönetilebilirlik gibi yazılımın kalitesini belirleyen temel nitelikleri optimize ederken tüm teknik ve yönetimsel gereksinimlerin de organize bir işleyişle yapılandırılmasını sağlayan bir çözüm modelinin tanımlanma aşmasıdır. Yazılım mimarisi, çok çeşitli bileşenlere dayanan bir dizi karar mekanizması içerir ve bu kararların her biri yazılımın kalitesi, performansı, sürdürülebilirliği ve genel başarısı üzerinde önemli etkiye sahiptir. Bir yazılım mimarisi bileşen grupları arasındaki bağlantıları, işleyişleri, etkileşimleri ve bileşenler arasındaki görev dağılımını net bir şekilde tanımlamalıdır [19].

Yazılım geliştirme sürecindeki tasarım aşaması ile yazılım mimarisi sıklıkla birbirine karıştırılmaktadır. Tasarım aşaması yazılımın tüm ayrıntılarını içerdiği halde mimari bize yazılımın dayandığı sistem ve arka plan hakkında bilgi verir. Başka bir deyişle mimari; tasarımın belli özelliklerini yansıtan bir alt aşamasıdır. Ancak bu tanımdan yazılım mimarisinin tasarımdan daha sonra hazırlandığı gibi bir durum anlaşılmaması gerekir. Bunun tam aksine yazılım mimarisi öncelikle hazırlanır ve tamamlandıktan sonra tasarım sürecine dâhil edilir. Başka bir deyişle, tasarım aşaması mimaride belirtilen bileşenlerin

detaylandırılmasını kapsar [19,20]. Sonuç olarak, yazılım mimarisi yazılım geliştirme sürecine yönelik olarak aşağıda belirtilen avantaj ve katkıları sağlar:

- Sistemin karmaşıklığını gidermek, bütünlüğünü korumak ve yönetimini kolaylaştırmak için, kontrol edilebilir bir yapı sunar.
- Yazılım mimarisi, sistem kurulmadan önce yazılım sistemlerinin davranışlarının analizi için gerekli temeli sağlar.
- Nesnelerin ve nesneler arasındaki davranışların, başka süreçlerde yeniden kullanımı kolaylaştıran esnek bir yapı sağlar.
- Bir sistemin geliştirilmesi, kurulumu ve bakım ömrünü etkileyen tasarım kararlarını desteklemektedir.
- Paydaşlarla olan iletişimi kolaylaştırarak ihtiyaçların daha iyi karşılanmasında sisteme katkı sağlar.
- Yazılım mimarisi, karmaşık BT projelerinde risk ve maliyetleri yönetmeyi kolaylaştıran önemli bir araçtır.

Yazılım mimarisinin hazırlanmasında rol oynayan en temel işlemlerden birisi de bir mimari stil (architectural style) ya da bir mimari kalıbın (architectural pattern) belirlenmesidir. Her yazılım projesinin amaç ve işlevine uygun yapıda bir yazılım mimarisi stili vardır. Mimari stiller ve kalıplar, bir uygulamayı şekillendiren farklı kurallar ve ilkeler kümesi olarak da düşünülebilir. Yani her mimari stil kendi karakteristiğine özgü kurallar ve ilkeler içerir. Bugüne kadar yazılım bilimciler tarafından çok sayıda mimari stil önerilmiş ve yazılım disiplinindeki yerini almıştır [19,20]. Farklı karakteristik özelliklere sahip bu mimari stillerin en yaygın olanları genel tanımlarıyla birlikte aşağıda verilmiştir:

İstemci/Sunucu Mimarisi (Client/Server Architecture): İstemci-sunucu modeli, görevleri ve iş yüklerini bir hizmet sağlayıcı kaynak (server) ve hizmet talep eden istemciler (clients) arasında bölen, dağıtılmış bir uygulama yapısıdır. Genellikle istemciler ve sunucular ayrı bir konumda bulunan bir bilgisayar ağı üzerinden iletişim kurarlar. Sunucu sistemler, kaynaklarını istemcilerle paylaşan bir veya daha fazla sunucu programını çalıştırır. Bir istemci ise kaynaklarının hiçbirini paylaşmaz ancak bir sunucudan içerik veya hizmet işlevleri sağlar. İstemci-sunucu mimarisine dayalı uygulamalara örnek olarak e-posta, ağ üzerinden yazdırma ve sosyal ağlar verilebilir [18,19-21].

Bileşen Tabanlı Mimari (Component-Based Architecture) : Bu mimari, yazılım sisteminin tasarlanması ve geliştirilmesine yönelik bir yazılım geliştirme yaklaşımıdır. Bu yaklaşımın temeli, yöntemler, olaylar ve özellikler gibi yazılım sistemini iyi tanımlayan iletişim arabirimlerinin birbirinden ayrı işlevsel veya mantıksal olarak iki temel bileşene ayrıştırılmasına dayanır. Bileşen tabanlı mimarinin birincil amacı bileşenlerin yeniden kullanılabilirliğini sağlamaktır. Bir bileşen, bir yazılım öğesinin işlevselliğini ve davranışlarını yeniden kullanabilir. Bu mimari model, nesne yönelimli mimari ilkelerinden daha yüksek bir modelleme seviyesi sağlar [18,19-22].

Etki Alanı Odaklı Tasarım (Domain Driven Design): Etki alanı odaklı tasarım, işletme etki alanlarındaki öğelere, öğelerin davranışlarına ve aralarındaki ilişkilere odaklı yazılım tasarlamak için kullanılan nesne yönelimli bir yaklaşımdır. Etki alanı modeli, daha sonra gerçekleştirilecek çözümlerin erken tahmini olarak görülebilir. Alan odaklı tasarım uygulamak için modellemek istenilen işletme alanının iyi anlaşılmış olması gerekir. Geliştirme ekibi, çoğu zaman etki alanı modellemesi için işletme etki alanına hâkim uzmanlarla birlikte çalışırlar. Alan odaklı tasarımda, tüm teknik ekip yalnızca işletme etki alanına odaklıdır ve farklı disiplinlerden olsalar da ekibin tamamı ortak bir teknik jargon kullanır [18,20,23].

Katmanlı Mimari (Layered Architecture): Katmanlı mimari, bir uygulama içindeki işlevsel yapının dikey olarak üst üste yığınlanmış ayrı katmanlar şeklinde gruplandırılması olarak tanımlanır. Her katmandaki işlevsellik ortak bir rol veya sorumluluk ile ilişkilidir. Katmanlar arasındaki iletişim açık ve esnek olarak birbirine bağlıdır. Uygulamanın uygun bir şekilde katmanlara ayrılması, sistemle ilgili doğabilecek aksaklıkların keskin bir şekilde birbirinden ayrılmasına yardımcı olur. Bu işlem sistemin esnekliğini ve bakım kolaylığını artırır. Katmanlı mimari stil, her katmanın kendinden alttaki katmanın sorumluluklarını ve soyutlamalarını topladığı ters çevrilmiş bir piramit olarak tasvir edilmeketdir. Keskin bir katmanlama süreci ile bir katmandaki bileşenler yalnızca kendi katmanındaki bileşenlerle veya sadece kendinden bir alttaki katmanda bulunan bileşenlerle doğrudan etkileşime girebilir. Daha esnek bir katmanlama durumunda ise bir katmandaki bileşenlerin aynı katmandaki veya herhangi bir alt katmandaki bileşenlerle etkileşime girmesine olanak sağlanır [18-20].

İleti Veri Yolu (Message bus): İleti veri yolu mimarisi, bir veya daha fazla iletişim kanalı kullanarak iletiler alıp gönderebilen bir yazılım sisteminin kullanım ilkelerini ortaya koyan bir mimaridir. Böylece uygulamalar birbirleri hakkında belirli detayları bilmeden birbirleriyle etkileşime girebilir. Bu model, uygulamalar arasındaki etkileşimin mesajların ortak bir veri yolu üzerinden iletilmesiyle (genellikle eşzamanlı olmadan) gerçekleştirildiği uygulamaları tasarlamak için kullanılan bir yaklaşımdır. Bu mimari yapıdaki bir uygulama, ortak şemaları kullanan ve ileti gönderip alma yoluyla iletişim kuran birbirinden bağımsız daha küçük uygulamaları içerir [19,20,24].

N Katmanlı Mimari (N-Tier Architecture) : Yazılım geliştirme sürecinde kullanılan N katmanlı mimari veya çok katmanlı mimari (multitier architecture), sunum, uygulama işleme ve veri yönetimi fonksiyonlarının fiziksel olarak ayrıldığı bir istemci-sunucu mimarisidir. N katmanlı mimari model, işlevselliğin katmanlı mimaride olduğu gibi katmanlara ayrılmasına dayanan bir modeldir. Farklı olarak bu modelde, sistemi oluşturan her bir bölüm fiziksel olarak farklı bir konumdaki bilgisayarda yer alan ayrı bir katman olarak ayrıştırılır. N katmanlı yazılım geliştirme mimarisi ileri düzeyde ölçeklenebilme, yönetim ve kaynak kullanımı imkânları sağlayarak uygulamaların ve servis bileşenlerinin fonksiyonel olarak ayrışmasını ve dağıtılmasını sağlar. Her katman, kendisinden bir üst ve bir alt seviyede bulunan katmanlar dışında diğer tüm katmanlardan tamamıyla bağımsızdır. Daha işlevsel bir ölçeklenebilirliği sağlayabilmek için katmanların birbirleri arasındaki iletişim çoğunlukla eşzamanlı yapılmaz. Bir uygulamanın katmanlara ayrılmasıyla geliştiriciler tüm uygulamayı yeniden düzenlemek yerine belirli bir katmanı değiştirme veya ekleme imkânına sahip olur. N katmanlı mimarinin en yaygın kullanımı üç katmanlı mimaridir. Üç katmanlı mimari genellikle bir sunum katmanı, bir etki alanı mantık katmanı ve bir veri depolama katmanı içerir. Bu katmanların her biri ayrı bir fiziksel sunucu üzerinde bulunur. Güvenliğin önemli olduğu bir finansal web uygulaması (internet bankacılığı), N katmanlı mimariye örnek olarak verilebilir [19,20,25].

Nesne Yönelimli Mimari (Object-Oriented Architecture): Nesne yönelimli mimari, bir uygulama veya sistem için sorumlulukların bölünmesine dayanan, her bir nesneye ilişkin verileri ve davranışları içeren, yeniden kullanılabilir ve kendi kendine yeterli nesneleri temel alan bir tasarım modelidir [19,26]. Nesne yönelimli bir tasarım, bir dizi kalıplaşmış talimat yerine kendi içerisinde işbirliği yapabilen bir dizi nesne olarak bir sistemi görüntüler. Nesneler ayrık, bağımsız ve esnek bağlanmıştır. Nesneler arabirimler

aracılığıyla yöntem çağırarak diğer nesnelerdeki özelliklere erişerek ve ileti gönderip alarak iletişim kurarlar [26]. Nesne yönelimli mimarinin yapısını oluşturan temel prensipler aşağıda verildiği gibidir:

- Soyutlama (Abstraction): Nesneyi daha önceden belirlenmiş özelliklere sahip olan ve bazı olayları gerçekleştirebilen bir veri biçimi olarak genelleştirmektir.
- Birleştirme (Composition): Nesneleri birbirleriyle birleştirme veya gruplandırma işlemidir.
- Sarmalama / Paketleme (Encapsulation): Tanımlanan sınıfın içerisinde yer alan yöntem ve özelliklerle ilgili bilgilerin bu sınıfı kullanması muhtemel kullanıcılardan gizlenmesi işlemidir.
- Miras Alma (Inheritance): Alt sınıfa ait nesnelerin türetildikleri üst sınıfa ait özellikleri almasıdır.
- Çok Biçimlilik (Polymorphism): Farklı nesnelerin aynı durum karşısında farklı davranış gösterebilme yeteneğidir.

Servis Odaklı Mimari (Service-Oriented Architecture - SOA): Servis odaklı mimari veya yaygın olarak bilinen adıyla hizmet yönelimli mimari, yazılım sistemlerinin fonksiyonel özelliklerini iş süreçleri etrafında gruplaştırarak sistemin tasarlanmasına ve geliştirmesine katkı sağlayan bir yazılım mimarisidir. Servis odaklı mimari, hazırlanmamış olmalarına rağmen iş süreçlerine dâhil edilen farklı yazılım uygulamalarının birbirleriyle iletişim sağlamak üzere veri alışverişinde bulunmalarını sağlar. Servis odaklı mimari, yazılım geliştiricilerin bir iletişim ağı üzerinden sundukları işlevlerin başka kullanıcılar tarafından kullanılması amacıyla ayrı birimlere ya da hizmetlere bölünmesini sağlar. Servis odaklı mimari, bilgi iletişiminde çeşitli protokoller ve veri formatları kullanarak iş süreçlerini birlikte çalışabilir hizmetler olarak sunabilir. Müşteriler ve diğer servisler, aynı katmanda çalışan yerel servislere veya bağlantılı bir ağ üzerinden farklı katmandaki uzak servislere erişebilir [27].

Sonuç olarak, yazılım geliştirme süreci oldukça karmaşık ve farklı uzmanlık alanlarını kapsayan, ileri düzey alan bilgisi gerektiren ve oldukça iyi planlanması gereken bir süreçtir. Bu bakımdan yazılım projesinin karakteristiklerine uygun bir mimarinin belirlenmesi, yazılımın başarımı açısından stratejik öneme sahip bir aşamadır [19,20,25]. Yazılım projesinin amaç ve işlevselliğine uygun mimari stillerin seçilmesi ve uygulanması,

projenin ileriye taşınmasında adeta bir kaldıraç görevi görür. Bu amaç ve kapsamda yazılım bilimciler tarafından birçok mimari model önerilmiştir. Bu modeller içerisinde yaygın olarak kullanılan mimarilerden bazıları yukarıda ifade edilmiştir. Yazılım mimarisindeki modellerin doğru anlaşılması ve uygun modelin yazılım projesine uygulanması, yazılım geliştirme sürecinin en önemli aşamalarından birisidir [19-27].

1.5.3. Yazılım Geliştirme Modelleri

Yazılım geliştirme sürecinde kullanılan modeller, bu sürece uygun bir bilgi sisteminin oluşturması, planlanması, yürütülmesi ve kontrol edilmesi için kullanılan sistematik bir yapı olarak değerlendirilir. Yazılım geliştirme modeli, bir yazılım odaklı sistemin tasarım ve geliştirme sürecini planlamak, yapılandırmak ve bu süreci kontrol etmek için kullanılan bir çerçevedir. Bu modeller, teknik açıdan herhangi bir deneyim veya kodlama bilgisi gerektirmeden sadece yazılım geliştirme sürecini modellemeye yöneliktir. Bu süreçte dikkate alınması gereken en önemli nokta, yüksek seviyede entegre yazılım sistemlerinin geliştirebilmesi için planlamanın doğru yapılması gerektiği noktasıdır. Bu modellerin temel amacı, projenin gereksinimlerine göre özelleştirilmiş, sistematik bir yazılım geliştirme sürecinin planlanmasına katkı sağlamaktır [18,25,28].

Her model belirli bir amaç için tasarlanmıştır ve her birinin göreceli avantajları ve dezavantajları vardır. Projenin karakteristiklerine uygun bir modelin seçimi, projenin niteliği, müşteri gereksinimleri, kullanılacak teknolojiler vb. gibi birçok faktöre bağlıdır. Buna ek olarak, yazılım projesine en uygun modelin seçimi, modellerin özeliklerini, avantajlarını ve dezavantajlarını da bilmeyi gerektirir [18,25,28]. Bu kapsamda, yaygın olarak kullanılan yazılım geliştirme modelleri ve bunların karakteristik özellikleri aşağıda verilmiştir:

Şelale (Waterfall) Modeli: Şelale modeli, yazılım geliştirme süreçlerinde yaygın olarak kullanılan geleneksel yazılım geliştirme modellerinden biridir. Bu yaşam döngüsü modeli çoğu zaman klasik yazılım geliştirme modeli olarak kabul edilir. Bu modelde, yazılım geliştirme süreci doğrusal sıralı bir iş akışına sahiptir. Bu doğrusal süreçte, geliştirme sürecindeki herhangi bir aşamanın başlayabilmesi için kendinden önceki

aşamanın tamamlanmış olması gerekir. Çünkü bu modelde herhangi bir işlem üzerinde değişiklik yapmak için önceki aşamaya geri dönüş yapılamaz [18,25,28].

Prototip (Prototype) Modeli: Prototip modeli, yazılım geliştiricilerin yazılımın işlevselliğini müşterilere göstermek ve gerçek uygulamayı geliştirmeden önce gerekli değişiklikleri yapmak için yalnızca çözümün prototipini oluşturmalarını sağlayan bir yazılım geliştirme modelidir. Bu modelin en bilinen özelliği, geleneksel bir şelale modelinde sıklıkla karşılaşılan birçok soruna çözüm getirmesidir [18,25,28].

Çevik (Agile) Yazılım Geliştirme Modeli: Çevik yazılım geliştirme modeli, müşteriye hızlı çözüm üretmek için uzun vadeli sabit planlamaya dayalı tasarım ve geliştirme süreçleri yerine, kısa aralıklarla yeni sürümler çıkarmaya dayalı yaklaşımlarla değişen koşullara hızlı adaptasyon sağlamayı amaçlayan bir modeldir. Yazılım geliştirme adımlarını raporlar yerine geliştirilen yazılımın yeni sürümleriyle takip etmesine olanak sağladığından kullanıcı tabanındaki gereksinimlere duyarlı dinamik bir modeldir. Bu yaklaşımda yazılım geliştirme projesinde sık sık değişiklik yapmaya izin veren sistematik yapı vardır. Bu modeldeki ilk öncelik kaliteli ve güvenli bir yazılımın kısa sürede müşteriye teslim edilebilmesidir. Projenin ilk aşamalarından itibaren sürekli yeni sürüm teslimleri yapılarak müşterinin yazılımı çok önceden kullanmaya başlamasına ve geliştirme sürecine katma değer sağlamasına olanak sağlanır. Değişiklikler projenin ileriki aşamalarında bile olsa sürece dâhil edilebilir. Amaç müşterinin ihtiyaçlarına cevap veren onlara fayda sağlayacak ve sisteme gerçek manada katma değer katacak yazılım üretmektir. Kısa süreli testler, dinamik tasarım, kapsamlı otomatik testler, eş zamanlı entegrasyon gibi sistematik adımlarla değişikliklerin getireceği maliyetler en aza indirilir ve yazılım geliştirme süreci değişikliklere hızlı bir şekilde adapte edilir [28,29].

Hızlı Uygulama Geliştirme (RAD) Modeli: Hızlı uygulama geliştirme (RAD)modeli, diğer yazılım geliştirme yöntemlerine kıyasla daha kısa sürede ve yüksek kalitede sonuçlar elde etmek için daha etkili bir yöntemdir. Bu modelin amacı tüm yazılım geliştirme sürecini hızlandırarak yazılım geliştirme sürecinde zaman ve kalite açısından avantaj sağlamaktır. RAD modelinde bileşenler veya işlevler sanki mini projeler gibi paralel olarak geliştirilir. Bileşenler belirli bir zaman çerçevesinde teslim edilir ve daha sonra bu bileşeler ve işlevler birleştirilerek amaçlanan sistemin çalışan bir prototipi hazır hale getirilir. Bu prototip ile müşteriye hızlı bir şekilde ulaşılıp geribildirim sağlanabilir. Bu modelde hedefe

kolayca erişilebilmek için geliştirme sürecinde aktif kullanıcı katılımına izin verilir. Bu model, artırımlı modelin farklı bir türüdür. Hızlı uygulama geliştirme modeli 4 temel aşamayı kapsamaktadır: gereksinimleri planlanması, geliştirme (tasarım, prototip, yapılandırma), test ve uygulamanın tamamlanması. Hızlı uygulama geliştirme modeli, 2-3 ay gibi kısa bir süre içinde modüler hale getirilebilen yazılım sistemlerinin oluşturulması için önerilen bir yaklaşımdır. Hızlı sonuçlar elde etmeyi amaçlayan bu modelde diğer geliştirme yaklaşımlarının da kullanılmasıyla hızlı ve etkin bir yazılım geliştirme süreci amaçlanır [30].

Artırımlı (Incremental) Model: Artırımlı modelde tüm gereksinimler çeşitli süreçlere ayrılmıştır. Burada çoklu bir şelale modeli döngüsünden bahsetmek mümkündür. Her bir bileşen analiz, tasarım, uygulama ve test süreçlerinden geçer. Sistem, her bir görevi tek başına üstlenen bağımsız artımlardan (increment) meydana gelmektedir. Müşteri gereksinimlerinin öncelikleri belirlendikten sonra yüksek önceliğe sahip gereksinimler ilk artımda gerçekleştirilir. Bu ilk artımda müşteriye en çok gerekli olan işlevleri içeren ilk ana sürüm elde edilir. Sonraki her bir artımda bu sürüme yeni bir işlev eklenir. Bu süreç, yazılım ürünü tüm işlevleriyle istenen aşamaya gelene kadar tekrar edilir. Her bir artım şelale modelinde yer alan eş değer aşamaları içerir. İlk artımlarda oluşturulan ürünün ilk sürümleri, prototip gibi düşünülerek müşterinin isteklerine hangi düzeye kadar cevap verdiği test edilir ve gerek duyulan değişiklikler erken aşamada tespit edilip uygulanabilir. Bu modelde gerek duyulan değişiklikleri güncellemek diğer geliştirme modellerine nazaran daha az maliyetli ve daha hızlı olur. Sisteme her yeni bir işlev eklenmesinde yani her bir artımda küçük bir güncelleme gerektiğinden sistemin test edilmesi ve hataların ayıklanması daha kolaydır ve bu şekilde ilk prototip üzerinde çok sayıda test yapma şansına sahip olunur. Bu modelin yazılım geliştirme sürecine uygulanabilmesi için yazılım fonksiyonlarının artımlara uygun hale getirilmesi gerekmektedir. Geliştirilen sisteme sürekli yeni fonksiyonlar eklendiği için genellikle uzun süreli projeler için önerilen bir modeldir [25,31].

V-Şekilli Model: Bu modelde uygulanan adımlar görüntü olarak V şeklini oluşturduğu için V-model olarak adlandırılır. V-modeli doğrulama (verification) ve onaylama (validation) modeli anlamına gelir. Tıpkı şelale modelinde olduğu gibi, V-model de yazılım geliştirme yaşam döngüsü süreçleri sıralı bir dizi içinde uygulanır. Bir sonraki aşamaya geçmeden önce bir önceki aşamanın mutlaka tamamlanmış olması gerekir. V-

modelinde o anda yürütülen geliştirme aşaması ile aynı aşamanın testi eş zamanlı yürütülmektedir [28,32]. Bu yaklaşımda aynı hizada olan her bir aşamanın doğrulama ve onaylama işlemleriyle birbirini tamamlaması esastır. Şöyle ki, her aşamanın tamamlanmasıyla o aşamanın test süreci gerçekleştirilir. Kodlama bittikten sonra birim testleri yapılır. Bundan sonra sistemin bütününe yönelik uyum testleri yapılır. Her bir bileşenin birbiri ile uyumluluğu test edilir. Analiz aşamasında ise ihtiyaçlar müşteriden istenir ve nelere ihtiyaç olunduğu belirlenir. Belirlenen ihtiyaçlara yönelik kabul testleri yapılır. "Evet, o ihtiyaçlar uygun şekilde karşılandı" cevabı alınana kadar ilgili test senaryoları hazırlanır ve uygulanır. V-model, aslında her aşamanın bir onaylanma kabulünün olduğu bir süreçtir. "Bir işlem yapıldı ama bunun doğruluğu onaylandı mı?" sorgulamasına yönelik olarak en altta kodlamaya kadar inip, daha sonra sistemi adım adım yukarıya doğru ilerleten bir işleyiş yapısı söz konusudur. Bir yazılım geliştirme sürecinde V- modelinin uygulanabilmesi için müşteriye olan güvenin yüksek seviyede olması gerekir. Hiçbir prototip üretilmediğinden müşteri beklentilerinin karşılanmasında yüksek bir risk söz konusudur. V-model, çoğunlukla şartların açıkça tanımlandığı ve sabitlendiği küçük ve orta ölçekli projeler için önerilmektedir [28,32].

Spiral Model: Spiral model diğer yazılım geliştirme modellerinden farklı olarak, yazılım geliştirme sürecini oluşturan aşamaların her bir döngüde (spiral) yeniden tekrarlanmasını ve her döngüde projenin daha da ilerlemesini sağlayacak işlemleri esas alır. Örneğin, projenin her döngüde yeni bir prototip çıkaracağı göz önüne alınırsa toplam 3 döngüde tamamlanan bir proje için 3 prototip elde edilir. Spiral modeli yaklaşımında planlama, analiz, geliştirme ve değerlendirme olmak üzere 4 temel aşama bulunur. Bir yazılım geliştirme projesinde her bir döngüde bu aşamalar tekrarlanır. Planlama aşamasından başlayarak temel spiral sırasıyla analiz, geliştirme ve değerlendirme aşamalarından geçerek ilk döngüyü tamamlar. Sonraki her spiral, temel spiral üzerine eklenerek devam eder [28,31].

Döngüsel (Iterative) Model: Döngüsel yazılım geliştirme modelinde gereksinimlerin tam olarak belirtilmesi gerekmez. Bunun yerine geliştirme aşamasında yazılımın yalnızca bir bölümü belirlenerek bu kısım kodlanır ve elde edilen prototip, daha sonraki gereksinimleri belirlemek için gözden geçirilir. Bu işlem daha sonra modelin her döngüsü için yazılımın yeni bir sürümünü üretmek üzere tekrarlanır. Dolayısıyla döngüsel modelde bütün ürün adım adım geliştirilebilmektedir. Her bir döngünün sonunda sisteme yeni

özellikler eklenir. Bu süreç sistem tüm gereksinimlere cevap verebilecek düzeye gelene kadar devem eder. Tasarlanan sistem karmaşık ve değişken bir yapıda ise bu durum döngü sayısını artırır. Döngüsel modelde döngüler arasındaki zaman aralığı artırımlı modele göre daha büyüktür. Buna şöyle bir örnek verilebilir. Bir araba piyasaya çıktıktan sonra birkaç ay sonra onun makyajlı modelinin sunulması artırımlı modele benzetilebilir. Birkaç yıl sonra aynı arabanın tamamen yeniden geliştirilmiş yeni modelinin piyasaya sunulması ise döngüsel modele dayalı geliştirme sürecine bir örnektir [2,8,31-33].

Scrum Model: Scrum, çevik yazılım geliştirme yaklaşımına dayanan yeni nesil modellerden birisidir. Bu geliştirme modelinin en temel özelliği gözlemciye, geliştirme uzmanına ve tekrara dayalı olmasıdır [2,8,29]. Kısa döngülü çıktı üretme ve geri bildirime dayalı güncelleme ilkesi benimsenmektedir. Projede yüksek önceliğe sahip gereksinimlere öncelik verilmesini esas alır. Proje zamanı boyunca ihtiyaç duyulan değişiklikler sonradan eklenebilir. Scrum modelin bir yenilikçi yönü de proje ilerlemesini net bir şekilde modelleyen kalan ihtiyaçlar/geçen gün sayısı ve kalan gün sayısını gösteren grafiği sunabiliyor olmasıdır. Bu grafik sayesinde yazılım geliştirme ekibi projenin durumunu ve ilerleyişini sürekli olarak kontrol edebilmektedir [2,8,29,31,32]. Bu modelde, yazılım projelerinin genellikle karmaşık ve kapsamlı bir süreç olduğu ve en baştan sistemin bir bütün olarak planlanmasının zor olacağı varsayımından hareket edilir. Tasarlanan yazılımın ürüne dönüştürülmesi, daha önceden hazırlanmış bir ihtiyaçlar listesinin sırasıyla gerçekleştirilmesi şeklinde yapılmaz. Bunun yerine müşterinin ihtiyaç duyduğu işlevler, iki ya da dört haftalık “sprint” adı verilen dönemler içerisinde tamamlanır ve test edilir. Bu kullanıcı odaklı gereksinimlerin tanımı “kullanıcı hikâyesi” olarak tanımlanır ve bu şekilde yapılacak işlemler listesine eklenir. Her sprint tamamlandığında yazılımın işlevsel bir bileşeni tamamlanarak müşteriye teslim edilebilecek hale getirilir. Özetle, scrum modeli çevik yazılım geliştirme yaklaşımlarını esas alan bir yöntemdir ve özellikle karmaşık projelerin yürütülmesi için önerilen bir modeldir [2,8,29,31-33].

1.5.4. Yeni Nesil Yazılım Geliştirme Eğilimleri

Günümüzün değişken yaşam koşullarında sayısız yazılım ürününün farklı amaçlarla günlük hayatın her aşamasında etkin bir şekilde kullanıldığı görülmektedir. Hatta şöyle ki, yazılım ürünlerinin kullanılmadığı bir alan nerdeyse yok gibidir. Yazılım teknolojilerine

dayalı yenilikçi ürünler, sürekli olarak hayatımızı değiştiren ve daha ileriye taşıyan önemli dinamiklerdir. Bu kapsamda yazılım endüstrisi, sürekli olarak değişen ve kendi ilerlemesine katkıda bulunan gelişime açık endüstrilerden birisidir [1,9,10,34,35].

Son dönemlerde yaşanan teknolojik gelişmeler doğrultusunda hızla değişen ve yenilenen yazılım geliştirme eğilimleri, bu alandaki teknik bilgi ve becerilerin de büyük bir hızla değişmesine yol açmıştır. Çünkü günümüzün modern yazılım geliştirme ortamlarında birçok teknik bilgi ve beceriyi bir arada gerekli kılan farklı yapıda platformlar yer almaktadır. İçinde bulunduğumuz şu dönemde yaşanan dördüncü sanayi devrimi, büyük ölçüde bilgi teknolojileri odaklı olup yazılım ve bilgi teknolojilerine dayalı endüstriler için çok yeni fırsatlar sunmaktadır. Bunların içinde web tabanlı sistemler, mobil uygulamalar, büyük veri, bulut tabanlı teknolojiler, internet nesneleri ve akıllı sistemler günümüzün yazılım geliştirme eğilimlerine yön veren önemli teknolojilerdendir. Daha geniş bir bakış açısıyla, son dönemde yazılım endüstrisine yön veren ve yazılım ürünlerine talep ve kaynak oluşturan yeni nesil eğilimler aşağıdaki gibi listelenebilir [1,4,9,34,35]:

- Mobil uygulamalar (Mobile applications)
- Büyük veri (Big data)
- Betik Programlama (Scripting programming)
- İş zekâsı (Business intelligence)
- Elektronik ticaret (E-commerce)
- Sohbet robotları (Chatbots)
- Sosyal ağ uygulamaları (Social network applications)
- Bulut tabanlı sistemler (Cloud based systems)
- Açık kaynak kodlu yazılım geliştirme (Open source development)
- Akıllı sistemler (Smart systems)
- Yapay zekâ (Artificial intelligence)
- Çevik yazılım geliştirme (Agile software development)
- Makine öğrenmesi yöntemleri (Machine learning methods)
- Sanal ve artırılmış gerçeklik (Virtual and augmented reality)
- Yazılım güvenliği (Software security)
- Veri güvenliliği ve gizliliği (Data security and privacy)
- Internet nesneleri (Internet of things)

- Ölçeklenebilir sistemler (Scaling systems)
- Gömülü sistemler (Embedded systems)
- Dağıtılmış sistemler (Distributed systems)
- Hizmet olarak kullanılan yazılım (Software as a service)
- Hizmet olarak kullanılan platform (Platform as a service)
- Hizmet odaklı mimariler (Service oriented architectures)
- Otomasyon (Automation)

Günümüzün bu popüler bilişim teknolojileri, çok kısa sürede ortaya çıkan ve geleneksel yazılım geliştirme süreçlerini tamamen değiştiren teknolojilerdir. Yeni nesil yazılım geliştirme süreçlerinde bulut tabanlı teknolojiler, büyük veri odaklı sistemler ve mobil uygulamalar en fazla talep ve ilgiyi gören eğilimlerin başında gelmektedir. Bu ve benzeri yenilikçi bilişim teknolojilerinin öncülüğünde yazılım geliştirme platformları ve bu platformlarda kullanılan süreçler, mimariler ve yöntembilimler oldukça hızlı bir evrim geçirmektedir [1,3,4,9,10,35].

Yazılım geliştirme teknolojilerinde yaşanan bu hızlı dönüşümün amacı, yazılım ürünlerinin fonksiyonelliğini ve verimliliğini artırarak gelişen teknolojilere ve taleplere cevap verebilecek koşulları ve işlevselliği sağlayabilmektir. Sürekli gelişen ve artan yazılım taleplerine cevap verebilmek için bu kapsamda ilk olarak yapılması gereken şey; bu niteliklere sahip yazılım uzmanlarının yetişmesini sağlamaktır. Bu sürecin başarıya ulaşabilmesi için öncelikle yazılım endüstrisindeki eğilim ve taleplerin güncel olarak analiz edilmesi ve doğru anlaşılması gerekir. Çünkü günümüzün dinamik yazılım geliştirme ortamlarında sadece kendini yenileyebilen, yeteneklerini geliştirilebilen, teknolojik dönüşüme ve yeni nesil eğilimlere ayak uydurabilen sistemler hayatta kalabileceklerdir [1,3,9,34,35].

1.5.5. Mobil Uygulama Geliştirme

Günümüzde yaşamın her alanında her yaştan büyük bir kesimin yaygın olarak kullandığı taşınabilir (mobil) cihazlar (tablet, akıllı telefon, vb.) için özel olarak tasarlanan ve kodlanan yazılımlara “mobil uygulama” denir. Mobil uygulamalar kullanılacak olan cihazın türüne ve uygulamanın kullanım amacına göre tasarlanır. Günümüzde mobil

uygulamalar dünya genelinde en yaygın kullanım oranına sahip mobil işletim sistemleri olan IOS ve Android'e uygun olarak tasarlanır ve bu çerçevede geliştirilir. Şöyle ki, geliştirilen bir mobil uygulama Iphone, Ipad gibi mobil cihazlarda çalıştırılacaksa o uygulamanın IOS uyumlu olarak geliştirilmesi gerekir. Eğer geliştirilen uygulamanın Android işletim sistemi kullanan bir mobil cihazda çalışması isteniyorsa o uygulamanın Android tabanlı olarak geliştirilmesi gerekmektedir. Ancak bu noktada yapılması gereken en doğru seçim bir mobil uygulamanın tüm platformlara ve her iki işletim sistemine uyumlu olacak şekilde geliştirilmesidir.

Mobil uygulamalar, taşınabilir cihazların en etkin şekilde kullanabilmesine olanak tanıyan küçük yazılımlardır. BT endüstrisinde farklı amaçlarla kullanılan hemen her türlü yazılımın mobil ortamlara uyarlanması mümkündür. Bu şekilde taşınması zor olan bilgisayarlar yerine taşınabilir cihazlar olan akıllı telefonlar ya da tablet bilgisayarlarla rutin yapılması gereken faaliyetler daha etkin bir şekilde yürütülebilir [1,4,9,34,35].

Mobil uygulamalar kullanım amaçlarına göre büyük bir çeşitliliğe sahiptir. Mobil uygulamalar her kullanıcıya farklı programlar aracılığıyla farklı yönlerden katkı sağlar. Bu uygulamalar sayesinde mobil kullanıcılar arama yapar, oyun oynar, görüntülü görüşme yapar, içerik paylaşır, konum bildirir, video izler, internete girer ve bunun gibi sayısız işlemi mobil uygulamalar aracılığıyla gerçekleştirir. Özellikle e-ticaret ve çevrimiçi alışveriş siteleri için geliştirilen mobil uygulamalar, artık işlevselliğinin yanı sıra aynı zamanda şirketler için bir saygınlık unsurudur. Mobil kullanıcılar, geliştirilen bu mobil uygulamalar sayesinde ilgili web sitesinin adresini kullanmaya gerek duymadan, telefon üzerindeki bir ikona tıklayarak doğrudan web sitesine veya ilgili mobil uygulamaya erişim sağlarlar. Bu şekilde kullanıcılar tarayıcı üzerinden bir web sitesini açmak yerine mobil uygulama ile daha hızlı ve etkin işlem yapma ayrıcalığına sahip olurlar [1,34,35].

Günümüzde teknoloji üreten şirketler mobil uygulamalar için çok büyük ar-ge bütçeleri ayırmaktadırlar. Şöyle ki, Whatsapp'ın 19 milyar dolara Facebook'a satılması, yine Facebook'un 1 milyar dolara İnstagram'ı satın alması, bunun yanında Microsoft'un 8.5 milyar dolara Skype yazılımını satın alması mobil uygulamalara yönelik olarak gerçekleştirilen önemli yatırımlardandır. Telaffuz edilen bu meblağlar mobil uygulama endüstrisinin günümüzde nerelere geldiğini ve gelecekte neler vaat ettiğini çok açık şekilde ortaya koymaktadır.

Mobil cihazların günlük yaşamdaki yeri ve önemi arttıkça mobil uygulamaların sayısı ve çeşitliliği de bu doğrultuda her geçen gün büyük bir hızla artmaktadır. Bu bakımdan oldukça dinamik ve gelişime açık bir çalışma disiplini olan mobil uygulamaların yakın gelecekte teknolojik ortamlardaki hâkimiyetini daha da artıracağı ve yeni fırsatlar sunacağı öngörülmektedir. Her geçen gün hızla artan akıllı telefon sayısı mobil uygulamalara duyulan ihtiyaçları da aynı oranda arttırmaktadır. 2013 yılında yayınlanan sektöre özgü bir analiz raporunda, Avrupa Birliği üyesi 28 ülkeyi kapsayan 529.000 BT odaklı iş ilanı içerisinde, %60'lık bir oranın mobil uygulama geliştirmeye yönelik iş ilanlarından oluştuğuna vurgu yapılmaktadır. Bu bakımdan mobil uygulamalardaki artan talep ve eğilimler doğrultusunda, bu alana yönelik yatırım ve ar-ge çalışmalarının yakın gelecekte daha da hız kazanacağı öngörülmektedir [1,4,9,17,34,35].

1.5.6. Büyük Veri Odaklı Yazılım Geliştirme

Yazılım geliştirme uzmanlığı, veri odaklı bir disiplin ve veri biliminin ayrılmaz bir parçasıdır. Büyük veri (big data), yüzbinlerce internete bağlı cihazın (bilgisayar, akıllı telefon, tablet, vb.) oluşturduğu milyonlarca verinin yeni nesil yöntemlerle depolanması, işlenmesi, analiz edilmesi ve yorumlanması gibi işlemleri içeren geniş kapsamlı bir süreçtir. Bu zorlu süreçleri kapsayan yeni nesil yazılım geliştirme modellerinin tasarlanması ve büyük veri yığınlarına uygulanması yazılım geliştirme disiplini açısından yüksek önceliğe sahip araştırma konuları arasındadır. Sosyal ağlar, nesnelerin interneti, mobil uygulamalar ve bilgi paylaşım platformları gibi güncel teknolojilerle birlikte büyük veri sistemlerinin ortaya çıkması, bu büyük veri setlerini işleyebilen yazılım uygulamalarının geliştirilmesi açısından, yazılım geliştirme uzmanlık alanlarına yeni bir çalışma sahası daha eklemiştir [1,4,17,35].

Endüstriyel stratejilerdeki küresel ölçekli gelişmeler ve yaşanan hızlı teknolojik dönüşüm, veri odaklı karar verme süreçlerine yeni bir bakış açısı kazandırmıştır. Şöyle ki, veri analizine dayalı karar destek sistemleri günümüz işletmeleri için stratejik bir öneme sahiptir. Bu nedenle farklı endüstriyel alanlarda büyük veri odaklı talep ve eğilimlerin büyük bir hızla artmasıyla, büyük veri odaklı yazılım geliştirme uzmanlığı bugünün teknoloji odaklı işgücü piyasasında yoğun ilgi çeken bir iş kolu haline gelmiştir. Böylelikle büyük veri odaklı uzmanlık alanlarında nitelikli işgücüne yönelik talep ve eğilimler her

geçen gün hızla artmaktadır. Araştırma şirketi Gartner yaptığı araştırmada, büyük veri odaklı sistemlerin 2015'e kadar 3.7 trilyon dolarlık ürün ve hizmet sağlayacağını ve buna bağlı olarak da 4.4 milyon kişilik yeni istihdam yaratacağına vurgu yapmaktadır [36]. Ayrıca 2018 yılına kadar sadece Amerika Birleşik Devletleri'nde veri odaklı analitik becerilere sahip BT uzmanlarına yönelik olarak 140.000 ile 190.000 arasında değişen bir işgücü ihtiyacı yaşanacağı öngörülmektedir [1,4,9,10,17,37].

Büyük veri odaklı yazılım geliştirme ortamlarında, yazılım uzmanları için en önemli süreç yazılım sistemlerinin tasarımı ile ilgilidir. Büyük veriler için tasarlanan yazılım uygulamaları ilk olarak daha küçük ölçekli veri sistemleri için geliştirilir ve bu ölçekte test edildikten sonra veri ölçeği artırılarak büyük veri odaklı yazılım sistemlerine uygulanır. Ölçeklenebilir yazılım geliştirme mimarisi olarak da adlandırılan bu yeni nesil yazılım geliştirme yaklaşımı, özellikle büyük veri odaklı yazılım geliştirme sürecinin temelini oluşturmaktadır. Bu bakımdan büyük veri odaklı uygulamaların geliştirilebilmesi için ölçeklenebilir yazılım mimarilerinin tasarlanması ve uygulanması, yazılım geliştirme uzmanlığı açısından farklı bilgi, beceri ve yetkinlikleri gerekli kılmaktadır [1,4,9,17,35,37].

1.5.7. Bulut Tabanlı Sistemler

Bulut teknolojisi, tüm uygulama, program ve verilerin hiçbir kuruluma ihtiyaç duyulmadan, internete bağlı uzak sistemler üzerinde depolanmasını, çalıştırılmasını ve yönetilmesini sağlayan araç ve hizmetlerin bütününü kapsayan genel bir kavramdır. Özellikle son dönemlerde yazılım geliştirme uzmanları için yeni kariyer fırsatları sunan bulut tabanlı uygulamalar her geçen gün daha fazla adından söz ettiren yenilikçi bir teknoloji olarak karşımıza çıkmaktadır [1,9,35,37,38].

Yapılan araştırmalar bulut tabanlı uygulama geliştirme becerilerinin yazılım uzmanlarının sahip olması gereken temel nitelikler arasında olduğuna dikkat çekmektedir. Yakın gelecekte bilgisayarlarda kullanılan hard disklerin yerini çevrim içi bulut sistemlerinin alacağı konusunda yaygın bir ön görü hâkimdir. Bundan dolayı bulut tabanlı uygulama geliştirme becerilerinin ilerleyen zamanlarda yazılım uzmanları için olmazsa olmaz nitelikler arasında olacağı öngörülmektedir [37,39].

1.5.8. Betik Programlama

Betik programlama (scripting programming) günümüzün en önemli yazılım geliştirme eğilimlerinden birisidir. Özellikle web ve mobil tabanlı uygulamaların yaygınlaşmasıyla derleyiciden bağımsız dinamik ve esnek bir uygulama geliştirme yaklaşımı sunan betik programlama modeli her geçen gün bu alandaki baskınlığını daha da arttırmaktadır. Betik programlama dilleriyle yazılan programlar, özel çalışma-zamanlı (run-time environment) sistemlerin yorumlayabileceği programlardır. Klasik programlama dilleri ile yazılan kodlar makine koduna çevrildikten sonra (derleme) ortaya çıkan program doğrudan çalıştırılırken betik programlama ile geliştirilen programlar betik yorumlayıcı tarafından doğrudan okunur, yorumlanır ve bu yorumlama işleminin sonunda yapılması gereken işlemler yorumlayıcının kendisi tarafından gerçekleştirilir [1,35,40,41].

Betik programlar herhangi bir metin editörü (not defteri vb.) ile düzenleyebileceğiniz metin dosyalarıdır. Bu metin tabanlı dosyalarda belli bir betik programlama dilinde yazılmış komutlar bulunur. Kullanım amacına göre çok sayıda ve farklı türlerde birçok betik programlama dili mevcuttur. Java gibi derleyici tabanlı bir programlama dili ile yazılan bir program ile Python gibi betik tabanlı bir programlama ile yazılmış bir program arasında ne tür farklılıklar vardır? Betik programlama dilleri ile yazılmış programlar bilgisayarda metin dosyaları olarak saklanır ve program her çalıştırıldığında bilgisayar tarafından yeniden yorumlanır. Derleyici tabanlı programlama dillerinde yazılmış bir program ise bir kez derlenir ve bu şekilde mikroişlemcinin anlayabileceği makine diline dönüştürüldükten sonra çok daha hızlı çalışırlar. Betik programlama dilleri derleme işlemine ihtiyaç duymazlar ve bu durum yazılım geliştiriciye zaman kazandıran önemli bir avantajdır. Ayrıca, betik programlama dillerine dayalı yazılım geliştirme becerileri öğrenilmesi ve uygulanması daha kolay olan temel becerilerdir [40,41].

Betik programlama dilleri genellikle yüksek seviyeli diller olarak da bilinir. Bu diller diğer dillere göre daha kolay ve daha az kaynak kullanan uygulamalar için tasarlanmıştır. Betik programlama çoklu platform desteği gerektiren uygulamalarda veya yoğun işlem gerektirmeyen hızlı ve işlevsel çözümlerde etkin olarak kullanılan bir yazılım geliştirme modelidir. Perl, Javascript, Python, Ruby gibi programlama dilleri yaygın olarak kullanılan betik programlama dilleri arasında yer almaktadır [1,35,40,41].

1.5.9. Programlama Dilleri ve Özellikleri

Programlama dili, yazılımı geliştiren kişinin yazılımla ilgili yapılması gereken tüm adımları bilgisayarın anlayacağı şekilde ifade etmesini sağlayan komut setleridir. Programlama dilleri, yazılımcının bilgisayara hangi aşamada hangi işlemi nasıl yapacağını sistematik olarak anlatmasını sağlayan araçlardır. Geçmişten günümüze gelene dek 2500'ü aşkın programlama dili kullanıma sunulmuştur. Bunların çoğu teknolojik gelişim karşısında yerini yeni nesil programlama dillerine bırakmıştır [42,43]. Programlama dilleri algı düzeyine bağlı olarak üç gruba ayrılır:

- Düşük seviyeli programlama dilleri: Makine kodlamasına ve makine dillerine oldukça yakın programlama dilleridir. Bu programlama dillerini kullanan yazılım geliştiriciler daha çok donanıma yönelik çalışan kişilerdir. Bu durum, mikro işlemciler hakkında yetkin düzeyde bilgi sahibi olmayı gerektirir [43].

- Orta seviyeli programlama dilleri: Oldukça esnek olan orta seviye programlama dilleri hem üst hem alt seviye (yazılım-donanım) programlamaya olanak sağlarlar. Düşük seviyeli dillere nazaran daha gelişmiş ve anlaşılması daha kolay dillerdir (C programlama dili gibi) [43].

- Yüksek seviyeli programlama dilleri: Olay (event) tabanlı programlama dilleri olarak da tanımlanırlar. Bilgisayar donanımına uzak olup daha çok uygulama geliştirmeye yönelik programlama dilleridir. Bu programlama dilleri yoğun olarak belirli fonksiyonlar ve alt yordamlar çerçevesinde işlem yürütürler ve bu özelliklerinden dolayı programlamanın kapsamını azaltırlar. En hızlı ve en işlevsel programlama dilleri bu kategoride yer almaktadır (java, pyton, c++ gibi). Bu diller diğer programlama dillerine kıyasla daha anlaşılır ve daha kolay uygulanabilir olduğundan yazılım geliştirmeye yeni başlayanlar için en uygun programlama dilleridir [43].

Yüksek seviyeli programlama dilleri ile geliştirilen yazılımların çalışabilmesi için makine diline dönüştürülmesi yani derlenmesi gerekir. Bunun için program hangi yüksek seviyeli dil ile kodlandıysa o dile uygun bir derleyici kullanılır. Böylece yüksek seviyeli programlama dilinde kodlanmış olan yazılım, makine dilinde kodlanmış uygulama

programına dönüştürülür. Geliştirilen yazılımın kaynak kodları üzerinde değişiklik yapmak mümkündür fakat derlenmiş olan uygulama programının üzerinde hiçbir değişiklik yapılamaz [42,43]. Günümüz modern yazılım geliştirme ortamlarında yaygın olarak tercih edilen programlama dilleri aşağıda verildiği gibidir:

Java: 1990'lardan beri var olan Java programlama dili açık kaynak kodlu, nesne yönelimli, sınıf temelli, platformdan bağımsız, çok fonksiyonlu ve adım adım yorumlanabilen yüksek seviyeli bir dildir. Java ilk zamanlar küçük cihazlar için tasarlanmış çoklu platform dili olarak düşünülmüştü. Çoklu platform ve tek tip kütüphane desteği ile daha üstün ve daha güvenli bir yazılım geliştirme olanağı sunan Java, zamanla tüm platformlarda kullanılmaya başlamıştır. Java, şu anda tüm platformlarda kullanılabilen esnek, dinamik ve güçlü bir dildir. Özellikle günümüzde kurumsal yazılımlarda, web ve mobil uygulamalarda son derece popüler olan Java, yeni sürümlerinin de kullanılmaya başlamasıyla masaüstü uygulamalar için de vazgeçilmez bir yazılım geliştirme aracı olmuştur [42-44].

Python: 1990'lı yıllarda ilk kez ortaya çıkan ve genel amaçlı bir programlama dili olan Python, son derece okunaklı ve anlaşılır bir dil olup yazılım geliştirmeye yeni başlayanlar için önemli kolaylıklar içermektedir. Python yenilikçi özellikleriyle birlikte, yorumlanabilir, modüler, nesne yönelimli ve yüksek seviyeli bir betik programlama dilidir. Çoklu platform (Android, Unix, IOS, Linux, Mac, Windows, Amiga, Symbian) desteğine sahip modüler bir programlama dili olan Python, hemen hemen her türlü platformda çalışabilir. Girintilere dayalı basit ve anlaşılır komut dizimi dilin öğrenilmesini açısından büyük kolaylık sağlar. Python günümüzde çok geniş bir kullanım alanına sahip olup sistem programlama, kullanıcı ara yüzü programlama, ağ programlama, mobil ve web tabanlı uygulama geliştirme, büyük veri odaklı uygulamalarda ve veri tabanı yazılımlarında etkin olarak kullanılan bir dildir. Bu nedenle günümüz yazılım endüstrisi, Python uzmanları için oldukça geniş kariyer olanakları sunmaktadır [42,45].

JavaScript: JavaScript, web tabanlı içeriklerin oluşturulmasında ve görsel olarak sunulmasında yaygın olarak web tarayıcılarında kullanılan nesne tabanlı ve sistematik bir betik programlama dilidir. JavaScript ile kodlanan istemci tarafında çalışan betik programlar sayesinde tarayıcı-kullanıcı etkileşiminin sağlanması, tarayıcının kontrol edilmesi, asenkron bir şekilde sunucu ile iletişim kurulması ve web sayfası içeriğinin

dinamik bir şekilde değiştirilmesi gibi önemli işlevler yerine getirilir. HTML ve CSS ile birlikte JavaScript web içerik üretiminin üç temel teknolojisinden birisi olmuştur. Web sayfalarını etkileşimli hale getirmesinde ve video oyunları da dâhil olmak üzere çevrimiçi uygulamaların geliştirilmesinde etkin olarak kullanılan bir teknolojidir. JavaScript, Node.js gibi yeni teknolojiler sayesinde sunucu tarafında da etkin bir şekilde kullanılmaktadır. JavaScript yüksek seviyeli, dinamik, yorumlanabilir, prototip tabanlı, çoklu platform desteğine sahip bir betik programlama dilidir. Nesne yönelimli bir dil olan JavaScript fonksiyonel programlama niteliklerine sahiptir. Genelde Java ile karıştırılsa da yapı olarak oldukça farklı dillerdir. JavaScript'in kodlama biçimi C programlama dilinden esinlenmiş olup tasarımı ve yorumlanması Self ve Scheme programlama dillerine benzemektedir [42,46].

Html5: Html5 ileri metin işaretleme dili, web içeriklerinin oluşturulmasını sağlayan çekirdek teknolojilerden html standardının beşinci sürümüdür. Güncel tarayıcılar tarafından yüksek oranda desteklenen bu teknoloji html'nin günümüzün zengin web içeriklerine uyarlanmış sürümüdür. Html5 kullanan sistemler tek kod temelli kullanım sayesinde hem kullanıcı ara yüzü (front-end) geliştirmeye hem de sunucu tabanlı (back-end) uygulama geliştirmeye olanak sağlamaktadır. Html5 teknolojisinin temelleri klasik html'ye dayansa da her platform için desteklenen sunucu tabanlı uygulama geliştirme eklentileri Html5'in getirdiği önemli yeniliklerdendir. Ses veya film etiketleri gibi birçok yeni özelliği ile Html5 sadece bir biçimlendirme dili olmaktan ziyade web uygulamalarının geliştirilmesinde etkin şekilde kullanılan bir programlama dili olmuştur. Css3 ve JavaScript ile birlikte kullanılan Html5 ile çok daha güçlü ve daha dinamik web ve mobil uygulamalar geliştirmek mümkündür. Bu yönüyle Html5, mobil uygulamalar için platformlar arası ortak bir geliştirme aracı olarak görülmektedir. Pek çok yeni söz dizimsel özelliklerle birlikte gelen Html5, grafiksel içerikler, multimedya (ses ve görüntü), ölçeklenebilir vektör grafikleri (SVG) ve matematiksel formüller için oldukça gelişmiş yenilikler sunmaktadır [42,47].

C#: Microsoft tarafından geliştirilen, basit, modern, genel amaçlı, nesne yönelimli ve orta seviyeli bir programlama dilidir. C#, yine Microsoft tarafından geliştirilen .NET platformuna yönelik olarak geliştirilen dillerden birisidir. C programlama dili bu mimarinin temelini oluşturur. C programlama dilini nesneye yönelimli programlama yapabilmek için yeni eklentilerle C++ programlama dili geliştirilmiş ve benzer şekilde C++

programlama diline yenilikçi eklentiler yapılarak bu mimari daha da ileriye götürülmüş ve tamamen nesneye yönelimli programlama için C# programlama dili geliştirilmiştir. Bu dilin tasarlanmasına Pascal, Delphi derleyicileri ve J++ programlama dilinin tasarımlarıyla bilinen Anders Hejlsberg öncülük etmiştir. Birçok alanda Java'yı kendisine örnek alan C# programlama dili, özellikle nesne yönelimli programlama kavramının gelişmesine katkı sağlayan işlevsel programlama dillerinden birisidir. C# orta seviyeli programlama dillerindendir. Yani hem makine diline hem de insan algısına eşit seviyededir. Şöyle ki; Visual Basic'in yüksek seviyeli bir dil olduğu göz önüne alınırsa, bu durum bu programlama dilinin insanların günlük konuşma biçimine yakın şekilde tasarlandığını gösterir. Dolayısıyla Visual Basic, C#'dan daha güçlü bir dildir denilemez [42,48].

Ruby: Ruby nesne yönelimli, dinamik, reflektif, fonksiyonel ve yüksek seviyeli bir programlama dilidir. Ruby betik programlama dili ilk olarak Yukihiro Matsumoto tarafından Japonya'da tasarlanmış ve geliştirilmiştir. Özgür ve açık kaynak kodlu bir programlama dili olan Ruby; GPL ve Ruby lisansı ile lisanslanmıştır. Aralık 1995'te 0.95 ile ilk genel sürümü yayınlanan Ruby'nin aynı ay içerisinde peş peşe üç sürümü daha yayınlanmıştır. İlk genel sürümden yaklaşık olarak bir yıl sonra, Aralık 1996'da Ruby 1.0 sürümü yayınlanmıştır. Anavatanı Japonya'da hızla gelişen ve tanınan Ruby, 1999'da İngilizce kaynaklarının yayınlanması ile tüm dünyaya açılma imkânına kavuşmuştur. 2000 yılında Japonya'da hızla büyüyen ve Python'dan daha popüler hale gelen Ruby, yüksek taşınabilirlik ve geniş kütüphane desteği ile betik programlama dilleri arasında adından sıkça söz ettiren bir teknoloji olmuştur. Ruby oldukça kolay, anlaşılır ve basit bir sözdizimine (syntax) sahiptir. Ruby'nin tasarımında model alınan diller Ada, Lisp, Eiffel, Perl ve Smalltalk'dur. Bundan dolayı betik programlama yapısına sahiptir. Ruby'nin kod yapısı ve sözdizimi günümüzde de yaygın olarak kullanılan Perl ve Python dilleri ile büyük ölçüde benzerlikler göstermektedir. Ruby on Rails, Ruby dili ile yazılmış açık kaynak kodlu bir web tabanlı uygulama geliştirme çatısıdır (framework) [42,49].

SQL: SQL, (Structured Query Language: Yapılandırılmış Sorgu Dili), verileri tasarlamak, yönetmek ve sorgulamak için kullanılan bir veri tabanı yönetim sistemidir. SQL, kendisi aslında bir programlama dili olmamasına karşın birçok otorite tarafından bir programlama dili olarak kabul edilir. SQL herhangi bir veri tabanı platformunda kullanılan bir alt dildir. SQL ile sadece veri tabanlarına yönelik işlemler gerçekleştirilebilir. SQL'e özgü komutlar kullanılarak veri tabanına yeni kayıtlar eklenebilir, var olan kayıtlar

değiştirilebilir, silinebilir ve bu kayıtlardan farklı veri listeleri oluşturulabilir. SQL dili ilişkisel veri yönetimi için tasarlanmış olup ilişkisel veri tabanı yönetim sistemlerinin tamamında yer alan standart bir dile dönüşmüştür. Bu nedenle veri odaklı bilgi sistemlerinde çalışan tüm bilişim uzmanları tarafından kesinlikle bilinmesi gereken bir dildir. Temel amacı verilerin ve veri setlerinin kullanım amacı doğrultusunda modellenmesine dayanan SQL, günümüzde dünya çapında kullanılan birçok veri tabanı ile yakından ilişkilidir. SQL ile kontrol edebilen başlıca veri tabanları: MySQL, Mssql, Microsoft SQL Server, Oracle, IBM Database 2 (IBM DB2) ve PostgreSQL şeklinde verilebilir [42,50].

R: Fonksiyonel bir programlama dili olan R, istatistiksel hesaplama, çıkarım ve grafikler için tasarlanmış bir uygulama ortamı olup aynı zamanda bir programlama dilidir. GNU platformunun bir parçası olan R özgür bir yazılımdır. Yeni Zelanda Auckland Üniversitesinden Ross Ihaka ve Robert Gentleman tarafından tasarlanan bu programlama dili, şu anda çekirdek bir ekip tarafından geliştirilmeye devam etmektedir. S programlama diline benzer özellikler taşıyan R, S'nin farklı bir uyarlaması olarak da değerlendirilmektedir. R, çok geniş kapsamda istatistiksel analizler ve grafiksel veri modelleme teknikleri sunmaktadır. R tam anlamıyla bir programlama dili olarak tasarlanmış olup kullanıcılara yeni fonksiyonlar tanımlama ve ek özellikler geliştirme olanağı sağlamaktadır. Yoğun hesaplamalar gerektiren analizler için C, C++ ve Fortran kodları da iş akışına eklenebilir. Hatta ileri düzey kullanıcılar R nesnelerinin farklı modellere uyarlanabilmesi amacıyla C kodu yazabilirler. R, kullanıcıların eklediği özel fonksiyonlar ve eklentilerle geliştirilmeye açık bir platformdur. Bu paketler belli temel disiplinlerle ilgili teknikler, gelişmiş grafik özellikleri ve birçok harici dosya biçimini (SPSS,Excel, Matlab, MySQL bağlantıları vb.) okuma-yazma yeteneği gibi birçok özelliği R programlama diline kazandıran eklentilerdir [51].

PHP: PHP, web içeriklerini sistematik bir biçimde oluşturmak ve yönetmek için geliştirilmiş sunucu taraflı bir betik programlama dilidir. PHP, html gibi derlenmez buna karşılık yalnızca sunucudaki php yorumlayıcısı tarafından yorumlanır. C programlama dilini bilen birisi için PHP'nin öğrenilmesi çok daha kolaydır, şöyle ki dosya işlemleri gibi karışık konularda PHP, C'den daha basittir. PHP ile yazılan kodlar hiçbir zaman kullanıcılar tarafından görülemez. Kullanıcı sayfanın kaynağını görüntülese bile göreceği sizin PHP kodlarınızın ürettiği html içerikleridir. PHP kodları, PHP yorumlama modülü

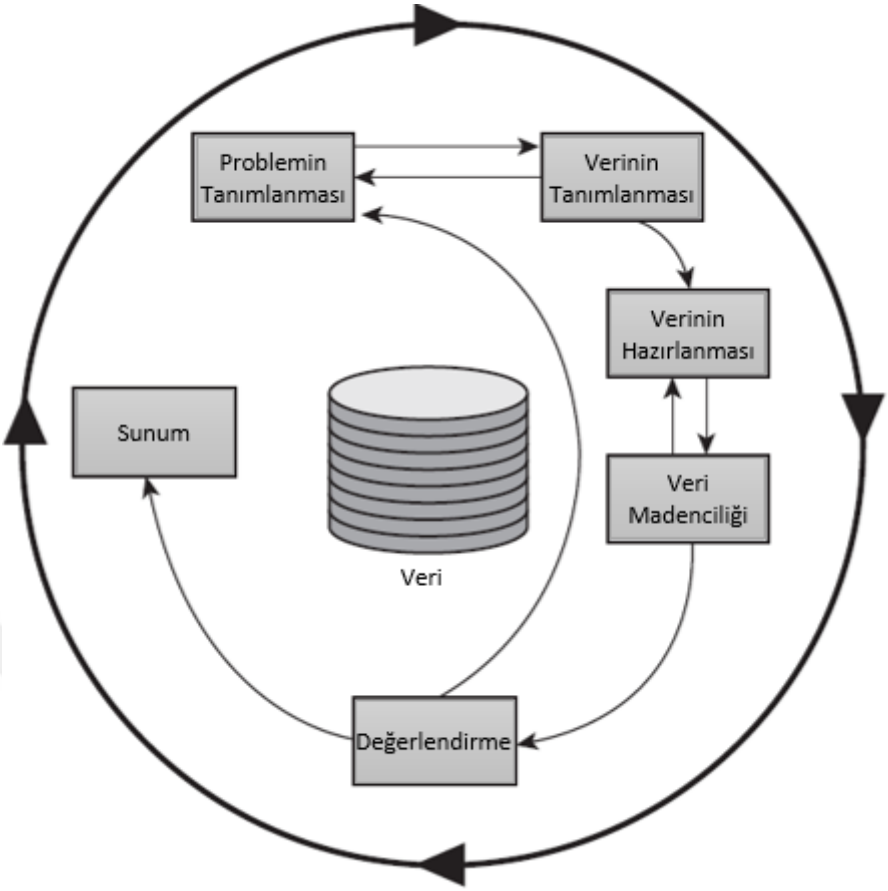
bulunan bir web sunucusu tarafından yorumlanır ve sonuçta çıktı olarak bir web sayfası üretilir. Bu PHP kodları veriyi işlemek üzere harici bir dosyaya kaydedilerek çağırılabildiği gibi doğrudan html kodlarının içine gömülü olarak da çalıştırılabilir. PHP özgür bir yazılım olup PHP lisansına sahiptir. Bu lisans kullanım şartları kısmında GNU Genel Kamu Lisansı ile tam olarak örtüşmese de, PHP tüm web sunucularına ve hemen hemen tüm işletim sistemlerine ve farklı uygulama geliştirme platformlarına ücretsiz olarak yüklenebilir [52].

1.6. Veri Madenciliği

Veri madenciliği (veriden bilgi keşfi, "data mining") büyük miktardaki veri yığınları içerisinden sistematik süreçlerle anlamlı bilgi ve örüntülerin elde edilmesidir (Knowledge Discovery in Databases). Başka bir deyişle veri madenciliği, makine öğrenimi, istatistik ve veri tabanı sistemlerinin birlikte kullanımını gerektiren yöntem ve analizleri içeren kapsamlı bir süreçtir. Veri madenciliği ile elde edilen bu anlamlı bilgi ve örüntüler karar destek sistemleri içerisinde ileriye yönelik stratejilerin belirlenmesi amacıyla etkin bir şekilde kullanılmaktadır [53].

1.6.1. Veri Madenciliği Aşamaları

Veri madenciliği birbirini takip eden adımlardan oluşan kapsamlı bir süreçtir. Bu süreç problemin ortaya konulması ile başlar ve veriler üzerinde istenen işlemlerin adım adım gerçekleştirilmesiyle problemin çözümüne kadar uzanır. Bu süreç Şekil 3'de akış şeması olarak gösterilmiştir. Veri madenciliği sürecine dâhil edilen problemin etkin bir şekilde çözümlenmesine kadar devam eden süreçte eldeki veri kümesine farklı aşamalarda farklı işlemler uygulanır. Veri madenciliği sürecini oluşturan bu aşamaların her birisinde veri kümesi, birbirini takip eden sıralı işlemlerle ilk olarak analiz için uygun hale getirilir. Daha sonra bu veri kümesine veri madenciliği algoritmaları uygulanarak tanımlanan problem için bir çözüm modeli oluşturulur. Bu süreç boyunca veri kümesine hangi aşamalarda hangi işlemlerin uygulandığı aşağıda sırasıyla verilmiştir [53]:



Şekil 3. Veri madenciliği aşamaları

- Problemin Tanımlanması: Ortaya konulan problem bu aşamada bir veri madenciliği projesi olarak tanımlanır ve problemin çözülmesi için gereken adımlar belirlenir. Problemin veri madenciliği projesine dönüştürüldüğü bu aşamada veri madenciliği uzmanları, çözüm talep eden taraflar ve alan uzmanları proje hedeflerini ve gereksinimleri bir iş takvimi çerçevesinde tanımlamak için birlikte çalışırlar. Bu şekilde tasarlanan projenin hedefleri bir veri madenciliği problemi haline dönüştürülürmüş olur [53,54].

- Veri Toplama: Veri toplama aşaması, problemin tanımlanmasından sonra bu problemin çözümü için ihtiyaç duyulan verilerin ve bu verilerin elde edilebileceği veri kaynaklarının tespit edildiği aşamadır. Tanımlanan problemin karakteristiklerine uygun veri kaynaklarının belirlenmesinden sonra ihtiyaç duyulan veriler bu kaynaklardan farklı yöntemlerle toplanarak analiz için gerekli veri seti oluşturulur. Verilerin toplanmasında problemin türüne bağlı olarak tek bir veri kaynağı yeterli olabildiği gibi farklı veri kaynaklarından elde edilecek

verilere de ihtiyaç duyulabilir. Problemin çözümü için uygun özelliklerde ve yeterli miktarda veri toplandıktan sonra bu verilerle deneysel veri seti oluşturulur ve uygun formatta veri tabanlarına kaydedilir [53,54].

- Veri Ayıklama ve Önişleme: Veri madenciliğinin bu önemli aşaması veri tabanında yer alan tutarsız ve hatalı verilerin temizlendiği aşamadır. Hatalı ve tutarsız veri gerekli analizlerin amacına uygun olarak yapılmasını ve dolayısıyla analizin başarımını olumsuz yönde etkiler. Bundan dolayı hatalı ve tutarsız veriler temizlenmeli ve çıkarılan veriler yenileriyle tamamlanmalıdır. Bu aşamada uygun hale getirilen veri ilerleyen aşamalardaki süreçleri doğrudan etkilemektedir. Bu nedenle veri önişleme aşaması analizin başarımını doğrudan etkileyen önemli bir süreçtir [53,54].

- Veri Bütünleştirme: Farklı veri kaynaklarından elde edilen verilerin veri seti içerisinde tutarsızlıklara yol açmasını önlemek amacıyla farklı veri türlerinin tek türe dönüştürüldüğü aşamadır. Bu işlem sayesinde farklı kaynaklardan elde edilen farklı türdeki veriler tek türde bütünleştirilerek homojen bir veri yapısının oluşturulması amaçlanır [53,54].

- Veri İndirgeme: Büyük veri setleri üzerinde gerçekleştirilen veri madenciliği çalışmalarında verinin tamamını analiz etmek zor ve maliyetli olabilir. Bu ve benzeri durumlarda veri indirgemenin sonuca etkisinin önemsenmeyecek kadar az olduğu tespit edilirse verinin miktarı veya değişkenlerin sayısı azaltılabilir [53,54].

- Veri Seçme: Veri seçme aşaması yapılacak analize uygun olan verilerin belirlendiği aşamadır. Bu aşamada yapılan işlemin amacı, seçilen verilerin uygulanacak analiz modeline uygun yapıda verilerden oluşmasını temin etmektir [53,54].

- Veri Dönüşümü: Kullanılacak analiz modeline göre verinin içeriğini koruyarak sadece biçiminin uygun formata dönüştürülmesi işlemidir. Veri dönüştürme işlemi, kullanılacak veri madenciliği modeline uygun biçimde gerçekleştirilmesi gereken bir aşamadır [53,54].

- Veri Madenciliği: Önceki aşamalardaki uygulanan işlemlerle hazır hale getirilen veri seti üzerinde veri madenciliği algoritmalarının uygulanması bu aşamada gerçekleştirilir. Tanımlanan problem için en uygun çözüm modelinin belirlenebilmesi, olabildiğince çok sayıda yöntemin denenmesi ile mümkün olabilmektedir. Veri madenciliği algoritmalarının deneysel veri seti üzerinde uygulandığı bu aşama, en ideal çözüm modeli elde edilinceye kadar tekrarlanan bir süreçtir [53,54].

- Örüntü Değerlendirme: Bu aşamada, uygulanan model uzmanlar tarafından değerlendirilir. Eğer uygulanan model beklentileri karşılamıyorsa, modelleme aşamasına geri dönülür ve ideal değerlere ulaşılıncaya kadar model parametreleri değiştirilerek model yeniden inşa edilir. Sonunda uygulanan model belirlenen hedefleri karşılıyorsa, elde edilen örüntüler değerlendirilir ve bu örüntülerin nerde nasıl kullanılacağına karar verilir [53,54].

Böylelikle yukarıda bahsedilen veri madenciliği aşamaların sonucunda, önceden belirlenen problem için bir çözüm modeli oluşturulur ve süreç bu şekilde tamamlanmış olur. Bu süreç sonunda elde edilen anlamlı bilgiler ve örüntüler gelecekteki iş stratejilerinin belirlenmesinde bir karar destek mekanizması olarak kullanılır [53,54].

1.6.2. Veri Madenciliği Modelleri

Genel olarak veri madenciliği modelleri iki başlık altında kategorize edilir. Birincisi, kestirime dayalı veri madenciliği modeli (predictive models), diğeri ise tanımlayıcı veri madenciliği modeli (descriptive models) olarak tanımlanır. Kestirime dayalı veri madenciliği; sınıflandırma, eğri uydurma, zaman serileri analizi gibi modelleri içermektedir. Tanımlayıcı veri madenciliği ise; kümeleme, ilişkilendirme kuralları, özetleme, anomali tespiti gibi modelleri içermektedir [53,54]:

Sınıflandırma (Classification): Sınıflandırma, veri madenciliğinde en çok kullanılan modeldir. Kestirime dayalı bir yaklaşım olan sınıflandırma modelinde, var olan veri setinin bir kısmı eğitim seti olarak kullanılır ve sınıflandırma için gerekli olan kurallar oluşturulur. Bu eğitim seti, oluşturulan kurallar yardımıyla yeni verilerin hangi sınıflara atanacağını tahmin etmeye çalışır. Bahsedilen bu sınıflandırma modeli makine öğrenmesi olarak da

bilinmektedir. Veri madenciliğinin sınıflandırma modeli içerisinde en sık kullanılan algoritmalar; karar ağaçları, Bayes sınıflandırıcı, en yakın komşu, yapay sinir ağları, destekçi vektör makinası, lojistik regresyon, diskriminant analizi gibi makine öğrenmesine dayalı yaklaşımlardır [53,54].

Eğri uydurma (Regression): Eğri uydurma modeli, veri ve veri kümeleri arasındaki ilişkileri tahmin etmek amacıyla var olan veriyi gerçel değerli bir fonksiyona dönüştürür. Bu şekilde veriler fonksiyonda yer alan değişkenlerin etki değerlerine göre modellenmiş olur. Bu model ile değişkenler arasındaki ilişkinin varlığı ve eğer ilişki var ise bu ilişkinin hangi düzeyde olduğunun belirlenmesi amaçlanır [53,54].

Zaman serileri analizi (Time series analysis): Zaman serileri analizi tahmin değerleri önceden bilinen olayları örnek alarak gelecek olayları tahmin etmeye yarayan bir kestirim modelidir. Bir zaman serisi, eldeki verinin zaman içerisinde sıralanmış ölçümlerinin bir kümesidir. Zaman serisi analizinin amacı, gözlem kümesinde temsil edilen değerlerin anlaşılmasını ve zaman serisindeki değişkenlerin zaman içindeki değerlerinin doğru bir şekilde tahmin edilmesini sağlamaktır [53,54].

Kümeleme (Clustering): Veri madenciliğinde yaygın olarak kullanılan kümeleme yaklaşımı verilerin kendi aralarındaki benzerliklere göre gruplandırılması işlemidir. Kümeleme yaklaşımları çoğunlukla veriler arasındaki uzaklıkları temel alarak işlem yapar. Veri madenciliğinde çok sayıda kümeleme yaklaşımı kullanılmaktadır. Bu yaklaşımlar, değişkenler arasındaki benzerliklerden ya da tam aksine farklılıklardan yararlanarak bir veri kümesini alt kümelere ayırmakta yani verileri gruplandırmakta kullanılmaktadır. Kümeleme analizinin amacı, yapılandırılmamış verileri benzerliklerine veya farklılıklarına göre gruplandırmak ve verinin daha anlaşılır bir biçimde özetlenmesini sağlamaktır [53,54].

İlişkilendirme kuralları (Association rules): Veri kümesi içinde yer alan verilerin birbiriyle olan ilişkilerini inceleyerek yaygın örüntüleri ve veriye ait nitelikler arasındaki ilişkileri ortaya koymaya çalışan veri madenciliği yaklaşımlarıdır. Bu yaklaşımlarla veriler arasındaki birliktelik kuralları belirlenerek hangi olayların eş zamanlı olarak birlikte gerçekleşebileceği tespit edilmeye çalışılır. Özellikle pazarlama alanında (market sepet analizleri) yaygın olarak kullanılan bu yaklaşım, birliktelik kurallarının perakende satışlar

üzerinde uygulanmasıyla müşterilerin satın alma eğilimlerinin belirlemesini sağlayan etkin bir modeldir [53,54].

Özetleme (Summarization): Bu yaklaşımda, veri kümesinin tamamını temsil etmek yerine veriyi en iyi şekilde ortaya koyan niteliklerin belirlenip verinin bu niteliklerle temsil edilmesi amaçlanır. Bu işlemle veri kümesi alt gruplara ayrılır. Her alt grubu temsil edecek nitelikler belirlenir. Bu işlem genelleştirme (generalization) ve nitelendirme (characterization) olarak adlandırılan farklı iki yöntemle gerçekleştirilir. Verinin özetlenmesi, özellikle büyük miktarlardaki veri yığınlarının daha kolay anlaşılmasına ve yorulmasına katkı sağlayan bir süreçtir [53,54].

Anomali tespiti (Anomaly detection): Veri madenciliğinde anomali (aykırılık veya sapma) tespiti, bir veri setindeki beklenen değerlere ve niteliklere uymayan verilerin, veriye ait niteliklerin, olayların veya gözlemlerin belirlenmesi işlemidir. Bu yöntemle ilginç olabilecek olağandışı veri kayıtlarının veya daha ileri düzeyde analizleri gerektiren veri olaylarının tespit edilmesi sağlanır [53,54].

Özetle, veri madenciliğinin kestirim (predictive) ve tanımlama (descriptive) olmak üzere iki temel işlevi vardır. Kestirime dayalı modeller, sonuçları bilinen verileri kullanarak yeni veriler için bir tahmin modeli oluşturur. Ortaya konulan bu model, sonuçları bilinmeyen öğelerin tahmin edilmesinde kullanılır. Tanımlayıcı modeller ise veri setinde yer alan veriler arasındaki ilişkileri, örüntü ve davranışları ortaya koymak amacıyla kullanılır [53,54].

1.7. Metinsel Verilerden Bilgi Keşfi

Metin madenciliği olarak da bilinen bu süreç genellikle büyük miktarlardaki metinsel verilerden faydalı bilgilerin çıkarılmasını işlemlerini kapsamaktadır. Başka bir deyişle, veri madenciliğinin özel bir alanı olan metin madenciliği, geleneksel veri madenciliği yöntemlerin doğal dil işleme yordamlarıyla birlikte dijital ortamlarda üretilen ve paylaşılan metinsel içerikler üzerinde uygulanması olarak da tanımlanabilir.

Metin madenciliği kapsamında yapılan çalışmalar, metinlerin kümelenmesi, özetlenmesi, sınıflandırılması ya da modellenmesi gibi farklı analizleri kapsamaktadır.

Metin madenciliğinin yeni bir uygulama alanı olan duygu ve fikir analizi ise belli konularda insanların bireysel ve toplumsal olarak düşüncelerinin ve eğilimlerinin metinsel veriler üzerinden tespit edilmesine yönelik bir çalışma alanıdır [55,56].

Metinsel veriler üzerinde bilgi keşfine dayalı analizlerin en çok kullanıldığı araştırma alanları: verilerin görselleştirmesi, metinlerin etiketlenmesi, yazar tanıma, varlık-ilişki tanımlama, metinsel örüntü tanıma, belgeler arasındaki benzerliklerin bulunması, metinlerin özetlenmesi, metinler üzerinde olay tespiti, kişilik tespiti, konu tespiti, metinsel veriler üzerinden bireysel ve toplumsal eğilimlerin tespiti, duygu ve fikir analizi, metinsel içerikler üzerinde konu modellemesi, vb. olarak sıralanabilir. Özellikle metin odaklı büyük veri yığınları içerisindeki gizli anlamsal yapıları keşfetmek amacıyla olasılıksal konu modellemeye dayalı gerçekleştirilen araştırma ve uygulamalar da günümüzün ilgi çeken çalışma alanlarıdır [55,56].

Metinsel verilerden bilgi keşfi, yapısal olmayan metinsel verilerin sistematik yaklaşımlarla analiz edilmesi sonucu anlamlı bilgilere erişilmesi ve verilerin anlamlı bilgi olarak yapısal halde sunulması süreçleridir. Metinsel veri yığınlarından bilgi çıkarımı doğal dil işlemede çalışılan önemli araştırma konularındandır. Sosyal medya uygulamalarının yaygın olarak kullanıldığı son beş yıl içerisinde metinsel verilerin miktarı ve çeşitliliğindeki büyük artış, metin analizine dayalı çalışmaların önemini daha da artırmıştır.

Metin madenciliğine dayalı çalışmaların amaçlarından bir tanesi de metinden anlamlı, nitelikli ve yorumlanabilir bilginin çıkarılmasıdır. Böylelikle metinsel veri setinin içerdiği temel içerik ortaya koyulabilir. Metin madenciliğindeki diğer temel amaç ise metinsel veri yığınlarının yapısal ve anlamsal olarak modellenmesine dayanır [55,56].

Bir metinsel veri kümesinden anlamlı bilgi edilinceye kadar gerçekleştirilen ardışık işlemler ya da daha çok bilinen tabirle ifade edilirse “metin madenciliği aşamaları” aşağıda verilen sıralı adımları içermektedir [55,56]:

- Metinsel veri kümesinin oluşturulması
- Metin önişleme
- Öznitelik çıkarımı

- Metinlerin vektörel temsili
- Terim ağırlıklandırma
- Metin analizi
- Değerlendirme ve yorum

Veri madenciliği yaklaşım ve yordamları sadece yapısal (analize uygun) veriler üzerinde uygulanabildiği için metinsel veri yığınlarının bu sürece dâhil edilmeden önce belli önişleme adımlarıyla yapısal hale getirilmesi gereklidir. Metinsel veriler yapısı gereği nitel verilerdir. Nitel yapıdaki metinsel veriler üzerinde nicel veri analizlerin gerçekleştirilebilmesi için metinsel verilerin yapılandırılması ve nicel veri analizi için uygun hale getirilmesi gerekir. Bu amaçla metinsel veriler önişleme aşamalarından geçirilerek yapısal hale dönüştürülür [1,55,56].

1.7.1. Metin Önişleme

Metin önişleme, metin madenciliğinde analiz öncesi metin belgelerinin yapıtaşı olan kelimelerle üzerinde uygulanan işlemleri içeren önemli bir süreçtir [56,57]. Metin önişleme adımları, deneysel analiz sürecinin başarılı ya da başarısız olmasına direk etki eden bir süreçtir [57]. Gerçekleştirilecek analiz öncesi deneyde kullanılacak metinlerin analiz sonrası çıkarımları en iyi şekilde yansıtacak dağılıma ve hassaslığa sahip olması beklenmektedir. Gerçek veri analizine dayanan uygulamalarda veri eksik, gürültülü veya tutarsız olabilir [56,57].

Metin önişleme adımları verinin eksik nitelik değerlerini tamamlama, metinlerdeki aykırılıkların bulunması, gürültülü verinin daha homojen hale getirilmesi, tutarsızlıkların giderilmesi, anlam ifade etmeyen verilerin silinmesi ve boyut indirgeme gibi birtakım sıralı işlemleri içermektedir. İşlenen metinlerin ve gerçekleştirilecek olan deneysel analizin türüne bağlı olarak metin önişleme adımları değişiklik gösterebilir. Eğer işlenen metinler web sayfalarından oluşuyorsa html etiketlerinin, sosyal medya mesajlarından oluşuyorsa o sosyal ağa özgü karakterlerin ve bazı özel terimlerin, elektronik postalardan oluşuyorsa kişisel bilgilerin anlamlı içeriklerden ayrıştırılması gerekir. Genel olarak, metinsel verilere uygulanan önişleme işlemleri aşağıda verildiği gibidir [1,56,57].

Metin önişleme aşamasının ilk adımı dizge parçalama (tokenization) işlemi ile başlar. Bu aşamada anlamlı özniteliklerin elde edilebilmesi için metinsel içerik kelimelere ayrılır. Böylelikle veri setinde yer alan her bir metinsel veri bir kelime vektörü ile temsil edilmiş olur. Bu işlemden sonra genel olarak metinler üzerinde veri temizleme işlemi gerçekleştirilerek noktalama işaretleri, web bağlantıları, özel etiketler ve anlamsız karakterler metinlerden silinir [1,56,57].

Metin önişleme aşamasında uygulanan önemli işlemlerden bir tanesi de elde edilen kelime uzayını küçültmek için metinlerdeki durak kelimelerinin (stop words) silinmesi işlemidir. Durak kelimeleri bir dilde yaygın olarak kullanılan ve genellikle tek başına kullanıldığında bir anlam ifade etmeyen kelimelerdir (örneğin, İngilizce için: “and”, “or”, “with”, “there”, “she” gibi kelimeler, Türkçe için: “ve”, “ile”, “veya”, “ne”, “için” gibi kelimeler). Bu nedenle metin analizine dayalı çalışmalarda genellikle metinsel içerikler durak kelimelerinden temizlenir. Ancak bu kelimeler çıkarılırken her dile özgü durak kelimelerini içeren bir listeye ihtiyaç duyulmaktadır. Durak kelimelerinin çıkarılması işlemi yapılan analizin türüne ve öznitelik çıkarım modeline bağlı olarak değişiklik göstermektedir [1,56,57].

Metin önişleme de uygulanan diğer bir işlem ise gövdeleme (stemming) işlemidir. Köke indirgeme olarak da bilinen bu işlem, hecelerle türetilen kelimelerin tek bir köke indirgenerek kelime uzayında tek bir kelime olarak örneklenmesini sağlamaktadır. Örneğin, gözlemlemek, gözledim, gözlemler, gözlemsel gibi kelimeler gözlem kökünden türetilmiş kelimelerdir. Bu haliyle kelime uzayında beş farklı kelime olarak temsil edilecek bu kelimeler, gövdeleme işlemi sonrası tek bir kelime (“gözlem”) ile temsil edilir. İngilizce metinler için Porter, Stemmer gibi gövdeleme algoritmaları yaygın olarak kullanılmakla birlikte, Türkçe gibi sondan eklemeli bir dil için başarı yüzdesi istenen seviyede değildir [1,56,57].

1.7.2. Öznitelik Çıkarımı

Metin madenciliğinde kullanılan algoritmaların metinler üzerinde uygulanabilmesi için metinlerin analize uygun formata dönüştürülmesi gerekmektedir. Metinsel veri kümeleri genel olarak içerdiği kelimelerle ve kelimelerin diziliminden ortaya çıkan kelime

gruplarıyla belirlenir. Metinsel analizlerde basitlik açısından genellikle kelimeler ya da kelime grupları göz önüne alınır ve kelimelerin sıralamaları ihmal edilir. Metinleri temsil etmek üzere öznitelik olarak seçilen kelime ya da kelime gruplarına terim (term, feature) adı verilir [1,58-60].

Metin madenciliğinde kullanılan öznitelik belirleme yöntemlerinden en yaygın olanı kelime torbası (BoW: Bag of Words) modeli olarak adlandırılır. BoW modelinde veri kümesindeki metinlerde yer alan tüm kelimelerin görülme sayıları hesaplanır ve bir havuzda toplanır. Bu şekilde veri kümesindeki metinlerin içerdiği tüm kelimelerin neler olduğu ve her bir kelimenin görülme frekansı belirlenmiş olur. BoW modeli, öznitelik olarak kelimelerin kullanıldığı bir model olup kelimelerin metin içerisindeki sırası bu modelde dikkate alınmaz. BoW modelinde kelimelerin dilbilimsel ve anlamsal özellikleri ile değil metin içeriğinde görülme frekansları kullanılarak istenen analizler gerçekleştirilir. BoW modelinde öznitelikler kelimelerle belirlendiği için bu model kullanılan dile bağımlıdır ve bu nedenle veri kümesindeki metinler üzerinde dile uygun önişleme uygulanmalıdır [1,58-60].

Metin madenciliğinde özniteliklerin oluşturulmasında kullanılan diğer bir yöntem ise N-gram modelidir. N-gram modelde, öznitelikler metin içerisinden çıkarılan farklı uzunluklardaki karakter grupları (karakter seviyesi) veya kelime gruplarından (kelime seviyesi) oluşmaktadır. Karakter seviyesindeki N-gram modeli karakter grubunun uzunluğu ile isimlendirilirken, kelime seviyesindeki N-gram modeli ise içerdiği kelime sayısı ile adlandırılır. Karakter seviyesindeki N-gram modeli, karakter uzunluğu iki ise "bigram", üç ise "trigram" olarak adlandırılır. Karakter sayıları üçten fazla olursa, 4 için "fourgram", 5 için "fivegram", vb. şeklinde adlandırma yapılır. Örneğin, karakter seviyesinde N-gram modeli ile trigram öznitelikleri çıkarılmak istenen kelime "yazılım" olsun. Bu durumda karakter seviyesinde elde edilen trigramlar: "yaz", "azı", "zıl", "ılı", "lım" şeklinde sıralanır [59,60].

Kelime seviyesindeki N-gram modelinde ise, sadece tek bir kelime ile oluşturulan N-gram'lar, "unigram" olarak adlandırılır. Birden fazla kelime içeren N-gram modellerinde ise adlandırma karakter seviyesindeki N-gram modeli ile aynı şekilde yapılmaktadır. Örneğin, kelime seviyesinde N-gram modeli ile öznitelikleri çıkarılmak istenen metin

“Yazılım geliştirme süreci fizibiliteyle başlar” olsun. Bu durumda kelime seviyesinde elde edilen N-gram’lar aşağıdaki gibi olur:

- Unigram : “Yazılım”, “geliştirme”, “süreci”, “fizibiliteyle”, “başlar”
- Bigram : “Yazılım geliştirme”, “geliştirme süreci”, “süreci fizibiliteyle”, “fizibiliteyle başlar”
- Trigram : “Yazılım geliştirme süreci”, “geliştirme süreci fizibiliteyle”, “süreci fizibiliteyle başlar”

Kelime seviyesindeki N-gram modeli, kelime torbası modeli ile aynı yaklaşıma dayanır her iki modelde de öznitelikler kelimelerden oluşur. Ancak veri önişleme sürecinde, kelime seviyesindeki N-gram modeli için kelimelere ayırma ve N’ sayısına göre kelimeleri gruplama dışında diğer önişleme adımları uygulanmaz. Metin madenciliği alanında öznitelikler hangi model ile elde edilmiş olursa olsun terim olarak tanımlanır [59,60].

1.7.3. Metinlerin Vektörel Temsili

Metin madenciliğinde kullanılan nicel analize dayalı algoritmaların nitel veri olan metinler üzerinde uygulanabilmesi için metinlerin analize uygun sayısal formata dönüştürülmesi gerekmektedir. Nicel metin analizlerinde yaygın bir yaklaşım olan vektör uzay modelinde, dokümanlar çok boyutlu vektör uzayında bir terim vektörü olarak temsil edilmektedirler. Dokümanlar kümesindeki ayrık terim (kelime) sayısı vektör uzayının boyutunu belirlemektedir. Bu yaklaşımda veri setindeki her bir metin bir vektör ile temsil edilir ve tüm metinler için oluşturulan her bir vektörün boyutu aynıdır. Bu vektörlerin boyutu terim uzayındaki toplam terim sayısına eşittir. Her bir metin için oluşturulan terim vektörü eşit boyutlu olmasına rağmen içerdikleri terimlere göre vektörlerin terim değerleri farklılık gösterir [1,58-60].

Veri kümesindeki metinlerin terim uzayı vektörel olarak modellendikten sonra her metne ait olan terim vektörleri ile cebirsel işlemler ve farklı içeriklerde analizler gerçekleştirilebilir. Örnek olarak, dokümanlar arasındaki benzerliği ölçmek için nokta çarpım benzerliği, kosinüs benzerliği veya farklı bir benzerlik yöntemi kullanılabilmektedir. Bunun dışında terim vektörleri üzerinde gerçekleştirilen farklı

işlemlerle kümeleme analizi, anlamsal analizler veya konu modelleme analizleri gerçekleştirilebilir [1,58-60]. Veri setindeki her bir metin için terim vektörü oluşturulduktan sonra bu vektörlerdeki her bir terim için, terim ağırlıklandırma (TW: Term Weighting) işlemi uygulanır.

1.7.4. Terim Ağırlıklandırma

Vektörler metinlerin temsilinde kullanılan en temel modelleme aracıdır. Metinler içindeki her bir kelime vektörlerin elemanıdır. Metinlerin vektör uzayına dönüştürülmesinde veya her metine ait bir terim vektörü oluşturulmasında değişik terim ağırlıklandırma (TA) yöntemleri kullanılmaktadır. Bunlardan ikili ağırlıklandırma yönteminde kelimelerin vektör için var veya yok olarak ifade edilmesi esas alınır. Bu yöntemde önemli olan, bir dokümanın her hangi bir terimi içerip içermediğidir. Eğer metin o terimi içeriyorsa hangi sıklıkla içerdiğine yani frekansına bakılmaz ve ağırlığı 1 kabul edilir. Eğer metin o terimi içermiyorsa ağırlığı sıfır kabul edilir [60,61].

Normal terim frekansına göre yapılan ağırlıklandırma da ise, terim frekansı yani o terimin metindeki görülme sıklığı esas alınır. Bu modelde veri kümesinde yer alan metinler içerdikleri terimlerin frekanslarıyla temsil edilir. Bu terimlerin frekansları belirlenirken kelimelerin kendileri, kökleri, karakter gramları ya da kelime gramları kullanılabilmektedir. Bu modele göre satırlarında metinlerin, sütunlarında terimlerin yer aldığı bir sayısal matris oluşturulur. Matrisin [i, j] hücresinde i. metinde j. kelimenin kaç kere geçtiğine ait frekans bilgisi tutulur. Matrisin satır sayısı doküman (metin) sayısına, sütun sayısı ise tüm metinlerde geçen ayrık terimlerin (kelimelerin) sayısına eşittir [60,61].

Metin madenciliğinde terimlerin hem sıklığını hem de tüm dokümanlar içindeki etkisini göz önüne alan Terim Frekansı- Ters Doküman frekansı (TF-TDF) (Term Frequency - Inverse Document Frequency, TF-IDF) yöntemi vektör ağırlıklandırılmasında yaygın olarak kullanılan bir başka modeldir. TF-TDF ile ağırlıklandırma yönteminde terim frekansı, bir metin içinde diğer terimlere kıyasla daha sık geçen bir terimin öneminin fazla olduğunu esas alan bir ölçüttür. Ters doküman frekansı ise metinler kümesinde daha az sayıda görülen bir terimin ayırt edici özelliğinin ön plana çıkarıldığı bir ölçüttür. TF-TDF ağırlıklandırma modelinde her terimin ağırlığı o terim için hesaplanan terim frekansı ile

ters doküman frekansının çarpımına eşittir [60,61]. TDF yöntemi, terimlerin ağırlıklarını sadece bulunduğu dokümanda değil tüm veri seti kapsamında değerlendirmeye tabi tutan bir yaklaşımdır. Bu yöntemle tüm veri setinde yaygın olarak kullanılan terimlerin ağırlığı düşürülürken, seyrek olarak gözlemlenen terimlerin ağırlığı yükseltilmeye çalışılır. Doküman frekansı (DF): bir c teriminin veri setinde gözlendiği doküman sayısını ve bununla birlikte d_i : i.dokümanı temsil etmek üzere TDF aşağıdaki eşitliklerle ifade edilir [60,61]:

$$W_{DF} = \sum_{i=1}^{n} TF(d_i, c) \tag{1}$$

$$W_{TDF}(c, d) = \log \frac{N}{DF(c, d)} \tag{2}$$

TF-TDF, bir terimin ağırlığını bir dokümandaki frekansıyla orantılı olarak artırır ancak veri seti genelindeki frekansıyla bunu dengelemeye çalışır. TDF yönteminde olduğu gibi amacı dokümanlarda ortak olarak kullanılan terimlerin ağırlığını düşürmektir ancak bunu yaparken terimin yerel frekansını da dikkate alır. TF-TDF ile ağırlıklandırma yöntemi için W ağırlıklandırma fonksiyonu Eşitlik 3 ile ifade edilir [60,61].

$$W_{TF-TDF} = TF(c, d).TDF(c, d) \tag{3}$$

Yukarıda bahsedilen terim ağırlıklandırma yöntemlerinin dışında, metin analizlerinde kullanılan diğer bir TA yaklaşımı da entropi ağırlıklandırmasıdır. Entropi ağırlıklandırmasında temel varsayım, eğer bir terim tüm dokümanlarda bir kez görülüyorsa ağırlığı 0, sadece bir dokümanda bir kez görülüyorsa o terimin ağırlığı 1 olacaktır. Entropi modeli, seyrek olarak gözlenen terimlerin ağırlığını yükselttiği için kullanışlı ve etkin bir ağırlıklandırma yöntemidir [60,61].

1.8. Olasılıksal Konu Modelleme Yaklaşımı

Olasılık teorisi veya olasılık kuramı, rastlantı ya da kesin olmayan olayları ve bu olayların gerçekleşebilme ihtimallerini belli kurallara göre matematiksel yöntemlerle inceleyen bilim dalıdır. Rastlantı olayı ya da rassal olay, gerçekleşmesi şansa bağlı olan

önceden kesinlikle bilinmeyen olaylardır. Olasılık kuramının esasları rassal değişkenler, rassal değişkenlerle ilgili süreçler ve rassal olaylara dayanır. Bir değişkenin gelecekteki bir gözlemde alacağı değer önceden bilinmiyorsa bu değişken "rassal değişken" olarak tanımlanır. Bir rassal değişkenin bir gözlem sırasında belli bir değeri almasına "rassal olay" denir. Bir madeni paranın yazı-tura denemesi için havaya atılması ve paranın tura gelmesi olayı bir rassal olaydır [62,63].

Rassal bir olayın modellenmesi sayısal değerlerle ifade edilen ve rassal değişken olarak adlandırılan değişkenler yardımıyla yapılır. Rassal bir deneyin ya da olayın sonuçlarını sayısal değerlerle ifade eden değişkene rassal (rastgele) değişken denir. Rassal değişken yardımıyla örnek uzayda tanımlanan olaylara sayısal değerler karşılık getirilir. Yani rassal deneyin sonuçlarıyla direk işlem yapmak yerine, onlara karşılık gelen sayısal değerler kullanılarak işlem yapılır. Rassal değişkenler aldıkları değerlere göre kesikli ve sürekli olmak üzere ikiye ayrılır. Değer kümesi sayılabilir olan rassal değişkenler kesikli, değer kümesi sayılamayan rassal değişkenler de sürekli olarak adlandırılır [62,63].

Örneğin, bir X rassal değişkeni bir kutudaki kusurlu kalem sayısı olarak tanımlanırsa, burada X rassal değişkeni kesikli olur. Bir başka Y rassal değişkeni yeni doğan bebeklerin ağırlıkları olarak tanımlanırsa, Y rassal değişkeni sürekli olur. Benzer şekilde, öğrencilerin boy uzunlukları, domateslerin ağırlığı, rüzgârın hızı, havanın sıcaklığı, yaşlı insanlardaki kolesterol miktarı sürekli rassal değişkenlere birer örnektir.

Rassal değişkenin aldığı değerlere karşılık, bu değerleri alması olasılıkları "olasılık dağılımı" olarak ifade edilir. Olasılık dağılımı bir rassal değişkene ilişkin tüm olasılıkların hesaplanabilmesini sağlar. Herhangi bir olasılık dağılımı $y=f(x)$ biçiminde tanımlanan matematiksel bir fonksiyondur (burada y, x' değerlerinin ortaya çıkma sıklığını gösterir). Bir fonksiyonun kesikli olasılık dağılım fonksiyonu olarak tanımlanabilmesi için aşağıdaki iki özelliği sağlaması gerekir:

- Birinci özellik, olasılık dağılımında yer alan tüm $P(X=x)$ olasılıklarının [0,1] aralığında olmasını gerektirir.

- İkinci özellik ise, olasılık dağılımında yer alan tüm olasılık değerlerinin toplamının 1' e eşit olmasını gerektirir $(\sum P(X=x)=1)$.

Kesikli rassal değişkenleri modellemek için kullanılan kesikli olasılık dağılımlarının en yaygın olanları Bernoulli dağılımı, Binom dağılımı ve Poisson dağılımıdır [62,63].

Sürekli rassal değişkenleri modellemek içinse sürekli olasılık dağılımlarından yararlanılır. Sürekli rassal değişkenlerle ilgili olasılıklar kesikli rassal değişkenlerde olduğu gibi tek tek hesaplanamaz. Sürekli rassal değişkenlerle ilgili olasılıkları belirlemek için alan kavramı kullanılır. Bu alanların hesaplanabilmesi için olasılık yoğunluk fonksiyonu veya olasılık dağılım eğrisinden yararlanılır. Değerlendirmeye alınan her bir sürekli rassal değişken için olasılık dağılım eğrisinin şekli değişir [62,63].

Kesikli rassal değişkenler için kullanılan olasılık dağılım fonksiyonunda olduğu gibi sürekli rassal değişkenler için kullanılan olasılık yoğunluk fonksiyonunun da belirli özellikleri sağlaması gerekir. Bu özellikler;

X sürekli rassal değişkeninin olasılık yoğunluk fonksiyonu *f(x)* olsun. Bu durumda,

- Her x için $f(x) \geq 0$'dır.
- *f(x)* eğrisi altında kalan ve x-ekseniyle sınırlandırılmış alan 1'e eşittir.

Sürekli rassal değişkenleri modellemek için kullanılan sürekli olasılık dağılımlarından bazıları; Normal dağılım, Standart normal dağılım, Düzgün dağılım, Üstel dağılım, Beta dağılımı, Gamma dağılımı ve Dirichlet dağılımı olarak sıralanabilir [62,63].

Olasılıksal konu modelleme, bir metin koleksiyonunda bulunan gizli anlamsal yapıları keşfetmek için yaygın olarak kullanılan bir istatistiksel modelleme türüdür [64]. Olasılıksal konu modelleme yaklaşımı, metin dokümanlarında geçen kelimeler üzerinde bir olasılık dağılım yüzdesine sahip konuların rassal olarak bir araya gelerek dokümanları oluşturduğu esasına dayanmaktadır [64,65]. Sezgisel olarak bir belgenin belirli bir konuyla ilgili olması nedeniyle belgede belirli kelimelerin daha sıklıkla görülmesi beklenir [65].

Konu modelleme yaklaşımları tarafından keşfedilen konular, aslında anlamca birbirine yakın veya birlikte sıklıkla kullanılan kelimelerin oluşturduğu anlamsal kümelerdir [66]. Olasılıksal konu modelleme de yapılan tüm hesaplamalar, dokümanlar dizisinin incelenmesine, her birinin hesaplanan istatistiklerine, her belgedeki konuların neler olabileceğine ve her bir belgenin konu dağılımının yüzdesinin ne olacağına karar veren işlemler ve süreçler dizisini içerir [64-66].

Olasılıksal konu modelleme yaklaşımı üç temel varsayıma dayanır. Bu üç temel varsayım konu modellemeye dayalı analizlerin temel araştırma hedeflerini oluşturur. Bu üç temel varsayım:

- Dokümanlar, gizli anlamsal yapıları (topic) içerirler ve bu anlamsal temalar dokümanlarda bağımsız olarak dağılırlar. Buna bağlı olarak her bir doküman birden fazla anlamsal konu içerebilir.

- Bir dokümandaki anlamsal yapılar, o dokümandaki kelimelerin görülme sıklığına bağlı olarak değişir. Her dokümandaki kelimelerin dağılımı birbirinden bağımsızdır.

- Her dokümandaki kelimelerin dağılımı, aynı zamanda bu dokümanlardaki konuların dağılımını belirler. Hangi konuya ait kelimeler yoğunluktuysa ilgili doküman yüksek oranda o konuyla ilgilidir. Şöyle ki, kelimelerin dağılımı konuları, konuların dağılımı da dokümanı modeller.

Olasılıksal konu modelleme için önerilen yordamlar, istatistiksel yaklaşımlar olup dokümanı meydana getiren kelimeleri analiz ederek bir sonuca varmayı amaçlar ve bu süreçte herhangi bir etiketleme adımına veya eğitim setine ihtiyaç duymazlar [64-66]. Burada konu (topic) ile ifade edilmek istenen kavram, bir dokümanda tartışılan ana temalardır ya da temel konulardır yani dokümanın değindiği ana temalardır. “Okunduğunuz bir kitap hangi konulara değinmiş?”, “Türkiye’yi tanıtan bir yazı hangi yönleriyle ülkeyi tanıtmış?”, “Otizmle ilgili yapılan araştırmalar daha çok hangi problemleri ele almaktadır?”, “Twitter’da bir resme yorum yapanlar, resmin hangi özelliklerine en çok yorum yapmışlardır?”, “Suriye göçmenleri ile ilgili tartışmalar hangi başlıklar altında yoğunlaşıyor?” şeklindeki soruları cevaplayabilmek için yapılacak analizler olasılıksal konu modellemenin kapsamına giren çalışmalardır.

Bu yönüyle konu modelleme yaklaşımı anlamsal analizlerde, bilgi çıkarımında ve içerik modelleme uygulamalarında sıklıkla başvurulan bir yöntemdir. İçinde bulunduğumuz bilgi ve iletişim çağında paylaşılan veri miktarı ve veri çeşitliliği her geçen gün katlanarak büyümektedir. Olasılıksal konu modelleme, yapılandırılmamış büyük metin koleksiyonlarının anlamsal ve istatistiksel olarak analiz edilmesinde önemli bir rol üstlenmektedir. Başlangıçta bir metin analiz aracı olarak geliştirilen konu modelleme

algoritmaları, günümüzde genetik bilgi, görüntüler, resimler, videolar ve sosyal ağlar gibi farklı türdeki veriler üzerinde de uygulanmaktadır [65]. Bahsedilen konu modelleme ve eğilim belirleme araştırmalarında kullanılan konu modelleme algoritmaları arasında en yaygın olanları Gizli Anlamsal Analiz (GAA), Olasılıksal Gizli Anlamsal Analiz (OGAA) ve Gizli Dirichlet Ataması (GDA) yaklaşımlarıdır.

1.8.1. Gizli Anlamsal Analiz

Gizli anlamsal indeksleme (Latent Semantic Indexing- LSI) olarak da bilinen Gizli Anlamsal Analiz (GAA), belgelerin temel anlamını veya içerdiği ana temaları bulmak için belgelerin anlamsal olarak analiz edilmesi anlamına gelir. Her sözcük yalnızca bir anlam içeriyorsa ve her anlam yalnızca bir sözcük tarafından karşılanıyorsa, sözcüklerden anlamlara basit bir eşleme olacağından GAA'nın uygulanması kolaydır. Fakat metin analizinde işler her zaman bu kadar kolay değildir. Çünkü doğal dil işlemede çok anlamlı ve eşanlamlı kelimeler işin içine girdiğinde metin analiz süreci farklı bir boyut kazanmakta ve bu noktada metinlerin çözümlenmesi ve kümelenmesi oldukça zor bir probleme dönüşmektedir [67].

GAA'da bir belge, vektör uzayındaki kelimelerin bir vektörü olarak temsil edilir, benzer şekilde diğer belgelerde aynı vektör uzayındaki farklı kelime vektörleriyle temsil edilirler. GAA yaklaşımı, orijinal herhangi bir X' terim-doküman matrisini tekil değer ayrıştırma (singular singular value decomposition-SVD) işlemi vasıtasıyla filtrelenmiş (boyutu indirgenmiş) bir terim-doküman matrisine indirgemektedir. Tekil değer ayrıştırmasının sağladığı temel fayda terim-doküman matrisinin anlamsal ilişkileri daha net vurgulayan ve gürültüyü azaltan daha düşük boyutlu bir matrise dönüştürmesidir [67].

Vektör uzayındaki belgelerin temsilini sağlayan terim-doküman matrisinin daha düşük boyuta indirgenebilmesi için tekil değer ayrıştırma sürecinde, X' terim-doküman matrisi ilk olarak üç özgün matrise ayrıştırılır. Şöyle ki, *t*: terim sayısı, *d*:doküman sayısı ve $r \leq \min(t,d)$, r: X' terim-doküman matrisinin rankı olmak üzere, X terim-doküman matrisinin tekil değer ayrışımı aşağıdaki gibi ifade edilir:

$$X = U \Sigma V^T \tag{4}$$

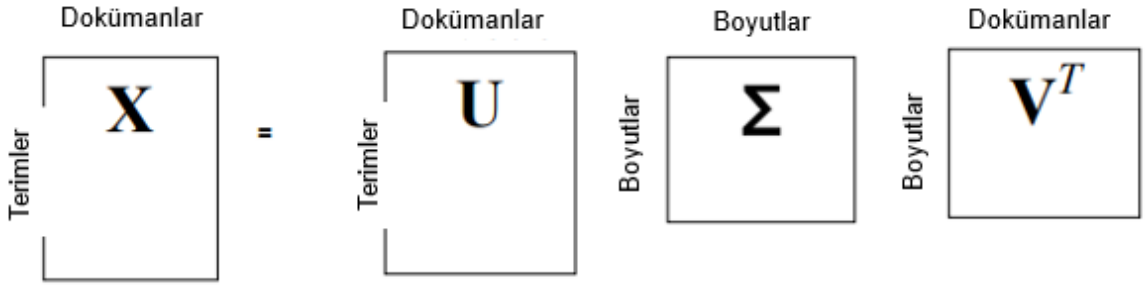
U : XX^T ' in öz vektörlerinin [t x r] boyutlu simetrik kare matrisidir.

$\sum$: U ve V' matrislerinin öz değerlerinin kareköklerini içeren [r x r] boyutlu köşegen matristir.

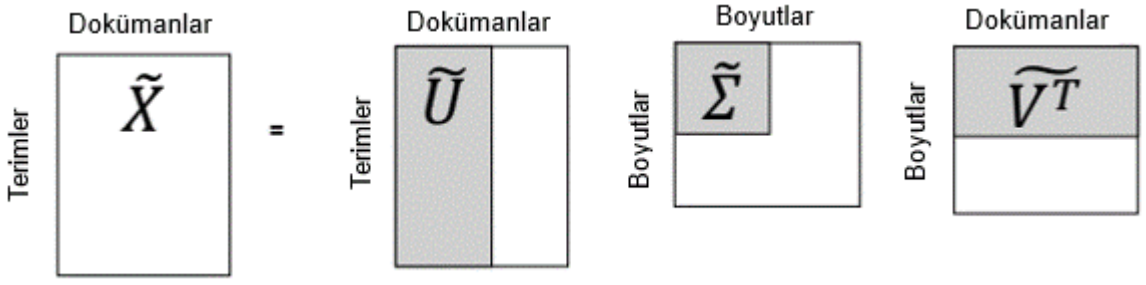
V : X^TX ' in öz vektörlerinin [d x r] boyutlu simetrik kare matrisidir.

Herhangi bir A matrisinin öz değerleri $Det(A-\lambda I) = 0$ eşitliği hesaplanır. Aynı A matrisinin öz değerlerine karşılık gelen öz vektörleri hesaplamak için, bulunan öz değerler tek tek $(A-I)x = 0$ denklem sisteminde yerine konulup, elde edilen yeni denklem sistemleri çözülerek öz değer sayısında öz vektör elde edilir.

Şekil 4'de gösterildiği üzere GAA metoduna giren X matrisi, tekil değer ayrışımı ile üç matrise ayrıştırılır. GAA yönteminden çıkan X matrisi Şekil 5'de gösterildiği gibi daha küçük boyutlu bir matrise indirgenmiş olur.



Şekil 4. Orijinal X' matrisinin tekil değer ayrışımı



Şekil 5. İndirgenmiş X' matrisinin tekil değer ayrışımı

GAA, konu modelleme de kullanılan bir yöntem olmasına rağmen, etkin sonuçların alınabilmesi için çalışılan veri seti açısından bazı koşulların analiz için uygun olması gerekmektedir [67]. Bu koşullar;

- Dokümanlar aynı yapıda ve aynı yazı sitilinde olması gerekir.
- Her bir doküman tek bir konuya odaklı olmalıdır.
- Her sözcük yüksek olasılıkla tek bir konuya ait olmalı ve diğer konulara ait olması olasılığı düşük olmalıdır.

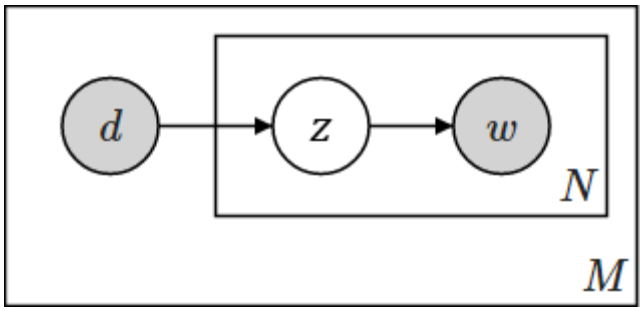
GAA, eşanlamlı ve çok anlamlı yapılar açısından vektör uzay modelini ileriye taşıyan bir yaklaşım olsa da, bazı sınırlamalar ve dezavantajlar içermektedir: Birincisi, bu yaklaşım basit bir matris boyut indirgeme işlemidir ve güçlü olasılıksal arka plana sahip değildir. İkincisi, faktörlerin (konuların) sayısı istatistiksel olarak belirlenemez ve faktör sayılarının en uygun şekilde belirlenmesi araştırmacının bakış açısına bağlıdır. Üçüncüsü, dikey eliminasyon özelliği nedeniyle GAA farklı konularda bir kelimenin birden çok kez ortaya çıkmasını önleme eğilimindedir ve bu nedenle çok anlamlılık problemlerini çözmek için GAA etkili bir yöntem değildir. Bu noktada, basit matris indirgeme ve çok anlamlılık sorununun üstesinden gelmek için üretken modeller geliştirilmiş ve bu yaklaşımın daha gelişmiş ve yenilikçi bir üst modeli olan olasılıksal gizli anlamsal analiz yöntemi önerilmiştir [67].

1.8.2. Olasılıksal Gizli Anlamsal Analiz

Metin madenciliğinde ve doğal dil işleme uygulamalarında sözcüklerin veya sözcük derlemlerinin değiştirilebilir olmasından dolayı, sözcüklerin varlığı ve görülme sıklığı olasılık teorisi ile modellenebilir. Olasılık teorisine dayalı modelde, oluşturulan bir terim-doküman matrisinin indirgenmesinden ziyade üretken model için önce bir yaklaşım belirlenir ve daha sonra, oluşturulan terim-doküman matrisi üzerinde olasılıksal modelin bilinmeyen parametreleri tahmin edilir. Bu nedenle, olasılık teorisine dayalı üretken modeller arasındaki en büyük fark, terim doküman matrisi oluşturulduktan sonra uygulanan modelin bilinmeyen parametrelerinin tahmin edilmesinde kullanılan yöntemlerdir [66,68].

Olasılıksal Gizli Anlamsal Analiz (OGAA), dokümanların üç ardışık aşamada oluşturulduğunu varsaymaktadır. İlk olarak, bir *d* dokümanı *P(d)* olasılığı ile seçilir. İkinci olarak, *z* etiketli konu (topik), *P(z | d)* olasılığı ile seçilir. Üçüncü olarak, bir konudaki her *w* kelimesi *P (w | z)* olasılığı ile üretilir. Bu süreç boyunca, OGAA modelinde verinin

rassal değişkenler olarak yönlendirilmiş doğrular boyunca hangi süreçlerden geçtiği Şekil 6'da ifade edilmiştir [68].



Şekil 6. OGAA modelinin grafiksel gösterimi

Bir üretken süreç iki varsayım üzerine kuruludur. Birincisi, oluşturulan sözcük-doküman çiftleri bağımsız olarak üretilir. İkincisi, *z* konusuna atanan kelimeler belirli bir *d* dokümanından bağımsızdır. Sonuç olarak, üç aşama birleştirilirse, *d* dokümanında, w kelimesinin bulunması olasılığı, aşağıda verilen 5 veya 6 numaralı eşitliklerle matematiksel olarak ifade edilir.

$$P(d \mid wn) = P(d)\, \Sigma z\, P(wn \mid z)\, P(z \mid d) \quad (5)$$

$$P(d \mid wn) = \Sigma z\, P(z)\, P(wn \mid z)\, P(z \mid d) \quad (6)$$

OGAA'nın matematiksel gösteriminde üçlü çarpım şeklinde gösterilen her bir parça GAA'nın üç parçasına yani doküman öz vektörlerine, öz değerlere ve kelime öz vektörlerine karşılık gelir. Bu eşleştirmeler aşağıdaki gibi ifade edilir:

$P(d \mid z) \rightarrow V$ (Doküman öz vektörleri) (7)

$P(z) \rightarrow \Sigma$ (Öz değerler) (8)

$P(w \mid z) \rightarrow$ U (Kelime öz vektörleri) (9)

Bu şekilde değerlerin yorumlanmasıyla GAA'daki parametrelerin tahmin edilen değerleri, OGAA'daki kelimelerin olasılıklarını belirtir ancak GAA'daki değerler olasılık anlamı olmayan faktörlerin yükleme değerlerini belirtir [68].

GAA'daki faktörlere benzer olarak OGAA'dan üretilen konular, kelimeler arasındaki eş anlamlılık ve çokanlamlılık ilişkilerini ortaya koyar. Genel olarak üretken modelde en yüksek marjinal olasılığa sahip kelimeler ilk seferinde bir konuya atanır ve sonra daha düşük marjinal olasılıklara sahip kelimeler bu konulara atanır. Bu nedenle başlık kelimeleri gibi genel olarak kullanılan kelimelerin bir konudaki olasılıklarının yüksek olması muhtemeldir. Bu özellik, üretken modellerin belgelerdeki benzersiz eğilimler yerine genel eğilimleri yakalamasına olanak tanır [66,68].

Kelimelerin eş anlamlarına dayanarak OGAA kelimelerin benzer anlamlara sahip olduğu farklı konuları ortaya koyabilir, buna karşın GAA kelimelerin anlamsal yapısını dikkate almaksızın bir faktörde yüksek sayıda geçen kelimeleri ortaya koyar. Kelimelerin çok anlamlı olmasına dayanarak OGAA'da aynı kelimeler aynı anda farklı konularda ve yüksek olasılıklarla ortaya çıkabilir. Bununla birlikte, OGAA'da dokümanların oluşturulması olasılıksal olarak modellemediğinden, *P(d)* kısmi üretken bir modeldir. Bundan dolayı doküman düzeyinde konuları tespit etmek ve üretken modeli daha etkin bir hale getirmek için GDA modeli önerilmiştir [64-66,68].

1.8.3. Gizli Dirichlet Ataması

Metin madenciliği uygulamalarında etkin olarak kullanılan konu modelleme yöntemlerinden birisi olan Gizli Dirichlet Ataması (GDA), (Latent Dirichlet Allocation-LDA), metin dokümanları gibi ayrık verileri anlamsal olarak analiz etmek ve dokümanları meydana getiren anlamsal temaları ortaya çıkarmak için kullanılan üretken ve olasılıksal bir yöntemdir [64,65,69]. Yöntemdeki "gizli" terimi, dokümanı oluşturan gizli anlamsal yapıları, yani içerikleri analiz ederek dokümanın anlamsal içeriğini bulmayı ifade etmektedir [65,69]. Üretken model (generative) ile kastedilen ise dirichlet dağılımına dayanan tekrarlı (iterative) bir olasılıksal süreç ile dokümanlardaki kelimelerin gizli (rassal) değişkenler olarak üretilmesiyle dokümanların anlamsal olarak modellenmesidir [65]. Dirichlet dağılımı, sonlu sayıdaki rassal değişkeni tanımlayan olasılık yoğunluk fonksiyonunu modellemenin bir yoludur. Dirichlet dağılımı, 19. yüzyılda yaşamış Belçikalı matematikçi Johann Dirichlet' ten adını almıştır.

GDA modelinde metin belgeleri konuların birleşimi olarak temsil edilmektedir. Bu temsil yapısı, konuların sözcükler üzerinde bir olasılık dağılımına sahip olduğu ve metin belgelerinin de konular üzerinde bir olasılık dağılımına sahip olduğu varsayımına dayanmaktadır. Her bir konu ise sabit bir kelime seti üzerinde bir olasılık dağılımı olarak modellenmektedir [69]. Örnek olarak, aşağıdaki belgeler kümesine sahip olduğunuzu varsayalım:

- Belge1: Ayşe balık[1] ve sebze[1] yer[1].
- Belge2: *Balık*[2] *evcil*[2] bir hayvandır.
- Belge3: Benim *kedim*[2] balık[1] yer[1].

Yukarıdaki üç belge göz önüne alındığında, "yemek" olarak etiketleyebileceğimiz Konu1 kümesi altında altı çizili kelimeleri (balık, sebze, yer) sınıflandıralım. Benzer şekilde italik yazılan kelimeleri (balık, evcil, kedi) "evcil hayvan" olarak etiketleyebileceğimiz ayrı bir Konu2 kümesi altında sınıflandıralım. GDA, bu belgelerin içerdiği konuları otomatik olarak keşfeden bir tekniktir. GDA, belgenin içerdiği her konuyu birer kelime grubu olarak tanımlar ve bu konular atanan kelimelerin dağılımına uygun olarak etiketlenir. GDA modelinin keşfedilen konuları kelime düzeyinde tanımlamasının 2 temel avantajı vardır:

1. Her bir belgenin içeriğini kelime sayımı ile çıkarabiliriz: Belge1: % 100 Konu1, Belge2: % 100 Konu2 ve Belge3: %33 Konu1 ve %67 Konu2'yi içermektedir.

2. Her kelimenin keşfedilen konulardaki geçiş oranlarını çıkarabiliriz. Örneğin, Konu1 kümesini oluşturan kelimeler aşağıdaki değerlerde görülme oranına sahiptir: balık (%40), yer (%40), sebze (%20).

GDA için üretken süreç yaklaşımı, bir doküman koleksiyonundaki konu dağılımını (yukarıdaki örneğe benzer şekilde) modellemek için aşağıdaki üç adımı sırasıyla icra eder [69]. Bu adımlar;

1.Adım: İlk olarak, analizin kapsamına bağlı olarak hedeflenen konu sayısı belirlenir. Daha önceden öğrenilmiş bir tahminde bulunabilir (örnek: bir önceki analizin sonuçları) veya sadece deneme-yanılma yoluyla konu sayısı belirlenebilir. Farklı tahminlerle

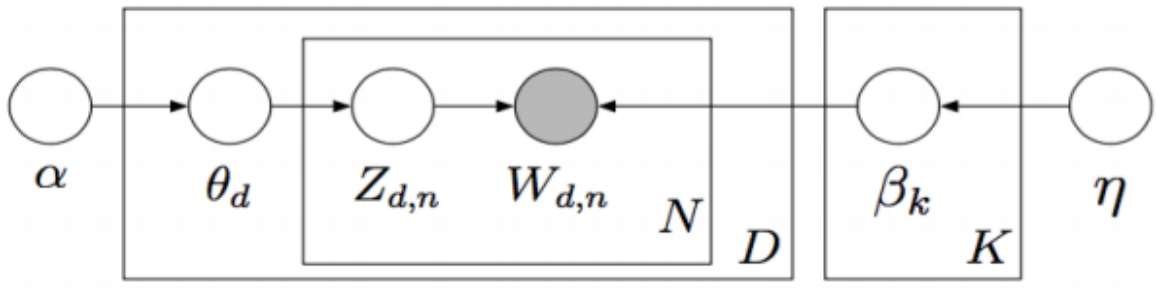
çalışırken istenilen anlamlılık düzeyini veya en yüksek istatistiksel kesinliği veren konu sayısı seçilebilir [69].

2.Adım: GDA her kelimeyi geçici bir konuya atar. Konu atamaları 3. Adımda güncellenecekleri için geçicidir. Geçici konular her kelimeye yarı-rastgele şekilde atanır. Bu durum, bir kelime iki kez görünürse her kelime farklı konulara atanabilir demektir [69].

3.Adım (Tekrarlı yaklaşım): GDA her belgedeki her bir kelime için konu atamalarını kontrol eder ve atamaları günceller. Her kelime için konu ataması iki kıstasa göre güncellenir [69]:

1. Konularda bu kelime ne kadar yaygın?
2. Belgede bu konular ne kadar yaygın?

GDA için üretken model özetlenecek olursa her doküman konuların rassal olarak karışımından meydana gelmekte ve dokümanı oluşturan kelimelerin her biri de konuların bir tanesinden seçilmektedir. Üretken süreç ilk olarak derlemi oluşturan sözlükteki kelimelerin konulara atanması ile başlar. Bir sonraki adımda, her bir doküman için konuların o dokümanda bulunma olasılığına bakılır ve konuların doküman içerisindeki dağılımı belirlenir. Son adımda, ilgili dokümanda yer alan her kelime için o kelimenin ataması yapılan konuda ne kadar yaygın olduğuna ve ataması yapılan konunun o belgede ne kadar yaygın olduğuna bakılarak atamalar güncellenir. Son adımdaki işlem derlemdeki tüm dokümanlar için tekrarlanır [69]. GDA modelinde verinin rassal değişkenler olarak yönlendirilmiş doğrular boyunca hangi süreçlerden geçtiği Şekil 7'de verilmiştir [69].



Şekil 7. GDA modelinin grafiksel gösterimi

Şekil 7'de gösterilen ve olasılık dağılımında kullanılan tüm parametreler ve açıklamaları Tablo 1'de verilmiştir.

Tablo 1. GDA için model parametreleri ve açıklamaları

Parametre	Açıklaması
D	Toplam doküman sayısı
K	Toplam gizli konu sayısı
V	Sözlükte bulunan toplam kelime sayısı
N_d	d. dokümandaki kelime sayısı
α	Doküman başına düşen konu dağılımları için önsel Dirichlet parametresi
η	Konu başına düşen kelime dağılımları için önsel Dirichlet parametresi
θ_d	Konuların d. dokümandaki dağılımı
β_k	Kelimelerin k. konudaki dağılımı
$z_{d,n}$	d. dokümandaki n. konumda bulunan kelimenin atandığı konu
$w_{d,n}$	d. dokümandaki n. konumda gözlemlenen kelime

Grafiksel gösterimde dokümanlardan konuların çıkarılması sürecinde elimizde sadece gözlenen değişken olarak kelimelerin dokümanlardaki dağılımı bilinmektedir. Bunun dışında kelimelerin konulara atanması, konuların dokümana atanması ve kelimelerin konulardaki dağılımları gizli dağılımlardır. Bu nedenle grafiksel gösterimde gözlenen değişkenler gri renkle gözlemlenemeyen gizli dağılımlar ise beyaz renkle gösterilmiştir.

Yukarıda açıklanan GDA modelinin grafiksel gösterimine göre ifade edilen tüm gözlemlenen ve gizli rassal değişkenlerin birleşik dağılımı matematiksel olarak tanımlanmış ve Eşitlik 10'da verilmiştir [69].

$$p(\beta_{1:K}, \theta_{1:D}, z_{1:D}, w_{1:D}\,;\,\eta, \alpha) = \prod_{i=1}^{K} p\,(\beta_i|\eta) \prod_{d=1}^{D} p\,(\theta_d|\alpha) \prod_{n=1}^{N} p\left(z_{d,n}|\theta_d\right)\; p\left(w_{d,n}|\beta_{1:K,} z_{d,n}\right) \tag{10}$$

Yukarıda verilen birleşik olasılık dağılımı $p(\beta, \theta, z, | w)$ çıkarımı için bir sonsal (posterior) dağılımı tanımlar. Daha farklı bir ifadeyle, GDA modelinin birleşik dağılımı *p(konular, oranlar, atamalar | kelimeler)* olasılık dağılımının sonsal dağılımı olarak

görülebilir. GDA modeli için tanımlanan birleşik olasılık dağılımın da ifade edilen konular (β), oranlar (θ) ve atamalar (z) için olasılık dağılımlarının hesaplanmasında Dirichlet dağılımı kullanılır. Dirichlet dağılımı, olasılık ve istatistikte sıklıkla kullanılan çok değişkenli sürekli olasılık dağılımlarından bir tanesi olup, α (pozitif reel sayılardan oluşan bir vektör) parametresi ile birlikte tanımlanır. Dirichlet dağılımı, beta dağılımının çok değişkenli genellemesidir ve çok terimli (multinomial) dağılım için de eşlenik önsel dağılım olarak ifade edilir [69,70].

Beta dağılımı;

$$p(p \mid \alpha, \beta) = \frac{1}{B(\alpha, \beta)} p^{\alpha-1}(1-p)^{\beta-1} \tag{11}$$

Çok terimli dağılım;

$$P(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{n!}{\prod_{i=1}^{d} x_i!} \prod_{i=1}^{d} \theta_i^{x_i}, \qquad n = \sum_{i=1}^{d} x_i, \quad \sum_{i=1}^{d} \theta_i = 1, \quad \theta_i \geq 0 \tag{12}$$

Dirichlet dağılımı;

$$p(\boldsymbol{x}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{i=1}^{d} \alpha_i)}{\prod_{i=1}^{d} \Gamma(\alpha_i)} \prod_{i=1}^{d} x_i^{\alpha_i - 1}; \quad \sum_{i=1}^{d} x_i = 1, \qquad x_i \geq 0 \tag{13}$$

Dirichlet dağılımının modellenmesinde bir veri noktasının çok terimli bir dağılıma sahip olması ve dağılım parametresinin önsel (prior) dağılımı (veri noktasını üreten olasılıkların vektörü) bir Dirichlet dağılımı ise, parametrenin sonsal (posterior) dağılımı da bir Dirichlet dağılımı olur. Bahsedilen olasılık dağılımları arasındaki bu ilişkilendirmeler Tablo 2'de bağıntı olarak verilmiştir. Sezgisel olarak böyle bir durumda veri noktasını gözlemlemeden önce değişken hakkında bildiklerimizden yola çıkarak bilgimizi veri noktasına dayalı olarak güncelleyebiliriz. Bu süreç, karmaşık matematiksel hesaplamalarla uğraşmadan bir defada bir tane gözlem ekleyerek bir parametrenin (değişkenin) bilgisini art arda güncelleyebileceğimiz anlamına gelmektedir [69,70].

Tablo 2. Olasılık dağılımlarının önsel ve sonsal eşlenikleri

Olasılık (likelihood) $p(y\|x)$	Önsel dağılım (prior) $p(x)$	Sonsal dağılım (posterior) $p(x\|y)$
Binomial (iki terimli) *Bin (n,x)*	Beta *Beta (α,β)*	Beta *Beta (α+y, β+n-y)*
Multinomial (çok terimli) $M_k\ (x_1, \ldots \ldots x_k)$	Dirichlet $D\ (\alpha_1, \ldots \ldots\ \alpha_k)$	Dirichlet $D\ (\alpha_1+y_1, \ldots \ldots\ \alpha_k+y_k)$

Yukarıda bahsedilen ve Tablo 2'de verilen süreç istatistikte Bayes çıkarımı olarak bilinir. Bayes çıkarımı, istatistikte ve özellikle matematiksel istatistikte sıkılıkla başvurulan önemli bir tekniktir ve bir dizi şeklindeki verilerin analizinde etkin olarak kullanılan bir yöntemdir. Bayes çıkarımı sonsal (posterior) olasılığı, önceden tespit edilmiş iki olasılığın bir sonucu olarak ve gözlemlenen veriler için istatistiksel bir modelden türetilen bir "olasılık fonksiyonu" olarak türetir. Bayes çıkarımı, Bayes teoremine göre sonsal (posterior) olasılığı hesaplar (Eşitlik 14) [69].

$$sonsal\ d.(posterior) = \frac{olasılık(likelihood)\ X\ önsel\ dağılım(prior)}{kanıt(evidence)} \tag{14}$$

Stokastik (rassal) bir süreç sırasında ortaya çıkan bir rassal *x* olayı ile bir diğer rassal *y* olayının (eğer *y* için kaybolmamış olasılık varsa) koşullu olasılıkları arasındaki ilişki Bayes teoremi (çıkarımı) olarak bilinir ve Eşitlik 15'de verildiği gibi ifade edilir. [62,69].

$$p(x|y) = \frac{p(y|x).p(x)}{p(y)} \tag{15}$$

Bayes teoreminde yer alan her bir terim için özel isimlendirmeler kullanılır:

x : Bilinmeyen değişkenler

y : Gözlem verisi

p(x|y) : Değişkenlerin sonsal dağılımı

p(x) : Değişkenlerin önsel dağılımı

p(y|x) : Olasılık fonksiyonu

p(y) : Delil (normalizasyon sabiti, kararı etkilemez)

GDA'da model parametrelerinin kestirimi için örnekleme yöntemlerinden yararlanılmaktadır. Bu işlem için yaygın olarak kullanılan Gibbs örneklemesi, doğrudan örneklemenin zor olduğu durumlarda daha önceden belirlenmiş çok değişkenli bir olasılık dağılımından yararlanarak yaklaşık bir gözlem dizisi elde etmek için kullanılan bir Monte Carlo Markov zinciri (MCMC) algoritmasıdır. Tipik olarak bazı değişkenler değerleri bilinen gözlemlere karşılık gelir ve bu nedenle örnekleme alınması gerekmez [70,71]. Gibbs örneklemesi bir istatistiksel örnekleme aracı olarak Bayes çıkarımında sıklıkla kullanılır. Rassal bir algoritma (yani, rastgele sayılar kullanan bir algoritma) olan Gibbs örneklemesi, diğer MCMC algoritmalarında olduğu gibi her biri yakındaki örneklerle ilişkilendirilen bir Markov zincir modeli üretir [70,71].

GDA modelinde, parametre kestirimi için bizi ilgilendiren temel değişkenler β_k (kelimelerin k. konudaki dağılımı) ve $\theta_{d,k}$ (k. konunun d. dokümandaki oranı) değişkenleridir. Gibbs örnekleme yöntemin de sık kullanılan bir teknik gözlemlenen sözcükler göz önüne alındığında sözcük-konu ataması için sonsal dağılımının (posterior) doğrudan (β ve θ değişkenleri dışarda tutularak) tahmin edilmesidir. Bu yöntem daraltılmış (collapsed) Gibbs örneklemesi olarak adlandırılır. Daraltılmış Gibbs örnekleme yönteminde β ve θ dışlanıp, sadece $z_{d,n}$ (d. dokümandaki n. konumda bulunan kelimenin atandığı konu) değişkeni üzerinden örnekleme gerçekleştirilir. Daraltılmış Gibbs örneklemesi ile belli bir kelimenin hangi konuya atanacağı Eşitlik 16' dan yararlanılarak bulunmaktadır [69-71].

$$P(z_i = j|\mathbf{z}_{-i}, w_i, d_i, \cdot) \propto \frac{C_{w_ij}^{WT} + \eta}{\sum_{w=1}^{W} C_{wj}^{WT} + W\eta} \frac{C_{d_ij}^{DT} + \alpha}{\sum_{t=1}^{T} C_{d_it}^{DT} + T\alpha} \quad (16)$$

z_i : i. kelimenin atandığı konu

d_i : i. kelimeyi içeren doküman

w_i : i. sıradaki kelime

z_{-i} : diğer tüm kelimelerin konu atamaları dizisidir

(.) : α ve η parametreleri gibi belirtilmeyen tüm parametreler

C^{WT} : kelime-konu (word-topic) için sayım matrisi

C^{DT} : doküman-konu (document-topic) için sayım matrisi

Bu işlemler doküman koleksiyonunda bulunan tüm dokümanların içerdiği tüm kelimeler için yapıldıktan sonra sayım matrisleri Eşitlik 16' da olduğu gibi kullanılarak, β_{ij} (i. kelimenin j. konuya atanması olasılığı) ve θ_{dj} (j. konunun d. dokümanda bulunması olasılığı) değişkenlerinin güncellemesi Eşitlik 17 ve 18'e göre gerçekleştirilir [69-71].

$$\beta_{ij} = \frac{C_{ij}^{WT} + \eta}{\sum_{k=1}^{W} C_{kj}^{WT} + W\eta} \tag{17}$$

$$\theta_{dj} = \frac{C_{dj}^{DT} + \alpha}{\sum_{k=1}^{T} C_{dk}^{DT} + T\alpha} \tag{18}$$

Tamamen denetimsiz bir yaklaşım olan GDA herhangi bir önbilgiye ihtiyaç duymamakla birlikte kelime torbası yaklaşımına dayalı çalışmaktadır. Bu modelde kelimelerin doküman içerisindeki yerleşimi göz ardı edilirken kelimelerin birlikte bulunması varsayımı temel alınmaktadır. GDA modelinde herhangi bir eğitim setine ihtiyaç duyulmadığı için çok büyük veri yığınlarının modellemeye dayalı analizleri, denetimli yöntemlere kıyasla daha kısa sürede gerçekleştirilir. Bu nedenle metin odaklı büyük veri setlerinin analizinde önerilen bir yaklaşımdır [65,69-71].

1.9. Literatür Çalışması

Olasılıksal konu modellemeye dayalı metin analizlerine yönelik bilimsel araştırma ve uygulamalar son yıllarda büyük bir artış göstermiştir. Konu modelleme analizleri, farklı çevrimiçi platformlarda ve sosyal ağlarda paylaşılan metin odaklı veri miktarındaki artışa bağlı olarak her geçen gün bu alandaki araştırmacıların daha fazla ilgisini çeken bir çalışma konusu olmuştur. Olasılıksal konu modellemeye dayalı çalışmalar geniş uygulama alanlarına sahiptir. Literatürde yapılan çalışmalar incelendiğinde farklı türde verilerin modellenmesine yönelik olarak gerçekleştirilen farklı içerikte çalışmalar göze çarpmaktadır. Bu bağlamda, yapılan çalışmanın amaç ve kapsamını daha anlaşılabilir bir şekilde ortaya koymak için konu modelleme analizlerine ve yazılım geliştirme eğilimlerinin belirlenmesine yönelik çalışmalar irdelenmiş ve yorumlanmıştır. Gerçekleştirilen literatür araştırması kapsamında birinci kısımda yazılım geliştirme

eğilimlerinin belirlenmesine yönelik çalışmalara, ikinci kısımda ise olasılıksal konu modellemeye dayalı olarak gerçekleştirilen çalışmalara yer verilmiştir.

1.9.1. Yazılım Geliştirme Eğilimlerine Yönelik Literatür Değerlendirmesi

Bilgi sistemlerine dayalı dördüncü sanayi devriminin yaşandığı son 10-15 yıl içerisinde hızla değişen bilgi teknolojileri, yazılım geliştirme eğilimleri ve bu alanlarda talep edilen uzman bilgi ve becerileri birçok bilimsel çalışmaya konu olmuştur. Özellikle son yıllarda iş ilanlarının analizine dayanan birçok çalışmada teknoloji odaklı iş ilanlarının farklı yordamlarla analizleri gerçekleştirilmiştir. Bu çalışmalarda genel olarak bilişim sistemleri ve yazılım geliştirme alanlarında ihtiyaç duyulan uzman bilgi ve becerilerinin belirlenmesi amaçlanmıştır [1]. Bunların dışında, bilgi teknolojilerindeki eğilimlerin belirlenmesine yönelik olarak farklı veri setleri üzerinde gerçekleştirilen çalışmalar da literatürde yer almaktadır [1,35,65].

Aken ve diğerleri [15], farklı çevrimiçi iş platformlarından topladıkları bilişim odaklı iş ilanlarının metinleri üzerinde gerçekleştirdikleri çalışmada K-means algoritmasına dayalı bir kümeleme analizi gerçekleştirmişlerdir. Bu çalışma ile araştırmacılar, bilgi sistemleri ve yazılım geliştirme alanlarında 20 farklı uzmanlık rolü ve bu rollere ilişkin bilgi ve becerileri ortaya koymuşlardır. Bu roller beş farklı uzmanlık sınıf altında (web uzmanlığı, yazılım uzmanlığı, veri tabanı uzmanlığı, yönetim uzmanlığı ve proje geliştirme uzmanlığı) gruplandırılarak sunulmuştur.

Prabhakar ve diğerleri [14], bilgi teknolojileri alanlarındaki iş ilanları üzerinde manuel sorgulamaya dayalı metin içerik analizi yordamıyla bir deneysel çalışma gerçekleştirmişlerdir. Bu çalışmanın sonucunda araştırmacılar, web programlama, Unix, C++, Java, SQL programlama ve Oracle veri tabanı gibi önemli uzmanlık becerilerine vurgu yapmışlar ve ayrıca bu alanlarda giderek artan taleplere ve nitelikli işgücü ihtiyacına dikkat çekmişlerdir.

Smith ve Ali [16], iş ilanları üzerinde anahtar kelime indeksine dayalı bir içerik analizi gerçekleştirmişlerdir. Bu çalışmada araştırmacılar, programlama dilleri üzerinde gelişen talep ve eğilimleri analiz ederek programlama dillerinin zaman içerisindeki eğilimlerini ortaya koymuşlardır. Ayrıca programlama dilleri arasında Java ve veri tabanı

olarak da SQL ve Oracle'ın gelecekte baskın teknolojiler olacağına vurgu yapmışlardır. Bu çalışma da elde bulgular araştırmacıların çalıştığı ilgili akademik birimdeki eğitim müfredatının piyasa taleplerine duyarlı bir şekilde planlanması amacıyla kullanılmıştır.

Harris ve diğerleri [72], 1970'li ve 2011'li yılların bilgi sistemleri iş piyasasını değerlendirmek için bir çalışma yapmışlardır. Bu kapsamda araştırmacılar Atlanta, Chicago, Los Angeles, Nashville ve New York'u içeren beş büyük şehirdeki bilgi sistemleri alanındaki çevrimiçi ilanlarını inceleyerek bu alanlardaki talep ve eğilimleri zamanla değişimini ortaya koymuşlardır. Çalışmanın bulgularına göre 1978/1979'da işverenler iş ilanları başına 2.63 beceri istiyorken, 2010/2011 döneminde işverenler iş ilanları başına ortalama 7.54 beceri istedikleri görülmüştür. Bu bulgular işverenlerin yakın zamanda talep ettikleri bilgi ve becerilerin daha da arttığını ve çeşitlendiğini ortaya çıkarmıştır. Bu bulgulara dayanarak araştırmacılar, bilgi sistemleri odaklı akademik birimlerin eğitim müfredatının piyasa deneyimleri ile örtüşen bir şekilde güncellemesi gerektiğine vurgu yapmışlardır.

Huang ve diğerleri [73], BT alanında gerekli olan bilgi ve becerileri bulmak için üç farklı veri seti üzerinde (akademik makaleler, uygulayıcı yayınlar ve çevrimiçi iş ilanları) bir çalışma gerçekleştirmişlerdir. Çalışmada bu üç veri seti kullanılarak iş piyasasında talep edilen bilgi ve beceriler tespit edilmiş ve sistematik bir taksonomi ile kategorize edilmiştir. Çalışmanın sonucunda araştırmacılar, akademik makalelerin iş piyasasındaki taleplerle ilgili gelişmeleri daha geriden takip ettiklerini tespit etmişlerdir. Bunun yanında çevrimiçi iş ilanları ile daha ayrıntılı teknik beceriler ortaya çıkarılmış ve işverenlerin teknik becerilerin yanında teknik olmayan becerileri de büyük oranda talep ettikleri görülmüştür. Ayrıca araştırmacılar, uygulayıcı yayınların iş becerileri konusunda ileriye dönük bir bakış açısı kazandırma eğiliminde olduğunu ve teknik beceriler açısından da zengin bir kaynak olduğunu belirtmişlerdir. Araştırmacılar bu çalışmada çevrimiçi iş ilanlarının, hem teknik hem de teknik olmayan bilgi ve becerileri en iyi yansıtan objektif ve güncel bir kaynak olduğuna kanaat getirmişlerdir.

Litecky ve diğerleri [74], BT alanındaki teknik becerilerle teknik olmayan beceriler arasındaki farklılıkları ortaya çıkarmak amacıyla iş ilanları üzerinde bir çalışma gerçekleştirmişlerdir. Bu çalışmanın sonucunda araştırmacılar BT alanlarındaki istihdam süreci için iki aşamalı bir model ortaya atmışlardır. Bu model, ilk işe alım aşamasında

teknik becerilerin daha çok ön planda tutulduğunu ve ikinci aşamada ise işverenlerin teknik olmayan becerileri işe devam etmenin koşulları olarak gördüğünü ortaya koymuştur. Böylelikle BT alanları için teknik olamayan becerilerin teknik beceriler kadar önemli olduğuna vurgu yapılmıştır.

Gallivan ve diğerleri [75], yaptıkları çalışmada son 17 yılda BT uzmanlarına yönelik bilgi ve becerilerdeki değişen eğilimleri bulmak için BT odaklı iş ilanları üzerinde içerik analizine dayanan bir araştırma gerçekleştirmişlerdir. Bu deneysel çalışma ile araştırmacılar, BT alanında kendilerinin tespit ettikleri eğilimlerin başka araştırmacılar tarafından öngörülen gelecekteki iş ve beceri eğilimleri ile örtüşüp örtüşmediğini araştırmışlardır. Çalışmada elde bulguların birçoğu daha önceki çalışmalarla tutarlı çıkmış ve işverenlerin yeni çalışanlardan sürekli artan sayıda beceri seti aradığı fikrini desteklemiştir. Buna ek olarak pek çok firmanın teknik alan bilgisinin yanı sıra işletme bilgisi ve teknik olmayan becerilere de vurgu yapmasına rağmen, işe alma sürecinde yaygın olarak teknik alan bilgi ve becerilerin ön plana çıktığı belirtilmiştir. Araştırmacılar, BT uzmanlarına yönelik değişen istihdam taleplerinden dolayı sadece BT uzmanları değil, aynı zamanda onları yetiştiren akademisyenlerin de ömür boyu bilgi ve becerilerini güncellemesi gerektiğine vurgu yapmışlardır.

Ahmed ve diğerleri [76], BT pozisyonları için 500 iş ilanını analiz ederek ilgili iş ilanlarında talep edilen teknik olmayan becerileri araştırmışlardır. Bu çalışmanın sonucunda araştırmacılar, hangi teknik olmayan becerilerin yazılım geliştirme ortamları için yüksek oranda talep gördüğünü ve hangilerinin önemine rağmen ihmal edildiğini ortaya koymuşlardır. Çalışmada elde edilen bulgular, yazılım endüstrisinde teknik olmayan becerilerin yüksek oranda talep edildiğini, ancak sınırlı bir düzeyde kaldığını göstermiştir. Ayrıca bu çalışmada, bu teknik olmayan becerilerin çalışanların mesleki yeteneği ve çalışma performansı içerisinde oynadığı role ilişkin anlayış eksikliğine de vurgu yapılmıştır.

Todd ve diğerleri [77], 1970-1990 yılları arasındaki 20 yıllık dönemde dört büyük gazetede yer alan BT uzmanlarına yönelik iş ilanlarını analiz ederek bu alanda talep edilen bilgi ve becerilerdeki zamansal değişimi ortaya koymuşlardır. Araştırmacılar bunun için üç meslek grubunu ele almıştır: programcılar, sistem analistleri ve bilgi sistemleri yöneticileri. Anahtar kelime analizine dayanan bu çalışmanın bulguları, programcılara özgü iş

ilanlarının çok az değiştiğini, bunun yanında teknik gereksinimlerin yüksek seviyede, işletme ve sistem bilgisine dayalı gereksinimlerin ise nispeten daha düşük seviyede talep edildiğini ortaya koymuştur. Ayrıca bu çalışmada, bilgi sistemleri yönetim pozisyonları için iş ilanlarının nispeten daha istikrarlı olduğu, işletme bilgisinin yüksek seviyede, teknik becerilerin ise daha düşük seviyede talep edildiği belirtilmiştir. Bu 20 yıllık süre boyunca talep edilen bilgi ve beceriler açısından en büyük değişim, sistem analistlerine yönelik işlerde gözlemlenmiştir. Beklentilerin aksine iş ilanlarında talep edilen teknik bilgi ve becerilerin oranının zamanla artarken, işletme ve sistem bilgisine yönelik taleplerin gittikçe azaldığı görülmüştür.

Garousi ve diğerleri [78], Türkiye yazılım endüstrisindeki yazılım uygulamalarının türlerini belirlemek amacıyla yaklaşık 200 aktif çalışan yazılım geliştirme uzmanı üzerinde 46 sorudan oluşan deneysel bir anket çalışması gerçekleştirmişlerdir. Araştırmacılar bu çalışmanın sonucunda, Türkiye'de yazılım endüstrisinin en etkin olduğu alanlar, yazılım geliştirmede kullanılan yöntem bilimler, yazılım ölçekleme yöntemleri, yazılım testi uygulamaları, yazılım gereksinimlerinin belirlenmesi gibi, Türk yazılım endüstrisine özgü önemli bulgulara ulaşmışlardır. Elde edilen bulgular Türkiye'deki yazılım endüstrisine yönelik gelişen eğilimleri yansıtması açısından önemli çıkarımlar sağlamıştır.

Debortoli ve diğerleri [17], büyük veri (big data) ve iş zekası (business intelligence) alanlarına yönelik çevrimiçi iş ilanları üzerinde Gizli anlamsal analiz yöntemine dayalı bir çalışma gerçekleştirmişlerdir. Bu çalışmada araştırmacılar, büyük veri ve iş zekâsı uzmanlığı için gerekli olan bilgi ve becerileri ortaya koymayı amaçlamışlardır. Gizli anlamsal analiz yordamına dayalı olarak gerçekleştirilen bu konu modelleme çalışmasında, büyük veri uzmanlığı için 15 farklı yetkinlik alanı ve aynı şekilde iş zekâsı uzmanlığı için de 15 farklı yetkinlik alanı tespit edilmiştir. Bu yetkinlikler, sistematik olarak gruplandırılarak işletme bilgisine yönelik yetkinlikler ve BT bilgisine yönelik yetkinlikler olarak iki başlık altında listelenmiştir.

Barua ve diğerleri [35], yazılım geliştiricileri için en aktif tartışma sitelerinden biri olan Stack Overflow platformunda paylaşılan soru ve cevaplar üzerinde GDA'ya dayalı bir olasılıksal konu modelleme analizi gerçekleştirmişlerdir. Araştırmacılar bu çalışma da, yazılım geliştirme uzmanlarının en çok tartıştığı konu ve temaları ortaya çıkarmayı amaçlamışlardır. GDA modeline dayalı olarak gerçekleştirilen bu analiz ile yazılım

geliştirme eğilimlerini en anlamlı düzeyde ortaya koyan 40 temel konu (topik) keşfedilmiştir. Çalışmanın sonucunda elde edilen bulgular, web ve mobil ortamlar için yazılım geliştirme eğilimleri, yazılım geliştirme araçları ve platformları, çoklu yazılım geliştirme alanları, açık kaynak yazılım geliştirme, bilgi paylaşım platformları, dokümantasyon oluşturma ve betik programlama gibi yazılım geliştirme eğilimlerine yönelik önemli temaları ortaya koymuştur. Gerçekleştirilen çalışma, özellikle yeni nesil yazılım geliştirme eğilimlerinin doğru anlaşılması ve yorumlanması açısından yazılım geliştiricilerin temel odak noktalarına ilişkin farklı çıkarımlar sağlamıştır.

Carreño ve Winbladh [79], Android market içerisinde yer alan üç farklı mobil uygulamaya yönelik olarak yapılan kullanıcı yorumları üzerinde GDA algoritmasına dayanan bir olasılıksal konu modelleme analizi gerçekleştirmişlerdir. Yapılan bu çalışmada yazılım gereksinimlerine yönelik olarak kullanıcıların bakış açılarının belirlenmesi amaçlanmıştır. Bu kapsamda GDA konu modelleme algoritması kullanılarak her bir uygulama için en çok tartışılan on konu tespit edilmiştir. Elde edilen bulgular, ilgili uygulamaların gelecek sürümlerinin tasarlanmasında yazılım geliştirme uzmanlarına yol gösterici olabilecek çıkarımlar ortaya koymuştur.

Lukins ve diğerleri [80], yazılım geliştirme aşamasında ve sonrasında karşılaşılan hataları kategorize etmek amacıyla açık kaynak kodlu üç farklı yazılımın (Mozilla, Eclipse, Rhino) geliştirilme süreci üzerinde GDA algoritması ile olasılıksal konu modellemeye dayalı bir çalışma gerçekleştirmişlerdir. Bahsedilen bu üç yazılımın farklı sürümleri için geliştirme ve derleme aşamasındaki hatalar (bugs) GDA ile analiz edilmiş ve gruplandırılmıştır. Çalışmanın bulguları, yazılım sisteminin tek bir yinelemesinde tüm hatalar arasında doğruluğun korunduğunu, iki yazılım sisteminin birden çok revizyonunda görülen çok sayıda hatanın ölçeklenebilir olduğunu göstermiştir. Ayrıca yapılan çalışmada, GDA tabanlı yaklaşımın doğruluğunun söz konusu yazılım sistemlerinin boyutundan veya kaynak kod tabanının yapısından etkilenmediğine de vurgu yapılmıştır. Bunun yanında GDA tabanlı yaklaşımın yazılım sistemlerine yönelik otomatik hata lokalizasyonunda etkin bir yöntem olarak kullanılabileceği belirtilmiştir.

Thomas ve diğerleri [81], yaptıkları çalışmada iyi bilinen ve iyi dokümantasyona sahip olan açık kaynak kodlu JhotDraw ve jEdit yazılım sistemlerinin kaynak kodları üzerinde GDA yöntemi ile bir konu modelleme analizi gerçekleştirmişlerdir. GDA ile

keşfedilen konular, yazılım geliştirme evrimine yönelik eğilimleri, kodlamada yaşanan değişimleri ve yenilenen yazılım geliştirme ortamlarındaki değişimleri ortaya koymuştur. Elde edilen bulgulara göre temalardaki (topiklerdeki) değişimin, yazılım geliştiriciler tarafından gerçekleştirilen gerçek kod değişiklikleri ile sıkı bir uyum içerisinde olduğu (%87-%89) görülmüştür. Çalışmanın sonucunda, GDA tabanlı konu modelleme yordamının yazılımın kodlama aşamasında yaşanan değişimlerin otomatik olarak keşfedilmesi ve özetlenmesi için etkili bir yöntem olduğuna özellikle vurgu yapılmıştır.

Gardiner ve diğerleri [82], büyük veri (big data) uzmanlık alanlarına yönelik iş ilanları üzerinde içerik analizine dayanan bir çalışma gerçekleştirmişlerdir. Bu kapsamda 1216 iş ilanı üzerinde gerçekleştirilen bu çalışma ile büyük veri uzmanlığına yönelik bilgi ve beceriler tespit edilmiş ve kategorilere ayrılarak sunulmuştur. Ayrıca bu çalışmada, büyük veri iş becerilerinin çok yönlü altyapısına dikkat çekilmiştir. Bunun yanında, büyük veri uzmanlık alanlarında talep edilen bilgi ve becerilerin genellikle analitik bilgi sistemlerinin geliştirilmesine yönelik teknik bilgi ve becerileri kapsadığına ve teknik olmayan becerilerin de yüksek oranda talep edildiğine vurgu yapılmıştır.

1.9.2. Olasılıksal Konu Modellemeye Yönelik Literatür Değerlendirmesi

Olasılıksal konu modelleme yaklaşımı, bir veri koleksiyonunda bulunan gizli anlamsal yapıları keşfetmek için kullanılan bir konu modelleme yordamıdır. Olasılıksal konu modelleme yordamları; metin dokümanlarında geçen kelimeler üzerinde bir olasılık dağılım yüzdesine sahip konuların veya temaların rassal olarak bir araya gelerek dokümanları oluşturduğu esasına dayanmaktadır [64-66]. Konu modelleme için önerilen yordamlar, istatistiksel yaklaşımlar olup dokümanı meydana getiren kelimeleri analiz ederek bir sonuca varmayı amaçlar ve bu süreçte herhangi bir etiketleme adımına veya eğitim setine ihtiyaç duymazlar [64-66].

Bu yönüyle olasılıksal konu modelleme yordamları anlamsal analizlerde, örüntü tanımada, bilgi çıkarımında ve doğal dil işlemede sıklıkla başvurulan yöntemlerdir. Bu ve benzeri araştırmalarda kullanılan konu modelleme algoritmaları arasında en yaygın olanları GAA, OGAA ve GDA yaklaşımlarıdır. Başlangıçta bir metin madenciliği aracı olarak geliştirilen konu modelleme algoritmaları, günümüzde genetik bilgi, görüntüler, resimler,

videolar ve ağlar gibi farklı türdeki veriler üzerinde de uygulanmaktadır. İlgili literatür incelendiğinde olasılıksal konu modelleme yaklaşımlarının farklı veri setleri üzerinde farklı amaçlarla uygulandığı bir çok çalışmaya rastlanmıştır [65].

GDA yaklaşımı normal metin içerikleri üzerinde gerçekleştirilen analizlerin dışında, açıklayıcı metinlerin (metadata) normal metinler hakkında içerdiği tanımlayıcı bilgiler (yazar, başlık, coğrafi konum, bağlantılar, vb.) üzerinde gerçekleştirilen anlamsal analizlerde de sıklıkla başvurulan bir yöntemdir [64,65]. Daha açık bir ifadeyle GDA yaklaşımı, farklı metinsel veri tabanları üzerinde, yazar tanımaya dayalı metin analizleri [83], isimlendirilmiş varlıkların belirlenmesi [84], coğrafi konum içeren metin analizleri [85,86], istenmeyen e-posta ve web sayfalarının filtrelenmesi [87] gibi konu modellemeye dayalı çalışmalarda da yaygın olarak kullanılan bir yaklaşımdır. Bunlara ek olarak GDA yaklaşımı, doğal dil işleme araştırmaların da [88-90], web ortamlarından bilgi çıkarımında [91,92] ve duygu analizinde de (sentiment analysis) [93,94] sıklıkla başvurulan bir modeldir.

Metin tabanlı verilerin yoğun olarak paylaşıldığı bir diğer önemli platform da sosyal ağlardır. Özellikle son yıllarda, sosyal ağlarda paylaşılan verinin miktarı ve çeşitliliğinde yaşanan artışla birlikte sosyal ağlar olasılıksal konu modelleme yaklaşımları için yeni bir araştırma ve uygulama sahası olmuştur. Birçok alanda (siyaset, eğitim, sağlık, vb.) toplumsal eğilimlerinin belirlenmesi amacıyla yapılan araştırmalarda, sosyal ağ analizlerinden büyük ölçüde faydalanılmaktadır. Bu açıdan bakıldığında, sosyal ağlarda paylaşılan verilerin analizi sonucunda elde edilen çıkarımlar, bireysel ve toplumsal eğilimlerin tespit edilmesi açısından birçok alanda karar verme süreçlerine önemli katkılar sağlamaktadır [95].

Temelde metin madenciliğine dayanan sosyal ağ analizleri, GDA tabanlı olasılıksal konu modelleme yaklaşımlarının bu araştırmalarda kullanılmasıyla önemli ölçüde gelişim göstermiştir [95]. Özellikle son dönemlerde sosyal ağlar üzerinde GDA tabanlı yaklaşımlarla, Twitter mesajlarının yazar-konu (author-topic) modeli ile gruplandırılması [96], sağlıkla ilgili tartışılan konuların belirlenmesi [97], Twitter mesajlarından çıkarılan olayların modellenmesiyle otomatik suç tahmini [98], Facebook üzerinde paylaşılan konuların cinsiyetlere göre modellenmesi [99], Twitter üzerindeki paylaşımlardan borsa

eğilimlerinin tahmini [100], sıra dışı toplumsal olayların ve mevsimsel eğilimlerin zamana bağlı olarak tespiti [101] gibi önemli çalışmalar gerçekleştirilmiştir [102].

GDA tabanlı olasılıksal konu modelleme yaklaşımının bir avantajı da model parametreleri, veri üreten dağıtım fonksiyonu seçenekleri ve algoritma çıkarımları üzerinde küçük değişikliklerle farklı veri türlerine uyarlanabilmesidir [65]. Bu bağlamda GDA yaklaşımı anket verileri, akademik yayınlar, kullanıcı tercihleri, ses ve müzik, yazılım kodları, ağ günlükleri, hata raporları, elektronik postalar, coğrafi konum bilgileri, resim, video, hasta tahlil raporları, genetik dokular ve sosyal ağlar da dâhil olmak üzere birçok farklı veri türüne uyarlanmıştır [65]. Metin odaklı veri analizlerinin dışında, GDA tabanlı konu modellerinin başarı ile uygulandığı diğer iki alan popülasyon genetiği [103] ve görüntü işleme [104] araştırmalarıdır.

Popülasyon genetiğinde, örnek bir bireyin genetik kökeninde atadan kalma özellikleri bulmak için yapılan araştırmalarda GDA tabanlı olasılıksal konu modelleri başarıyla uygulanmıştır [103]. Bu varsayım, her bireyin genotipinin bir veya daha fazla atanın soyundan geldiği fikrine dayanır. GDA tabanlı modeller kullanılarak biyologlar hem örnek popülasyonlardaki genetik kalıpları (temalar, konular) karakterize edebilir, hem de her genetik kalıbın bireyleri hangi oranda ifade ettiğini belirleyebilir (konu oranları). Popülasyon genetiğinde, GDA tabanlı modeller güçlü ve üretken modellerdir çünkü ata popülasyonlarındaki genetik kalıplar, onlardan "saf" örnekler alınamadığı durumlarda bile varsayımla modellenebilir [103,104].

GDA tabanlı modellerin başarıyla uygulandığı bir diğer alan ise bilgisayarla görme ve görüntü işleme araştırmalarıdır [104]. Bu alandaki araştırmalarda GDA tabanlı yöntemler, görüntülerin alımı, sınıflandırması ve organizasyonunda doğal görüntülere başarıyla uygulanan bir modeldir [65,104]. Bilgisayarla görme alanında çalışan araştırmacılar, görüntülerden belgeye doğrudan bir örnekleme yapmışlardır. Doküman analizinde, dokümanların birden fazla konu içerdiği ve dokümanları içeren derlemin de aynı konuları içerdiği varsayılır. Görüntü analizinde de buna benzer şekilde, her bir görüntünün bir görsel doku içerdiği ve aynı görsel dokuların bir görüntü koleksiyonunda da aynı şekilde tekrarlandığı varsayılır. Şöyle ki, bir görüntü işleme analizinin veri önişleme aşamasında, görüntüler "görsel kelimeler" koleksiyonları olarak analiz edilir [65,104]. GDA yaklaşımı bilgisayarla görme ve görüntü işleme araştırmalarında,

görüntülerin sınıflandırması [105], resim ve altyazı ilişkilendirmesi [106], görüntülerinin içerdiği temaların belirlenmesi [107] ve görüntü hiyerarşilerinin oluşturması [65,104], vb. analizlerin gerçekleştirilmesinde etkin olarak kullanılan bir yöntemdir [104].

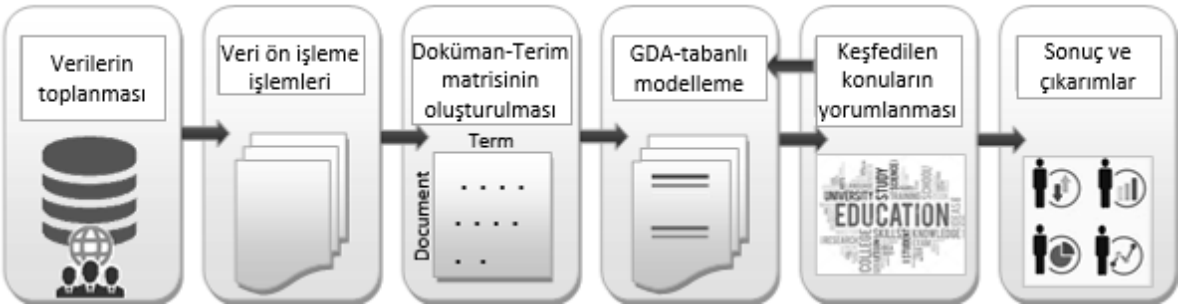
Olasılıksal konu modellemeye dayalı analizlerin yeni alanlarda uygulanmasıyla hiyerarşik karma modelleme yordamlarına yönelik yeni yaklaşımlar geliştirilmiştir. Olasılıksal konu modelleri, verilerin değiştirilebilir olduğunu yani bir koleksiyondaki belgelerin sıralamasının ve buna bağlı olarak da bir belgedeki kelime sıralamasının önemli olmadığını varsayıyordu. Bu durum, birçok problem için çok kısıtlayıcı bir durum oluşturuyordu. Bu varsayımı daha esnek hale getirmek ve daha etkin bir konu modeli oluşturmak için olasılıksal konu modelleme yaklaşımı, hiyerarşik karma modelleme yaklaşımı ile genişletilmiştir [108-110].

2. YAPILAN ÇALIŞMALAR

Bu bölümde, gerçekleştirilen bu tez çalışması kapsamında kullanılan veri setleri, yöntem ve analizler açıklanmıştır. İlk olarak, çalışılan konuya özgü standart bir veri kümesi mevcut olmadığından verilerin toplanması, verilere uygulanan önişleme süreçleri, veri setinin hazırlanması ve doküman-terim matrisinin oluşturulması aşamaları anlatılmıştır. Daha sonra, yöntem ve analiz başlığı altında önerilen yöntemin kavramsal tanım ve özellikleri, GDA tabanlı olasılıksal konu modelleme yaklaşımının veri seti üzerinde uygulanması, yöntemde kullanılan parametrelerin belirlenmesi ve deneysel analizlerin gerçekleştirilmesi aşamaları açıklanmıştır.

2.1. Çalışmanın Mimarisi

Çalışmanın mimarisi metin tabanlı nitel veri seti üzerinde GDA-tabanlı konu modelleme yaklaşımı ile gerçekleştirilen anlamsal analizin aşamalarını içermektedir. Gerçekleştirilen tez çalışmasının amaç ve kapsamına göre tasarlanan mimari, Şekil 8'de gösterildiği gibi birbirini takip eden altı sıralı aşamadan oluşmaktadır.



Şekil 8. Çalışma mimarisinin akış şeması

Birinci aşamada, çevrimiçi iş platformları üzerinden elde edilen verilerle deneysel veri seti oluşturulmuştur. Ardından, boyut azaltma ve analizin başarısını artırmak amacıyla veri seti üzerinde önişleme adımları uygulanmıştır. Bir sonraki aşamada, nitel veri kümesi üzerinde sayısal analizlerin gerçekleştirebilmesi için veri seti doküman-terim matrisine (DTM) dönüştürülmüştür. Bu işlemden sonra, gizli anlamsal yapıları keşfetmek amacıyla

DTM üzerinde GDA-tabanlı olasılıksal konu modelleme yaklaşımına dayalı bir anlamsal analiz gerçekleştirilmiştir. Daha sonra, deneysel analizden elde edilen konuların anlam düzeyleri tartışılmış ve istenilen düzeyde anlam düzeyi elde edilinceye kadar deneysel analiz farklı parametrelerle yeniden uyarlanmıştır. Son aşamada ise gerçekleştirilen analiz ile elde edilen sonuç ve çıkarımlara yer verilmiş ve bu sonuçlar yapılan benzer çalışmalar ışığında tartışılmıştır. Bundan sonraki bölümlerde çalışmanın mimarisini oluşturan aşamalar ayrıntılı olarak açıklanmıştır.

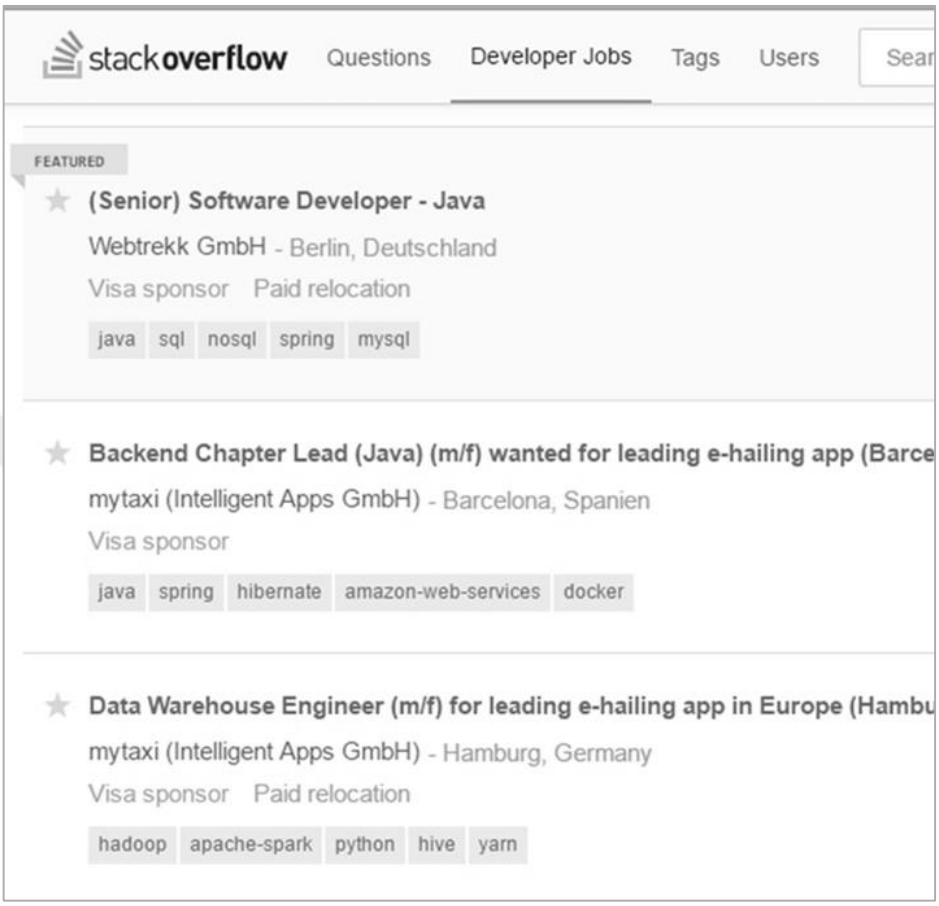
2.2. Verilerin Toplanması

Günümüzün teknolojik ortamlarında çevrimiçi iş platformları işverenler ve çalışanlar tarafından etkin bir kullanıma sahiptir. Bu çevrimiçi iş platformları (web siteleri) işverenlerle çalışanlar arasındaki etkileşimi sağlayan en önemli iletişim araçlarındandır. Bu kapsamda, veri kaynağının belirlenmesi amacıyla internette bulunan birçok iş platformu incelenmiştir. Deneysel analiz için veri kaynağının belirlenmesinde iki önemli kıstas dikkate alınmıştır: Birinci kıstas, seçilen veri kaynağının yoğun olarak yazılım geliştirme uzmanlık alanlarına odaklı bir iş platformu olmasıdır. İkinci kıstas ise seçilen veri kaynağının dünya çapında bir iş platformu olması ve farklı ülkelerden yazılım geliştirme uzmanlık alanlarına yönelik iş ilanlarını içeriyor olmasıdır.

Bu kıstaslar dikkate alınarak Stack Overflow iş platformu (Stack Overflow Developer Jobs) veri kaynağı olarak seçilmiştir [111]. Stack Overflow iş platformunda bulunan iş ilanları doğrudan yazılım geliştirmeye yönelik (developer jobs) çalışma alanlarını kapsamaktadır. Stack Overflow platformu [112], yazılım geliştirme alanında önemli bir iş platformu olmasının yanında yazılım geliştiricilerin birbirleri arasında soru-cevap şeklinde etkileşim kurmasını sağlayan önemli bir bilgi paylaşım platformudur. Bu yönüyle Stack Overflow yazılım geliştiriciler için stratejik öneme sahip bir platformdur. Bu nedenle, bu platformda yayınlanan iş ilanları çok sayıda yazılım geliştirme uzmanı tarafından takip edilmekte ve yorumlanmaktadır.

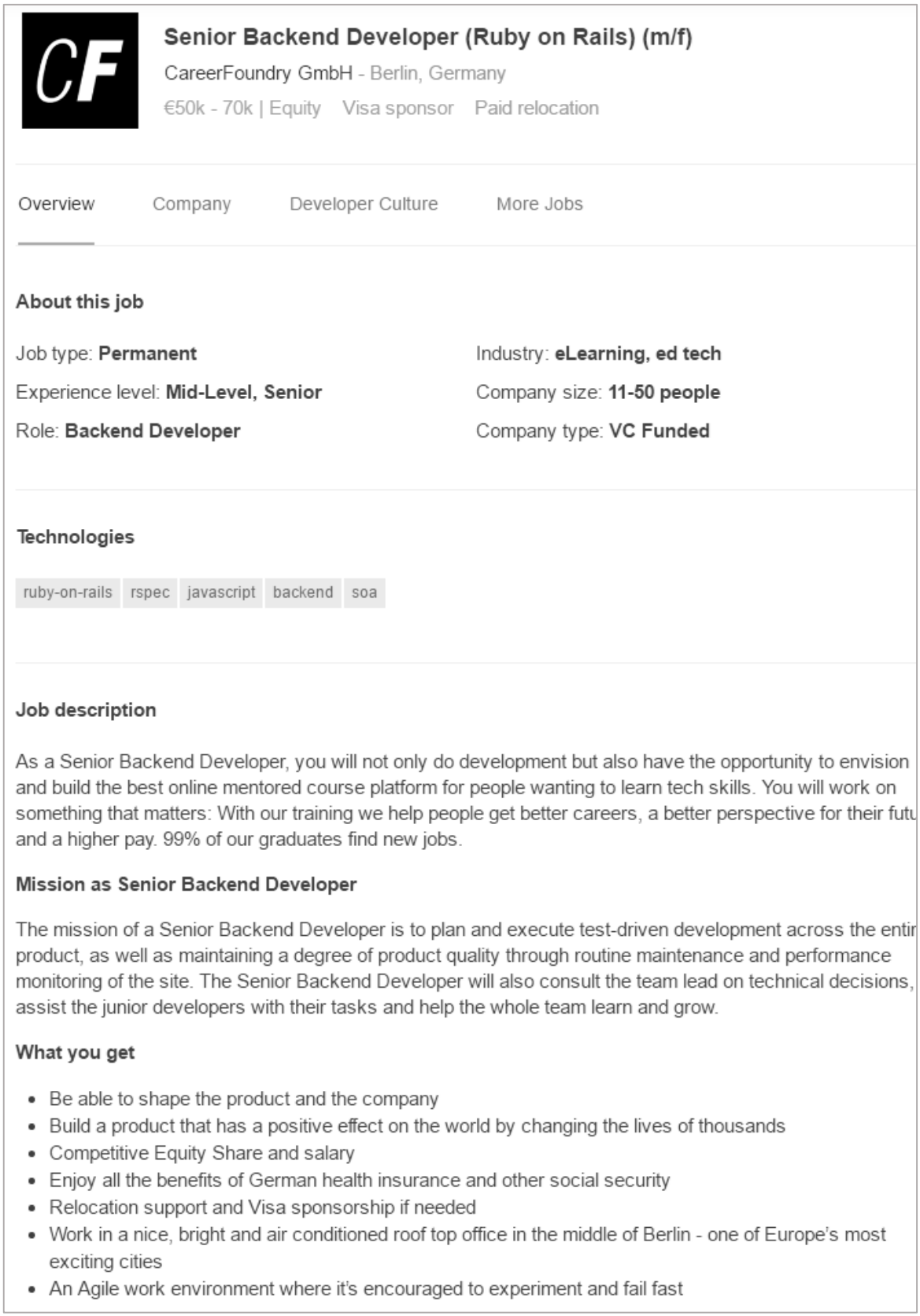
Stack Overflow platformunda iş aramayı basitleştirmek için kapsamlı arama sorgusu seçenekleri yer almaktadır. Ancak, verilerin toplanması aşamasında herhangi bir arama sorgusu kullanılmadan farklı ülkelerden yazılım geliştirme odaklı iş ilanlarının geneli

alınarak veri setine dâhil edilmiştir. İlgili iş platformunda yer alan ve veri setine dâhil edilen iş ilanlarını gösteren bir örnek Şekil 9'da verilmiştir.



Şekil 9. Stack Overflow platformunda yer alan iş ilanlarını gösteren bir örnek

Bu kapsamda Ocak 2016 ile Kasım 2016 tarihleri arasındaki 11 aylık dönemde yayınlanan İngilizce iş ilanları ile oluşturulan veri seti 4128 adet yazılım geliştirme odaklı iş ilanı içermektedir. Oluşturulan veri setindeki her bir iş ilanı tipik olarak, iş unvanı, tarih, konum, görev tanımı, talep edilen bilgi, beceri, gereksinim ve yetkinlikler gibi bilgileri içermektedir. Veri setindeki her bir iş ilanı için dört temel kayıt alanı oluşturulmuştur. Bu alanlar iş ilanının başlığını, ilanın açıldığı tarihi, ilanın tam metnini ve ait olduğu ülkeyi kapsamaktadır. Örnek bir iş ilanının veri setine dâhil edilmeden önce ilgili web sitesinde görüntülen içeriği Şekil 10'da verilmiştir.



Senior Backend Developer (Ruby on Rails) (m/f)

CareerFoundry GmbH - Berlin, Germany

€50k - 70k | Equity Visa sponsor Paid relocation

Overview Company Developer Culture More Jobs

About this job

Job type: **Permanent**

Industry: **eLearning, ed tech**

Experience level: **Mid-Level, Senior**

Company size: **11-50 people**

Role: **Backend Developer**

Company type: **VC Funded**

Technologies

ruby-on-rails rspec javascript backend soa

Job description

As a Senior Backend Developer, you will not only do development but also have the opportunity to envision and build the best online mentored course platform for people wanting to learn tech skills. You will work on something that matters: With our training we help people get better careers, a better perspective for their futu and a higher pay. 99% of our graduates find new jobs.

Mission as Senior Backend Developer

The mission of a Senior Backend Developer is to plan and execute test-driven development across the entir product, as well as maintaining a degree of product quality through routine maintenance and performance monitoring of the site. The Senior Backend Developer will also consult the team lead on technical decisions, assist the junior developers with their tasks and help the whole team learn and grow.

What you get

- Be able to shape the product and the company
- Build a product that has a positive effect on the world by changing the lives of thousands
- Competitive Equity Share and salary
- Enjoy all the benefits of German health insurance and other social security
- Relocation support and Visa sponsorship if needed
- Work in a nice, bright and air conditioned roof top office in the middle of Berlin - one of Europe's most exciting cities
- An Agile work environment where it's encouraged to experiment and fail fast

Şekil 10. Örnek bir iş ilanının görünümü

Şekil 10'un devamı

Outcomes for 2018

Main outcomes in 2018 for the whole developer team potentially include but are not limited to:

- Refactor code and improve the application's test coverage
- Move our application to service oriented architecture
- Separate the backend from the frontend and serve the backend as an API
- Start splitting out services into microservices
- Upgrade from Rails 4 to Rails 5

Skills & Requirements

- You have a minimum of 4-5 years professional experience with backend development in Ruby on Rails
- TDD/BDD with RSpec is the only way you write code and you are comfortable teaching junior develope best testing practices.
- You have a general understanding of JavaScript, HTML and CSS.
- You know your way around GitHub, Heroku, and have server administration/dev ops, and strong troubleshooting skills.
- You have experience building RESTful APIs and SOA.
- You think analytically and are able to make informed decisions regarding efficiency and organizational design patterns.
- You write clean code and proactively support and teach others best practices and refactoring technique

Nice to haves

- You are familiar with Agile Methodology or Scrum.
- You have worked with some or all of these technologies before: React, Firebase, jQuery, Bootstrap, ActionCable.
- You have experience with coding bootcamps or online education.

Your Competencies

- *Flexible/Adaptable:* Since we are a startup, you should be able to adjust quickly to changing priorities a roll with the changes.
- *Analytical Skills:* Able to process qualitative or quantitative data and draw insightful conclusions.
- *Communicating and Listening:* We are a multi-cultural office with people from all over the world--so that means listening to people and striving to make connections.
- *Proactive:* Acts without being told what to do. Feels confident making suggestions and bringing new ide
- *Strategic thinking:* Able to see and communicate the bigger picture. Also able to determine opportunities and deficiencies via comprehensive analysis.
- *Growth mindset:* Believe in everybody's potential for growth, don't see them as a list of skills, but as a process, believe in the power of "yet."
- High standards, hard work ethic, and persistence.

Buna ek olarak, programlama dillerine yönelik eğilimleri tespit etmek ve programlama dilleri açısından Dünya ölçeğindeki eğilimlerle Türkiye ölçeğindeki eğilimleri karşılaştırabilmek amacıyla Türkiye'deki yazılım geliştirme odaklı iş ilanlarını kapsayan ikinci bir deneysel veri seti oluşturulmuştur. Bu ikinci deneysel veri seti Indeed

Türkiye [113] çevrimiçi iş platformu üzerinden elde edilen 425 adet yazılım geliştirme odaklı iş ilanı ile oluşturulmuştur. Indeed Türkiye platformunda yer alan yazılım geliştirme odaklı iş ilanlarını gösteren bir örnek kesit Şekil 11'de verilmiştir.

Şekil 11. Indeed Türkiye platformunda yer alan iş ilanlarını gösteren bir örnek

Indeed Türkiye iş platformu genel kapsamlı bir portal olduğu için yazılım geliştirme odaklı iş ilanlarına erişmek için bazı arama ölçütleri ile iş ilanları filtrelenmiştir. Arama bölümünde "ne" kısmına "software or developer or application" şeklinde bir arama sorgusu girilerek sadece yazılım geliştirme odaklı İngilizce iş ilanlarının filtrelenmesi sağlanmıştır. Bu veri setinde yer alan veriler Ocak 2017 ile Mart 2017 tarihleri arasındaki 2 aylık dönemde yayınlanan yazılım geliştirme odaklı iş ilanlarından oluşmakta olup 425 adet iş İngilizce iş ilanını içermektedir.

Belirtilen ölçütlere göre oluşturulan her iki veri seti ilk olarak Excel veri tabanına kaydedilmiş, daha sonra CSV formatına dönüştürülerek bu format üzerinde deneysel analizler gerçekleştirilmiştir. Stack Overflow platformu üzerinden elde edilen verilerle oluşturulan 1.veri setindeki ilk 25 kaydı gösteren örnek veriler Şekil 12'de verilmiştir.

	A	B	C	T
1	**Date**	**Title**	**Location**	**Full Text**
2	4.01.2016	Frontend Developer	San Francisco, CA	At least 3 years' experience with Ruby on Rail
3	4.01.2016	Data Engineer	Jupiter, FL	Knowledgeable of data warehouse concepts, i
4	4.01.2016	UI Developer	Cambridgeshire, UK	The specific strong UI skills we are looking for
5	4.01.2016	Web Developer	München, Deutschland	You are a software developer who is passiona
6	5.01.2016	Software Engineer	Berlin, Deutschland	A degree in computerscience and/or at least o
7	5.01.2016	Java Developer	London, UK	A Bachelor's degree or equivalent in a comput
8	5.01.2016	Data Engineer	Berlin, Deutschland	A university degree in computerscience, Math
9	5.01.2016	System Engineer	Warszawa, Poland	Thrive in Linux environment, Understand confi
10	6.01.2016	Android Developer	Den Haag, Nederland	Bachelor's degree or equivalent, or significant
11	6.01.2016	iOS Developer	Berlin, Deutschland	A degree in computerscience or equivalent pra
12	6.01.2016	DevOps Engineer	Boston, MA	BA/ BS or commensurate experience A minimu
13	6.01.2016	Software Engineer	Brisbane, Australia	Ruby 2.2 - Rails 3 & 4 RESTful API design Perfo
14	7.01.2016	Software Developer	Bedford, MA	Bachelor's Degree or equivalent experience i
15	7.01.2016	Software Engineer	New York, NY	BS or advanced degree in computerscience, IT
16	7.01.2016	Data Engineer	Naperville, IL	Experience in managing MySQL and SQL Serve
17	7.01.2016	Frontend Developer	Berlin, Deutschland	You are passionate about programming and ha
18	8.01.2016	Software Engineer	Oakland, CA	8+ years of full stack engineering experience,
19	8.01.2016	Frontend Developer	Mountain View, CA	A passion for Frontend web design and a burn
20	8.01.2016	Software Developer	Scottsdale, AZ	Your first task is to familiarize yourself with ou
21	8.01.2016	Software Engineer	Berlin, Deutschland	B.Sc/B.Tech or higher Solid knowledge of obje
22	8.01.2016	Frontend Developer	Berlin, Deutschland	You have at least two years of work experienc
23	9.01.2016	Backend Developer	London, UK	At least 3 years+ strong experience with mode
24	9.01.2016	Android Developer	Berlin, Deutschland	Knowledge of professional software engineeri
25	9.01.2016	Java Developer	Barcelona, Spain	B.S. or Masters in computer engineering or eq

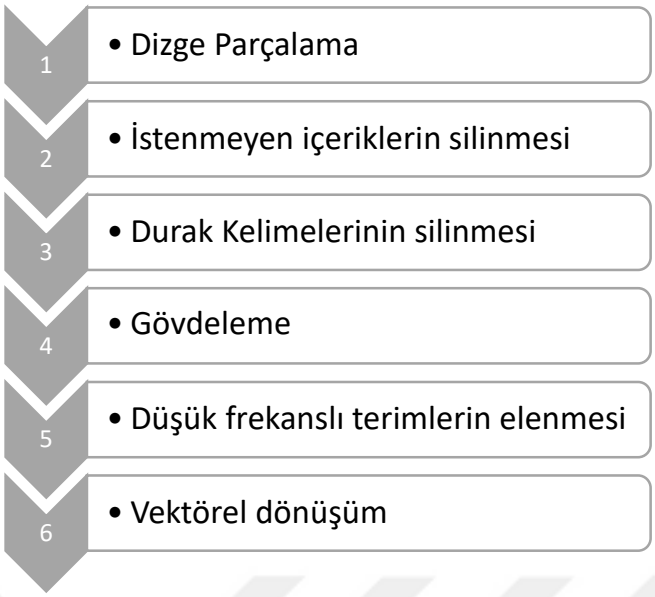
Şekil 12. Veri setindeki ilk 25 kaydı gösteren örnek veriler

2.3. Metin Önişleme ve Vektörel Dönüşüm

Metin önişleme, metin madenciliğinde analiz öncesi uygulanan en önemli işlemlerden birisidir [1,57]. Metin önişleme adımları, metin odaklı veri analizlerinin başarımını doğrudan etkileyen bir süreçtir [57]. Metin önişleme sürecindeki temel amaç, deneysel analizde kullanılacak metinlerin analiz sonrası çıkarımları en iyi şekilde yansıtacak dağılıma ve hassaslığa sahip olmasını sağlamaktır. Metin analizine dayanan uygulamalarda genellikle metinler yapılandırılmamış veriler olduğu için metin önişleme süreci bu tür analizlerin başarımı açısından oldukça önemlidir [1,56,57]. Metin önişleme süreci, özellikle yapısal olmayan (düzensiz) web tabanlı metinlerin ve sosyal ağlardan elde edilen metinsel içeriklerin analizinde mutlaka uygulanması gereken bir işlemdir [60].

Metin önişleme süreci genel olarak, dizge parçalama, metnin temizlenmesi, uyumsuz ve tamamlanmamış metinlerin veri setinden çıkarılması, durak kelimelerinin silinmesi, gövdeleme (köke indirgeme), frekans indirgeme gibi sıralı işlemleri içermektedir. İşlenen verinin ve gerçekleştirilecek olan deneysel analizin türüne bağlı olarak veri önişleme adımları değişiklik gösterebilir [1,57,60]. Örneğin, bu çalışmada veri setinde yer alan metinler yüksek oranda teknik bir jargon içerdiği için bu metinler üzerinde gövdeleme (stemming) işlemi uygulanmamıştır.

Bu çalışmada sadece Stack Overflow platformundan elde edilen ve 4128 iş ilanını içeren veri seti (1.veri seti) üzerinde önişleme adımları uygulanmıştır. Indeed Türkiye platformundan elde edilen ve 425 iş ilanını içeren veri seti (2.veri seti) az sayıda veri içerdiğinden ve üzerinde konu modelleme analizi yapılmadığından bu veri seti için önişleme sürecine gerek duyulmamıştır. Bahsedilen 2.veri seti sadece programlama dillerindeki eğilimlerin belirlenmesi için kullanılmıştır. Bundan sonraki süreçlerde aksi belirtilmedikçe veri setinden anlaşılması gereken veri kümesi 1.veri setidir. Ayrıca, veri setleri ile ilgili tablolarda verilen istatistiki bilgiler 1.veri setini kapsamaktadır. Bu kapsamda, 1.veri setine uygulanan metin önişleme adımları birbirini takip eden 6 aşamadan oluşmaktadır. Bu aşamalar aşağıda açıklanmış ve Şekil 13'de verilmiştir.



Şekil 13. Veri setine uygulanan metin önişleme aşamaları

Veri önişleme aşamasının ilk adımı olan dizge parçalama (tokenization) aşamasında gerçekleştirilecek olan metin analizinin türüne göre metinsel içerikler üzerinde yapılacak işlemler değişiklik gösterebilmektedir. Bu aşamada anlamlı özniteliklerin elde edilebilmesi için metinsel içerik kelimelere ayrılmıştır. Böylelikle veri setinde yer alan her bir metinsel veri bir kelime uzayı ile temsil edilmiş olur [1,57,60].

Bir sonraki önişleme adımında, noktalama işaretleri, web bağlantıları, özel etiketler ve anlamsız karakterler metinlerden silinmiştir. Bu işlemden sonra tüm kelimeler küçük harfe dönüştürülmüştür. Bu işlemden sonra sadece kelimelerden oluşan bir içerik elde edilmiştir [1,57,60].

Bu aşamada, elde edilen kelime uzayını küçültmek için metindeki durak kelimeleri (stop words) çıkarılmıştır. Durak kelimeleri bir dilde yaygın olarak kullanılan ve genellikle tek başına kullanıldığında bir anlam ifade etmeyen kelimelerdir. Bu nedenle metin analizine dayalı çalışmalarda genellikle durak kelimeleri metinlerden çıkarılır. Ancak bu kelimeler çıkarılırken her dile özgü, durak kelimelerini içeren bir listeye ihtiyaç duyulmaktadır. Durak kelimelerinin çıkarılması işleminin yapılan analiz türüne ve öznitelik çıkarım modeline bağlı olarak yapılabilirliği değişiklik göstermektedir. Bu çalışmada oluşturulan veri seti İngilizce metinlerden oluştuğu için, İngilizce dilinde yaygın

kullanılan durak kelimeleri (and, or, with, there, she, with, are, the, vb.) metinlerden çıkarılmıştır [1,57,60].

Deneysel analiz için oluşturulan veri seti teknik bir jargon içerdiği için veri setini modelleyen kelime uzayı da çok sayıda teknik kelimeden oluşmaktadır. Bu bakımdan, herhangi bir anlam kaybına sebebiyet vermemek için oluşturulan kelime uzayı üzerinde herhangi bir stemming (gövdeleme) işlemi uygulanmamıştır. Şöyle ki, geliştirme (development) kelimesi ile geliştirici (developer) kelimeleri aynı kökten türetilmesine karşın farklı anlamlar içermektedir. Bundan dolayı bu kelimelerin anlamsal analizde farklı konulara atanması gerekebilir. Bu kelimelerin "geliştir" (develop) şeklinde tek bir köke indirgenmesi anlamsal analizde bu kelimeler için bir anlam kaybına neden olacaktır ve bu durum anlamsal analizin tutarlılığını azaltacaktır. Bu nedenle oluşturulan kelime uzayı üzerinde herhangi bir gövdeleme işlemi uygulanmamıştır. Sadece çoğul olan İngilizce kelimeler tekile (developers→developer gibi) indirgenmiştir [1,57,60].

Bütün bu işlemlerden sonra en son adımda tüm veri setinde görülme frekansı 5'ten küçük olan kelimeler (anlamsal katkısı yok denecek kadar az olan kelimeler), kelime uzayından çıkarılarak vektör boyutu daha da indirgenmiş ve önişleme süreci bu şekilde tamamlanmıştır. Uygulanan önişleme adımlarının sonunda veri seti üzerinde gerçekleştirilen indirgemeler Tablo 3'de verilmiştir.

Tablo 3. Önişleme adımları ile veri setinde yapılan indirgemeler

Önişleme Adımları	**Eşsiz Kelime Sayısı**	**Toplam Kelime Sayısı**
Önişleme öncesi kelime sayıları	5911	505550
Durak kelimeleri silindikten sonra	5527	325520
Frekansı 5'ten küçük kelimeler silindikten sonra	3479	316990

Uygulanan önişleme adımlarının sonunda kelime uzayının boyutu önemli ölçüde indirgenmiş ve analiz için daha anlamlı bir kelime uzayı oluşturulmuştur. Böylelikle başlangıçta 5911 olan kelime sayısı bu önişleme adımlarının sonucunda indirgenerek 3479 terimden (kelimeden) oluşan bir kelime uzayı elde edilmiştir. Başlangıçta 505550 olan toplam kelime sayısı da önişleme adımları sonunda 316990 kelimeye indirgenmiştir. Bu

sürecin sonucunda, analizde kullanılacak veri setinin daha düşük boyutlu bir doküman-terim matrisi ile temsil edilmesi sağlanmıştır.

Veri setinde yer alan her metne ait kelime vektörlerinin oluşturulmasıyla veri setini modelleyen kelime uzayı da elde dilmiş olur. Elde edilen bu kelime uzayında en sık görülen ilk 20 kelime Tablo 4'de verilmiştir. Tabloda 1. sütunda kelimeler, 2.sütunda kelimelerin frekansları, 3.sütunda frekansların yüzdesi, 4.sütunda kelimelerin kaç tane veride görüldüğü, 5.sütunda ise kelimenin görüldüğü veri sayısının tüm veri seti içindeki yüzde oranı verilmektedir.

Tablo 4. Veri setindeki en yüksek frekanslı ilk 20 kelimenin dağılımı

Kelimeler	**Frekans**	**Frekans (%)**	**Veri Sayısı**	**Veri Sayısı (%)**
Experience	14838	4,68%	3788	90,58%
Development	4676	1,48%	2480	59,30%
Knowledge	3774	1,19%	2064	49,35%
Skill	3668	1,16%	2126	50,84%
Software	3610	1,14%	2080	49,74%
Year	3446	1,09%	2474	59,16%
Developer	3324	1,05%	2716	64,95%
Java	3300	1,04%	1830	43,76%
System	3242	1,02%	1624	38,83%
Design	2998	0,95%	1648	39,41%
Python	2926	0,92%	1642	39,26%
Web	2902	0,92%	1834	43,85%
Working	2624	0,83%	1686	40,32%
Application	2534	0,80%	1692	40,46%
Strong	2402	0,76%	1528	36,54%
JavaScript	2378	0,75%	1600	38,26%
Team	2328	0,73%	1580	37,78%
Work	2322	0,73%	1488	35,58%
Ability	2154	0,68%	1238	29,60%
Html	2134	0,67%	1282	30,66%

Tablo 4'de verilen kelimeler aslında yazılım geliştirme uzmanlığının oturduğu temel çerçeveyi az çok ortaya koymaktadır. Bu kelimelerle bile yazılım geliştirme eğilimlerine yönelik küçük öngörüler veya analizden elde edilecek sonuçlara ilişkin ön tahminler

belirlenebilir. Bunlara ek olarak, veri seti ile ilgili genel istatistiksel bilgiler de Tablo 5'de verilerek analiz öncesi 1.veri seti ile ilgili son değerlendirmeler ortaya konulmuştur.

Tablo 5. Veri seti ile ilgili istatistiksel bilgiler

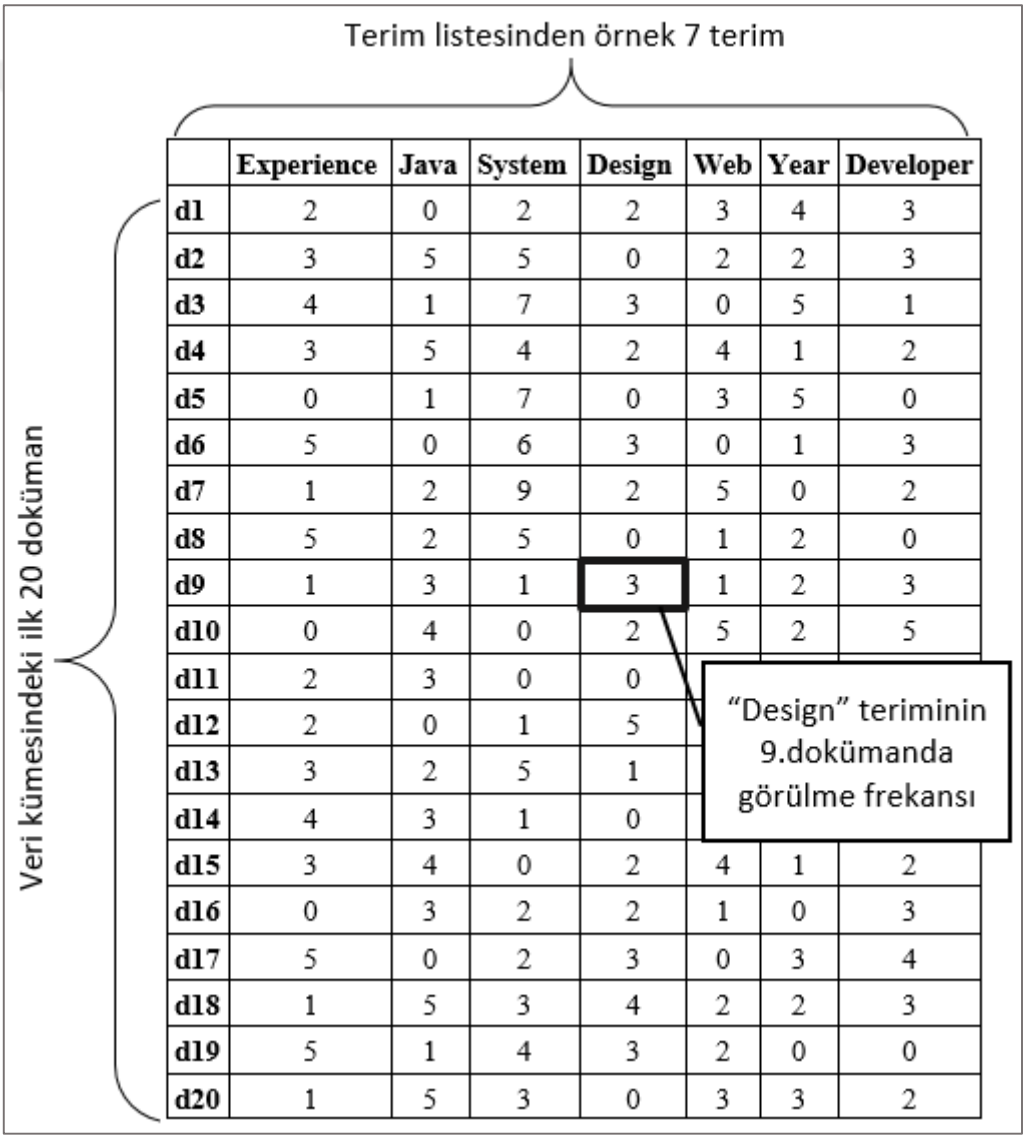
İstatistiksel Bilgiler	**Değerler**
Toplam veri sayısı	4128
Önişleme sonucunda veri setindeki eşsiz kelime sayısı	3479
Önişleme sonucunda veri setindeki toplam kelime sayısı	316990
Önişleme ile veri setinden çıkarılan toplam kelime sayısı	188560
Veri setindeki en yüksek frekanslı ilk 20 kelimenin toplam geçiş sayısı	71580
Önişleme ile veri setinden çıkarılan toplam kelime sayısının veri setindeki toplam kelime sayısına oranı	37,3%
Veri setindeki en yüksek frekanslı ilk 20 kelimenin sayısının toplam kelime sayısına oranı	22,6%
Veri setindeki en yüksek frekanslı ilk 50 kelimenin sayısının toplam kelime sayısına oranı	35,8%
Veri setindeki en yüksek frekanslı ilk 100 kelimenin sayısının toplam kelime sayısına oranı	47,5%

Metin önişleme sürecinin tamamlanmasıyla veri setinde yer alan metinler bir kelime vektörü olarak tanımlanmış olur. Terimlerin (kelimelerin) dokümanlar üzerinde dağılımını modellemeye yarayan vektör uzay modelinde tüm sayısal analizler kelime vektörleri üzerinde gerçekleştirilir. Her metne ait kelime vektörünün boyutu benzersiz kelime sayısı olan 3479'a eşittir. Vektör sayısı ise doküman sayısına yani 4182'ye eşittir. Bu şekilde veri setindeki tüm metinler vektör uzay modelinde bir kelime vektörü olarak modellenmiştir.

Metinlerin bu şekilde bir kelime vektörü olarak temsili, veri seti üzerinde sayısal analizlerin gerçekleştirilebilmesi için yapılması gereken bir dönüşümdür. Özetle, metinler nitel veriler olduğundan sayısal analizlerin yapılabilmesi için metinleri sayısal olarak temsil edilmesi yani kelime vektörüne dönüştürülmesi gerekir. Diğer türlü nitel veriler üzerinde nicel analizlerin yapılabilmesi mümkün değildir. Veri setinin tamamını modellemek için de her dokumana ait olan terim (kelime) vektörleri bir araya getirilerek bir dokuman-terim matrisi oluşturulur [1,56,58,60].

2.4. Doküman-Terim Matrisinin Oluşturulması

Metin odaklı veri analizlerinde metin belgeleri, terimlerin (kelimelerin) vektörleri olarak temsil edilir ve bu belgelerin oluşturduğu veri seti de bir DTM ile temsil edilir [94]. Önişleme adımları DTM oluşturulmadan önce tamamlanmalıdır. DTM'nin oluşturulma süreci, metin analizinin başlangıç noktasını oluşturur [1,58]. Bu kapsamda, önişleme adımlarının tamamlanmasıyla elde edilen terimlerle bir DTM oluşturulmuştur. Oluşturulan bu DTM'den ilk 20 dokümanı kapsayan örnek bir kesit Şekil14'de verilmiştir.



	Experience	Java	System	Design	Web	Year	Developer
d1	2	0	2	2	3	4	3
d2	3	5	5	0	2	2	3
d3	4	1	7	3	0	5	1
d4	3	5	4	2	4	1	2
d5	0	1	7	0	3	5	0
d6	5	0	6	3	0	1	3
d7	1	2	9	2	5	0	2
d8	5	2	5	0	1	2	0
d9	1	3	1	3	1	2	3
d10	0	4	0	2	5	2	5
d11	2	3	0	0			
d12	2	0	1	5			
d13	3	2	5	1			
d14	4	3	1	0			
d15	3	4	0	2	4	1	2
d16	0	3	2	2	1	0	3
d17	5	0	2	3	0	3	4
d18	1	5	3	4	2	2	3
d19	5	1	4	3	2	0	0
d20	1	5	3	0	3	3	2

Şekil 14. Oluşturulan DTM'den örnek bir kesit

Oluşturulan DTM bir metin koleksiyonundaki terimlerin frekansını gösteren sayısal bir matristir. Bir DTM'deki her satır, veri setindeki bir dokümanı temsil eder ve her sütun ise dokümanlarda görülen benzersiz bir kelimeyi temsil eder [96]. Matristeki her hücre ise satır tarafından temsil edilen dokümanda sütun tarafından temsil edilen kelimenin kaç kez görüldüğünü gösterir. Kelime torbası (bag of words) yaklaşımına göre oluşturan DTM, kelimelerin sırasını dikkate almadan dokümandaki kelimelerin frekansları hakkında bilgi verir [1,58,60]. Bu analiz için oluşturulan DTM 4182 satır ve 3479 sütun içermektedir. Daha açık bir ifadeyle, oluşturulan DTM 4182 metin dokümanının (iş ilanının) 3479 terimden oluşan bir kelime uzayı ile temsil edildiğini göstermektedir.

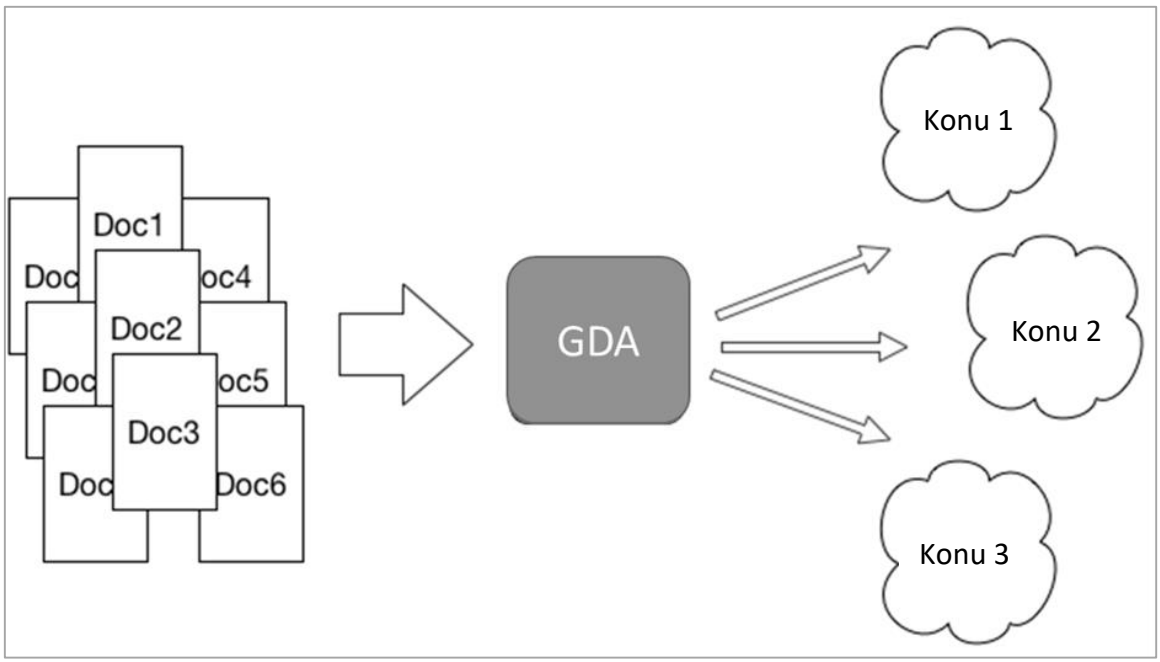
2.5. GDA-Tabanlı Konu Modelleme Analizi

Olasılıksal konu modelleme yaklaşımı, büyük bir koleksiyondaki metin dokümanlarının anlamsal yapısını modellemek ve gizli anlamsal yapıları keşfetmek amacıyla kullanılan üretken ve olasılıksal bir yaklaşımdır [64-66]. Metin dokümanları, konu (topic) olarak adlandırılan gizli anlamsal yapıları içerirler. Her konu sabit bir kelime kümesindeki olasılık dağılımı ile tanımlanır [65,66,69]. Olasılıksal konu modelleme yaklaşımına göre bir metin dokümanı farklı oranlara sahip birden fazla konuyu içerebilir. Bu konu oranlarının belirlenmesi olasılıksal konu modelleme yaklaşımının temelini oluşturur [65,66]. Bu yönüyle olasılıksal konu modelleme yaklaşımları metin koleksiyonları içerisindeki anlamsal yapıların keşfedilmesinde kullanılan etkin yöntemlerdir.

GDA konu modelleme yaklaşımı, denetimsiz öğrenmeye dayalı bir yöntem olup herhangi bir eğitim setine gerek duymadan büyük metin koleksiyonları üzerinde etkin olarak uygulanabilen yöntemdir [65]. Bu yöntemde, herhangi bir ön öğrenmeye gerek duyulmadığından, çok büyük metinsel veri setleri kısa sürede analiz edilebilir [65,66,69,70]. Bu özelliklerinden dolayı GDA yöntemi, bu tez çalışması kapsamında gerçekleştirilen deneysel analizlerde temel yöntem olarak uygulanmıştır [1].

GDA [69], bir veri setindeki gizli anlamsal yapıların keşfedilmesinde kullanılan etkin bir olasılıksal konu modelleme algoritmasıdır. GDA modelinde, her bir metin belgesi bir dizi konunun birleşimi ile temsil edilir ve benzer şekilde her konu sıkça birlikte geçen

kelimeler ile temsil edilir [65,66,69]. GDA olasılık tabanlı konu modelleme algoritması, kelimeler arasındaki anlamsal ilişkileri ortaya çıkarmak için DTM üzerinde gerçekleştirilen bir dizi matris işlemlerinden oluşmaktadır. Bu süreçte, aralarında yüksek oranda birliktelik ilişkilerine sahip kelimeler gruplandırılır ve bu kelime gruplarının her biri bir "konu" (topic) olarak adlandırılır. GDA yaklaşımı ile dokümanlardan anlamsal konuların elde edilmesine ilişkin süreç Şekil 15'de verilmiştir.



Şekil 15. GDA ile dokümanlardan konuların elde edilmesi

Bu kapsamda, İngilizce iş ilanlarından oluşan 1.veri seti üzerinde GDA modeline dayalı bir konu modelleme analizi gerçekleştirilmiştir. GDA-tabanlı olasılıksal konu modelleme yaklaşımına dayalı olarak gerçekleştirilen bu deneysel analiz sürecinde, iş ilanlarından oluşan metin koleksiyonundaki anlamsal konuların dağılımını modellemek amacıyla aşağıdaki üç adım sırasıyla icra edilmiştir:

1.Adım: İlk olarak, analizin kapsamına bağlı olarak hedeflenen konu sayısı belirlenmiştir. Konu sayısını belirlemek için daha önceden öğrenilmiş bir tahminde bulunabilir (örnek: bir önceki analizin sonuçları) veya sadece deneme-yanılma yoluyla konu sayısı belirlenebilir. Farklı tahminlerle çalışırken, istenilen seviyede anlamlı görülen veya en yüksek istatistiksel kesinliği veren konu sayısı seçilebilir. Bunun için konu sayısı 40 ile 75 arasındaki değerlerle test edilmiş her defasında konu sayısı beş artırılarak

deneysel analiz tekrarlanmıştır. En anlamlı düzey elde edilinceye kadar keşfedilen konular her defasında yeniden yorumlanmış ve deneysel analiz farklı konu sayılarıyla yeniden uygulanmıştır. Sonuçta olarak, konu sayısı K= 52 için en anlamlı düzeyde konular elde edilmiştir.

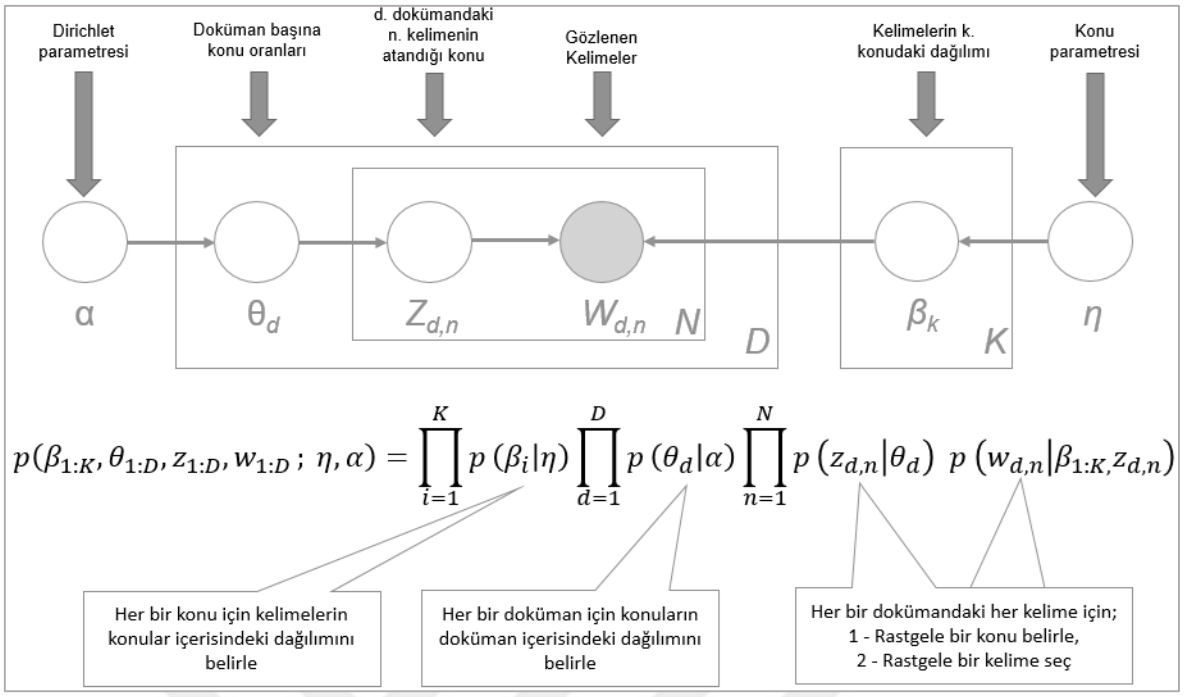
Konu modelleme analizlerinde konuların anlam düzeyi tamamen araştırmacının bakış açısı ile ilgilidir. K ile belirtilen konuların sayısı, ortaya çıkan konuların ayrıntısını veya ayrıntı seviyesini belirlemek amacıyla araştırmacı tarafından belirlenen önemli bir parametredir [1,35]. Keşfedilen konuların ayrıntı düzeyi ne kadar yüksek olursa, içerik analizi o kadar detaylı olur ve her konu daha açık ve daha belirleyici kelimelerle temsil edilir. Konu sayısı belirleyen K-değerinin artması, daha ince taneli (daha özel) konuların üretilmesini sağlarken, K-değerinin düşürülmesi daha kaba taneli (daha genel) konuların üretilmesini sağlar [1,35,69].

2.Adım: Konu sayısını temsil eden K parametresi belirlendikten sonra GDA modeli, her kelimeyi geçici bir konuya atar. Konu atamaları, 3.adımda yeniden güncelleneceği için yapılan bu ilk konu-kelime atamaları geçicidir. Geçici konulara, her kelime yarı-rastgele şekilde atanır. Bu durum, bir kelime iki kez görünürse, her kelime farklı konulara atanabilir demektir [69].

3.Adım (Tekrarlı yaklaşım): Bu adımda GDA modeli, her dokümandaki her bir kelime için konu atamalarını kontrol eder ve birliktelik ilişkilerine göre atamaları yeniden günceller. Bu adımdaki işlem, derlemdeki tüm dokümanlar için tekrar edilir [69]. Bu aşamada her kelime için yapılan konu ataması iki ölçüte göre güncellenir:

1) Konularda bu kelime ne kadar yaygın?
2) Belgede bu konular ne kadar yaygın?

Yukarıda özetlenen üç adım ile gerçekleştirilen GDA-tabanlı analiz süreci boyunca gizli değişkenlerin ve parametrelerin olasılık dağılım fonksiyonunda kullanımına ilişkin akış şeması Şekil 16'da verilmiştir [65,69]. Bu gösterimle, GDA-tabanlı konu modelleme sürecinde DTM'de yer alan rassal değişkenlere uygulanan olasılık dağılımları matematiksel olarak ifade edilmiştir.



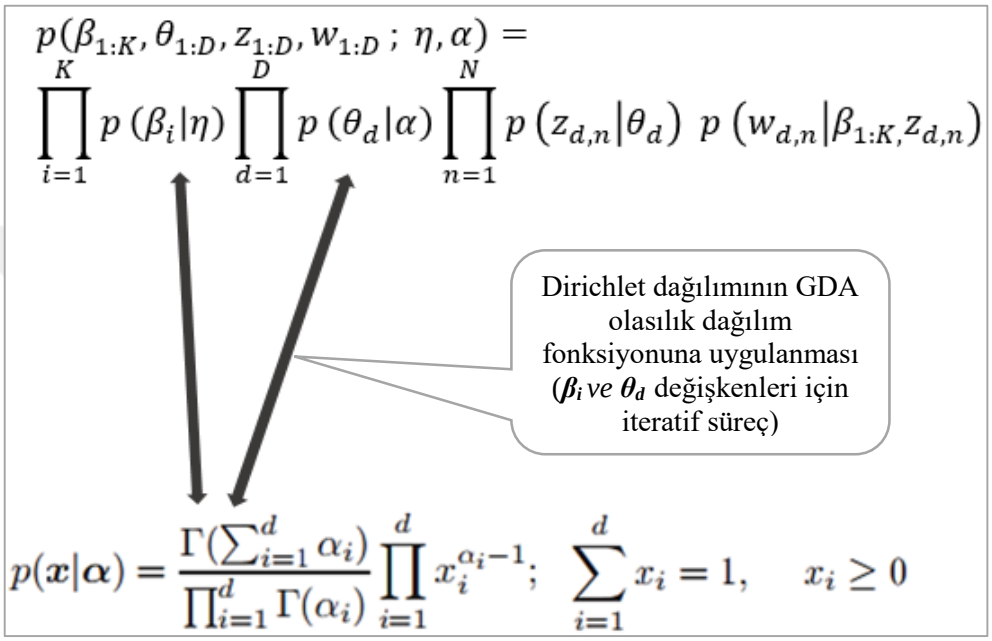
Şekil 16. GDA-tabanlı konu modelleme analizi için akış şeması

Grafiksel gösterimde dokümanlardaki konuların keşfedilmesi sürecinde, elimizde sadece gözlenen değişken olarak kelimelerin dokümanlardaki dağılımı bilinmektedir. Bunun dışında kelimelerin konulara atanması, konuların dokümana atanması ve kelimelerin konulardaki dağılımları gizli dağılımlardır. Şekil 16'da gösterilen ve olasılık dağılımında kullanılan tüm parametreler ve açıklamaları Tablo 6'da verilmiştir [65,69].

Tablo 6. GDA için önerilen model parametreleri, açıklamaları ve değerleri

Parametre	Açıklaması
D	Toplam doküman sayısı (D=4128)
K	Toplam gizli konu sayısı (K=52)
V	Sözlükte bulunan toplam kelime sayısı (V=3479)
N_d	d. dokümandaki kelime sayısı
α	Doküman başına düşen konu dağılımları için Dirichlet parametresi (α=50/K)
η	Konu başına düşen kelime dağılımları için Dirichlet parametresi (η=0,1)
θ_d	Konuların d. dokümandaki dağılımı
β_k	Kelimelerin k. konudaki dağılımı
$z_{d,n}$	d. dokümandaki n. konumda bulunan kelimenin atandığı konu
$w_{d,n}$	d. dokümandaki n. konumda gözlemlenen kelime

GDA modeli için verilen olasılık dağılım fonksiyonunda kelimelerin konulardaki dağılımı (β_k) ve konuların dokümanlardaki dağılımının (θ_d) hesaplanması aşamasındaki iteratif süreçte Dirichlet dağılımı kullanılır. Bu süreçteki işlemler veri kümesindeki tüm kelimeler ve tüm dokümanlar için tekrar edilir. Bu aşamada uygulanan işlem Şekil 17'de verilmiştir [69,70].



Şekil 17. GDA-tabanlı konu modelleme analizi için akış şeması

Dirichlet dağılımda kullanılan değişkenler, model parametreleri ve bunların açıklaması aşağıda verildiği gibidir:

α : θ'nın ortalama şekli ve seyrekliğini kontrol eden Dirichlet parametresi

d : Kullanılan uzayın boyutu: ***α*** için ***K*** (konu sayısı), ***η*** için ***V***(kelime sayısı)

$\boldsymbol{x_i}$: i.sıradaki kelimenin görülmesi olasılığı $P(w_i)$

Γ( ) : gama dağılımı , Γ(n)=(n-1)!

GDA modelinde, parametre kestirimi için bizi ilgilendiren temel değişkenler β_k (kelimelerin k. konudaki dağılımı) ve $\theta_{d,k}$ (k. konunun d. dokümandaki oranı) değişkenleridir. GDA algoritmasında model parametrelerinin kestirimi için çoğunlukla Gibbs örneklemesinden yararlanılmaktadır [71]. Bu kapsamdaki deneysel analizler için GDA modelinin Gibbs örneklemesine dayalı ölçeklenebilir bir uygulaması olan MALLET

[114] aracı kullanılmıştır. Konu modelleme uygulamalarında yaygın olarak kullanılan MALLET aracı, Gibbs örnekleme [71] algoritmasına dayalı konu modellemesi yapan hızlı ve ölçeklenebilir bir Java uygulamasıdır [1,35]. Mallet uygulaması ile gerçekleştirilen analizde GDA algoritması 1000 Gibbs örnekleme tekrarı (iterasyon) ile 1.veri seti üzerinde uygulanmış ve analiz sonucunda en anlamlı düzeyde 52 konu elde edilmiştir.

2.6. Programlama Dillerine Yönelik İçerik Analizi

Yazılım geliştirme uzmanlık alanlarına yön veren en önemli eğilimlerden birisi de programlama dilleridir. Programlama dillerinde yaşanan gelişmeler, yenilikler ve yeni eğilimler yazılım geliştirme uzmanlık alanlarına direk etki eden faktörlerdir. Programlama dillerine yönelik bilgi ve beceriler yeni nesil yazılım geliştirme uzmanlık alanlarında en çok talep edilen yetkinliklerdir. Özellikle programlama dillerinin birlikte kullanımını gerektiren yeni nesil yazılım geliştirme becerileri bu alanda sıklıkla talep edilen önemli nitelikler arasında yer almaktadır [1,13,16,35].

Bu bağlamda, programlama dillerine yönelik talep ve eğilimleri ortaya koymak amacıyla iki farklı veri seti üzerinde (dünya ölçeği için 1.veri seti üzerinde, Türkiye ölçeği için 2.veri seti üzerinde) anahtar kelime indeksine dayalı iki aşamalı bir içerik analizi gerçekleştirilmiştir. Bu içerik analizinde programlama dillerinin belirlenmesinde ve programlama dillerine yönelik üçlü grupların oluşturulmasında veri setindeki kelimelerin görülme frekansları esas alınmıştır.

Analizin 1.aşamasında, programlama dillerine yönelik eğilimleri Dünya ve Türkiye ölçeğinde ortaya koymak amacıyla veri setleri üzerinde içerik analizi gerçekleştirilmiş ve her iki ölçekte en çok talep edilen 20 programlama dili tespit edilmiştir. Analiz kapsamında ilk olarak Dünya ölçeğinde en çok talep edilen ilk 20 programlama tespit edilmiştir. Daha sonra bu 20 programlama dilinin Türkiye ölçeğini temsil eden 2.veri setindeki geçiş frekansları belirlenerek bu programlama dillerinin Türkiye ölçeğindeki talep sıralaması ortaya konulmuştur.

Analizin 2.aşamasında ise programlama dillerinin birlikte kullanımına yönelik eğilimleri Dünya ve Türkiye ölçeğinde ortaya koymak amacıyla veri setleri üzerinde anahtar kelime indekslemeye dayalı içerik analizi gerçekleştirilmiştir. Analizin bu

aşamasında, daha önceki aşamada elde edilen en yüksek frekanslı 20 programlama dili ile oluşturulan farklı üçlü grupların ilk olarak 1.veri setindeki görülme frekansları hesaplanarak bu üçlü grupların dünya ölçeğindeki talep sıralaması tespit edilmiştir. Daha sonra aynı üçlü grupların 2.veri setindeki görülme frekansları hesaplanarak bu üçlü grupların Türkiye ölçeğindeki talep sıralaması tespit edilmiştir. Bu üçlü gruplar içerisinde Dünya ve Türkiye ölçeğinde en çok talep edilen ilk 20 grup yüzde oranları ile birlikte tespit edilmiş ve bulgular bölümünde verilmiştir.

3. BULGULAR

Bu tez kapsamında, yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerinin belirlenmesi amacıyla İngilizce iş ilanlarından oluşturulan veri seti üzerinde, GDA-tabanlı olasılıksal konu modellemeye dayalı deneysel analizleri içeren bir çalışma gerçekleştirilmiştir. Deneysel analizlerin sonucunda, yeni nesil yazılım geliştirme eğilimlerine yönelik teknik alan bilgi ve becerileri, yazılım geliştirme ve programlama becerileri, analitik ve mantıksal beceriler ve teknik olmayan beceriler ortaya konulmuştur. Bu kapsamda ayrıca, yazılım geliştirme uzmanlık rolleri, iş tanımları ve sorumlulukları, programlama dillerine yönelik eğilimler ve birlikte talep edilen programlama dilleri arasındaki birliktelik ilişkileri tespit edilmiştir. Deneysel analizlerden elde edilen bulgular, yazılım geliştirme uzmanlık alanlarındaki talep ve eğilimlerin çok geniş kapsamda disiplinler arası bilgi, beceri ve yetkinlikleri gerektirdiğini ortaya koymuştur.

3.1. Yazılım Geliştirme Eğilimlerine Yönelik Bilgi ve Beceriler

Konu modelleme analizinin bu aşamasında, sistem farklı konu sayılarıyla denenmiş ve bunun sonucunda, yazılım geliştirme eğilimlerine yönelik bilgi ve becerileri en anlamlı düzeyde ortaya koyan 52 konu keşfedilmiştir. İş ilanlarına özgü metinler, teknik bir jargon kullanan ve içerisinde alan bilgisi, iş tecrübesi, eğitim, beceri, yetkinlik, deneyim, vb. gibi çok çeşitli bilgileri kapsayan zengin içerikli metinlerdir. Bu zengin içerik, analiz sonucunda keşfedilen konuların sayısını ve kapsamını artırmıştır.

Bu kapsamda elde edilen ve yazılım geliştirme uzmanlık alanlarına yönelik eğilimleri ortaya koyan 52 konu, temel disiplinlerle ve yazılım geliştirme uzmanlık alanlarıyla ilişkilendirilerek alan bilgi ve becerilerine özgü bir taksonomi oluşturulmuştur. Bu süreçle yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerileri için dört ana yetkinlik alanı belirlenmiştir. Sonrasında, analiz sonucunda elde edilen 52 konunun içeriğine bakılarak konuların bu alanlara ataması gerçekleştirilmiştir. Bu atama işlemi yapılırken konuların içerdiği anahtar kelimeler ve yüzde oranları da dikkate alınmıştır. Konuların atandığı bu yetkinlik alanları aşağıdaki şekilde sıralanmıştır:

1. Teknik alan bilgi ve becerileri
2. Yazılım geliştirme ve programlama becerileri
3. Analitik ve mantıksal beceriler
4. Teknik olmayan beceriler

Yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerileri için belirlenen dört ana yetkinlik alanının konuların yüzdelik oranlarına göre dağılımı: teknik alan bilgi ve becerileri (%23); yazılım geliştirme ve programlama becerileri (%52); analitik ve mantıksal beceriler (%12) ve teknik olmayan beceriler (%13) şeklindedir. Bu dağılımlar Tablo 7 ve Şekil 18'de verilmiştir. Ayrıca konuların yetkinlik alanlarına sayı olarak dağılım oranları ise: teknik alan bilgi ve becerileri 13 konu; yazılım geliştirme ve programlama becerileri 22 konu; analitik ve mantıksal beceriler 10 konu ve teknik olmayan beceriler 7 konu şeklindedir.

Elde edilen bulgular, ilk bakışta yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerin farklı disiplinleri içeren geniş bir bilgi alanını kapsadığını göstermektedir. Elde edilen bulgulara göre, yazılım geliştirme uzmanlık alanları için kapsamlı yazılım geliştirme becerilerin yanı sıra teknik bilgi alanları, analitik ve mantıksal beceriler ve ayrıca iletişim becerilerine dayalı bireysel (teknik olmayan beceriler) becerilerin önemli ölçüde talep edildiği görülmüştür.

Tablo 7. Yazılım geliştirme uzmanlığı için yetkinlik alanları ve yüzdelik dağılımları

Yazılım geliştirme uzmanlığı için yetkinlik alanları	**Konu sayısı**	**Toplam ağırlığı**
Teknik alan bilgi ve becerileri	13 (%25,0)	%23
Yazılım geliştirme ve programlama becerileri	22 (%42,3)	%52
Analitik ve mantıksal beceriler	10 (%19,2)	%12
Teknik olmayan beceriler	7 (%13,5)	%13

Şekil 18. Yazılım geliştirme yetkinlik alanlarının dağılım grafiği

Yeni nesil yazılım geliştirme eğilimlerine yönelik bilgi ve becerilerin daha iyi anlaşılması ve bu kapsamda belirlenen konuların (temel bilgi ve beceriler) ve yetkinlik alanlarının daha ayrıntılı olarak ortaya konulabilmesi amacıyla elde edilen bulgular dört alt başlık altında verilmiştir.

3.1.1. Teknik Alan Bilgi ve Becerileri

Keşfedilen 52 konu içerisinde 13 tanesinin (%25), "teknik alan bilgi ve becerileri" kapsamında olduğu gözlemlenmiştir. Bu kapsamda değerlendirilen konuların frekans yüzdelerinin toplamı %23'tür. Teknik alan bilgi ve becerileri yetkinlik alanına giren bu 13 konu, atanan konu adları (Türkçe ve İngilizce), betimleyici anahtar kelimeler ve yüzde oranları ile birlikte Tablo 8 ve Şekil 19'da verilmiştir.

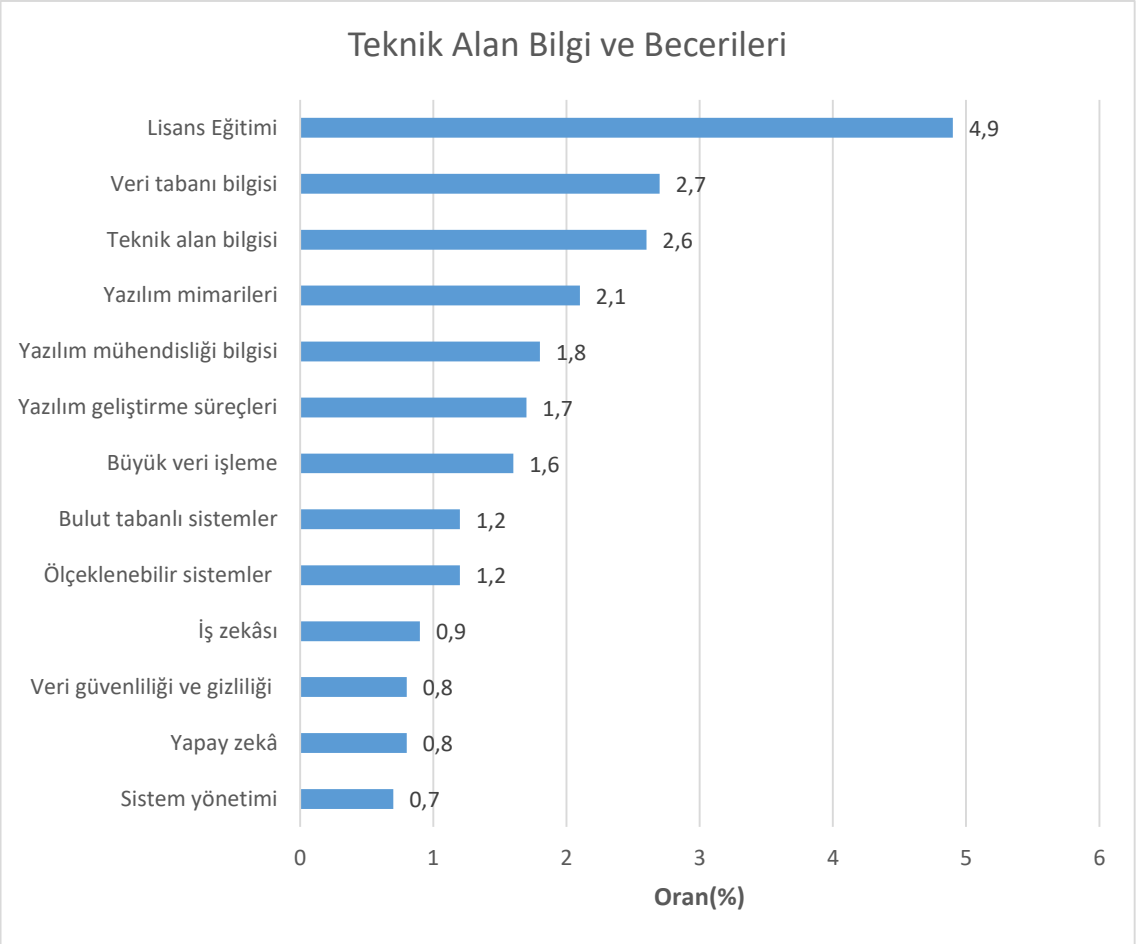
Teknik alan bilgi ve becerileri kapsamındaki bulguların, yazılım uzmanlığına yönelik olarak sahip olunması gereken teorik bilgi alanlarına vurgu yaptığı görülmektedir. Özellikle alanla ilgili lisans eğitimi en fazla vurgulanan nitelik olarak göze çarpmaktadır. Buradan elde edilen çıkarımla, yazılım geliştirme uzmanlığındaki talep ve eğilimlerin

üniversiteler tarafından değerlendirilmesi ve bu kapsamda ele alınması gereken bir konu olduğunu söylemek mümkündür.

Bunun yanında en fazla vurgulanan ilk beş konu yüzde sırasına göre "lisans eğitimi", "veri tabanı bilgisi", "teknik alan bilgisi", "yazılım mimarileri" ve "yazılım mühendisliği bilgisi" olarak sıralanmıştır. Tabloda konularla birlikte verilen betimleyici anahtar kelimeler, o konuda en sık görülen kelimeleri temsil etmektedir. Ayrıca yüzdelik oranlarla temsil edilen değerler, o konunun tüm doküman koleksiyonundaki görülme sıklığını belirtmektedir.

Tablo 8. Teknik alan bilgi ve becerilerine yönelik konular ve yüzdeleri

Konu Adları	**Tanımlayıcı Anahtar Kelimeler**	**Oran**
Lisans Eğitimi (Educational requirements)	degree science computer bachelor related field discipline skill	4,9%
Veri tabanı bilgisi (Database knowledge)	database sql knowledge nosql server oracle mongodb mysql	2,7%
Teknik alan bilgisi (Deep domain knowledge)	deep domain software knowledge background related training	2,6%
Yazılım mimarileri (Software architectures)	architecture software oop client server building soa design	2,1%
Yazılım mühendisliği bilgisi (Software Eng. knowledge)	engineering development software app field knowledge ground	1,8%
Yazılım geliştirme süreçleri (Software dev. processes)	analysis design modelling software process requirement test	1,7%
Büyük veri işleme (Big data processing)	big data processing integration hadoop tool apache file hdfs	1,6%
Ölçeklenebilir sistemler (Scaling systems)	system building scalable highly designing multi scala	1,2%
Bulut tabanlı sistemler (Cloud-based systems)	service cloud system data mapping aws paas vpc saas	1,2%
İş zekâsı (Business intelligence)	business knowledge manage intelligence domain industry	0,9%
Yapay zekâ (Artificial intelligence)	artificial intelligence modeling prolog learning decision lisp	0,8%
Veri güvenliliği ve gizliliği (Data security and privacy)	data privacy security etl warehouse storage consistent control	0,8%
Sistem yönetimi (System administration)	system network web server apache noc linux data center voip	0,7%
Toplam		**23%**



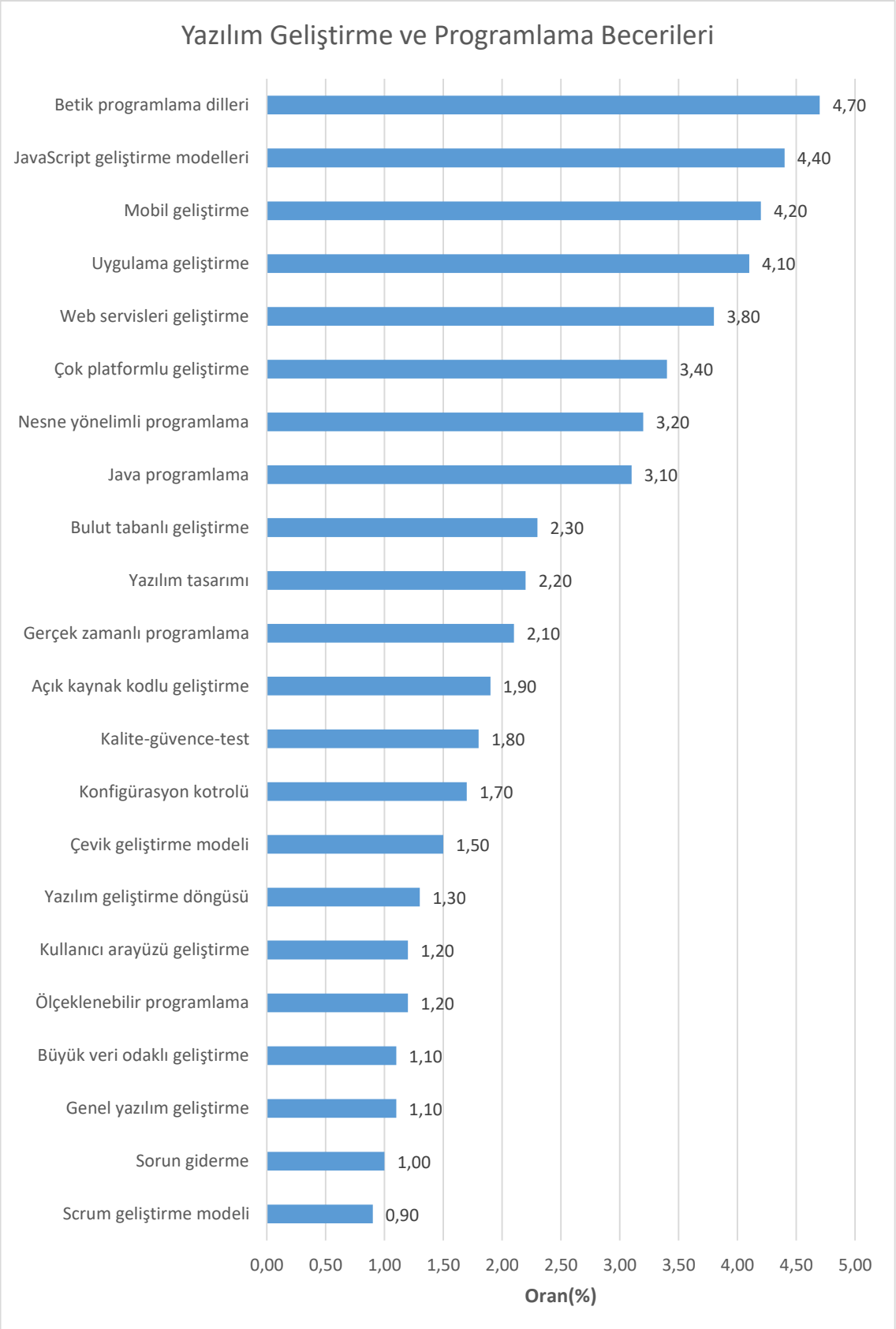
Şekil 19. Teknik alan bilgi ve becerilerinin dağılım grafiği

3.1.2. Yazılım Geliştirme ve Programlama Becerileri

Keşfedilen 52 konu içerisinde 22 tanesinin (%42,3) "yazılım geliştirme ve programlama becerileri" kapsamında olduğu gözlemlenmiştir. Bu kapsamda değerlendirilen konuların frekans yüzdelerinin toplamı %52' dir. Yazılım geliştirme ve programlama becerileri yetkinlik alanına dâhil edilen bu 22 konu, atanan konu adları (Türkçe ve İngilizce), betimleyici anahtar kelimeler ve yüzde oranları ile birlikte Tablo 9 ve Şekil 20'de verilmiştir.

Tablo 9. Yazılım geliştirme ve programlama becerilerine yönelik konular ve yüzdeleri

Konu Adları	**Tanımlayıcı Anahtar Kelimeler**	**Oran**
Betik programlama dilleri (Scripting programming languages)	java python javascript language ruby programming scripting	4,7%
JavaScript geliştirme modelleri (JavaScript frameworks)	javascript html web jquery frontend nodejs ajax bootstrap	4,4%
Mobil geliştirme (Mobile development)	android mobile ios developer swift platform objective-c	4,2%
Uygulama geliştirme (Application development)	app development application software mobile usage platform	4,1%
Web servisleri geliştirme (Web Services development)	web service api rest json backend xml restful develop	3,8%
Çok platformlu geliştirme (Multi-platform Development)	multi multiple platform stage web android mobile	3,4%
Nesne yönelimli programlama (Object oriented programming)	object programming oriented coding java python model	3,2%
Java programlama (Java programming)	java series programming hibernate jvm docker eclipse	3,1%
Bulut tabanlı geliştirme (Cloud-based Development)	cloud developer azure aws java python sql service	2,3%
Yazılım tasarımı (Software design)	software design plan modelling development app sampling	2,2%
Gerçek zamanlı programlama (Real-time programming)	real time data programming java flow stream integration	2,1%
Açık kaynak kodlu geliştirme (Open-source development)	open source system built debug software github lab	1,9%
Kalite-güvence-test (Quality-assurance-testing)	quality testing assurance qa automation metric bug	1,8%
Konfigürasyon kotrolü (Configuration Control)	control puppet chef configuration support operations plan	1,7%
Çevik geliştirme modeli (Agile development model)	development agile lean scrum practice kanban framework	1,5%
Yazılım geliştirme döngüsü (Software development cycle)	development life cycle model planning stage analysis	1,3%
Ölçeklenebilir programlama (Scalable programming)	scalable programming scala java spark size type	1,2%
Kullanıcı arayüzü geliştirme (User Interface development)	ux ui user interface frontend design javascript development	1,2%
Genel yazılım geliştirme (Full-stack development)	full stack frontend backend multi experience developer fsd	1,1%
Büyük veri odaklı geliştirme (Big data development)	big data hadoop pig hive nosql mongodb etl hbase spark	1,1%
Sorun giderme (Troubleshooting)	remote troubleshooting technical service helpdesk solution	1,0%
Scrum geliştirme modeli (Scrum development model)	scrum framework model agile tdd process design project	0,9%
Toplam		**%52**



Şekil 20. Yazılım geliştirme ve programlama becerilerinin dağılım grafiği

Yazılım geliştirme ve programlama becerileri kapsamındaki bulguların yazılım uzmanlığı alanlarındaki uygulama geliştirme, programlama ve kodlama becerisi ve bu becerilere özgü bilgi alanlarına vurgu yaptığı görülmektedir. Özellikle, yeni nesil betik programlama ve bu kapsamda ön plana çıkan Java, Pyton, Javascript gibi programlama dilleri en fazla vurgulanan nitelikler olarak göze çarpmaktadır.

Bunun yanında en fazla vurgulanan ilk beş konu yüzde sırasına göre "betik programlama dilleri", "Javascript geliştirme modelleri", "mobil geliştirme", "uygulama geliştirme" ve "web servisleri geliştirme" olarak sıralanmıştır. Tabloda konularla birlikte verilen betimleyici anahtar kelimeler, o konuda en sık görülen kelimeleri temsil etmektedir. Ayrıca yüzdelik oranlarla temsil edilen değerler, o konunun tüm doküman koleksiyonundaki görülme sıklığını belirtmektedir.

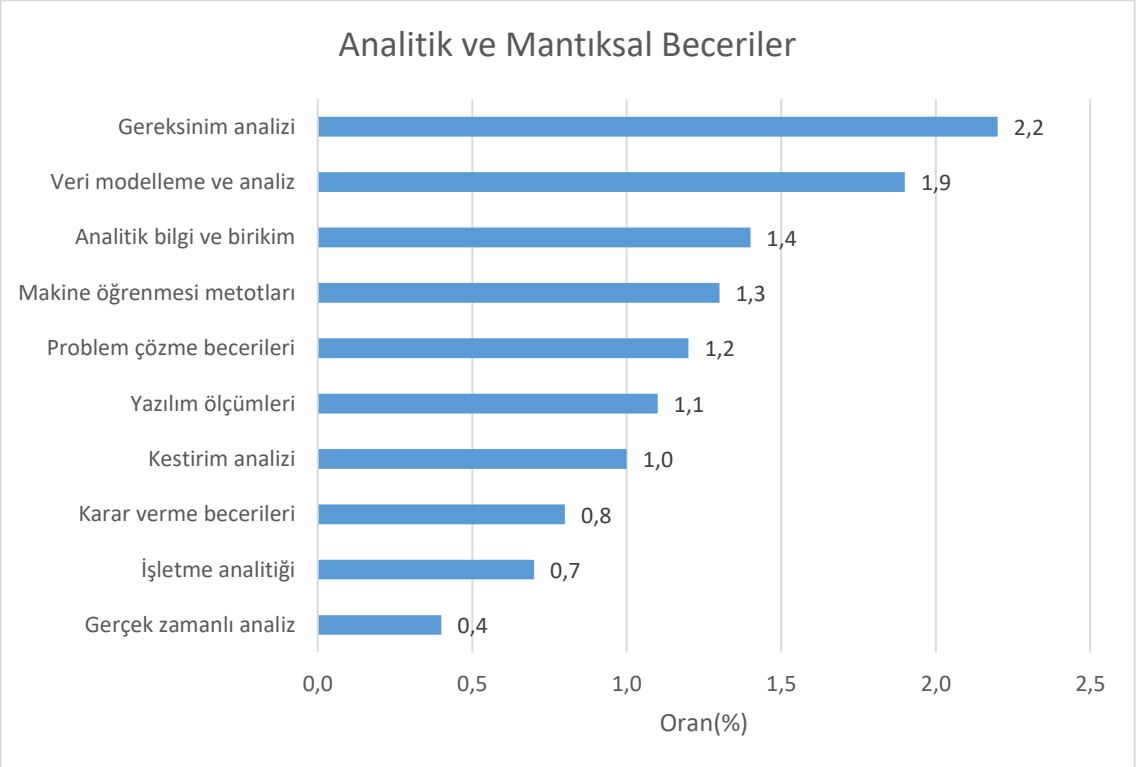
3.1.3. Analitik ve Mantıksal Beceriler

Keşfedilen 52 konu içerisinde 10 tanesinin (%19,2), "analitik ve mantıksal beceriler" kapsamında olduğu gözlemlenmiştir. Bu kapsamda değerlendirilen konuların frekans yüzdelerinin toplamı %12' dir. Analitik ve mantıksal beceriler yetkinlik alanına dâhil edilen bu 10 konu, atanan konu adları (Türkçe ve İngilizce), betimleyici anahtar kelimeler ve yüzde oranları ile birlikte Tablo 10 ve Şekil 21'de verilmiştir.

Analitik ve mantıksal beceriler kapsamındaki bulguların, yazılım geliştirme uzmanlık alanlarına yönelik olarak sahip olunması gereken veri modelleme, veri görselleştirme ve yazılım geliştirme süreçlerinde kullanılan farklı analiz becerilerine vurgu yaptığı görülmektedir. Özellikle, yazılım geliştirme sürecinin en önemli aşamalarından olan "gereksinim analizi" en fazla vurgulanan analiz becerisi olarak göze çarpmaktadır. Bunun yanında en fazla vurgulanan ilk beş konu yüzde sırasına göre "gereksinim analizi", "veri modelleme ve analizi", "analitik bilgi ve birikim", "makine öğrenmesi metotları" ve "problem çözme becerileri" olarak sıralanmıştır. Tabloda konularla birlikte verilen betimleyici anahtar kelimeler, o konuda en sık görülen kelimeleri temsil etmektedir. Ayrıca yüzdelik oranlarla temsil edilen değerler, o konunun tüm doküman koleksiyonundaki görülme sıklığını belirtmektedir.

Tablo 10. Analitik ve mantıksal becerilere yönelik konular ve yüzdeleri

Konu Adları	**Tanımlayıcı Anahtar Kelimeler**	**Oran**
Gereksinim analizi (Requirements Analysis)	requirement need constraint software analysis control basic	2,2%
Veri modelleme ve analizi (Data modeling and analysis)	data modeling strong analysis ability processing focus	1,9%
Analitik bilgi ve birikim (Analytical background)	analytical knowledge background concept model experience work	1,4%
Makine öğrenmesi metotları (Machine learning methods)	machine learning model supervised method process training	1,3%
Problem çözme becerileri (Problem solving skills)	problem solving case explain skill mentality intuitive	1,2%
Yazılım ölçümleri (Software metrics)	software metric measure application test configuration check	1,1%
Kestirim analizi (Predictive analytics)	predictive estimate judgment analytics calculate guess tool	1,0%
Karar verme becerileri (Decision-making skills)	hypothesis support decision judgement skill rule experience	0,8%
İşletme analitiği (Business analytics)	business analytics risk intelligence data information analysis	0,7%
Gerçek zamanlı analiz (Real-time analytics)	real time process data analytics model integration stream	0,4%
Toplam		**12%**



Şekil 21. Analitik ve mantıksal becerilerin dağılım grafiği

3.1.4. Teknik Olmayan Beceriler

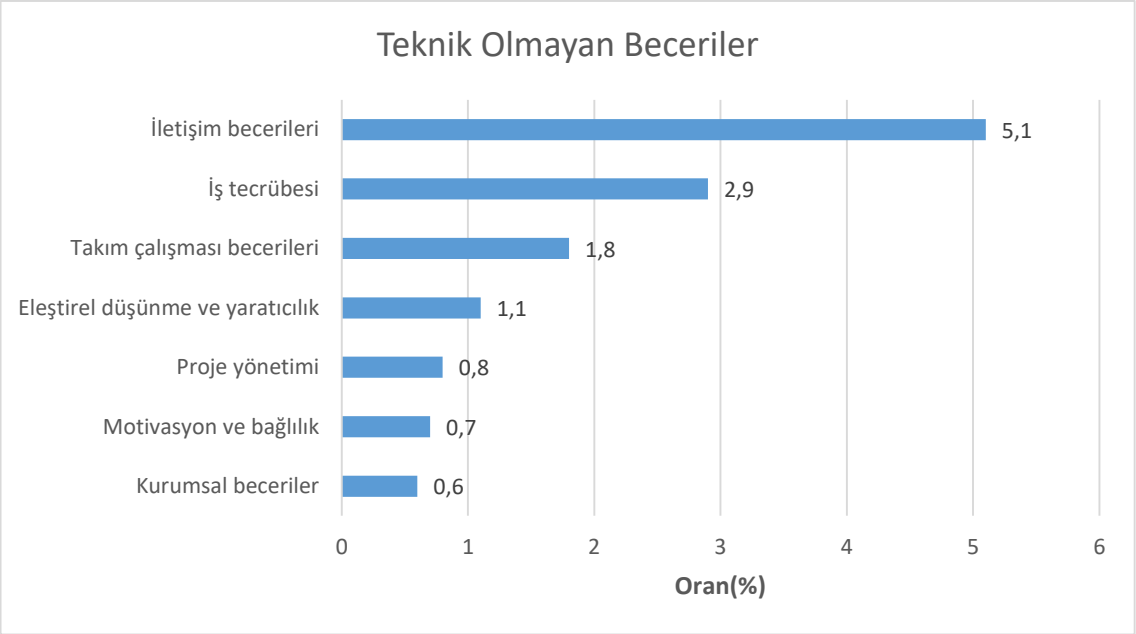
Keşfedilen 52 konu içerisinde 7 tanesinin (%13,5) "teknik olmayan beceriler" kapsamında olduğu gözlemlenmiştir. Bu kapsamda değerlendirilen konuların frekans yüzdelerinin toplamı %13' dir. Teknik olmayan beceriler yetkinlik alanına dâhil edilen bu 7 konu, atanan konu adları (Türkçe ve İngilizce), her konu için betimleyici anahtar kelimeler ve yüzde oranları ile birlikte Tablo 11 ve Şekil 22'de verilmiştir.

Teknik olmayan bireysel beceriler kapsamındaki bulguların, yazılım geliştirme uzmanlık alanlarında sahip olunması gereken iletişim becerileri, iş tecrübesi, takım çalışması becerileri, eleştirel düşünme ve yaratıcılık gibi farklı bireysel becerilere vurgu yaptığı görülmektedir. Özellikle iletişim becerisi, keşfedilen 52 konu içerisinde en yüksek orana sahip bireysel beceri olarak göze çarpmaktadır.

Bunun yanında teknik olmayan beceriler kategorisinde en fazla vurgulanan ilk beş konu yüzde sırasına göre "iletişim becerileri", "iş tecrübesi", "takım çalışması becerileri", "eleştirel düşünme ve yaratıcılık" ve "proje yönetimi" olarak sıralanmıştır. Tabloda konularla birlikte verilen betimleyici anahtar kelimeler, o konuda en sık görülen kelimeleri temsil etmektedir. Ayrıca yüzdelik oranlarla temsil edilen değerler, o konunun tüm doküman koleksiyonundaki görülme sıklığını belirtmektedir.

Tablo 11. Teknik olmayan becerilere yönelik konular ve yüzdeleri

Konu Adları	**Tanımlayıcı Anahtar Kelimeler**	**Oran**
İletişim becerileri (Communication skills)	skill communication written verbal excellent oral presentation	5,1%
İş tecrübesi (Working experience)	work experience practice year expertise knowledge competency	2,9%
Takım çalışması becerileri (Team work skills)	team player work collaboration group mutual task	1,8%
Eleştirel düşünme ve yaratıcılık (Critical thinking and creativity)	critical rigorous thinking versatile creativity vision imagination	1,1%
Proje yönetimi (Project management)	project management control process team plan assignment	0,8%
Motivasyon ve bağlılık (Motivation and Engagement)	motivation engagement inspiration self-control motive aim	0,7%
Kurumsal beceriler (Organizational skill)	organizational logistic skill office work executive desicion	0,6%
Toplam		**13%**



Şekil 22. Teknik olmayan becerilerin dağılım grafiği

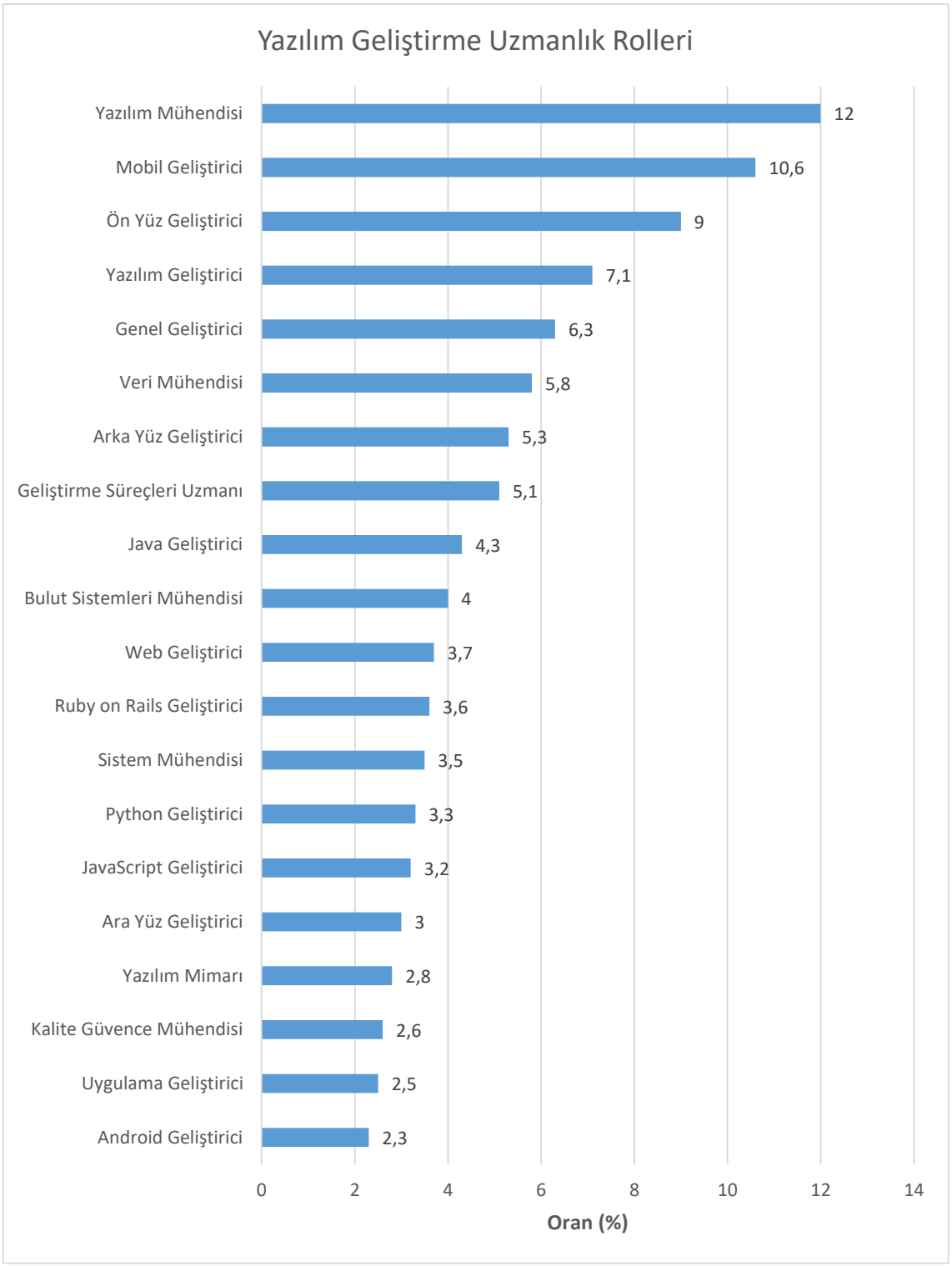
Elde edilen bulgular, yeni nesil yazılım geliştirme eğilimlerinde iletişim becerisinin ve takım çalışması becerilerin önemi de ortaya koymuştur. Şöyle ki, yeni nesil yazılım geliştirme eğilimleri yoğun olarak işbirlikçi takım modeline dayandığı için yazılım geliştirme ekibinde yer alan takım oyuncuları (yazılım geliştirme uzmanları) arasındaki etkileşim, işbirliği ve görev dağılımına dayalı beceriler bu alanda sıklıkta talep edilen yetkinlikler olarak karşımıza çıkmaktadır.

3.2. Yazılım Geliştirme Uzmanlık Rolleri

Yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerini yansıtan önemli göstergelerden birisi de bu alandaki uzmanlık rolleridir. Bu kapsamda gerçekleştirilen analiz sonucunda, yazılım geliştirme uzmanlık rolleri (iş unvanları) için 20 temel rol belirlenmiştir. Yazılım geliştirme uzmanlık rollerine yönelik bulgular Tablo 12 ve Şekil 23'de verilmiştir. Elde edilen roller görülme sıklığına göre büyükten küçüğe doğru sıralanmıştır. Elde edilen bulgulara göre, en fazla talep edilen ilk beş uzmanlık rolü sırasıyla: "Yazılım Mühendisi" (Software Engineer, %12,0), "Mobil Geliştirici" (Mobile Developer, % 10,6), "Ön Yüz Geliştirici" (Frontend Developer, %9,0), "Yazılım Geliştirici" (Software Developer, %7,1) ve "Genel Geliştirici" (Full Stack Developer, %6,3) olarak tespit edilmiştir.

Tablo 12. Yazılım geliştirme uzmanlık rolleri ve bu rollere ait beceriler

Uzmanlık Rolleri	**Temel Beceriler**	**Oran (%)**
Yazılım Mühendisi (Software Engineer)	java, python, c#, javascript, c++	12,0%
Mobil Geliştirici (Mobile Developer)	android, ios, java, swift, objective-c	10,6%
Ön Yüz Geliştirici (Frontend Developer)	javascript, html5, css, java, jquery	9,0%
Yazılım Geliştirici (Software Developer)	java, c#, c++, javascript, python	7,1%
Genel Geliştirici (Full Stack Engineer)	java, python, javascript, c++, c#	6,3%
Veri Mühendisi (Data Engineer)	sql, mysql, oracle, hadoop, java	5,8%
Arka Yüz Geliştirici (Backend Developer)	php, java, mysql, python, nodejs	5,3%
Geliştirme Süreçleri Uzmanı (DevOps Engineer)	linux, aws, devops, puppet, java	5,1%
Java Geliştirici (Java Developer)	java, spring, sql, hadoop, javascript	4,3%
Bulut Sistemleri Mühendisi (Cloud Systems Engineer)	cloud, linux, java, c#, aws	4,0%
Web Geliştirici (Web Developer)	html5, javascript, css, angularjs, jquery	3,7%
Ruby on Rails Geliştirici (Ruby on Rails Developer)	ruby on rails, ruby, python, backbonejs, nodejs	3,6%
Sistem Mühendisi (System Engineer)	linux, windows, python, ruby, java	3,5%
Python Geliştirici (Python Developer)	python, django, postgresql, php, jquery	3,3%
JavaScript Geliştirici (JavaScript Developer)	javascript, html5, css, angularjs, nodejs	3,2%
Ara Yüz Geliştirici (UI/UX Developer)	ui, javascript, ux, html5, css	3,0%
Yazılım Mimarı (Software Arcitecture)	agile, scrum, java, python, oop	2,8%
Kalite Güvence Mühendisi (Quality Assurance Engineer)	qa, sql, testing, java, c++	2,6%
Uygulama Geliştirici (Application Developer)	java, c++, .net, ruby, linux	2,5%
Android Geliştirici (Android Developer)	android, java, swift, c#, html5	2,3%
Toplam		**100%**



Şekil 23. Yazılım geliştirme uzmanlık rollerinin dağılım grafiği

Görülme sıklığı oranı %2'den daha düşük olan veya eksik yazılmış uzmanlık rolleri kendisine en yakın gruba atanmıştır. Elde edilen bulguların tabloda ve grafikte sunulan yazılım geliştirme uzmanlık rolleri ve bu rollere ait temel beceriler için tanımlayıcı olduğu öngörülmüştür. Bu nedenle, yazılım geliştirme uzmanlık rolleri için ek bir tanımlamaya ihtiyaç duyulmamıştır. Bu kapsamda, tabloda ve grafikte verilen uzmanlık rollerine ait temel beceriler, bu roller için tanımlayıcı bilgiler içermekte olup büyük ölçüde ait oluğu rolün kapsamını ortaya koymaktadır.

Yazılım geliştirme uzmanlık alanları için talep edilen becerilerin dağılımı göz önüne alındığında, elde edilen bulgular, programlama dillerinin yazılım geliştirme uzmanlık alanları için gerekli becerilerin temelini oluşturduğunu göstermiştir. Uzmanlık rollerine yönelik beceriler, her bir rol için görülme sıklığına göre büyükten küçüğe doğru sıralanmış ve bu şekilde tabloda verilmiştir. Örneğin, "Yazılım Mühendisi" rolü için gerekli olan beceriler "java", "python", "c #", "javascript" ve "c ++" olarak sıralanmıştır. Buna göre, yazılım mühendisleri için "java" en önemli birinci beceri, "python" en önemli ikinci beceri ve "c #" ise en önemli üçüncü beceridir.

3.3. Programlama Dillerine Yönelik Eğilimler

Bu aşamada, programlama dillerine yönelik talepleri ortaya koymak amacıyla iki farklı veri seti üzerinde gerçekleştirilen deneysel analiz ile Dünya ve Türkiye ölçeği için programlama dillerine yönelik eğilimler tespit edilmiştir. Elde edilen bulgular, Dünya ve Türkiye ölçeği için iki ayrı başlık altında verilmiştir.

3.3.1. Dünya Ölçeği İçin Programlama Dillerine Yönelik Eğilimler

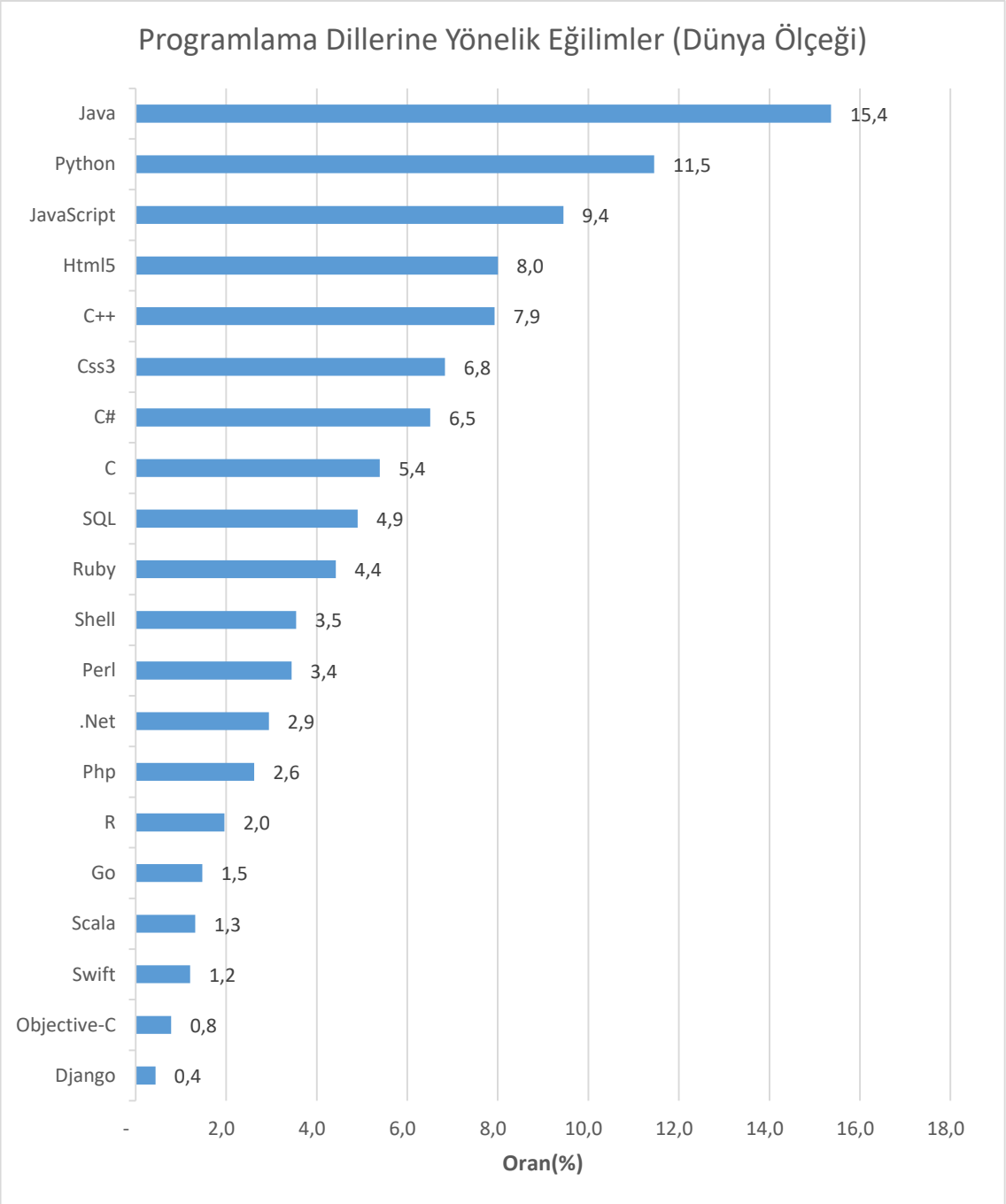
Analizin bu aşamasında, 1.veri seti üzerinde gerçekleştirilen kelime indekslemeye dayalı içerik analizi ile programlama dillerine yönelik talep ve eğilimler dünya ölçeği için tespit edilmiştir. Bu kapsamda en çok talep edilen 20 programlama dili ortaya konulmuştur. Elde edilen bulgular, programlama dillerine yönelik talep oranları ile birlikte Tablo 13 ve Şekil 24'de verilmiştir.

Tablo 13. Programlama dillerine yönelik eğilimlerin dağılımı (Dünya ölçeği)

No	Programlama Dilleri	Oran (%)
1	Java	15,4%
2	Python	11,5%
3	JavaScript	9,4%
4	Html5	8,0%
5	C++	7,9%
6	Css3	6,8%
7	C#	6,5%
8	C	5,4%
9	SQL	4,9%
10	Ruby	4,4%
11	Shell	3,5%
12	Perl	3,4%
13	.Net	2,9%
14	Php	2,6%
15	R	2,0%
16	Go	1,5%
17	Scala	1,3%
18	Swift	1,2%
19	Objective-C	0,8%
20	Django	0,4%

Elde edilen bulgular dikkate alındığında, Java programlama dilinin son dönemlerde görülen baskın ve lider rolünün devam ettiği gözlemlenmiştir. Bunu yanında son birkaç yılda Python programlama dilindeki inanılmaz yükselişte oldukça dikkat çekicidir. Şu anda Java ile birincilik yarışına giren Python programlama dilindeki yükseliş eğilimi bu şekilde devam ederse Python'un birincilik konusunda Java'yı zorlayacağı görülüyor.

Bunun yanında en çok talep edilen programlama dillerinin ilk 5 sırasında: Java, Python, Javascript, Html5 ve C++ yer almaktadır. Ayrıca, çoklu platformlar için uygulama geliştirme olanağı sunan Ruby, Shell, Go, Scala, Swift gibi programlama dilleri de sıralamanın içerisinde yer almaktadırlar.



Şekil 24. Programlama dillerine yönelik eğilimlerin dağılım grafiği (Dünya ölçeği)

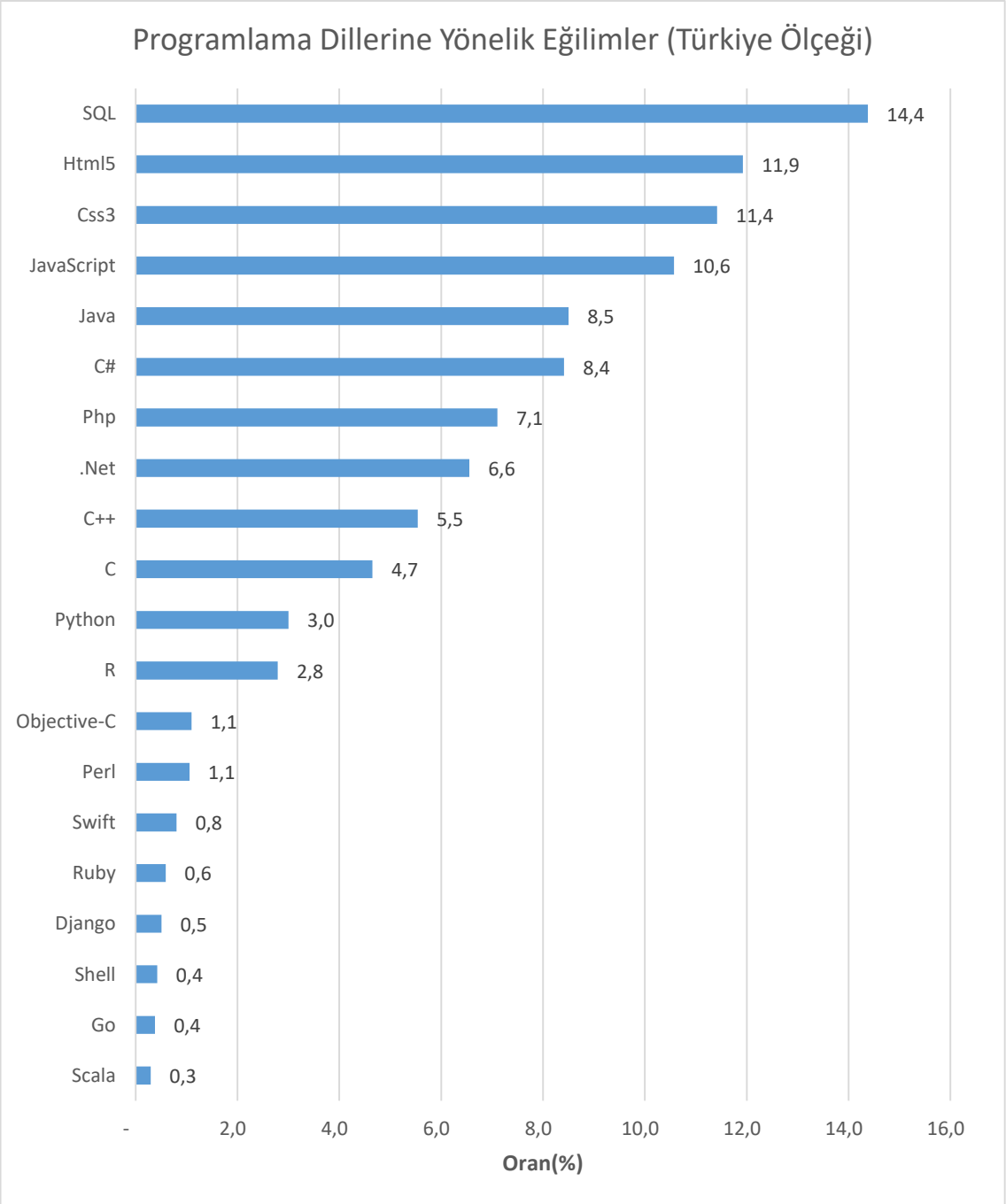
Özellikle derleyiciye ihtiyaç duymayan betik programlama dillerinin popülaritesi giderek artmaktadır. Hem mobil programlama hem de web uygulamaları açısından önemli avantajlar sunan betik programlama dilleri, esnek ve platformdan bağımsız yapısıyla yazılım geliştirme uzmanlık alanlarında yoğun olarak talep edilen beceriler arasında yer almaktadır.

3.3.2. Türkiye Ölçeği İçin Programlama Dillerine Yönelik Eğilimler

Deneysel analizin bu aşamasında, Türkiye ölçeği için programlama dillerine yönelik eğilimlerin tespit edilmesi amacıyla 2.veri seti üzerinde kelime indekslemeye dayalı bir içerik analizi gerçekleştirilmiştir. Analizin sonucunda dünya ölçeği için elde edilen ilk 20 programlama dilinin Türkiye ölçeğindeki dağılımı tespit edilmiştir. Bu kapsamda elde edilen bulgular Tablo 14 ve Şekil 25'de verilmiştir. Elde edilen bulgular programlama dilleri açısından dünyadaki eğilimlerden daha farklı bir sonuç ortaya koymuştur. Dünya ölçeğinde Java birinci sırada olmasına rağmen, Türkiye ölçeğinde bir sorgulama dili olan SQL birinci sırada yer almaktadır. Bunu yanında son birkaç yılda Python programlama dilindeki inanılmaz yükselişe rağmen bu programlama dilinin Türkiye ölçeğinde ilk 10'a girememesi diğer dikkat çekici bir sonuç olmuştur.

Tablo 14. Programlama dillerine yönelik eğilimlerin dağılımı (Türkiye ölçeği)

No	Programlama Dilleri	Oran (%)
1	SQL	14,4%
2	Html5	11,9%
3	Css3	11,4%
4	JavaScript	10,6%
5	Java	8,5%
6	C#	8,4%
7	Php	7,1%
8	.Net	6,6%
9	C++	5,5%
10	C	4,7%
11	Python	3,0%
12	R	2,8%
13	Objective-C	1,1%
14	Perl	1,1%
15	Swift	0,8%
16	Ruby	0,6%
17	Django	0,5%
18	Shell	0,4%
19	Go	0,4%
20	Scala	0,3%



Şekil 25. Programlama dillerine yönelik eğilimlerin dağılım grafiği (Türkiye ölçeği)

Bunun yanında en çok talep edilen programlama dillerinin ilk 5 sırasında: SQL, Html5, Css3, JavaScript, Java yer almaktadır. Ayrıca, çoklu platformlar için uygulama geliştirme olanağı sunan Ruby, Shell, Go, Scala, Swift gibi yeni programlama dillerinin sıralamayı çok gerilerden takip ettiği görülmüştür.

3.4. Programlama Dilleri Arasındaki Birliktelik İlişkileri

Günümüzün yazılım geliştirme ortamlarında sadece tek bir programlama dilini bilmek, yazılım endüstrisindeki ortaya çıkan taleplerin karşılanması açısından yetersiz kalmaktadır. Bundan dolayı yeni nesil yazılım geliştirme eğilimlerine yönelik bilgi ve beceriler, aynı anda birden fazla programlama diline hâkim olmayı zorunlu kılmaktadır.

Bu bağlamda, programlama dillerinin birlikteliklerine yönelik talepleri ortaya koymak amacıyla iki farklı veri seti üzerinde gerçekleştirilen deneysel analiz ile Dünya ve Türkiye ölçeği için talep ve eğilimler tespit edilmiştir. Elde edilen bulgular, Dünya ve Türkiye ölçeği için iki ayrı başlık altında verilmiştir.

3.4.1. Dünya Ölçeği İçin Programlama Dilleri Arasındaki Birliktelik İlişkileri

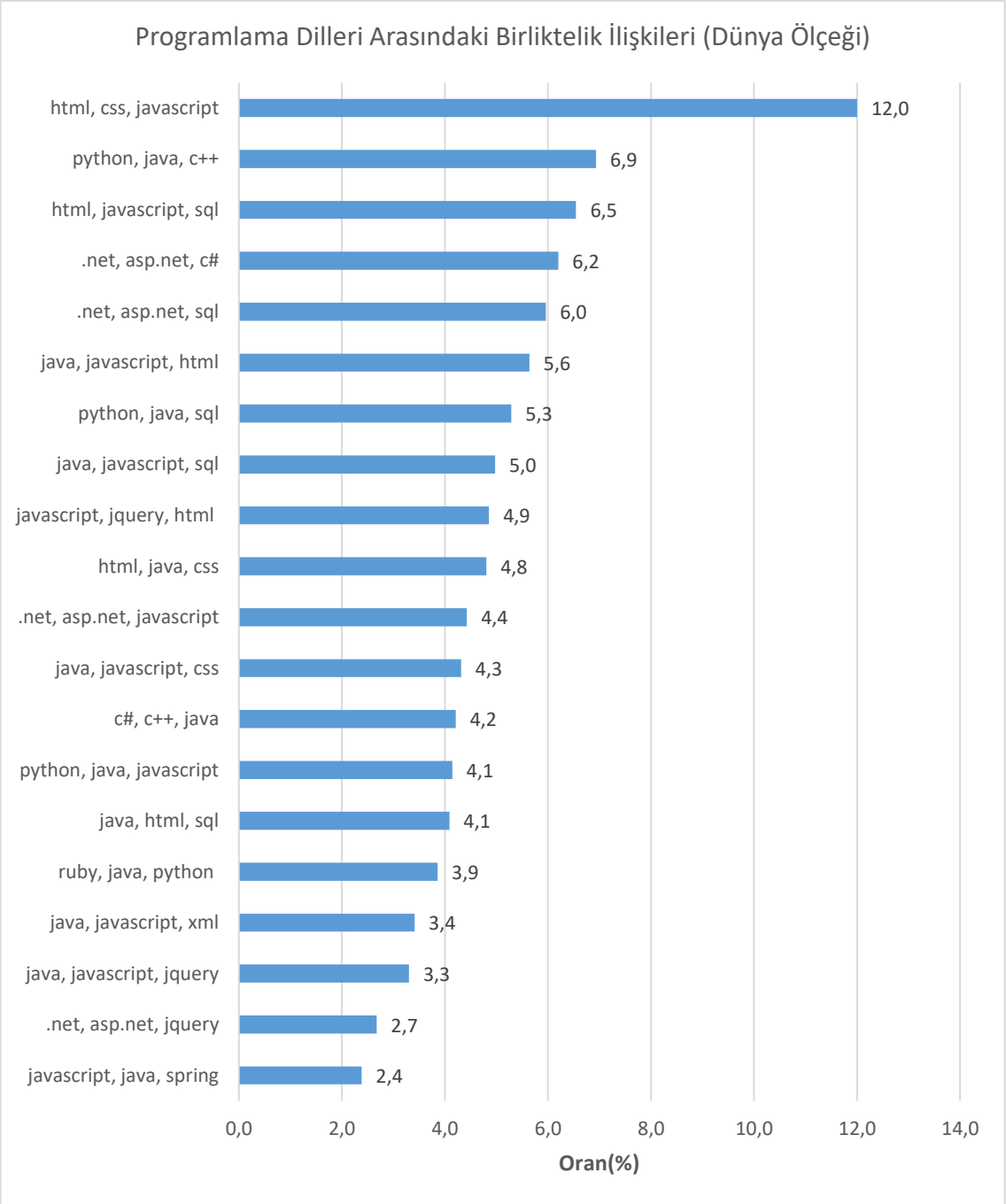
Gerçekleştirilen deneysel analizin bu aşamasında, dünya ölçeği için programlama dilleri arasındaki birliktelik ilişkilerinin tespit edilmesi amaçlanmıştır. Bunun için önceki aşamada elde edilen 20 programlama dili ile oluşturulan üçlü grupların 1.veri seti üzerindeki geçiş sayıları hesaplanmıştır. Bu kapsamda gerçekleştirilen analiz ile dünya ölçeği için programlama dillerinin birlikte kullanımına yönelik eğilimler tespit edilmiş ve en çok talep edilen ilk 20 birliktelik eğilimi Tablo 15 ve Şekil 26'da ayrıntılı olarak verilmiştir

Elde edilen bulgular dikkate alındığında, "html, css, javascript" üçlüsünün açık arayla birinci sırada yer aldığı gözlenmiştir. Özellikle css ve javascript'in birlikte kullanımı ile birlikte daha da güçlenen html5 ve bunların oluşturduğu üçlü grubun yazılım geliştirme uzmanlık alanları için önemli bir beceri olduğu gözlemlenmiştir. Hem web hem de mobil ortamlar için uygulama geliştirme olanağı sağlayan bu üçlü grup, yazılım geliştirme becerileri arasında çoğunlukla üst sıralarda yer almaktadır. Bunun yanında programlama dillerine yönelik olarak en çok talep edilen üçlü grupların ilk 5 sırasında: "html, css, javascript", "python, java, c++", "html, javascript, sql", ".net, asp.net, c#" ve ".net, asp.net, sql" yer almaktadır.

Tablo 15. Dünya ölçeği için programlama dilleri arasındaki birliktelik dağılımları

No	Programlama Dilleri Grubu	Oran
1	html, css, javascript	12,0%
2	python, java, c++	6,9%
3	html, javascript, sql	6,5%
4	.net, asp.net, c#	6,2%
5	.net, asp.net, sql	6,0%
6	java, javascript, html	5,6%
7	python, java, sql	5,3%
8	java, javascript, sql	5,0%
9	javascript, jquery, html	4,9%
10	html, java, css	4,8%
11	.net, asp.net, javascript	4,4%
12	java, javascript, css	4,3%
13	c#, c++, java	4,2%
14	python, java, javascript	4,1%
15	java, html, sql	4,1%
16	ruby, java, python	3,9%
17	java, javascript, xml	3,4%
18	java, javascript, jquery	3,3%
19	.net, asp.net, jquery	2,7%
20	javascript, java, spring	2,4%

Ayrıca üçlü gruplara yönelik bulgular, özellikle web ve mobil tabanlı uygulama geliştirme ortamlarının daha ön planda olduğunu ortaya koymaktadır. Bu nedenle yeni nesil yazılım geliştirme ortamlarında çoklu platformlara özgü programlama dillerinden oluşan kombinasyonlar daha çok talep edilmektedir.



Şekil 26. Dünya ölçeği için programlama dilleri birliktelik dağılımları grafiği

3.4.2. Türkiye Ölçeği İçin Programlama Dilleri Arasındaki Birliktelik İlişkileri

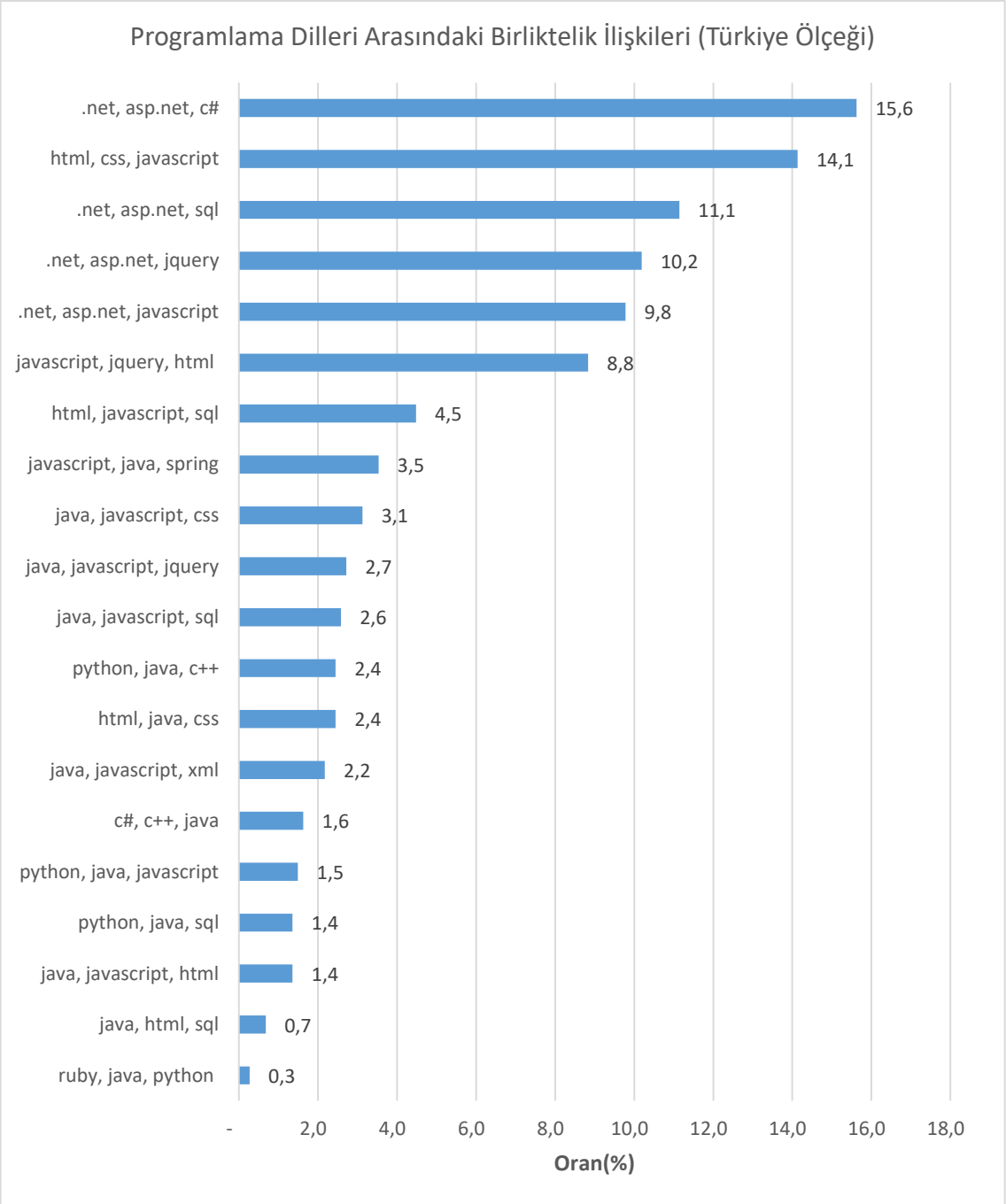
Analizin bu aşamasında ise Türkiye ölçeği için programlama dilleri arasındaki birliktelik ilişkilerinin tespit edilmesi amacıyla programlama dillerinden oluşan üçlü gruplara yönelik olarak en çok talep edilen ilk 20 eğilim ortaya konulmuştur. Bunun için

önceki aşamada Dünya ölçeği için elde edilen üçlü grupların Türkiye ölçeğindeki dağılımları 2.veri seti üzerinde analiz edilmiştir. Analiz sonucunda elde edilen bulgular, Tablo 16 ve Şekil 27'de ayrıntılı olarak verilmiştir.

Elde edilen bulgular dikkate alındığında, ".net, asp.net, c#" üçlüsünün birinci sırada ve "html, css, javascript" üçlüsünün birinciye yakın bir oranla ikinci sırada yer aldığı gözlemlenmiştir. Türkiye ölçeğinde ".net", "asp.net" ve "c#" programlama dillerine dayalı eğilimlerin baskınlığı açık bir şekilde görülmektedir. Bunun yanında programlama dillerine yönelik olarak en çok talep edilen üçlü grupların ilk 5 sırasında: ".net, asp.net, c#", "html, css, javascript", ".net, asp.net, sql", ".net, asp.net, jquery #" ve ".net, asp.net, javascript" yer almaktadır. Elde edilen bulgular aynı zamanda Türkiye ölçeğindeki web tabanlı eğilimlerin mobil tabanlı eğilimlere göre daha baskın olduğunu ortaya koymuştur.

Tablo 16. Türkiye ölçeği için programlama dilleri arasındaki birliktelik dağılımları

No	Programlama Dilleri Grubu	Oran
1	.net, asp.net, c#	15,6%
2	html, css, javascript	14,1%
3	.net, asp.net, sql	11,1%
4	.net, asp.net, jquery	10,2%
5	.net, asp.net, javascript	9,8%
6	javascript, jquery, html	8,8%
7	html, javascript, sql	4,5%
8	javascript, java, spring	3,5%
9	java, javascript, css	3,1%
10	java, javascript, jquery	2,7%
11	java, javascript, sql	2,6%
12	html, java, css	2,4%
13	python, java, c++	2,4%
14	java, javascript, xml	2,2%
15	c#, c++, java	1,6%
16	python, java, javascript	1,5%
17	java, javascript, html	1,4%
18	python, java, sql	1,4%
19	java, html, sql	0,7%
20	ruby, java, python	0,3%



Şekil 27. Türkiye ölçeği için programlama dilleri birliktelik dağılımları grafiği

4. İRDELEME

Yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerinin olasılıksal konu modelleme yordamıyla belirlenmesi amacıyla gerçekleştirilen bu çalışmada, yazılım endüstrisindeki güncel eğilimlerin anlaşılmasına yönelik önemli çıkarımlar elde edilmiştir.

Web ve mobil tabanlı uygulama geliştirme ortamlarının yazılım geliştirme uzmanları için hedef platform olarak ortaya çıktığı, ancak mobil platformlara yönelik yazılım geliştirme araçları açısından bir olgunlaşma sürecine ihtiyaç duyulduğu görülmüştür. Elde edilen bulgular, özellikle büyük veri ve bulut tabanlı sistemlere dayalı yazılım geliştirme becerilerine yönelik talep ve eğilimlere belirgin bir şekilde vurgu yapmaktadır. Bu bağlamda, çalışmadan elde edilen bulgular içerisindeki yüksek orana sahip eğilimler geçmişteki benzer çalışmaların ışığında yeniden değerlendirilmiş ve yorumlanmıştır.

4.1. Web Tabanlı Uygulamaların Hâkimiyeti

Elde edilen bulgular, yazılım geliştirme bilgi ve becerileri açısından web tabanlı uygulamaların bu alandaki hâkimiyetini belirgin bir şekilde ortaya koymaktadır. Hem elde edilen konular içerisindeki dağılım bakımından (Betik programlama dilleri %4.7, JavaScript geliştirme modelleri %4.2, Web servisleri geliştirme %3.8) hem de programlama dilleri (javascript, html5, css3, php, asp.net) açısından bakıldığında web tabanlı uygulama geliştirme becerilerinin yazılım geliştirme uzmanlık alanlarında oldukça baskın beceriler olduğu söylenebilir. Ayrıca web tabanlı uygulama geliştirme ortamlarında betik programlama dillerini ve javascript kütüphanelerini kapsayan becerilerin de yoğun olarak talep edildiği gözlemlenmiştir [1,15,35,115,116].

Ayrıca web tabanlı geliştirilen uygulamaların html5, javascript ve css3 desteğiyle direk olarak mobil ortamlara uygulanabilir olması, mobil kullanım açısından da web tabanlı uygulamalara olan uzmanlık taleplerini önemli ölçüde artırmaktadır. Yeni nesil geliştirme araçları ile tasarlanan web uygulamaları aynı zamanda birer mobil uygulama olarak ta kullanılabilmektedir. Elde edilen bulgular dikkate alındığında, birlikte kullanılan

programlama dilleri arasında birinci sırada “html, css, javascript” üçlüsünün yer alması web tabanlı uygulamaların hâkimiyetini ortaya koyan bir başka önemli deneysel çıkarımdır [1,15,35,115,117].

4.2. Betik Programlama Dillerine Geçiş

Çalışmada elde edilen önemli çıkarımlardan birisi de, web ve mobil tabanlı uygulamalarda görülen hızlı teknolojik dönüşüm neticesinde, kullanılan yazılım geliştirme mimarilerinin ve programlama dillerinin önemli ölçüde değişiyor olmasıdır. Yazılım geliştirme ve programlama becerileri içerisinde “Betik programlama dilleri” konusunun %4.7 oranla birinci sırada yer alması bu öngörüyü destekler niteliktedir. Bu durum, yeni nesil yazılım geliştirme ortamlarında derleyici tabanlı programlama dillerinin artık yerini betik programlama dillerine bıraktığını göstermektedir [1,13,15,35,40-42].

Özellikle derleyiciye ihtiyaç duymayan betik programlama dillerinin popülaritesi giderek artmaktadır. Hem mobil programlama hem de web uygulamaları açısından önemli avantajlar sunan betik programlama dilleri, esnek ve platformdan bağımsız yapısıyla yazılım geliştirme uzmanlık alanlarında yoğun olarak talep edilen beceriler arasında yer almaktadır [1,13,15,35,40-42].

Web ve mobil tabanlı uygulamalarda görülen hızlı teknolojik dönüşüm sadece programlama dillerinde değil, aynı zamanda yazılım geliştirme mimarilerinde de değişikliğe yol açmıştır. Elde edilen bulgular, çevik (agile) ve scrum geliştirme modellerinin yüksek oranda talep edilen beceriler arasında olduğunu göstermiştir. Bu yeni nesil yazılım geliştirme modelleri, geleneksel yazılım geliştirme yöntemlerinden daha esnektir ve hızlı değişen ortamlara daha iyi uyum sağlar. Bu yönüyle çevik geliştirme modelleri, web ve mobil tabanlı yazılım geliştirme ortamları için yüksek oranda talep edilen beceriler arasındadır [1,35,42].

4.3. Mobil Uygulamalara Yönelim

Günümüzde yaşanan teknolojik dönüşümle birlikte masaüstü ve web tabanlı uygulamalar yerini mobil uygulamalara bırakmaya başlamıştır. Bu bakış açısıyla, ilerleyen

zamanlarda web tabanlı uygulamaların mobil cihazlar ile tamamen bütünleşeceği net olarak söylenebilir. Mobil uygulamaların geniş kullanım alanlarına sahip olması, bu alandaki uygulamaların çeşitliliğini her geçen gün artırmaktadır. Hemen hemen her yaştan insanın hayatında yer alan çok farklı özelliklerdeki mobil uygulamalar, bu alanlarda talep edilen teknik bilgi ve beceriler açısından, yazılım geliştirme uzmanları için büyük bir istihdam potansiyeli ve yeni kariyer fırsatları oluşturmaktadır [1,9,35,115,118].

Çalışmanın bulgularından da anlaşılacağı üzere, iş platformlarında mobil geliştiricilere (mobile developers) yönelik kapsamlı kariyer seçenekleri yer almaktadır. Elde edilen bulgulara göre yazılım geliştirme uzmanlık rolleri içerisinde Mobil Geliştirici (Mobile Developer) uzmanlık rolünün %10,6'lık oranla ikinci sırada yer aldığı görülmüştür. Farklı mimari ve yaklaşımlarla mobil yazılım geliştirme imkânı sunan programlama araçları da (swift, java, javascript, objective-c, vb.) yazılım geliştirme uzmanlığı için yüksek oranda talep edilen beceriler arasında yer almaktadır [1,35,115,118].

Buna ek olarak, html5, javascript, css3 programlama dillerinin mobil tabanlı web uygulamalarının içeresinde aktif şekilde yer alması, web yazılım uzmanlarının mobil tabanlı web uygulamaları geliştirmesine de olanak sağlamıştır. Bu nedenle web geliştiricilerin aynı zamanda mobil geliştiriciler olduğu göz önüne alındığında, mobil uygulama geliştirmeye yönelik eğilim ve taleplerin görünenden daha fazla olduğu söylenebilir.

Mobil tabanlı uygulamalar, mobil cihazların hesaplama gücü sınırlı olduğu için daha küçük ölçekli ve daha az kaynak kullanan uygulamalardır. Bu nedenle mobil tabanlı uygulamaların geliştirilmesi daha farklı teknik yaklaşımlar ve beceriler gerektirir. Bu bağlamda mobil uygulamalar, yazılım geliştirme uzmanları için büyük kariyer potansiyeline sahip ve gelişime açık bir alan olarak görülmektedir. Elde edilen bulgular, mobil tabanlı yazılım geliştirme becerileri açısından ciddi bir talebin varlığını ortaya koymaktadır [115,118].

4.4. Büyük Veri Odaklı Uygulama Geliştirme

Yazılım geliştirme uzmanlığı, veri odaklı bir disiplin ve veri biliminin ayrılmaz bir parçasıdır. Sosyal ağlar, nesnelerin interneti, mobil uygulamalar ve bilgi paylaşım

platformları gibi güncel teknolojilerle birlikte büyük veri sistemlerinin ortaya çıkması, bu büyük veri setlerini işleyebilen yazılım uygulamalarının geliştirilmesi açısından yazılım geliştirme uzmanlık alanlarına yeni bir çalışma sahası daha eklemiştir [1,17,35,82,119].

Büyük veri odaklı uygulamalarda, yazılım geliştirme uzmanları için en önemli süreç yazılım sistemlerinin tasarımı ile ilgilidir. Büyük veri odaklı uygulamalar için ölçeklenebilir yazılım mimarilerinin tasarlanması ve uygulanması, yazılım geliştirme uzmanlığı açısından farklı bilgi ve beceriler gerektirmektedir [1,17].

Çalışmada elde bulgular da bu öngörümüzü destekler niteliktedir. Çünkü analiz sonucunda elde konular içerisinde altı tanesinin veri tabanları ve büyük veri odaklı becerilere dayandığı gözlemlenmiştir. Bu kapsamda elde edilen bulgular: Veri tabanı bilgisi (Database knowledge) %2,7; Büyük veri işleme (Big data processing) %1,6, Veri güvenliliği ve gizliliği (Data security and privacy) %0,8; Gerçek zamanlı programlama (Real-time programming) %2,1 ;Ölçeklenebilir programlama (Scalable programming) %1,2 ve Büyük veri odaklı geliştirme (Big data development) %1,1 olarak sıralanmaktadır. Büyük veriyle ilgili bu konuların toplam ağırlığı ise %9,5 olarak hesaplanmıştır. Ayrıca "analitik ve mantıksal beceriler (%13)" başlığının da önemli ölçüde büyük veri odaklı olduğunu göz önüne alırsak, veri odaklı becerilerin toplam ağırlığının yaklaşık %23'e kadar çıktığı görülür. Bu çıkarımlar, büyük veri odaklı bilgi ve becerilen yazılım geliştirme uzmanlık alanları için önemini net bir şekilde ortaya koymaktadır [1,17,35,82,119].

4.5. Bulut Tabanlı Uygulama Geliştirme

Bulut teknolojisi, tüm uygulama, program ve verilerinizin hiçbir kuruluma ihtiyaç duymadan, internete bağlı uzak sistemler üzerinde depolanmasını, çalıştırılmasını ve yönetilmesini sağlayan hizmetlerin bütünüdür. Özellikle son dönemlerde yazılım geliştirme uzmanları için yeni kariyer fırsatları sunan bulut tabanlı uygulamalar her geçen gün daha fazla adından söz ettiren bir teknoloji olarak karşımıza çıkıyor [1,37-39,119].

Bu kapsamdaki deneysel bulgular, bulut tabanlı uygulama geliştirme becerilerinin, yazılım uzmanlarının sahip olması gereken temel nitelikler arasında olduğunu göstermiştir. Bulut tabanlı uygulama geliştirmeye yönelik eğilimleri ortaya koyan bulgular: Bulut tabanlı sistemler (Cloud-based systems) %1,2, Çok platformlu geliştirme (Multi-platform

Development) %3,4, Bulut tabanlı geliştirme (Cloud-based Development) %2,3 şeklinde sıralanmakta olup bu konuların toplam ağırlığı yüzde %7'ye eşittir.

Elde edilen bulgular, yeni nesil yazılım geliştirme eğilimlerinden olan "bulut tabanlı yazılım geliştirme" becerisinin, yazılım geliştirme uzmanları tarafından sahip olunması gereken önemli nitelikler arasında olduğunu göstermiştir. Buna göre gelecekte, bilgisayar hard disklerinin yerini çevrim içi bulut sistemlerin alacağı konusunda yaygın bir ön görü hâkimdir. Bundan dolayı bulut tabanlı uygulama geliştirme becerilerinin ilerleyen zamanlarda, yazılım uzmanları için olmazsa olmaz nitelikler arasında olacağı öngörülmektedir [119,120].

4.6. Programlama Dillerine Yönelik Talep ve Eğilimler

Yazılım geliştirme uzmanlık alanlarına yön veren en önemli eğilimlerden birisi de programlama dilleridir. Programlama dillerinde yaşanan gelişmeler, yenilikler ve yeni eğilimler yazılım geliştirme uzmanlık alanlarını yakından ilgilendiren olaylardır. Elde edilen bulgulara dayanarak, dünya genelinde Java programlama dilinin son dönemlerde görülen baskın ve lider rolünün devam ettiği söylenebilir [1,35,40-43].

Bunu yanında son birkaç yılda Python programlama dilindeki inanılmaz yükseliş yazılım uzmanları için oldukça önemli bir gelişmedir. Şu anda Java ile birlikte zirveyi zorlayan Python programlama dilindeki yükseliş eğilimi bu şekilde devam ederse, Python'un birincilik konusunda Java'yı geçebileceği öngörülmektedir. Bunun yanında dünya ölçeğinde en çok talep edilen programlama dillerinin ilk 5 sırasında: Java, Python, Javascript, Html5 ve C++ yer almaktadır. Ayrıca, çoklu platformlar için uygulama geliştirme olanağı sunan Perl, Ruby, Objective-C, Shell, Go, Scala, Swift gibi yeni programlama dilleri de yazılım geliştirme uzmanlarının sahip olması gereken önemli beceriler arasında yer almaktadır [1,35,40-43].

Bulut, mobil ve büyük veri uygulamalarında yaşanan talep patlamasına cevap verebilmek için oluşturulan programlama dilleri sadeleştikçe, yazılım kodlamada yeni bir yapılanma döneminin başladığı söylenebilir. Ayrıca Html5'in programlama dilleri içerisinde dördüncü sıraya yerleşmesi klasik Html'nin son yıllardaki yükselişinin göstergesidir. Css3 ve Bootstrap teknolojileriyle birlikte yeni bir evrim geçiren Html5,

günümüzün en önemli uygulama geliştirme araçlarından birisi olmuştur. Daha önceden bir biçimlendirme aracı olarak kabul edilen Html, bu süreç sonunda etkin olarak kullanılan programlama dilleri arasındaki yerini almıştır.

Elde edilen bulgular, yeni nesil yazılım geliştirme süreçlerinde ve iş akışlarında yazılım geliştirme uzmanlığı açısından sadece tek bir programlama dilini bilmenin, günümüzün teknolojik ortamlarında sürekli gelişen ve büyüyen yazılım odaklı taleplerin karşılanması açısından yetersiz kalmadığını göstermiştir. Bu kapsamdaki bulgular, "html, css, javascript" üçlüsünün açık arayla birinci sırada olduğunu göstermiştir. Hem web hem de mobil ortamlar için uygulama geliştirme olanağı sağlayan bu üçlü grup, yazılım geliştirme becerileri arasında en çok talep edilen nitelikler arasında yer almaktadır.

Bunun yanında programlama dillerine yönelik olarak en çok talep edilen üçlü grupların ilk 5 sırasında: "html, css, javascript", "python, java, c++", "html, javascript, sql", ".net, asp.net, c#" ve ".net, asp.net, sql" yer almaktadır. Üçlü gruplara yönelik bulgular, özellikle web ve mobil tabanlı uygulama geliştirme ortamlarının daha ön planda olduğunu ortaya koymaktadır. Ayrıca deneysel bulguların, derleyiciden bağımsız olarak kullanılabilen betik programlama dillerine ve bu kapsamdaki programlama becerilerinin önemine de vurgu yaptığı gözlemlenmiştir [1,35,40-43].

5. SONUÇLAR

Bu tez çalışmasında, yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerinin olasılıksal konu modelleme yordamıyla belirlenmesi üzerine çalışılmıştır. Çalışmanın amacı, son yıllarda yazılım endüstrisinde yaşanan teknolojik dönüşüm doğrultusunda, sürekli olarak yenilenen yazılım geliştirme ortamları için talep edilen uzman bilgi alanları, teknik beceriler, yetkinlikler ve yazılım geliştirme uzmanlığının temelini oluşturan programlama dillerine yönelik talep ve eğilimlerin tespit edilmesine yöneliktir. Bu amaç kapsamında, GDA tabanlı olasılıksal konu modelleme yordamı kullanılarak yazılım geliştirme uzmanlığına yönelik çevrimiçi iş ilanlarından oluşturulan veri seti üzerinde anlamsal analize dayalı bir çalışma gerçekleştirilmiştir.

Çalışmanın sonucunda, yeni nesil yazılım geliştirme eğilimlerine yönelik bilgi ve becerileri en anlamlı düzeyde ortaya koyan 52 temel konu keşfedilmiştir. Yazılım endüstrisindeki güncel talep ve eğilimleri ortaya koyan bu 52 konu, temel disiplinlerle ve yazılım geliştirme uzmanlık alanlarıyla ilişkilendirilerek, alan bilgi ve becerilerine özgü bir taksonomi oluşturulmuştur. Bu şekilde, yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerileri için dört ana yetkinlik alanı belirlenmiştir. Bu yetkinlik alanlarına atanan konular: teknik alan bilgi ve becerileri (%23); yazılım geliştirme ve programlama becerileri (%52); analitik ve mantıksal beceriler (%12) ve teknik olmayan beceriler (%13) şeklinde bir frekans dağılımı göstermiştir.

Yeni nesil yazılım geliştirme eğilimlerine yönelik uzman bilgi ve becerilerini yansıtan önemli göstergelerden birisi de bu alandaki uzmanlık rolleridir. Bu bakış açısıyla, yazılım geliştirme uzmanlık alanları için 20 temel rol belirlenmiş ve bu rollere ait temel bilgi ve beceriler tespit edilmiştir. En fazla talep edilen ilk beş uzmanlık rolü sırasıyla: “Yazılım Mühendisi” (Software Engineer) (%12,0), "Mobil Geliştirici" (Mobile Developer) (% 10,6), “Ön Yüz Geliştirici” (Frontend Developer) (%9,0), “Yazılım Geliştirici” (Software Developer) (%7,1) ve “Genel Geliştirici” (Full Stack Developer) (%6,3) şeklinde bir dağılım göstermiştir.

Yazılım geliştirme uzmanlık alanlarına yön veren en önemli eğilimlerden birisi de programlama dilleridir. Bu bağlamda, programlama dillerine yönelik eğilimleri ortaya koymak amacıyla iki farklı veri seti üzerinde gerçekleştirilen deneysel analiz ile Dünya ve Türkiye ölçeğinde en çok talep edilen 20 programlama dili tespit edilmiştir. Dünya ölçeği için, en çok talep edilen ilk 5 programlama dili sırasıyla: Java, Python, Javascript, Html5 ve C++ olarak tespit edilmiştir. Türkiye ölçeği içinse, en çok talep edilen ilk 5 programlama dili sırasıyla: SQL, Html5, Css3, JavaScript, Java olarak tespit edilmiştir.

Yazılım geliştirme uzmanlık alanlarına yön veren diğer önemli bir eğilim de farklı programlama dillerinin birlikte talep edilmesidir. Bu kapsamda, programlama dillerinin birlikteliklerine yönelik talepleri ortaya koymak amacıyla Dünya ve Türkiye ölçeği için programlama dillerinden oluşan üçlü gruplara yönelik en çok talep edilen ilk 20 eğilim tespit edilmiştir. Dünya ölçeği için programlama dillerine yönelik en çok talep edilen ilk 5 üçlü grup: “html, css, javascript”, “python, java, c++”, “html, javascript, sql”, “.net, asp.net, c#” ve “.net, asp.net, sql” olarak tespit edilmiştir. Türkiye ölçeği içinse, programlama dillerine yönelik en çok talep edilen ilk 5 üçlü grup: “.net, asp.net, c#”, “html, css, javascript”, “.net, asp.net, sql”, “.net, asp.net, jquery #” ve “.net, asp.net, javascript” olarak tespit edilmiştir.

6. ÖNERİLER

Tüm bilimsel çalışmalarda olduğu gibi yapılan bu deneysel çalışmada da belirli kısıtlamalarla karşılaşılmıştır. İlk olarak, deneysel veri setindeki metinlerin miktarı ve çeşitliliğinin keşfedilen konuların anlam düzeyini yansıtması açısından analizin başarısını yakından etkilediği görülmüştür. Olasılıksal konu modellemeye dayalı anlamsal analizde, verilerin miktar ve uygunluğunun, analizin başarı oranını yakından etkilediği görülmüştür. İkincisi, bu deneysel çalışma yalnızca İngilizce metinlerle sınırlıdır. Uygulanan yöntembilim, çalışmada yapılan analizin niteliğinden dolayı çoklu dillere uygulanamamıştır. Bu kapsamda, uygulanan yöntem ve süreçlerin diğer dillere yönelik gelecek çalışmalar için bir temel oluşturabileceği öngörülmektedir.

Diğer bir kısıt, gerçekleştirilen çalışma birçok değişken parametrenin farklı kombinasyonlarını kullanan deneysel analize dayanmaktadır. Bu nedenle, optimal parametrelerin seçimi GDA için hala çözülmesi gereken bir problem olarak ortaya çıkmaktadır. Bu konuda daha fazla deneysel çalışmaya ihtiyaç olduğu görülmektedir. Özellikle, en anlamlı düzeyde konu sayısının (K) belirlenmesi çok sayıda test amaçlı analizi gerektirmektedir. Bu nedenle, deneysel parametreler ve elde edilen sonuçların daha iyi yorumlanması açısından bu alanda daha fazla destekleyici çalışmalara gerek duyulmaktadır.

Bu sınırlamaların ötesinde yazılım geliştirme uzmanlık alanları, teknolojik gelişmeler ve bu alandaki sürekli değişen eğilimler doğrultusunda yeniliğe açık bir araştırma ve uygulama alanıdır. Yapılan çalışma farklı vektör azaltma teknikleri, ön işleme aşamaları, anlamsal analiz yaklaşımları ve yeni hiyerarşik yöntemlerle desteklenerek daha farklı kapsamda çalışmalar yapılabilir. Uygulanan yöntembilim, farklı endüstriler (finans, eğitim, sağlık, vb.) veya teknik alanlara yönelik eğilimleri ve talepleri araştırmak için yeni destekleyici yaklaşımlarla geliştirilebilir. Bu çalışmada kullanılan yöntembilim, bloglar, sosyal ağlar ve forumlar gibi farklı çevrimiçi platformlardan bilgi çıkarımına dayalı çalışmalara da uyarlanabilir.

7. KAYNAKÇA

1. Gurcan, F. ve Kose, C., Analysis of Software Engineering Industry Needs and Trends: Implications for Education, International Journal of Engineering Education, 33,4 (2017) 1361-1368.

2. Abrahamsson, P., Salo, O., Ronkainen, J. ve Warsta, J., Agile software development methods: Review and analysis, arXiv preprint arXiv:1709.08439, 2017.

3. Akman, G. ve Yilmaz, C., Innovative capability, innovation strategy and market orientation: an empirical analysis in Turkish software industry, International Journal of Innovation Management, 12,01 (2008) 69-111.

4. Lee, K., ve Mirchandani, D., Dynamics of the importance of IS/IT skills, Journal of Computer Information Systems, 50,4 (2010) 67-78.

5. Dinh, H. T., Lee, C., Niyato, D. ve Wang, P., A survey of mobile cloud computing: architecture, applications, and approaches, Wireless communications and mobile computing, 13,18 (2013) 1587-1611.

6. Brynjolfsson, E. ve Hitt, L. M., Beyond computation: Information technology, organizational transformation and business performance, The Journal of Economic Perspectives, 14,4 (2000) 23-48.

7. Harter, D. E., Krishnan, M. S. ve Slaughter, S. A., Effects of process maturity on quality, cycle time, and effort in software product development, Management Science, 46,4 (2000) 451-466.

8. Pressman, R. S., Software engineering: a practitioner's approach, Palgrave Macmillan, 2005.

9. Moreno, A. M., Sanchez-Segura, M. I., Medina-Dominguez, F. ve Carvajal, L., Balancing software engineering education and industrial needs, Journal of systems and software, 85,7 (2012) 1607-1620.

10. Lee, D. M., Trauth, E. M. ve Farwell, D., Critical skills and knowledge requirements of IS professionals: a joint academic/industry investigation, MIS quarterly, (1995) 313-340.

11. Kim, Y., Hsu, J. ve Stern, M., An update on the IS/IT skills gap, Journal of Information Systems Education, 17,4 (2006) 395.

12. Lee, S., Koh, S., Yen, D. ve Tang, H. L., Perception gaps between IS academics and IS practitioners: an exploratory study, Information & Management, 40,1 (2002) 51-61.

13. Chen, Y., Dios, R., Mili, A., Wu, L. ve Wang, K., An empirical study of programming language trends, IEEE software, 22,3 (2005) 72-79.

14. Prabhakar, B., Litecky, C. R. ve Arnett, K., IT skills in a tough job market, Communications of the ACM, 48,10 (2005) 91-94.

15. Aken, A., Litecky, C., Ahmad, A. ve Nelson, J., Mining for computing jobs, IEEE software, 27,1 (2010) 78-85.

16. Smith, D. ve Ali, A., Analyzing computer programming job trend using web data mining, Issues in Informing Science and Information Technology, 11 (2014) 203-214.

17. Debortoli, S., Müller, O. ve vom Brocke, J., Comparing business intelligence and big data skills, Business & Information Systems Engineering, 6,5 (2014) 289-300.

18. Ruparelia, N. B., Software development lifecycle models, ACM SIGSOFT Software Engineering Notes, 35,3 (2010) 8-13.

19. Medvidovic, N. ve Taylor, R. N., Software architecture: foundations, theory, and practice. In Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, Mayıs 2010, Cape Town, ACM, 2, 471-472.

20. Fuggetta, A., ve Di Nitto, E., Software process, In Proceedings of the on Future of Software Engineering, Mayıs 2014, ACM, 1-12

21. Hanson, M. D., The Client/Server Architecture. Server Management, 3, 2000.

22. Heineman, G. T. ve Councill, W. T., Component-based software engineering, Putting the pieces together, 5, addison-westley, 2001.

23. Schmidt, D. C., Model-driven engineering, Computer-IEEE Computer Society, 39,2 (2006) 25.

24. Namiot, D. ve Sneps-Sneppe, M., On micro-services architecture, International Journal of Open Information Technologies, 2,9 (2014) 24-27.

25. Gorton, I., Essential software architecture, Springer Science & Business Media, 2006.

26. Booch, G., Object-oriented development, IEEE transactions on Software Engineering, 2 (1986) 211-221.

27. Marks, E. A. ve Bell, M., Service Oriented Architecture (SOA): a planning and implementation guide for business and technology, John Wiley & Sons, 2008.

28. Magdaleno, A. M., Werner, C. M. L., ve De Araujo, R. M., Reconciling software development models: A quasi-systematic review, Journal of Systems and Software, 85,2 (2012) 351-369.

29. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ve Kern, J., Manifesto for agile software development, 2001.

30. Beynon-Davies, P., Carne, C., Mackay, H. ve Tudhope, D., Rapid application development (RAD): an empirical review, European Journal of Information Systems, 8,3 (1999) 211-223.

31. Davis, A. M., Bersoff, E. H., ve Comer, E. R., A strategy for comparing alternative software development life cycle models, IEEE Transactions on software Engineering, 14,10 (1988) 1453-1461.

32. Munassar, N. M. A. ve Govardhan, A., A comparison between five models of software engineering, IJCSI International Journal of Computer Science Issues, 7,5 (2010) 94-101.

33. Stoica, M., Mircea, M., ve Ghilic-Micu, B., Software development: Agile vs. traditional, Informatica Economica, 17,4 (2013) 64.

34. McWherter, J., ve Gowell, S., Professional mobile application development, John Wiley & Sons, 2012.

35. Barua, A., Thomas, S. W. ve Hassan, A. E., What are developers talking about? an analysis of topics and trends in stack overflow, Empirical Software Engineering, 19,3 (2014) 619-654.

36. Gartner Says Big Data Creates Big Jobs: 4.4 Million IT Jobs Globally to Support Big Data by 2015 https://www.gartner.com/newsroom/id/2207915 13 Ekim 2017

37. Bughin, J., Chui, M. ve Manyika, J., Clouds, big data, and smart assets: Ten tech-enabled business trends to watch, McKinsey Quarterly, 56,1 (2010) 75-86.

38. Pallis, G., Cloud computing: the new frontier of internet computing, IEEE internet computing, 14,5 (2010) 70-73.

39. Rimal, B. P., Choi, E., ve Lumb, I., A Taxonomy and Survey of Cloud Computing Systems, NCM, 9 (2009) 44-51.

40. Bissyandé, T. F., Thung, F., Lo, D., Jiang, L. ve Réveillere, L., Popularity, interoperability, and impact of programming languages in 100,000 open source projects, In Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual, Temmuz 2013, Kyoto, IEEE, 303-312.

41. Chen, Y., Dios, R., Mili, A., Wu, L., ve Wang, K., An empirical study of programming language trends, IEEE software, 22,3 (2005) 72-79.

42. Pierce, B. C., Types and programming languages, MIT press, 2002.

43. Elrad, T., Filman, R. E., ve Bader, A., Aspect-oriented programming: Introduction, Communications of the ACM, 44,10 (2001) 29-32.

44. Java (programming language) – Wikipedia https://en.wikipedia.org/wiki/Java_ (programming _language) 15 Ekim 2017

45. Python (programming language) – Wikipedia https://en.wikipedia.org/wiki/Python_ (programming_language) 15 Ekim 2017

46. JavaScript – Wikipedia https://en.wikipedia.org/wiki/JavaScript 16 Ekim 2017

47. HTML5 – Wikipedia https://en.wikipedia.org/wiki/HTML5 18 Ekim 2017

48. C Sharp (programming language) – Wikipedia https://en.wikipedia.org/wiki/C_ Sharp_ (programming_language) 19 Ekim 2017

49. Ruby (programming language) – Wikipedia https://en.wikipedia.org/wiki/Ruby_ (programming_language) 22 Ekim 2017

50. SQL – Wikipedia https://en.wikipedia.org/wiki/SQL 24 Ekim 2017

51. R (programming language) – Wikipedia https://en.wikipedia.org/wiki/R_ (programming_ language) 25 Ekim 2017

52. PHP – Wikipedia https://en.wikipedia.org/wiki/PHP 28 Ekim 2017

53. Kantardzic, M., Data mining: concepts, models, methods, and algorithms, John Wiley & Sons, 2011.

54. Han, J., Pei, J., ve Kamber, M., Data mining: concepts and techniques, Elsevier, 2011.

55. Weiss, S. M., Indurkhya, N., Zhang, T., ve Damerau, F., Text mining: predictive methods for analyzing unstructured information, Springer Science & Business Media, 2010.

56. Feldman, R., ve Sanger, J., The text mining handbook: advanced approaches in analyzing unstructured data, Cambridge university press, 2007.

57. Vijayarani, S., Ilamathi, M. J., ve Nithya, M., Preprocessing techniques for text mining-an overview, International Journal of Computer Science & Communication Networks, 5,1 (2015) 7-16.

58. Figueiredo, F., Rocha, L., Couto, T., Salles, T., Gonçalves, M. A., ve Meira Jr, W., Word co-occurrence features for text classification, Information Systems, 36,5 (2011) 843-858.

59. Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A. ve Chanona-Hernández, L., Syntactic n-grams as machine learning features for natural language processing, Expert Systems with Applications, 41,3 (2014) 853-860.

60. Çoban Ö., Metin Sınıflandırma Teknikleri ile Türkçe Twitter Duygu Analizi, Yüksek Lisans Tezi, Atatürk Üniversitesi, Fen Bilimleri Enstitüsü, Erzurum, 2016.

61. Salton, G. ve Buckley, C., Term-weighting approaches in automatic text retrieval, Information processing & management, 24,5 (1988) 513-523.

62. Gnedenko, B. V., The theory of probability and the elements of statistics, 132, American Mathematical Soc., 2005.

63. Liptser, R. ve Shiryaev, A. N., Statistics of random Processes: I. general Theory, 5, Springer Science & Business Media, 2013.

64. Blei, D. M., & Lafferty, J. D., Topic models, Text mining: classification, clustering, and applications, 10,71 (2009) 34.

65. Blei, D. M., Probabilistic topic models, Communications of the ACM, 55,4 (2012) 77-84.

66. Steyvers, M. ve Griffiths, T., Probabilistic topic models, Handbook of latent semantic analysis, 427,7 (2007) 424-440.

67. Landauer, T. K., Latent semantic analysis, John Wiley & Sons Ltd, 2006.

68. Hofmann, T., Probabilistic latent semantic analysis, In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence, Temmuz 1999, Stockholm, Morgan Kaufmann Publishers Inc., 289-296.

69. Blei, D. M., Ng, A. Y. ve Jordan, M. I., Latent dirichlet allocation, Journal of machine Learning research, 3 (2003) 993-1022.

70. Wallach, H. M., Topic modeling: beyond bag-of-words, In Proceedings of the 23rd international conference on Machine learning, Haziran 2006, Pittsburgh, ACM, 977-984.

71. Geman, S. ve Geman, D., Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, IEEE Transactions on pattern analysis and machine intelligence, 6 (1984) 721-741.

72. Harris, A. H., Greer, T. H., Morris, S. A. ve Clark, W. J., Information systems job market late 1970's-early 2010's, Journal of Computer Information Systems, 53,1 (2012) 72-79.

73. Huang, H., Kvasny, L., Joshi, K. D., Trauth, E. M. ve Mahar, J., Synthesizing IT job skills identified in academic studies, practitioner publications and job ads, In Proceedings of the special interest group on management information system's 47th annual conference on Computer personnel research, Mayıs 2009, Limerick, ACM,121-128.

74. Litecky, C. R., Arnett, K. P. ve Prabhakar, B., The paradox of soft skills versus technical skills in IS hiring, Journal of Computer Information Systems, 45,1 (2004) 69-76.

75. Gallivan, M. J., Truex III, D. P. ve Kvasny, L., Changing patterns in IT skill sets 1988-2003: a content analysis of classified advertising, ACM SIGMIS Database, 35,3 (2004) 64-87.

76. Ahmed, F., Capretz, L. F. ve Campbell, P., Evaluating the demand for soft skills in software development, IT Professional, 14,1 (2012) 44-49.

77. Todd, P. A., McKeen, J. D. ve Gallupe, R. B., The evolution of IS job skills: a content analysis of IS job advertisements from 1970 to 1990, MIS quarterly, (1995) 1-27.

78. Garousi, V., Coşkunçay, A., Betin-Can, A. ve Demirörs, O., A survey of software engineering practices in Turkey, Journal of Systems and Software, 108 (2015) 148-177.

79. Carreño, L. V. G. ve Winbladh, K., Analysis of user comments: an approach for software requirements evolution, In Software Engineering (ICSE), 2013 35th International Conference on Software Engineering, Mayıs 2013, San Francisco, IEEE, 582-591.

80. Lukins, S. K., Kraft, N. A. ve Etzkorn, L. H., Bug localization using latent dirichlet allocation, Information and Software Technology, 52,9 (2010) 972-990.

81. Thomas, S. W., Adams, B., Hassan, A. E. ve Blostein, D., Studying software evolution using topic models, Science of Computer Programming, 80 (2014) 457-479.

82. Gardiner, A., Aasheim, C., Rutner, P. ve Williams, S., Skill Requirements in Big Data: A Content Analysis of Job Advertisements, Journal of Computer Information Systems, (2017) 1-11.

83. Rosen-Zvi, M., Griffiths, T., Steyvers, M. ve Smyth, P., The author-topic model for authors and documents, In Proceedings of the 20th conference on Uncertainty in artificial intelligence, Temmuz 2004, Bannf, AUAI Press, 487-494.

84. Guo, J., Xu, G., Cheng, X. ve Li, H., Named entity recognition in query, In Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, Temmuz 2009, Boston, ACM, 267-274.

85. Eisenstein, J., O'Connor, B., Smith, N. A. ve Xing, E. P., A latent variable model for geographic lexical variation, In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, Ekim 2010, Association for Computational Linguistics, 1277-1287.

86. Yin, Z., Cao, L., Han, J., Zhai, C. ve Huang, T., Geographical topic discovery and comparison, In Proceedings of the 20th international conference on World wide web, Mart 2011, Hyderabad, ACM, 247-256.

87. Bíró, I., Szabó, J. ve Benczúr, A. A., Latent dirichlet allocation in web spam filtering, In Proceedings of the 4th international workshop on Adversarial information retrieval on the web, Nisan 2008, Beijing, ACM, 29-32.

88. Newman, D., Lau, J. H., Grieser, K. ve Baldwin, T., Automatic evaluation of topic coherence, In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Haziran 2010, Los Angeles, Association for Computational Linguistics, 100-108.

89. Haghighi, A. ve Vanderwende, L., Exploring content models for multi-document summarization, In Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Mayıs 2009, Boulder, Association for Computational Linguistics, 362-370.

90. Mimno, D., Wallach, H. M., Naradowsky, J., Smith, D. A. ve McCallum, A., Polylingual topic models, In Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, Ağustos 2009, Singapore, Association for Computational Linguistics, 2,880-889.

91. Wang, X., McCallum, A. ve Wei, X., Topical n-grams: Phrase and topic discovery, with an application to information retrieval, In Data Mining, 2007, ICDM 2007, Seventh IEEE International Conference on, Ekim 2007, Omaha, IEEE, 697-702.

92. Schmitz, M., Bart, R., Soderland, S. ve Etzioni, O., Open language learning for information extraction, In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Temmuz 2012, Jeju Adası, Association for Computational Linguistics, 523-534.

93. Lin, C. ve He, Y., Joint sentiment/topic model for sentiment analysis, In Proceedings of the 18th ACM conference on Information and knowledge management, Kasım 2009, ACM, Hong Kong, 375-384.

94. Lu, B., Ott, M., Cardie, C. ve Tsou, B. K., Multi-aspect sentiment analysis with topic models, In Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Aralık 2011, Vancouver, IEEE, 81-88.

95. Zeng, D., Chen, H., Lusch, R. ve Li, S. H., Social media analytics and intelligence, IEEE Intelligent Systems, 25,6 (2010) 13-16.

96. Hong, L. ve Davison, B. D., Empirical study of topic modeling in twitter, In Proceedings of the first workshop on social media analytics, Temmuz 2010, Washington, ACM, 80-88.

97. Prier, K. W., Smith, M. S., Giraud-Carrier, C. ve Hanson, C. L., Identifying health-related topics on twitter, In International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction, Mart 2011, College Park, Springer,18-25.

98. Wang, X., Gerber, M. S. ve Brown, D. E., Automatic Crime Prediction Using Events Extracted from Twitter Posts, SBP, 12 (2012) 231-238.

99. Wang, Y. C., Burke, M. ve Kraut, R. E., Gender, topic, and audience response: an analysis of user-generated content on facebook, In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Nisan 2013, Paris, ACM, 31-34.

100. Si, J., Mukherjee, A., Liu, B., Li, Q., Li, H. ve Deng, X., Exploiting Topic based Twitter Sentiment for Stock Prediction, ACL, 2 (2013) 24-29.

101. Chae, J., Thom, D., Bosch, H., Jang, Y., Maciejewski, R., Ebert, D. S. ve Ertl, T., Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition, IEEE Conference on Visual Analytics Science and Technology (VAST) 2012, Ekim 2012, Seattle, IEEE, 143-152.

102. Kossinets, G. ve Watts, D. J., Empirical analysis of an evolving social network, Science, 311,5757 (2006) 88-90.

103. Pritchard, J. K., Stephens, M. ve Donnelly, P., Inference of population structure using multilocus genotype data, Genetics, 155,2 (2000) 945-959.

104. Blei, D., Carin, L. ve Dunson, D., Probabilistic topic models, IEEE Signal Processing Magazine, 27,6 (2010) 55-65.

105. Fei-Fei, L. ve Perona, P., A bayesian hierarchical model for learning natural scene categories, In Computer Vision and Pattern Recognition, 2005, CVPR 2005, IEEE Computer Society Conference on, Haziran 2005, IEEE II: 524-531.

106. Blei, D. M. ve Jordan, M. I., Modeling annotated data, In Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval, Temmuz 2003, Toronto, ACM, 127-134.

107. Bart, E., Welling, M. ve Perona, P., Unsupervised organization of image collections: taxonomies and beyond, IEEE transactions on pattern analysis and machine intelligence, 33,11 (2011) 2302-2315.

108. Du, L., Ren, L., Carin, L. ve Dunson, D. B., A bayesian model for simultaneous image clustering, annotation and object segmentation, In Advances in neural information processing systems, (2009) 486-494.

109. Pruteanu-Malinici, I., Ren, L., Paisley, J., Wang, E. ve Carin, L., Hierarchical Bayesian modeling of topics in time-stamped documents, IEEE transactions on pattern analysis and machine intelligence, 32,6 (2010) 996-1011.

110. An, Q., Wang, C., Shterev, I., Wang, E., Carin, L. ve Dunson, D. B., Hierarchical kernel stick-breaking process for multi-task image analysis, In Proceedings of the 25th international conference on Machine learning, Temmuz 2008, Helsinki, ACM, 17-24.

111. Stack Overflow Developer Jobs https://stackoverflow.com/jobs 20 Ekim 2017

112. Stack Overflow - Where Developers Learn, Share, & Build Careers https://stackoverflow.com/ 29 Mayıs 2017

113. Indeed Türkiye https://tr.indeed.com/jobs? 20 Ekim 2017

114. McCallum A (2002) MALLET: a machine learning for language toolkit. http://mallet.cs.umass.edu 21 Haziran 2017

115. Charland, A. ve Leroux, B., Mobile application development: web vs. native. Communications of the ACM, 54,5 (2011) 49-53.

116. Guinard, D., Trifa, V., Mattern, F. ve Wilde, E., From the internet of things to the web of things: Resource-oriented architecture and best practices, Architecting the Internet of things, (2011) 97-129.

117. Anttonen, M., Salminen, A., Mikkonen, T., ve Taivalsaari, A., Transforming the web into a real application platform: new technologies, emerging trends and missing pieces, In Proceedings of the 2011 ACM Symposium on Applied Computing, Mart 2011, TaiChung, ACM, 800-807.

118. Gavalas, D., ve Economou, D., Development platforms for mobile applications: Status and trends, IEEE software, 28,1 (2011) 77-86.

119. Assunção, M. D., Calheiros, R. N., Bianchi, S., Netto, M. A. ve Buyya, R., Big Data computing and clouds: Trends and future directions, Journal of Parallel and Distributed Computing, 79 (2015) 3-15.

120. Lin, A. ve Chen, N. C., Cloud computing as an innovation: Percepetion, attitude, and adoption, International Journal of Information Management, 32,6 (2012) 533-540.

ÖZGEÇMİŞ

1976 yılında Konya'da doğan Fatih GÜRCAN, ilköğrenimini Atatürk İlkokulunda, ortaöğrenimini Mehmet Akif Ersoy Ortaokulu'nda ve lise eğitimini Kulu Lisesi'nde tamamladı. Lisans eğitimini 2000 yılında Karadeniz Teknik Üniversitesi İstatistik ve Bilgisayar Bilimleri Bölümünde tamamladıktan sonra 2001 yılında Karadeniz Teknik Üniversitesi Enformatik Bölümünde öğretim görevlisi olarak göreve başladı. 2009 yılında Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim dalında yüksek lisans eğitimini tamamladı. 2011 yılında Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim dalında doktora eğitimine başlayan Fatih GÜRCAN; halen Karadeniz Teknik Üniversitesi Uzaktan Eğitim Uygulama ve Araştırma Merkezinde Öğretim Görevlisi olarak görev yapmaktadır. Evli ve bir çocuk babası olan Fatih GÜRCAN, iyi derecede İngilizce bilmektedir.

Gurcan, F. ve Kose, C., Analysis of Software Engineering Industry Needs and Trends: Implications for Education, International Journal of Engineering Education, 33,4 (2017) 1361-1368.