Mühendislik Bilimleri ve Tasarım Dergisi
8(5), 45-51, 2020
e-ISSN: 1308-6693
Araştırma Makalesi

Journal of Engineering Sciences and Design
DOI: 10.21923/jesd.826251
Research Article

Özel Sayı: Uluslararası Mühendislikte Yapay Zeka ve Uygulamalı Matematik Konferansı (UMYMK 2020)
Special Issue: International Conference on Artificial Intelligence and Applied Mathematics in Engineering (ICAIAME 2020)

YİNELENEN YAZILIM HATA KAYITLARININ MAKİNE ÖĞRENMESİ VE DERİN ÖĞRENME YÖNTEMLERİ İLE TESPİT EDİLMESİ

Azmi YÜKSEL[1*], Aydın ÇETİN[2]

[1] Kalite Test ve Süreç Yönetimi Müdürlüğü, STM A.Ş., Ankara, Türkiye
[2] Gazi Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği, Ankara, Türkiye

Anahtar Kelimeler
Doğal dil işleme, Hata kaydı, Hata raporu, Yinelenen hata kaydı tespiti. Yazılım hataları

Öz

Bir yazılım, geliştirme, bakım veya kullanım aşamasındayken beklenilen şekilde çalışmaması durumunda ortaya çıkan hatalar teknik ekip veya son kullanıcılar tarafından raporlanmaktadır. Raporlanan hata kayıtları, hatayı raporlayan kişiler tarafından farklı şekillerde sisteme girilse bile aynı hatayı işaret edebilir. Dolayısıyla, raporlanacak olan bir hata kaydının sistemde daha önceden bulunma ihtimali oldukça yüksektir. Hatayı düzeltecek olan geliştiricinin ilgili hata kaydının sisteme daha önce girilmiş olup olmadığını tespit etmesi oldukça yüksek çaba gerektirmektedir. Sisteme girilecek bir hatanın daha önce sistemde var olup olmadığını tespit etmek için otomatik bir tespit mekanizması gerekmektedir. Bu çalışmada, 3 farklı açık kaynak proje için hata kayıtları kullanılarak, yinelenen hata kayıtlarını makine öğrenmesi ve derin öğrenme yöntemleri ile tespit eden farklı modeller geliştirilmiştir. Çalışmada, kullanılan veri setleri için makine öğrenmesi algoritmalarının ve derin öğrenme yöntemlerinin başarımları karşılaştırmalı olarak incelenmiştir ve birleşik bir yöntem önerilmiştir. Önerilen birleşik yöntem tekil yöntemlere göre başarıyı en az %7.2 oranında artırmıştır.

DETECTING DUPLICATE SOFTWARE BUG RECORDS USING MACHINE LEARNING AND DEEP LEARNING METHODS

Keywords
Natural language processing, Bug record, Bug report, Duplicate bug report detection. Software bugs

Abstract

Bugs that occur when a software does not work as expected while it is in development, maintenance or use, are reported by the technical team or end users. Reported bug records can point to the same error even if they are entered into the system in different ways by the people reporting the bug. Therefore, it is highly likely that a bug record to be reported already exists in the system. The developer who will fix the error requires a lot of effort to determine whether the relevant error record has been entered into the system before. An automatic detection mechanism is required to detect whether a bug to be entered into the system has already existed in the system. In this study, different models have been developed that detect duplicate bug records with machine learning and deep learning methods by using bug records for 3 different open source projects. In the study, the performances of machine learning algorithms and deep learning methods for the data sets used were examined comparatively and an ensemble method is proposed. Proposed ensemble method increased the accuracy by at least 7.2% compared to the singular mehods.

Alıntı / Cite

Yüksel, A., Çetin, A., (2020). Yinelenen Yazılım Hata Kayıtlarının Makine Öğrenmesi ve Derin Öğrenme Yöntemleri İle Tespit Edilmesi, 8(5), 45-51.

Yazar Kimliği / Author ID (ORCID Number)	Makale Süreci / Article Process	
Yazar1 A. Yüksel, 0000-0000-0000-0000	**Başvuru Tarihi / Submission Date**	15.11.2020
Yazar2 A. Çetin, 0000-0002-8669-823X	**Revizyon Tarihi / Revision Date**	16.12.2020
	Kabul Tarihi / Accepted Date	16.12.2020
	Yayım Tarihi / Published Date	29.12.2020

[*] İlgili yazar/Corresponding author: azyuksel@stm.com.tr, +90-506-321-5876

1. Giriş (Introduction)

Yazılım bakım faaliyetleri, yazılım kalitesi için kritik olan yazılım projelerinin yaşam döngüsünün önemli bir parçasıdır (Pressman, 2005). Hata raporları yazılım bakım faaliyetlerinde önemli bir rol oynar (Penny vd., 2003). Yazılım hataları kaçınılmazdır, yazılımdaki hatayı düzeltmek, hata düzeltme süresini tahmin etmek, önce hangi hatanın düzeltilmesi gerektiğine karar vermek, belirli bir hatayı kimin düzeltmesi gerektiğine karar vermek gibi işlemlerde geliştiricilere rehberlik etmek için hata raporları kullanılabilir. Ayrıca hata raporları yazılım projesinin durumu hakkında değerli bilgiler de sağlayabilir.

Genellikle, hatalar yazılım projelerindeki yazılım geliştirme ve bakım faaliyetlerini yönetmek için kullanılan bir hata izleme sistemine rapor edilir (Anvik vd., 2005; Lientz ve Swanson, 1980). Hata izleme sistemleri yalnızca geliştiricilerin, test uzmanlarının ve müşterilerin sistemlerde bulunan sorunları tanımlamaları ve raporlamaları için değil aynı zamanda hataları ve hatalarla ilgili bilgileri depolamak ve bununla ilgili görevlerin durumlarını izlemek için tasarlanmıştır (Anvik vd., 2005; Anvik vd., 2006). Bu nedenle, bir hata izleme sistemi, hataları ortaya çıkarma verimliliğini artırabilirken yazılım kalitesini daha da iyileştirebilir.

Sistemde daha önce bulunan bir hatayı farklı şekilde tarif eden hata raporları yinelenen hata raporu olarak tanımlanır (Anvik vd., 2005). Yinelenen hata raporları sisteme girildikçe sistemdeki hata sayılarının sayısı artar. Çalışmalarda gösterildiği gibi, toplam hata raporları içerisinde yinelenen hata raporlarının oranı % 25-30'a kadar olabilmektedir (Anvik vd., 2005; Anvik vd., 2006; Jalbert ve Weimer, 2008). Yinelenen hata raporları, aynı hata için benzer bir nedenden dolayı hatayı yeniden üreten ve düzelten birden fazla geliştiriciye aynı hataların atanmasına yol açmaktadır. Bunun sonucunda da yazılımın geliştirme için harcanan çaba artmakta, süre ve maliyet kaybı ortaya çıkmaktadır.

Literatürde yer alan bazı çalışmalar (Runeson vd., 2007; Cavalcanti vd., 2013; Cavalcanti vd. 2010; Bettenburg vd., 2008), yinelenen hata raporlarını tespit etme konusunda ortaya çıkan durumları aşağıdaki gibi özetlemektedirler:

- Farklı geliştiricilere yinelenen hata raporları atandığında, geliştiricilerin zamanını ve çabasını boşa harcar.
- Bir hata düzeltildiğinde, yinelenen hata raporlarını bağımsız bir hata olarak ele almak zaman kaybıdır.
- Yinelenen hata raporlarının belirlenmesi, hataların giderilmesinde de yar-dımcı olabilir, çünkü bazı yinelenen hata raporları daha faydalı bilgiler sağlayabilir.

Günümüzde *triager* adı da verilen bir analist sisteme yeni girilen bir hata raporunun bir hata veya önceden bildirilen bir hata olup olmadığını anlamak için önceden girilmiş hata raporlarını kontrol ederek yinelenen hata raporlarını tespit etmektedir. Analist bu işlemi yaparken hafızasına, tecrübesine ve kullanılan hata izleme sisteminin arama özelliklerine güvenir (Anvik vd., 2005). Sistemde bulunan hata sayısı arttıkça analist için bu süreç sıkıcı hale gelmekte ve ayrıca önemli miktarda çaba ve zaman kaybına neden olduğundan motivasyonu düşürmektedir (Reis ve Fortes, 2002). Bu nedenle yinelenen hata raporu tespiti işlemini otomatik hale getirme ihtiyacı bulunmaktadır. İşlemlerin otomatik hale gettirilmesiyle birlikte yeni gelen hata raporları, benzer hata raporlarını belirlemek için sisteme daha önceden girilmiş olan hata raporları ile karşılaştırılabilir, bu sayede bir insanın raporları okuma, anlama ve arama için zaman harcamasına gerek kalmaz, bu da hata raporlarının işlenme süresini kısaltabilir. Bu otomatikleştirme işlemleri sonucunda yazılım bakım faaliyetlerinin verimliliğinin artmasına önemli düzeyde katkı sağlanabilir.

Yinelenen hata raporlarının otomatik olarak tespit edilmesi aşağıdaki nedenlerden dolayı zor bir problemdir (Sureka ve Jalote, 2010; Naumann ve Herschel, 2010):

- Hata raporları doğal dil metninde ifade edilir. Bu raporlar, aynı yazılım hatasından kaynaklanan farklı anormal davranışların ortaya çıkması koşul-larında aynı hatayı farklı kelimelerle tanımlayabilir. Doğal dil anlayışında dil belirsizliği problemi ve dil değişkenliği problemi vardır, bu da benzer raporların tanımlanmasını zorlaştırır.
- Hata izleme sistemindeki hata raporu sayısı fazladır.
- Hata raporlarının bir kısmının kalitesi yeterli değildir, örneğin hata raporu zayıf, yetersiz, eksik hatta yanlış bilgi içerebilir.

Son zamanlarda, yinelenen hata raporlarının otomatik olarak tespit edilmesi konusunda önemli çalışmalar yapılmış olsa da, bu çalışmalar halen bu soruna tam olarak ya da kalıcı bir çözüm üretememiştir.

Bu çalışmada, yinelenen hata raporlarının otomatik olarak tespit edilmesi amacıyla makine öğrenmesi ve derin öğrenme yöntemleri kullanılarak farklı yön-temlerdeki başarımların karşılaştırması yapılmıştır. Bu bağlamda öncelikle Mozilla Thunderbird, Eclipse JDT ve Eclipse Platform hata veri setleri (Logpai, 2020) incelenerek veriler üzerinde bir ön işleme yapılmıştır. Ön işlemesi yapılan bu verilere istatistiksel, makine öğrenmesi ve derin öğrenme tabanlı kelime veya cümle temsili yöntemleri uygulanarak vektörler oluşturulmuştur. Daha sonra oluşturulan bu vektörler arasındaki uzaklıklar ölçülerek kosinüs benzerlikleri hesaplanmıştır. Daha sonra, bu benzerlikler kullanılarak yöntemlerin hatırlama değerleri hesaplanmış ve yöntemlerin başarımları karşılaştırılmıştır.

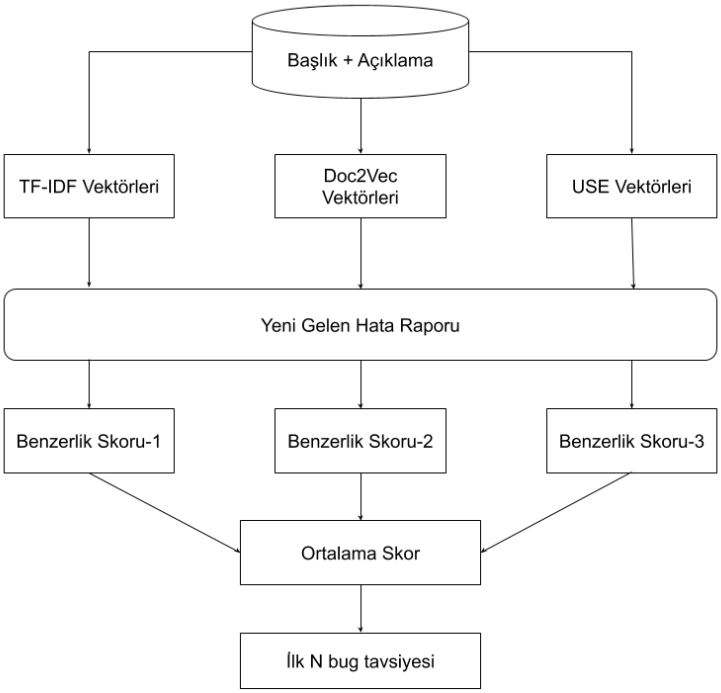
2. Kaynak Araştırması (Literature Survey)

Yinelenen hata raporlarının tespiti üzerine yapılan öncü çalışmalardan biri Buckley ve arkadaşları (Buckley vd., 1998) tarafından yapılmıştır. Çalışmada, önce bir hata raporunda Porter stemmer algoritması (Porter vd., 1980) kullanılarak her bir kelime ve Cornell SMART durma listesi (stop-word list) kullanılarak durma kelimeleri (stop-words) önceden işlenmiş, daha sonra, Terim Frekansı-Ters Belge Frekansı (TF-IDF) kullanılarak temizlenen metin bir kelime vektörüne dönüştürülmüştür (Raghavan ve Wong, 1986). Hata raporları için belge vektörleri oluşturulduktan sonra, kosinüs benzerliği kullanarak benzer iki hata raporunun birbirlerine ne kadar benzediğini ölçmüşlerdir. X. Wang ve arkadaşları (Wang vd., 2008) kosinüs temelli bir benzerlik ölçümü ile benzerliği hesaplamak için TF-IDF ile bir kelime vektörüne çalıştırma bilgisi eklemişlerdir. N. Jalbert ve arkadaşları (Jalbert ve Weimer, 2008) TF-IDF ve kosinüs temelli benzerlik ölçümü ile oluşturulan hata raporlarının belge vektörleri üzerinde doğrusal regresyon ile eğitilmiş bir sınıflandırıcı kullanmışlar, sınıflandırıcıları, yeni bir hata raporu için bir DUPLICATE veya NEW etiketi döndürebilmiştir. A. Sureka ve arkadaşları (Sureka ve Jalote, 2010) bir hata raporunun metnini karakter seviyesi n-gram temsiline aktarmak için karakter n-gram (karakterlerin sırası) kullanarak bir yaklaşım önermişlerdir. C. Sun ve arkadaşları (Sun vd., 2011) BM25F tabanlı özellik ağırlıklandırma şemasıyla benzerliği hesaplamak için hem metin içeriğini hem de hata raporlarının kategorik bilgilerini kullanan REP adlı bir yaklaşım, A. T. Nguyen ve arkadaşları ise (Nguyen vd., 2012) hem IR (Information Retrieval) tabanlı özelliklerden (BM25F) hem de konu tabanlı özelliklerden (Latent Dirichlet Allocation) yararlanan bir yaklaşım önermişlerdir. Tian ve arkadaşları (Tian vd., 2012) BM25F'den yararlanarak, yeni bir ürün tabanlı özellik ekleyerek ve en üstteki en benzer raporlardan göreceli benzerlik özelliğini kullanarak Jalbert ve Weimer'in çalışmalarını (Jalbert ve Weimer, 2008) daha da ileriye taşımışlardır. A. Alipour ve arkadaşları (Alipour vd., 2013) metinsel bir karşılaştırma olarak oluşturulmuş ve projenin hata açıklamalarında denetimli etiketli Gizli Dirichlet Tahsisi (etiketli-LDA) yardımıyla elle oluşturulan kelime listelerini ve konuları kullanan hata raporları arasındaki bağlamsal benzerliği değerlendirdirmişlerdir. A. Budhiraja ve arkadaşları (Budhiraja vd., 2018b) birden fazla kelime temsili yöntemini deneyerek bu konuda en iyi sonuç veren yöntemi bulmaya çalışmışlar, diğer bir çalışmalarında ise (Budhiraja vd., 2018c) LDA ve kelime temsil yöntemelerini birleştiren bir yaklaşım önermiştir. A. Budhiraja ve arkadaşları daha sonra (Budhiraja vd., 2018a) hata raporu ikililerini kelime temsillerine çevirerek birleştirmişler, bu vektörleri bir derin öğrenme modeline girdi olarak vererek bir model eğitmişlerdir. Model, 2 hata raporu verildiğinde iki hata raporunun duplicate olma ihtimalini hesaplayarak 0 ile 1 arasında bir çıktı vermektedir. L.Poddar ve arkadaşları (Poddar vd., 2019) kısmen denetimli bir derin öğrenme modeli geliştirmişlerdir. Geliştirilen derin öğrenme modeli önce hata raporunu topiklere kümeleyen daha sonra sınıflandıran 2 aşamalı bir modeldir. He ve arkadaşları (He vd., 2020) çalışmalarında çift kanallı evrişimsel sinir ağları kullanmışlardır. Kukkar ve arkadaşları (Kukkar vd., 2020) derin öğrenme tekniğine dayalı yinelenen hata raporu tespit ve sınıflandırma sistemi geliştirmişlerdir.

3. Materyal ve Yöntem (Material and Method)

3.1. Önerilen Yöntem (Proposed Method)

Çalışmada istatistiksel bir yöntem olan TF-IDF (Term Frequency - Inverse Document Frequency), makine öğrenme tabanlı bir yöntem olan Doc2Vec ve derin öğrenme tabanlı bir yöntem olan Universal Sentence Encoder (USE) kelime temsil yöntemleri kullanılmıştır. Önerilen sistemin mimarisi Şekil 1'de verilmiştir.



Şekil 1. Sistem Mimarisi (System Architecture)

Adından da anlaşılacağı gibi, TF-IDF, bir belgedeki her kelime için değerleri, belirli bir belgedeki kelimenin sıklığının, kelimenin göründüğü belgelerin yüzdesi ile ters orantılı olarak hesaplar. Yüksek TF-IDF numaralarına sahip kelimeler, göründükleri belge ile güçlü bir ilişki anlamına gelir; bu, bir kelimenin bir sorguda görünmesi durumunda belgenin kullanıcıyı ilgilendirebileceğini gösterir (Ramos vd., 2003).

Word2Vec yöntemi, Mikolov ve arkadaşları (Mikolov vd., 2013) tarafından kelimeler için yüksek kaliteli düğünleri öğrenmek için etkili bir yapay sinir ağı modeli olarak önerilmiştir. Doc2Vec yöntemi Word2Vec yönteminin doküman temsillerini öğrenmek amaçlı geliştirilmiş bir uzantısıdır (Lau ve Baldwin, 2016).

USE kapsamında cümlelerin sabit bir vektörle temsil edilmesini sağlayan iki adet derin öğrenme modeli geliştirilmiştir (Cer vd., 2018). Bu modellerden birincisi Transformer modeli (Vaswani vd., 2017) diğeri ise DAN (Deep Average Network) (Iyyer vd., 2015) modelidir. Her iki model de TensorFlow'da uygulanmış açık kaynak kodlu olup TensorFlow Hub'dan indirilebilir: https://tfhub.dev/google/universal-sentence-encoder/1. Bu çalışma kapsamında TensorFlow Hub'da bulunan önceden eğitilmiş DAN modeli kullanılmıştır.

Çalışmada, yukarıda bahsi geçen yöntemler ön işlemesi yapılan verilere uygulanarak öncelikle vektör temsilleri oluşturulmuştur. Oluşturulan bu vektörler kullanılarak hata raporları arasındaki kosinüs benzerlikleri ayrı ayrı hesaplanmıştır. Daha sonra hesaplanan kosinüs benzerliklerinin ortalamaları alınarak yeni bir benzerlik matrisi oluşturulmuştur. Oluşturulan yeni matris, bir hataya ait en benzer hataların bulunması için kullanılmıştır.

3.2. Veri Setleri (Datasets)

Çalışmada Eclipse Platform, Eclipse JDT ve Mozilla Thunderbird hata veri setleri \cite{logpai} hazır olarak kullanılmıştır. Verilerin tümü eğitim veri seti olarak kullanılırken, DUPLICATE olarak etiketlenen veriler test veri seti olarak kullanılmıştır. Kullanılan veri setlerinin özellikleri Tablo 1'de verilmiştir.

Tablo 1. Verisetlerinde bulunan verilerin özellikleri (Properties of data contained in datasets)

VERİ SETİ	ZAMAN ARALIĞI	HATA SAYISI	YİNELENEN HATA SAYISI	ORAN(%)
Eclipse Platform	10.10.2001 – 30.12.2003	85156	14404	16.9
Eclipse JDT	10.10.2001 – 30.12.2003	45296	7688	17
Mozilla Thunderbird	12.04.2000 – 30.12.2003	32551	12501	38.4

Hata raporları sisteme girilirken zorunlu veya opsiyonel olmak üzere belirli alanlar doldurulmaktadır. Kullanılan veri setlerinde, bir hata raporuna ait alanlar Tablo 2'de verilmiştir.

Tablo 2. Hata raporuna ait alanlar ve açıklamaları (Titles and explanations of the bug report fields)

ALAN ADI	AÇIKLAMA
Issue_id	Her bir rapora ait benzersiz numara
Priority	Hataların önemini bildiren öncelik değeri
Component	Hatanın ortaya çıktığı modül
Duplicated_issue	Eğer hata yinelenen hata olarak işaretlenmişse, eşlendiği hata numarası
Title	Hatanın başlığı
Description	Hatanın detaylı açıklaması
Status	Hatanın durumu
Resolution	Hatanın çözüm durumu
Version	Hatanın tespit edildiği yazılım versiyonu
Created_time	Hata raporunun oluşturulma zamanı
Resolved_time	Hatanın giderildiği zaman

Çalışmada, Tablo 2'de bulunan alanlardan başlık ve açıklama bilgileri kullanılmıştır. Başlık alanı, hatanın kısa bir özetini ifade etmektedir. Açıklama alanı ise hatanın detaylı olarak anlatıldığı adımdır. Veri ön işlemesi için ilk olarak her bir hata raporunun başlığı ve açıklaması tek bir alan olarak birleştirilmiştir.İkinci aşamada bu alan öncelikle terimlerine parçalanmış ve devamında bu terimlerden noktalama işaretleri, rakamlar ve durma kelimeleri (stop-words) çıkarılmıştır. Son aşamada ise, öznitelik boyutlarını azaltmak ve benzer kelimeleri ortak bir şekilde birleştirmek için kalan terimler kök formlarına dönüştürülmüştür.

4. Deneysel Sonuçlar (Experimental Results)

Sistemin değerlendirilmesinde daha önceki çalışmalarda da en yaygın olarak kullanılan metrik olan hatırlama (recall) metriği kullanılmıştır. Hatırlama, modelin, sistemde pozitif olarak bilinen değerlerin ne kadarını pozitif olarak tahmin edebildiğini ölçmede kullanılan bir metriktir.
Hatırlama metriğinin hesaplanma yöntemi Eşitlik (1)'de verilmiştir:

$$Hatırlama = TP/(TP + FN) \qquad (1)$$

Eşitlik (1)'de TP, doğru tespit edilen yinelenen hata raporu sayısını, FN, yanlış tespit edilen yinelemeyen hata raporu sayısını ifade etmektedir.

Sistemin değerlendirilmesi için yinelenen hataları bulunan verilerden yeni bir test seti oluşturulmuştur. Bu veri setindeki her bir hata için yinelenen hata raporu numarasının, en benzer ilk-k hata raporu numaraları listesinin içinde olup olmadığı kontrol edilmiştir. Önerilen yöntem tüm veri setlerine ayrı ayrı uygulanarak, k=1, 10 ve 20 en benzer hata raporları için hatırlama değerleri hesaplanmıştır.

Çalışmada önerilen birleşik yöntemin başarımını değerlendirmek, elde edilen sonuçların farklı veri setleri ile olan ilişkilerini gözlemleyebilmek için testler üç farklı veri seti üzerinden gerçekleştirilmiştir. Tablo 3'te önerilen birleşik yönteme ait sonuçlar verilmiştir.

Tablo 3. Önerilen birleşik yöntem için sonuçlar (Results for the proposed combined method)

VERİ SETİ	k=1	k=10	k=20
Mozilla Thunderbird	0.158	0.396	0.48
Eclipse JDT	0.16	0.387	0.46
Eclipse Platform	0.154	0.37	0.433

Diğer yandan çalışmada önerilen birleşik yöntemin başarımını ve etkinliğini diğer yöntemlerle kıyaslama yapabilmek için aynı test işlemler üç farklı veri seti üzerinden gerçekleştirilmiştir. Farklı veri setleri kullanılarak ayrı ayrı kelime temsil yöntemleri ve önerilen birleşik yöntem ile alınan sonuçlar Tablo 4'te sunulmaktadır.

Tablo 4. Test edilen yöntemlerin farklı veri setlerinden elde edilen hatırlama değerleri (Recall values obtained from different data sets of the tested methods)

VERİ SETİ	Mozilla Thunderbird			Eclipse JDT			Eclipse Platform		
YÖNTEM	k=1	k=10	k=20	k=1	k=10	k=20	k=1	k=10	k=20
TF-IDF	0.118	0.304	0.382	0.14	0.318	0.379	0.132	0.344	0.361
Doc2Vec	0.092	0.21	0.261	0.094	0.2	0.243	0.09	0.191	0.228
USE	0.102	0.271	0.332	0.108	0.26	0.315	0.096	0.236	0.287
Önerilen Yöntem	0.158	0.396	0.48	0.16	0.387	0.46	0.154	0.37	0.433

Sonuçlar incelendiğinde önerdiğimiz yöntemin, ilk 20 benzer hata raporunda, tek başına en iyi yöntem olan TF-IDF'e göre Mozilla Thunderbird, Eclipse JDT ve Eclipse Platform veri setlerinde sırasıyla yaklaşık %9.8, %8.1 ve %7.2 oranında bir iyileştirme ortaya koyduğu gözlenmektedir.

5. Sonuç ve Tartışma (Result and Discussion)

Yinelenen hata raporlarının otomatik olarak tespit edilmesi, yazılım geliştirme sürecinin daha hızlı ilerlemesinde önemli bir rol oynamaktadır. Bu çalışmada, yinelenen hata kayıtlarını otomatik olarak tespit etmek için yeni bir yaklaşım önerilmiştir. Önerilen yaklaşımda üç farklı tabana (istatistiksel, makine öğrenmesi ve derin öğrenme) sahip TF-IDF, Doc2Vec ve Universal Sentence Encoder kelime temsil yöntemleri birleştirilmiştir. Yöntemde önceden işlenmiş hata raporlarındaki başlık ve açıklama bilgileri yukarıda bahsi geçen 3 farklı kelime temsil yöntemini kullanarak kelime vektörlerine dönüştürülmekte ve benzerlik skorları oluşturulmaktadır. Oluşturulan benzerlik skorlarının ortalaması alınarak, yeni bir matris oluşturulmakta ve bu matrise göre en benzer hatalar tespit edilmektedir. Önerilen birleşik yaklaşımla, tek başına kullanılan istatistiksel, makine öğrenmesi ya da derin öğrenme yöntemlerine kıyasla elde edilen sonuçlarda yinelenen hata tespitinde asgari %7.2 oranında bir iyileştirme elde edilmiştir. Bu hatırlama değerlerinin farklı kelime temsil yaklaşımları ve/veya Doc2Vec model parametrelerinin optimizasyonu ile arttırılması için çalışmalar devam etmektedir.

Teşekkür (Acknowledgement)

Bu çalışma STM A.Ş. tarafından desteklenmektedir.

Çıkar Çatışması (Conflict of Interest)

Yazarlar tarafından herhangi bir çıkar çatışması beyan edilmemiştir. No conflict of interest was declared by the authors.

Kaynaklar (References)

Alipour, A., Hindle, A., and Stroulia, E., 2013. A contextual approach towards more accurate duplicate bug report detection. In2013 10th Working Conference on Mining Software Repositories (MSR), pages 183–192. IEEE.

Anvik, J., Hiew, L., and Murphy, G. C., 2005. Coping with an open bug repository. In Proceedings of the 2005 OOPSLA workshop on Eclipse technology eXchange, pages 35–39.

Anvik, J., Hiew, L., and Murphy, G. C., 2006. Who should fix this bug? In Proceedings of the 28th international conference on Software engineering, pages 361–370.

Bettenburg, N., Premraj, R., Zimmermann, T., and Kim, S., 2008. Duplicate bug reports considered harmful. . . really? In2008 IEEE International Conference on Software Maintenance, pages 337–345. IEEE.

Buckley, C., Walz, J., Cardie, C., Mardis, S., Mitra, M., Pierce,D., and Wagstaff, K., 1998. The smart/empire tipster ir system. In TIPSTER TEXT PROGRAM PHASE III: Proceedings of a Workshop held at Baltimore, Maryland, October 13-15, 1998, pages 107–121.

Budhiraja, A., Dutta, K., Reddy, R., and Shrivastava, M., 2018a. Dwen: deep word embedding network for duplicate bug report detection in software repositories. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, pages 193–194.

Budhiraja, A., Dutta, K., Shrivastava, M., and Reddy, R., 2018b. Towards word embeddings for improved duplicate bug report retrieval in software repositories. In Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval, pages 167–170.

Budhiraja, A., Reddy, R., and Shrivastava, M., 2018c. Lwe: Lda refined word embeddings for duplicate bug report detection. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceedings, pages 165–166.

Cavalcanti, Y. C., de Almeida, E. S., da Cunha, C. E. A.,Lucredio, D., and de Lemos Meira, S. R., 2010. An initial study on the bug report duplication problem. In2010 14th European Conference on Software Maintenance and Reengineering, pages 264–267. IEEE.

Cavalcanti, Y. C., Neto, P. A. d. M. S., Lucredio, D., Vale,T., de Almeida, E. S., and de Lemos Meira, S. R. (2013). The bug report duplication problem: an exploratory study. Software Quality Journal, 21(1):39–66.

Cer, D., Yang, Y., Kong, S.-y., Hua, N., Limtiaco, N., John, R. S., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., et al., 2018. Universal sentence encoder. arXiv preprint arXiv:1803.11175.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daum´e III, H., 2015. Deep unordered composition rivals syntactic methods for text classification .In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 1681–1691.

Kukkar, A., et al., 2020. Duplicate Bug Report Detection and Classification System Based on Deep Learning Technique. IEEE Access, 8, 200749-200763.

Jalbert, N. and Weimer, W., 2008. Automated duplicate detection for bug tracking systems. IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), pages 52–61. IEEE.

Lau, J. H. and Baldwin, T., 2016. An empirical evaluation of doc2vec with practical insights into document embedding generation. arXiv preprint arXiv:1607.05368.

Lientz, B. P. and Swanson, E. B., 1980. Software maintenance management. Addison-Wesley Longman Publishing Co., Inc.

He, J. et al 2020. Duplicate bug report detection using dual-channel convolutional neural networks. In Proceedings of the 28th International Conference on Program Comprehension, pages 117-227.

Mikolov, T., Chen, K., Corrado, G., and Dean, J., 2013. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Naumann, F. and Herschel, M., 2010. An introduction to duplicate detection. Synthesis Lectures on Data Management, 2(1):1–87.

Nguyen, A. T., Nguyen, T. T., Nguyen, T. N., Lo, D., and Sun,C., 2012. Duplicate bug report detection with a combination of information retrieval and topic modeling. In2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, pages 70–79. IEEE.

Penny, G. et al., 2003. Software maintenance: concepts and practice. World Scientific.

Poddar, L., Neves, L., Brendel, W., Marujo, L., Tulyakov, S., and Karuturi, P., 2019. Train one get one free: Partially supervised neural network for bug report duplicate detection and clustering. arXiv preprint arXiv:1903.12431.

Porter, M. F. et al., 1980. An algorithm for suffix stripping. Program, 14(3):130–137.

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

Raghavan, V. V. and Wong, S. M., 1986. A critical analysis of vector space model for information retrieval. Journal of the American Society for information Science, 37(5):279–287.

Ramos, J. et al., 2003. Using tf-idf to determine word relevance in document queries. In Proceedings of the first instructional conference on machine learning, volume 242, pages 133–142. Piscataway, NJ.

Reis, C. R. and de Mattos Fortes, R. P., 2002. An overview of the software engineering process and tools in the mozilla project. In Proceedings of the Open Source Software Development Workshop, pages 155–175.

Runeson, P., Alexandersson, M., and Nyholm, O., 2007. Detection of duplicate defect reports using natural language processing. In 29th International Conference on Software Engineering (ICSE'07), pages 499–510. IEEE.

Sun, C., Lo, D., Khoo, S.-C., and Jiang, J., 2011. Towards more accurate retrieval of duplicate bug reports. In 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), pages 253–262. IEEE.

Sureka, A. and Jalote, P. (2010). Detecting duplicate bug report using character n-gram-based features. In 2010 Asia Pacific Software Engineering Conference, pages 366–374. IEEE.

Team, L. Bug repo Github: logpai, 2020.

Tian, Y., Sun, C., and Lo, D., 2012. Improved duplicate bug report identification. In 2012 16th European Conference on Software Maintenance and Reengineering, pages 385–390. IEEE.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L.,Gomez, A. N., Kaiser, L., and Polosukhin, I., 2017. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008.

Wang, X., Zhang, L., Xie, T., Anvik, J., and Sun, J., 2008. An approach to detecting duplicate bug reports using natural language and execution information. In Proceedings of the 30th international conference on Software engineering, pages 461–470.