T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SOSYAL MEDYA LOKASYON ANALİZİ

YÜKSEK LİSANS TEZİ

Yahya ALALI

Bilgisayar Mühendisliği Anabilim Dalı

TEMMUZ 2024

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

SOSYAL MEDYA LOKASYON ANALİZİ

YÜKSEK LİSANS TEZİ

Yahya ALALI

Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı: Prof.Dr. Nilüfer YURTAY

TEMMUZ 2024

Yahya ALALI tarafından hazırlanan "SOSYAL MEDYA LOKASYON ANALİZİ" adlı tez çalışması 05.07.2024 tarihinde aşağıdaki jüri tarafından oy birliği/oy çokluğu ile Sakarya Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Anabilim Dalı'nda Yüksek Lisans tezi olarak kabul edilmiştir.

Tez Jürisi

Jüri Üyesi:	**Prof.Dr. Nilüfer YURTAY** (Danışman) Sakarya Üniversitesi	.....
Jüri Üyesi :	**Dr. Öğr. Hüseyin DEMİRCİ** Sakarya Üniversitesi	.....
Jüri Üyesi :	**Doç. Dr. Halit Öztekin** Sakarya Uygulamalı Bilimler Üniversitesi	.....

ETİK İLKE VE KURALLARA UYGUNLUK BEYANNAMESİ

Sakarya Üniversitesi Fen Bilimleri Enstitüsü Lisansüstü Eğitim-Öğretim Yönetmeliğine ve Yükseköğretim Kurumları Bilimsel Araştırma ve Yayın Etiği Yönergesine uygun olarak hazırlamış olduğum "SOSYAL MEDYA LOKASYON ANALİZİ" başlıklı tezin bana ait, özgün bir çalışma olduğunu; çalışmamın tüm aşamalarında yukarıda belirtilen yönetmelik ve yönergeye uygun davrandığımı, tezin içerdiği yenilik ve sonuçları başka bir yerden almadığımı, tezde kullandığım eserleri usulüne göre kaynak olarak gösterdiğimi, bu tezi başka bir bilim kuruluna akademik amaç ve unvan almak amacıyla vermediğimi ve 20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince Sakarya Üniversitesi'nin abonesi olduğu intihal yazılım programı kullanılarak Enstitü tarafından belirlenmiş ölçütlere uygun rapor alındığını, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun ortaya çıkması halinde doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi beyan ederim.

(05 /07 /2024).

(imza)

Yahya Alalı

Hasta babama, rahmetli anneme ve henüz tanışmadığım eşime adanmıştır.

TEŞEKKÜR

Prof. Dr. Nilüfer Yurtay'a hocam, bu tezin tamamlanması sürecinde göstermiş olduğu sonsuz sabır, rehberlik ve desteği için en içten teşekkürlerimi sunarım. Değerli bilgileriniz, teşvikiniz ve inancınız olmadan bu çalışmayı tamamlayabilmem mümkün olmazdı.

Ayrıca, bu çalışmanın gerçekleşmesine olanak tanıyan Sakarya Üniversitesi Lisansüstü Eğitim Enstitüsü'ne ve tez sürecim boyunca yanımda olan ve desteklerini esirgemeyen tüm arkadaşlarıma ve aileme teşekkür ederim.

Yahya Alalı

İÇİNDEKİLER

Sayfa

KISALTMALAR

API : Application Programming Interface
APP : Application
CDN : Content Delivery Network
CMD : Command Prompt
DOM : Document Object Mode
EXIF : Exchangeable Image File Format
FCM : Firebase Cloud Messaging
GCL : Geo Context Locator
GCS : Google Cloud Storage
GIS : Geographic Information Systems
GUI : Graphical User Interface
GPS : Global Positioning System
ID : Identity Document
IDE : Integrated Development Environment
IOS : iPhone Operating System
JSON : JavaScript Object Notation
NoSQL : Not only Structured Query Language
NPM : Node Package Manager
OCR : Optical Character Recognition
OpenCV : Open-Source Computer Vision Library
RGBA : Red, Green Blue Alpha
SDK : Software Development kit
UI : User Interface
URL : Uniform Resource Locator
XML : Extensible Markup Language

ŞEKİL LİSTESİ

Sayfa

SOSYAL MEDYA LOKASYON ANALİZİ

ÖZET

Bu araştırma, popüler sosyal medya platformu Instagram'ın içeriğinin analizini amaçlamaktadır. Web tarama tekniklerini kullanarak bu platformdan belirli veriler toplanmıştır ve bu verilerin toplanması, Instagram kullanıcılarının davranışları ve eğilimleri hakkında daha derin bir anlayış sağlamayı amaçlamaktadır. Bu çalışma, uygulamalı uygulamasında Python programlama diline dayanmaktadır.

Şu anda, sosyal medya platformları, insanlar arasında kolay iletişim ve etkileşimi teşvik etmek için önemli bir araç haline gelmiştir. Milyonlarca kişi, deneyimlerini ve içeriklerini bu platformlar aracılığıyla paylaşır, bu da zengin bir veri kaynağı oluşturmaktadır.

"SOSYAL MEDYA LOKASYON ANALİZİ" adlı çalışma, veri madenciliği tekniklerine dayalı bir masaüstü uygulaması geliştirmeyi amaçlamaktadır. Ayrıca, Instagram sosyal medya platformu kullanıcılarının coğrafi konum bilgilerini çıkarmak için görsel analiz de kullanılmaktadır. Bu, kullanıcıların daha önce paylaştıkları gönderilerin analizi yoluyla gerçekleştirilir. Bu uygulamanın amacı, Instagram kullanıcılarının, gönderilerinden çıkarılan coğrafi konum bilgilerine dayanarak arkadaşlarının bulunduğu yerleri tahmin etmelerine yardımcı olmaktır.

Bu çalışma, temel olarak Python programlama dilindeki instaloader kütüphanesine dayanmaktadır. Bu kütüphane, Instagram hesaplarına giriş yapma sürecini doğrulamak ve giriş yapılan hesap için kimlik doğrulama işlemi tamamlandıktan sonra takipçi listesini almak için kullanılmaktadır. Bu listeden bir kullanıcı belirlendikten sonra, hedef kullanıcı olarak bilinen bu kullanıcının gönderi bilgileri kazınır. Bu bilgiler iki türe ayrılır: JPG formatında olan fotoğraflar ve meta verileri (Post Metadata) olarak bilinen günlük bilgileri, bu JSON formatında bir dosyadır. Daha sonra bu veriler, özel bir dış veri tabanında saklanır. Bu çalışma, bu konuda popüler olan Firebase platformuna dayanmaktadır.

Bu çalışma, her gönderinin konum verilerini çıkarmak için üç farklı yöntem kullanmaktadır. Birincisi, gönderi, kullanıcının kendisinin paylaştığı ve instaloader kütüphanesinin gönderilerin meta verileri (Post Metadata) içinde kazıdığı "yer etiketi" konum verilerini içeriyorsa, bu durumda, aynı gönderiye ait fotoğrafların görsel analizine gerek kalmaz. Bu bilgiler, JSON dosyasında bulunan bilgilerdir. İkincisi, kullanıcı tarafından "yer etiketi" coğrafi konum verileri paylaşılmamışsa, "Google Cloud Vision" aracı, fotoğraflardaki ünlü yer işaretlerini tespit etmek ve bu yer işaretlerine ait coğrafi konum verilerini, yani enlem ve boylam koordinatlarını çıkarmak için görsel analizde kullanılır. Bu, konum verilerini "yer etiketi" içermeyen aynı gönderiye ait fotoğrafların analiz edilmesi anlamına gelir. Üçüncüsü, "Google Cloud Vision" aracı, aynı gönderinin fotoğraflarından herhangi bir konum bilgisi çıkaramazsa, üçüncü yönteme geçilir ve bu, her bir konum bilgisi içermeyen fotoğrafın görsel analizinde "OpenCV + TensorFlow" kütüphanesinin kullanılmasıdır. Aynı zamanda ünlü yer işaretlerinin tespit edilmesi ve bu yer işaretlerine ait koordinatların

çıkarılması anlamına gelir. Eğer önceki üç yöntemle herhangi bir coğrafi konum verisi bulunamazsa, bu, gönderinin herhangi bir konum verisi içermediği anlamına gelir ve bir sonraki gönderiye geçilir. Bu işlem, tüm gönderilerin analizi tamamlanana kadar devam eder.

Sonrasında, gönderilerden çıkarılan konum verileri (koordinatlar) analiz edilir, ardından hedef kullanıcının bulunması beklenen en sık tekrar eden ülke ve şehir belirlenir. Bundan sonra, bu konuda bir rapor sunulur, bu konumlar, kullanıcıya daha net bir görsel içerik sunmak için haritada (Google Maps) gösterilmektedir.

Örneğin, hedef kullanıcının sayfasındaki gönderiler arasında Türkiye'den beş, Almanya'dan üç ve Suudi Arabistan'dan bir gönderi varsa, programın oluşturduğu listede Türkiye, hedef kullanıcının bulunması beklenen ülke olarak ilk sırada yer almaktadır, ardından Almanya ve sonra Suudi Arabistan gelmektedir. Daha sonra, şehirler benzer şekilde ülkeler gibi sıralanır ve sınıflandırılır. Örneğin, Türkiye'deki gönderiler arasında İstanbul'dan üç, Sakarya'dan bir ve Kocaeli'den bir gönderi varsa, kullanıcıya, hedef kullanıcının bulunması beklenen en sık tekrar eden şehir hakkında bir rapor sunulmaktadır, bu İstanbul olmaktadır.

SOCIAL MEDIA LOCATION ANALYSIS

SUMMARY

This research is aimed at analyzing the content of one of the most famous social media platform Instagram, by collecting specific data from the platform using web crawling techniques. Web crawling techniques are very effective tools for scraping data which can be benefited from in numerous ways. The motivation behind this research is to understand the complex mechanisms that these techniques follow during the process of scraping and collecting data from the web, especially from sites like Instagram and Facebook, that seek to prevent their data from being scraped by preventing robots or crawling spiders from accessing and indexing this data. The aim of collecting customized data from is to provide an accurate and in-depth understanding of the behavior and trends of Instagram users.

The study relies in its practical application on Python programming language because it contains ready-made tools and libraries that are able to facilitate the practical implementation of this research, especially in the field of web crawling, as it is considered the leading language in this field.

With the continuous evolution of contemporary society, it cannot be denied that social media platforms have evolved to become an important tool for enhancing easy communication and interaction among people. These digital platforms are now considered the primary channels through which many people share their diverse experiences and content, collectively contributing to the creation of a vast, invaluable data repository that feeds the modern digital landscape.

The study "Social Media Location Analysis" aims to develop a desktop application that relies on data mining techniques and visual analysis to extract geographical location information of Instagram users, by analyzing the posts they have previously shared. The application will seek to help Instagram users guess the whereabouts of their friends based on the geographical location information extracted from their posts.

The practical application of this study begins with creating the user interface, relying on the Tkinter library. This interface is a login page, providing easy access to the personal Instagram accounts of users, through their username and password. It somewhat resembles the main Instagram interface. This study primarily relies on the Instaloader library (a Python library) in the process of authenticating logins into Instagram accounts. After successful authentication is completed in the login process, the user is directed to a second page which is the followers list of the account that has been logged into. The Instaloader library is responsible for fetching this list. After that, a username is selected from this list or entered manually for a username not on the follower list. The interface also allows targeting accounts that are not on the followers or following list of the logged in account as long as they are not private accounts. The name that is chosen or manually entered is known as the targeted user.

After that, the program is smoothly directed to the profile page of the targeted user (this process occurs in the background without appearing to the user), where the process of scraping information from this user's posts begins. The Instaloader library is also responsible for the process of scraping and saving this information in a special folder that bears the same name as the targeted user and is within the project folders. The collected information is divided into two types; 1) images that are in JPG format, and 2) daily information known as Post Metadata, which is a JSON file. The Instaloader library gives the JSON file of the Post metadata and the images that follow the same post the same name, with only the extension differing. For ease of handling the information of each post separately, name to be given to the files will be the date of sharing the post separately. After that, this information is transferred and saved in a special external database., In this study, Firebase platform was selected to serve that purpose based on its reliability.

This study follows three methods in extracting location data for each post. First, if the post contains the geographical data of the location "location tag" which the user shares himself/herself, the instaloader library scrapes it also within the descriptive data (Post Metadata) of the posts. In such cases where the information in the JSON file is available, there is no need to perform visual analysis of the images that follow the same post. Therefore, the process of analyzing and extracting coordinate information for this post ends here and does not move on to the remaining methods. Second, if the geographical location data "location tag" is not shared by the user, "Google Cloud Vision" tool is used, which is a paid service provided by Google, similar in its mechanism to Google Lens, where it performs visual analysis and detects famous landmarks in the images and extracts geographical location data that follow these landmarks., i.e., longitude and latitude coordinates, where the images that follow the same post that does not contain descriptive data on location information "location tag" are analyzed, which is every image that carries the same name as the JSON file for this post. If location data is found, the image analysis process stops here, and does not move on to the third method in the analysis. Third, if the "Google Cloud Vision" tool was not able to extract any location information from the same post images, the program moves on to the third method, which is using the "OpenCV + TensorFlow" library. This method relies on the use of machine learning models dedicated to visually analyzing of images by detecting famous landmarks and extracting the coordinates of these landmarks. If no location data is found using any of the previous three methods, it means that the post does not contain any location data, and the program moves on to the next post, until all posts are analyzed.

After that, the extracted location data (coordinates) from the posts are analyzed, where these coordinates are converted into addresses using the library GeoPy. The geographical data of interest in this study and the object is the identification of the country and city. Upon completion of the analysis of the targeted user's posts, it becomes possible to anticipate the residency location of the user where the user is expected to be residing in the most frequent city/country in the user posts. These results are displayed on a special page using the library Matplotlib with the names of the most frequently occurring country and city shown at the end of the page to serve as a final report presented to the user. In addition, the coordinates are then pinned as markers on the map (Google Maps) to provide the user with a clearer visual content.

For example, if there are five posts from Turkey, three from Germany, and one from Saudi Arabia on the targeted user's page, Turkey will be first in the list generated by

the program, the country where the targeted user is most likely located, followed by Germany and then Saudi Arabia. Similarly, cities are arranged and ranked in the same manner as countries. For instance, if there are three posts from Istanbul, one from Sakarya, and one from Kocaeli, Turkey, the report will show Istanbul as the most frequent city where the target user is expected to be located.

1. GİRİŞ

Web Tarama, internet üzerinde otomatik olarak gezinerek web sitelerinin içeriğini tarayan ve toplayan bir işlemdir. Bu işlemi gerçekleştiren yazılımlara 'Web Crawler' denir. Web crawler'lar, web sitelerini ziyaret eder, sayfaları indirir, içeriklerini analiz eder ve bu verileri daha sonra indeksleme, veri madenciliği, arama sonuçları oluşturma gibi amaçlarla kullanabilirler. Ayrıca, web crawler'lar büyük veri analizi ve elektronik madencilikte yoğun olarak kullanılmakta olup, değerli bilgiler ve veri setlerinden desenler çıkarmak için kullanılırlar, bu da ticari işletmeler ve akademik araştırmalar için değerli içgörüler sağlar.

1.1. Web Tarayıcı

1.1.1. Web crawler'ların çalışma mekanizması

Çeşitli arama motorları, örneğin Google, Bing, Yahoo!, DuckDuckGo, Baidu ve Yandex gibi, web sayfalarını indekslemek amacıyla örümcek botlarını kullanır. Bu botlar, tarama işlemlerine genellikle yüksek trafikli web sitelerinden başlar ve temel amaçları, her web sayfasının içeriğini özetlemektir. Böylece, web örümcekleri bu sayfalardaki kelimeleri tarayarak, kullanıcıların arama yapmaları durumunda arama motorlarının kullanacağı pratik bir kelime listesi oluşturur. İnternet üzerindeki tüm sayfalar, hiperlinkler aracılığıyla birbirine bağlıdır. Bu nedenle, site örümcekleri bu bağlantıları keşfedebilir ve ardışık sayfalara kadar izleyebilir. Web botları, tüm içeriği ve bağlantılı web sitelerini taradıklarında işlemlerini durdurur ve elde edilen bilgiler, dünya genelindeki sunuculara gönderilerek bir arama indeksi oluşturulur. Bu süreç, gerçek dünyadaki bir örümcek ağı gibi karmaşık bir yapıdadır. Sayfalar indekslendikten sonra bile, arama motorları, sayfalarda meydana gelen değişiklikleri tespit etmek için düzenli olarak web örümceklerini kullanmaya devam eder. Herhangi bir değişiklik algılandığında, ilgili arama motorunun indeksi buna göre güncellenir.

1.1.2. Web örümcekleri (web crawler) çalışma süreci

1. URL Başlatma: Web örümceği, belirlenen bir başlangıç URL'si ile tarama sürecine başlar. Bu URL, taranacak olan web sitesini veya sayfayı işaret eder.

2. Sayfa İndirme: Web örümceği, ilgili URL'yi ziyaret eder, sayfanın kaynak kodunu indirir ve içeriğini elde eder.
3. Veri Analizi: İndirilen sayfa içeriği, metin, resimler, bağlantılar ve diğer veri türleri açısından analiz edilir.
4. Bağlantı Keşfi: Sayfada bulunan bağlantılar tespit edilir ve bu bağlantılar, diğer sayfaları tarayabilmek için kullanılır.
5. Yineleme: Keşfedilen yeni bağlantılar, tarama sürecinin yeni başlangıç noktaları olarak kullanılır ve süreç yukarıda belirtilen adımlarla tekrarlanır. Böylece web örümceği, genişleyen bir ağ yapısını keşfeder ve verileri toplar.

Web örümcekleri, internet üzerindeki geniş veri yığınlarını toplamak ve bu verileri işlenebilir hale getirmek için hayati öneme sahiptir. Ancak, bu süreç aynı zamanda web site sahiplerinin gizlilik ve güvenlik konularındaki endişelerini de beraberinde getirebilir. Bu nedenle, web örümceklerinin kullanımı etik kurallar ve yasal düzenlemeler çerçevesinde gerçekleştirilmelidir.

1.1.3. Web tarama türleri

Web tarayıcıları, yalnızca arama motoru örümcekleriyle sınırlı değildir; çeşitli web tarama türleri mevcuttur. İşte bazı örnekler:

1. E-posta Tarama (Email Crawling): E-posta adreslerini toplamak amacıyla kullanılır. Bu tarayıcılar, web sitelerinde veya forumlarda bulunan e-posta adreslerini derleyerek veritabanlarına veya listelere ekler.
2. Haber Tarama (News Crawling): Haber sitelerini tarayarak güncel haber başlıkları, içerikleri ve diğer ilgili verileri toplar. Bu tarayıcılar, haber ajansları veya medya siteleri tarafından kullanılabilir.
3. Görüntü Tarama (Image Crawling): Görsel içerikleri toplamak için kullanılır. Web tarayıcıları, görsel dosyaları indirerek veya görsel içeriklerin URL'lerini takip ederek bu verileri toplar.
4. Sosyal Medya Tarama (Social Media Crawling): Sosyal medya platformlarını tarayarak kullanıcı gönderileri, profiller ve diğer sosyal verileri toplar. Bu tarayıcılar, trend analizi, duygu analizi veya kullanıcı davranış analizi için kullanılabilir.

Şekil 1.1. Sosyal medya örümcekleri

5. Video Tarama (Video Crawling): Video içeriklerini toplamak için kullanılır. Web tarayıcıları, video paylaşım sitelerindeki videoların başlıklarını, açıklamalarını ve ilgili diğer verileri toplar.

Bu çeşitli tarama türleri, özel veri ihtiyaçlarını karşılamak ve farklı amaçlar için tasarlanmıştır. Her tür, belirli bir veri türünü veya içeriği hedef alarak, veri toplama ve analiz işlevini yerine getirir.

1.1.4. Web tarayıcılarının örnekleri

Arama motorları, web sayfalarını indekslemek ve içerikleri analiz etmek için kendi özel tarayıcı botlarını kullanır. İşte en yaygın web tarayıcılarından bazıları:

1. Googlebot: Google'ın arama motoru için web sayfalarını taramak ve indekslemek üzere tasarlanmıştır.
2. Bingbot: Bing arama motoru için web sayfalarını taramak ve indekslemek amacıyla kullanılır.
3. Yandex Bot: Yandex arama motorunun içerikleri taraması için geliştirilmiştir.
4. Baidu Spider: Çin'in önde gelen arama motoru Baidu için web sayfalarını taramak üzere kullanılır.
5. Mozilla's GeckoView: Mozilla tarafından geliştirilen bir web tarayıcı motorudur.
6. Applebot: Apple'ın arama teknolojileri için web sayfalarını taramak amacıyla kullanılır.

7. Facebook Crawler: Facebook'un platformunda paylaşılan içerikleri indekslemek için kullanılır.
8. Twitterbot: Twitter'da paylaşılan içerikleri indekslemek için kullanılır.
9. LinkedIn Bot: LinkedIn'ın platformunda paylaşılan içerikleri indekslemek amacıyla kullanılır.
10. Ahrefs Bot: SEO analizi için web sayfalarını taramak ve indekslemek üzere kullanılır.
11. Semrush Bot: Dijital pazarlama analizleri için web sayfalarını taramak amacıyla kullanılır.
12. Scrapy: Açık kaynaklı bir web tarama çerçevesidir.
13. Alexabot: Alexa sıralaması için web sayfalarını taramak ve indekslemek üzere kullanılır.
14. Yahoo! Slurp Botu: Yahoo! arama motoru için web sayfalarını taramak ve indekslemek amacıyla kullanılır.

Bu listelenenler, internette bulunan çok sayıda web tarayıcısından sadece birkaçıdır. Her biri, belirli özelliklere ve amaçlara göre tasarlanmıştır ve internetin derinliklerindeki bilgileri keşfetmek için önemli araçlardır.

1.1.5. Web crawler ve web kazıyıcı (scraper) arasındaki farklar

Web Crawler ve Web Kazıyıcı (Scraper), web sitelerinden veri toplamak için kullanılan iki temel araçtır. Her ne kadar işlevsellikleri benzer gibi görünse de aslında farklı görevleri yerine getirirler. İşte bu iki araç arasındaki temel farklar:

1. Web crawler (web örümceği):

- Web örümcekleri, internet üzerinde otomatik olarak gezinerek web sayfalarını ve sitelerini tararlar.
- Temel amacı, arama motorlarının dizinlerini oluşturmak ve web sitelerini indekslemektir.
- Bağlantıları takip ederek bir siteden diğerine geçer ve metin, resim, bağlantılar gibi verileri toplar.
- Örneğin, Google'ın web örümcekleri, web sitelerini tarayarak Google'ın indekslerini oluşturur.

2. Web kazıyıcı (scraper):

- Web kazıyıcılar, belirli web sitelerinden hedeflenen verileri çekmek için kullanılır.

- Toplanan veriler genellikle belirli bir amaç doğrultusunda analiz edilir veya işlenir.
- Bir web sayfasının HTML kodundan belirli öğeleri (metin, resim, tablo vb.) çıkarır.
- Örneğin, bir e-ticaret sitesinden ürün fiyatlarını toplamak için web kazıyıcılar kullanılabilir.

1.1.6. Özel web tarayıcıları

Genel arama motorlarının ötesinde, belirli bir amaca hizmet etmek üzere tasarlanmış ve özelleştirilmiş yazılımlar olan özel web tarayıcıları, belirli web sitelerini veya içerikleri özel kriterlere göre tarayabilir, veri toplayabilir ve analiz edebilirler. Örneğin, bir şirket rekabet analizi yapmak istediğinde, rakip firmaların web sitelerini düzenli olarak tarayarak fiyatları, ürün özelliklerini ve kampanyalarını izlemek için özel bir web tarayıcı kullanabilir. Genel arama motorları bu tür spesifik verilere ulaşmak için yeterli olmayabilir, çünkü daha özelleştirilmiş bir yaklaşım gerekebilir. İşte bu noktada özel web tarayıcıları devreye girer.

Özel bir web tarayıcı geliştirmek, genellikle belirli bir programlama dili veya yazılım teknolojisi kullanılarak yapılır. Bu tarayıcılar, hedeflenen web sitelerine erişebilir, sayfaları indirebilir, içerikleri çıkarabilir ve gerektiğinde bu verileri analiz edebilir. Böylece, belirli bir projenin veya analizin ihtiyaçlarına uygun veriler elde edilir.

Genel olarak, özel web tarayıcıları, belirli bir görevi yerine getirmek için tasarlanmıştır ve güçlü araçlardır. Ancak, kullanımları sırasında etik ve yasal düzenlemelere uyulması büyük önem taşır.

1.2. Tezin Amacı

İnternetin yaygınlaşması ve çok sayıda web sayfası, sosyal medya sitesi ve diğer kaynakları içermesiyle birlikte, geniş veri yığınları arasından belirli bilgilere ulaşmak giderek daha zor bir hale gelmiştir. İnternet, içinde gizli bir hazine olarak kabul edilebilir ve özelleştirilmiş tarama robotları aracılığıyla belirli bilgilerin çıkarılması mümkündür. Bu robotlar, genellikle programcılar tarafından belirli görevleri yerine getirebilmek için geliştirilmiştir.

Bu araştırmanın temel amacı, Instagram kullanıcılarının etkileşimlerini ve mekânsal davranışlarını anlamak üzere günlük gönderilerden coğrafi konum verilerini çıkarmaktır. Geliştirilecek olan robot, bu verileri toplayarak ve analiz ederek,

kullanıcıların Instagram'daki aktivitelerini ve bu aktivitelerle nasıl etkileşime girdiklerini ayrıntılı bir şekilde inceleyecektir. Elde edilen bilgiler, pazarlama stratejilerinin daha etkin bir şekilde hedef kitlelere ve mekânlara yönlendirilmesine katkıda bulunabilir. Ayrıca, bu çalışma, Instagram'daki kullanıcı davranışlarını ve bu platformdan nasıl faydalanılabileceğini daha iyi anlamak isteyen şirketler ve araştırmacılar için değerli veriler sunmayı hedeflemektedir. Ancak, bu süreçte kullanıcı verilerinin gizliliğine ve veri koruma yasalarına büyük önem verilmesi gerekmektedir.

1.3. Literatür Araştırması

Çoğu önceki çalışma, sosyal medya platformlarının kullanıcılarının, gönderilerinde veya profil bilgilerinde paylaştıkları coğrafi konum bilgilerindeki konumlarını tahmin etmeye dayanmaktadır. Bu çalışmaların çoğu, şu anda "X" platformu olarak adlandırılan ünlü Twitter platformu etrafında dönmektedir, çünkü bu platform, web örümceklerinin ve tarayıcılarının ondan veri elde etmeleri için örneğin profil bilgileri, meta veriler ve tweet metni, hashtag bilgileri ve diğer bilgiler gibi olanaklar sağlar.

(Ishida, K. (2015)) tarafından yapılan çalışmada, "X" gibi mikroblog platformu kullanıcılarının konumları, kullanıcıların blog metinlerinden coğrafi konumla ilgili terimleri ve frekansları çıkararak tahmin edilmiştir. (Wang ve diğerleri (2018)) da Ishida, K'nın çalışmasına benzer bir çalışma yapmış ve kullanıcıların konumlarını tahmin etmek için Çin'in ünlü Weibo platformundaki küçük blogların metin içeriğini analiz etmeye ve bu içerikle ilgili coğrafi etiketleri çıkarmaya dayanmıştır. (Williams, E (2016)) tarafından yapılan çalışma ise, "X" platformunun kullanıcılarının konumlarını tahmin etmek için tweetleri iki farklı yönteme analiz etmiştir: Birincisi, kullanıcının tweetlerine eklediği GPS koordinat bilgilerini çıkararak; ikincisi, bu tweetlerin meta verilerini analiz ederek ve bunları daha önce hazırlanmış GCL modeline geçirerek tweetlerin konumlarını tahmin etmek. Öte yandan, (Khan ve diğerleri (2023)) "X" platformunun kullanıcılarının konumlarını tahmin etmek için tweet bilgilerine eklenen coğrafi koordinat GPS bilgilerine ve ayrıca profil analizine ve kullanıcının eklediği halinde ülke ve şehir adını çıkarmaya dayanmıştır.

(Xu ve diğerleri (2014)), Facebook'taki ünlü platformda kullanıcıların konumlarını tahmin edilmiştir. Bu tahmini yapmak için iki yöntem kullanılmıştır: Birincisi, kullanıcıların metin içeriklerinin analizi ve coğrafi etiketlerle ilgili bilgilerin

çıkarılmasına dayanmaktadır. İkincisi, bu Kullanıcının arkadaşlarının profil bilgilerinin incelenmesi ve bu profillerde şehir düzeyinde konum bilgileri içeren bilgiler bulunması durumunda kullanıcının konumunun tahmin edilmesi. Bu yöntemde, kullanıcıların çoğunlukla kendi konumlarına yakın kişileri takip etmeyi tercih ettiği varsayılmıştır.

Bu çalışma diğerlerinden farklı kılan birçok özellik bulunmaktadır. İlk olarak, Instagram platformuyla ilgilenmektedir. Çoğu çalışma, bu platformun kullanıcı verilerine erişimi ve kazımayı engellemeye çalışan Meta şirketinin sıkı kısıtlamaları nedeniyle bu platformla ilgilenmeyi tercih etmez. Bu çalışma, kullanıcıların paylaştığı tüm gönderi bilgilerine erişim sağlayarak ve bu bilgilerden konum bilgilerini çıkararak, bu kısıtlamaları Instagram Uygulama Programlama Arayüzü'nü kullanarak aşan kütüphane (instaloader) tarayıcı robotlarına dayanmaktadır.

İkincisi, bu çalışma sadece kullanıcıların doğrudan gönderilerine ekledikleri yer etiketi bilgilerini kazıma ve analiz etmeye dayanmamaktadır, buna ek olarak kullanıcıların yer etiketi eklemeyerek paylaştıkları gönderilerin fotoğraflarının görsel analizine de dayanmaktadır. Görsel analiz süreci, bu fotoğraflardaki ünlü yer işaretlerini keşfetmek ve onlardan coğrafi koordinat bilgilerini geri almak için öncelikle Google Cloud Vision API aracına dayanmaktadır. Google Cloud Vision API aracı bu fotoğraflardan yerle ilgili bilgileri keşfedemezse, ayrıca bu çalışmada kullanılan diğer yönteme, yani "OpenCV + TensorFlow" kütüphanesine geçirilir, bu kütüphane de fotoğrafların görsel analizini yapar ve bulunduğunda coğrafi koordinat bilgilerini geri verir.

Bu çalışmada kullanılan üç yöntem (gönderilerin meta verilerinden konum bilgilerini (yer etiketi) çıkarma, Google Cloud Vision API kullanarak fotoğraflar görsel analiz ve "OpenCV + TensorFlow" kullanarak görsel analiz), bu çalışmanın doğruluk oranını diğer çalışmalardan daha yüksek hale getirmiştir.

2. GENEL BİLGİLER

2.1. Sosyal Medya Kazıma

Web kazıma kavramına hâkim olunduğunda, sosyal medya kazıma kavramına kolayca hâkim olunur. Sosyal medya veri kazıma, genellikle kazıyıcı olarak adlandırılan otomatik bir veri kazıma aracı kullanılarak (Facebook, X (Twitter), Instagram vb.) gibi sosyal medya sitelerinden veri çıkarma işlemidir.

2.2. Sosyal Medya Kazıma Süreci

Sosyal medya kazıma genellikle bir dijital kazıyıcının (bir kod parçası) bir sosyal medya sitesinde nasıl çalışır. Dijital kazıma aracı çalışma adımları aşağıdaki (Şekil 2.2.)'de gösterilmektedir.

Şekil 2.1. Dijital kazıma aracı çalışma adımları

1. Adım:

Sosyal medya kazıma işlemine başlamak üzere, kazıma aracı, hedeflenen sosyal medya sitesinin belirtilen URL adresine istek gönderir. Bu isteğe yanıt olarak, site istenen bilgileri HTML formatında kazıma aracına geri gönderir.

Sosyal medyayı kazımaya başlamak için kazıma aracı, belirtilen URL adresini kullanarak hedef olan sosyal medya sitesinden kazımak istediği içeriği talep eder. Bu talebe yanıt olarak, istenen bilgileri HTML biçiminde kazıma aracına geri gönderir.

2. Adım:

Sunucu tarafından istenen bilgi HTML dosyası olarak geri gönderildiğinde, kazıyıcı, hedef verileri ayıklamak için HTML içeriğini "parse" etmeye, yani analiz etmeye başlar. Eğer hedef sayfa, veri kazıma işlemlerini engellemek için güvenlik mekanizmaları kullanıyorsa, kazıyıcının DOM ağacının karmaşık yapısını aşarak doğru verileri bulması ve ayıklaması gerekebilir. Bu süreç, Ajax teknolojisi gibi ileri düzey tekniklerin kullanılmasını gerektirebilir.

3. Adım:

Ayıklanan veriler daha sonra işlenir ve belirli bir formatta standartlaştırılır. Daha sonra, analiz ve kullanım için genellikle bir merkezi veri tabanında saklanır.

Bu süreç, sosyal medya platformlarından veri toplama ve analiz etme konusunda derinlemesine bir anlayış sağlar ve bu verilerin nasıl işleneceği ve kullanılacağı konusunda önemli bilgiler sunar.

2.3. Sosyal Medyadan Veri Kazıma: Ne Kazılmalı Ne Kazılmamalı?

Sosyal medya platformlarından kazılacak verilerin seçimi, projenin amacına göre değişiklik gösterir. Fotoğraflar, videolar, meta veriler, yorumlar ve iletişim bilgileri (e-posta, telefon numarası vb.) gibi öğeler, sosyal medya kazıma sürecinde hedeflenebilecek veri türlerinden bazılarıdır. Ancak, sosyal medya kazıma işlemine başlamadan önce, hedef platformun işleyişini anlamak ve etkileşim mekanizmalarını kavramak için iyi bir plan yapmak esastır.

Bu süreç, sadece teknik bir beceri seti gerektirmez, aynı zamanda yasal ve etik normlara uygun bir şekilde veri toplama konusunda da dikkatli olmayı gerektirir. Kazıma işlemi sırasında, kullanıcıların gizliliğine saygı göstermek ve platformun kullanım koşullarına dikkat etmek önemlidir. Bu nedenle, kazıma işlemi yapılırken, hangi verilerin toplanabileceği ve hangilerinin toplanamayacağı konusunda dikkatli bir değerlendirme yapılmalıdır.

2.4. Sosyal Medya Kazımanın Yararları

Sosyal medyadan veri kazımanın birçok faydası bulunmaktadır. Büyük şirketler, müşterilerinin görüşlerine büyük önem verir ve bu bilgilere ulaşmanın en etkili yolu, sosyal medyadaki paylaşımları, sohbetleri ve yorumları sosyal medya kazıma araçlarıyla analiz etmektir. Bu süreç, şirketler ile kullanıcılar arasındaki ilişkileri güçlendirir ve elde edilen verilere dayanarak kullanıcıların ruh hallerini anlamaya yardımcı olur. Çünkü elde edilen veriler somut ve değerli bilgiler içermektedir.

Şekil 2.2. Sosyal medya kazımanın yararları

2.5. Sosyal Medya Hesaplarının Kazımadan Korunması

Sosyal medya platformlarından sorumlu şirketler, kullanıcı hesaplarını sosyal medya kazıma faaliyetlerinden korumak için çeşitli yöntemler geliştirmiştir. Bu yöntemler arasında:

1. İki Faktörlü Kimlik Doğrulama (2FA): Kullanıcı hesaplarının güvenliğini artırmak için iki faktörlü kimlik doğrulama yöntemi zorunludur. Bu yöntem aktif olduğunda, kullanıcıdan şifre girişi sonrasında ek bir onay kodu (SMS veya uygulama üzerinden) gibi ikinci bir bilgi istenir.
2. Olağandışı Etkinliklerin İzlenmesi: Sosyal ağlar, olağandışı giriş denemeleri veya beklenmedik etkileşimler gibi şüpheli faaliyetleri tespit etmek için gelişmiş algoritmalar kullanır.

3. Davranış Analizi: Şirketler, kazıma girişimlerini işaret edebilecek anormal aktiviteleri belirlemek için kullanıcı davranışlarını analiz eder. Bu, faaliyet modellerindeki ani değişiklikler veya beklenmeyen coğrafi bölgelerden erişim girişimleri olabilir.
4. Veri Koruma: Kullanıcı verilerinin korunması amacıyla, altyapı ve veri merkezleri düzeyinde sıkı güvenlik önlemleri alınmaktadır.
5. Robot Tespiti: Otomatik programlar ve robotlar, kullanıcı hesaplarına erişim sağlamaya çalışırken tespit edilir ve bu tür erişimler engellenir.

Bu önlemler, sosyal medya kazıma tehditlerine karşı tam bir koruma sağlamasa da, riskleri önemli ölçüde azaltmaya yardımcı olur ve sürekli olarak güvenlik zorluklarına uyum sağlamak için geliştirilmektedir.

Şekil 2.3. Sosyal Medya Hesaplarının Kazımadan Korunması

2.6. Veri Kazıma ve Web Taraması Arasındaki Farklar

Veri kazıma (Data Scraping) ve veri taraması (Data Crawling), internet üzerinden bilgi toplama süreçlerini tanımlamak için kullanılan iki farklı terimdir. Her ne kadar bazen birbirlerinin yerine kullanılsalar da aslında aralarında belirgin farklar vardır.

Veri Kazıma (Data Scraping):

Bu işlem, web sayfalarından otomatik olarak veri çıkarmayı ifade eder. Otomatik programlar veya araçlar kullanılarak gerçekleştirilen veri kazıma, genellikle belirli bilgileri—fiyatlar, kullanıcı değerlendirmeleri veya diğer özel veriler gibi—hedef alır.

Bu süreç, metin veya görsel içerikten veri çıkarmak için gelişmiş analiz tekniklerinin kullanılmasını da içerebilir.

Veri Taraması (Data Crawling):

Bu terim, web sitelerini otomatik olarak taramak ve içeriklerini sistemli bir şekilde incelemek anlamına gelir. Bir veri tarayıcısı, web sayfaları arasında gezinmek ve bağlantılar ile içerikleri toplamak için kullanılır. Bu süreç, internetin yapısını ve organizasyonunu anlamak ve temel verileri—bağlantılar, adresler gibi—toplamak amacıyla yapılır.

Genel olarak, veri kazıma işlemi, daha geniş bir veri tarama sürecinin bir parçası olarak görülebilir. Burada, veri tarama yoluyla erişilen web sayfalarından istenen bilgileri çıkarmak için veri kazıma teknikleri devreye girer.

3. MATERYAL VE YÖNTEM

3.1. Materyal

Uygulamalı "Sosyal Medya Lokasyon Analizi" çalışmasını gerçekleştirmek için, PyCharm geliştirme ortamı ve Python programlama dili kullanılmaktadır.

3.1.1. Python programlama dili

Yüksek seviyeli bir programlama dili olarak, yazımı, okunması ve öğrenilmesi kolaydır. Bu dil açık kaynaklıdır, geliştirilebilir ve Nesne Yönelimli Programlama (OOP) paradigmalarını destekler. 1986 yılında "Guido Van Rossum" tarafından geliştirilmiştir ve 1991 yılında herkesin erişebileceği ilk versiyonu yayınlanmıştır. Web siteleri, web ve masaüstü uygulamaları geliştirmeden, oyun yazılımları, yapay zekâ, veri bilimi ve makine öğrenimi gibi geniş bir uygulama yelpazesi geliştirmek için kullanılan kapsamlı ve çok yönlü bir programlama dilidir. Aşağıdaki (Şekil 3.1.)'de Python simgesi görünmektedir:

Şekil 3.1. Python simgesi

Özel amaçlar için birçok programlama kütüphanesine sahiptir, örneğin "PyGame" kütüphanesi oyun programlamak için bir dizi fonksiyon sağlar, ayrıca en önemli veri tabanlarıyla etkileşim için hazır arayüzler sunar ve bilgi güvenliği ve etik hacking alanında ilgilenenler tarafından kullanılan en önemli dillerden biridir.

Bu çalışmada Python dilinin tercih edilmesinin birkaç önemli nedeni vardır:

1. Web kazıma (Web Scraping) işlemini ve veri çıkarmayı kolaylaştıran bir dizi kütüphane ve araca sahiptir. Örneğin (BeautifulSoup, Scrapy) ve bu çalışmada temel olarak kullanılacak kütüphane olan (instaloader).
2. (C, C++, Java) gibi diğer programlama dilleriyle kolayca etkileşim kurabilir. Ayrıca, çoğu programlama dili de onunla etkileşimi destekler.
3. (Windows, Mac OS, Linux, Unix) gibi tüm işletim sistemlerinde çalışır.
4. Geliştiricilere (Android, IOS) mobil uygulamalar geliştirme imkânı sağlar. Bunun için önemli kütüphaneler arasında (Kivy, PyQt, PySide) bulunmaktadır.

3.1.2. PyCharm

Python programlama dili ile çalışmak için tasarlanmış bir Tümleşik Geliştirme Ortamı (IDE)'dir. JetBrains şirketi tarafından geliştirilmiştir. PyCharm, geliştiricilerin Python'da kod yazmasını, düzenlemesini, çalıştırmasını ve test etmesini kolaylaştıran bir dizi özellik sunar. Bu özellikler arasında otomatik tamamlama (autocomplete), hata analizi (error analysis), dosyalar ve projeler arasında hızlı geçiş yapabilme, dilbilgisi ve yazım hatalarını düzeltme ve Git gibi Sürüm Kontrol Sistemleri (Version Control Systems) ile entegrasyon bulunmaktadır. Ayrıca Django kütüphanesi ile web geliştirmeyi, Anaconda1 kütüphanesi ile de veri bilimi uygulamaları üzerinde çalışmayı desteklemektedir. PyCharm, Linux, Mac OS ve windows üzerinde çalıştırılabilen çok platformlu bir programdır.

Kısacası, PyCharm, Python uygulamalarını geliştirmek için kullanışlı ve güçlü bir araçtır ve çeşitli programlama projelerinde etkin bir şekilde kullanılabilir. Aşağıdaki (Şekil 3.2.)'de PyCharm simgesi görünmektedir:

Şekil 3.2. PyCharm simgesi

3.2. Yöntem

3.2.1. Ana sayfa

Bu çalışma için uygulamanın ana sayfası, (tkinter) kütüphanesi kullanılarak oluşturulmuştur. Python'daki Grafik Kullanıcı Arayüzü (GUI) kütüphanesi olan tkinter, geliştiricilere temel kullanıcı arayüzü bileşenlerini (düğmeler, menüler, metin kutuları, resimler vb.) kullanarak grafik arayüzler oluşturma imkânı sağlamaktadır.

Ancak oluşturulacak uygulama bir mobil uygulama olarak kullanılacaksa, (tkinter) kütüphanesi mükemmel bir araç değildir. Olası seçeneklerden biri, cep telefonları da dahil olmak üzere farklı platformlarda çalışan tek bir uygulamanın oluşturulmasına izin verebilecek (kivy) gibi çok platformlu bir çerçeve kullanmaktır.

Ana sayfa, popüler sosyal medya platformu Instagram'ın giriş sayfasına benzer bir giriş sayfasından oluşmaktadır. Bu uygulamanın özel bir simgesi, kullanıcı adı için bir diyalog kutusu ve şifre girişi sırasında şifrenin gösterilmesini ve gizlenmesini sağlayan bir şifre diyalog kutusu içerir. Ayrıca, giriş işlemini gerçekleştirmek için bir düğme bulunmaktadır. Ana sayfa tasarlandıktan sonra aşağıdaki (Şekil 3.3)'de gösterildiği gibi görünmektedir:

Şekil 3.3. Ana sayfa

Bu arayüzü kullanabilmek için kullanıcının Instagram platformunda bir hesabının olması gerekmektedir, çünkü bu sayfa yeni bir hesap oluşturmayı veya mevcut şifreyi değiştirmeyi desteklememektedir. Bunu gerçekleştirmek için, (Instagram) platformunun ana sayfasına gitmek gerekmektedir.

Bu sayfa, Instagram hesaplarına giriş işlemini doğrulamak için tasarlanmıştır ve doğrulama işlemi, (instaloader) kütüphanesi kullanılarak gerçekleştirilir.

3.2.1.1. Instaloader kütüphanesi

Python dilinde açık kaynaklı bir programlama kütüphanesidir ve Instagram'dan içerik indirmeyi amaçlamaktadır. İlk olarak 2017 yılında yayımlanmıştır. Instaloader, Instagram verilerini analiz etmek isteyen araştırmacılar ve geliştiriciler için güçlü bir araçtır Instaloader Documentation (2023). Aşağıdakiler de dahil olmak üzere birçok fonksiyon sunar:

- Profillerden fotoğraf ve videoların indirilmesi.
- Gönderilerin yorumlar ve beğeniler gibi meta verilerini toplaması.
- Hikayeleri ve onlarla ilişkilendirilmiş meta verileri indirilmesi.
- Kullanıcıların profil fotoğraflarını indirleyebilme yeteneği.
- Instaloader, genel ve özel hesaplardan içerik indirmeyi destekler, ayrıca belirli hashtag'lerden ve sitelerden içerik indirmeyi destekler.

Giriş işlemi (instaloader) kütüphanesi kullanılarak gerçekleştirilir, bu kütüphaneden bir (**ig**) nesnesi oluşturulur ve giriş işleminden sorumlu olan (login) fonksiyonu çağrılır.

```
ig = instaloader. Instaloader ()

ig.login (user = username_data, passwd = password_data)
```

Ana sayfada kullanıcı tarafından girilen kullanıcı adı ve şifre olan (password_data) ve (username_data), (login) fonksiyonundaki (user) ve (passwd) ile karşılaştırılmaktadır.

Instaloader programlama kütüphanesindeki (login) fonksiyonu doğrudan Instagram sunucularıyla iletişim kurar. Kullanıcı adı ve şifre kullanılarak giriş yapılırken, kullanıcının girdiği kullanıcı adı ve şifreyi içeren bir istek Instagram'a gönderilir.

Instagram, kullanıcı adının ve şifrenin doğruluğunu kontrol eder. Eğer veriler doğruysa, Instagram giriş işleminin başarılı olduğunu onaylayan bir yanıt gönderir. Ancak, veriler yanlışsa, Instagram giriş işleminin başarısız olduğunu belirten bir yanıt gönderir. Bu durumda, (login) fonksiyonu bir (instaloader.BadCredentialsException) istisnası döndürür.

Dolayısıyla, (login) fonksiyonu kullanıcı adının ve şifrenin doğru olup olmadığını kendisi bilmez. Bunun yerine, giriş işleminin başarılı olup olmadığının belirlenmesi Instagram'dan gelen yanıta bağlıdır.

Giriş işlemi başarıyla gerçekleştirildiğinde, (Takipçi listesi) sayfasına yönlendirilecektir.

3.2.2. Takipçi listesi sayfası

Giriş işleminin başarılı bir şekilde tamamlanmasından sonra bu sayfa göründüğü için takipçi listesini göstermekten sorumlu sayfadır ve bu sayfa kütüphane (tkinter) kullanılarak ana sayfada olduğu gibi tasarlanmıştır.

Bu sayfada, öncelikle giriş yapılan hesabın profil fotoğrafı ve bu kullanıcıya hoş geldiniz mesajı görüntülenir. Kullanıcının profil fotoğrafı, bundan sorumlu olan (profile_pic_url) fonksiyonunu kullanılarak (instaloader) kütüphanesi aracılığıyla getirilir.

```
profile = instaloader.Profile.from_username(ig.context, username_data)

image_url = profile.profile_pic_ur
```

Daha sonra, öncelikle giriş yapılan hesabın takipçi listesi alınır. Instagram platformu, her hesap için takipçi listesi ve takipçi listesi olmak üzere iki tür liste içerir ve takipçiler, bu hesabın takip ettiği hesapların listesidir ve takipçi listesi, bu hesabı takip eden hesapların listesidir. Instagram'da bir Instagram hesabının günlüğüne erişmek için takipçi listesinde olması gerekmektedir. Bu çalışma kullanıcıların günlüklerine erişmeye dayandığı için takipçi listesi getirilmektedir.

Takipçilerin listesi, bunu gerçekleştirmekten sorumlu olan (instaloader) kütüphanesini kullanarak (get_followees) fonksiyonu kullanılarak getirilmektedir.

```
profile = instaloader.Profile.from_username(ig.context, username_data)

followers = list(profile.get_followees())
```

Bu hesabın herkese açık olduğu durumunda Instagram kullanıcılarının günlüklerine erişmek mümkündür, Instagram'da herkese açık ve gizli hesaplar olmak üzere iki tür hesap bulunmaktadır. herkese açık hesaplar, herkesin ünlülerin hesapları gibi

paylaşımlarına erişebileceği hesaplardır. Gizli hesaplar, yalnızca takipçilerin paylaşılan günlüklere erişebildiği hesaplardır.

Bu çalışma, herkese açık hesapların manuel girişini de destekler, bu nedenle, kullanıcının takipçi listesinden seçerek veya hesap genel türdeyse manuel olarak girebileceği, yerini tahmin etmek istediği hesabı seçmesine olanak sağlayan bir diyalog kutusu eklenmiştir.

3.2.2.1. Instagram'dan günlük paylaşımların kazınması

Takipçi listesinden yerini tahmin etmek istediği kullanıcı adını seçildikten veya herkese açık bir hesap için manuel olarak girildikten sonra veri kazıma işlemi başlamaktadır.

Bilindiği üzere, Instagram'da çok sayıda kazınabilecek veri bulunmaktadır (hesap bilgileri, profil fotoğrafı, fotoğraflar, videolar, hikayeler vb.). Bu çalışma, her gönderinin bilgilerini ve yalnızca fotoğrafları kazımaya odaklanmaktadır.

Bu bilgilerin kazılması için, (instaloader) kütüphanesi bu konuda birçok kolaylaştırır. Bu Instagram hesabından tüm verileri kazıyan (download_profile) fonksiyonu aracılığıyla gerçekleşir.

ig. download_profile (username)

Ancak, (download_profile) fonksiyonu doğrudan kullanıldığında, hedef hesaptan mevcut tüm bilgiler kazınacaktır.

Bu çalışma, tüm bilgilere ihtiyaç duymamaktadır, örneğin (videolar, kullanıcının gönderisine eklediği yorum, (post_metadata_txt_pattern) fonksiyonunun topladığı ve (txt) türünde bir dosya olarak kazıdığı, hesap sahibinin gönderilerine arkadaşlarının eklediği yorumlar ve diğer bilgiler) kazınmasına gerek yoktur. Bu bilgilerin kazınması biraz zaman, bellek ve internet gerektirir. Bu nedenle, (instaloader) kütüphanesi, bu çalışmada ihtiyaç duyulmayan verilerin kazınmasını devre dışı bırakma özelliği sunmaktadır.

ig.download_videos = False

ig.download_video_thumbnails = False

ig.compress_json = False

ig.download_geotags = False

ig.download_comments = False

ig.post_metadata_txt_pattern = ""

(download_profile) fonksiyonunun niteliklerine eklenen bu özellikler, ihtiyaç duyulmayan verilerin kazınmasını engeller ve sadece her gönderinin bilgilerini ve ayrıca fotoğrafları kazımaktadır.

Verilerin kazınması istenen kullanıcı adı belirlendikten sonra, kazıma işlemi, takipçi listesi sayfasına eklenen (Lokasyon Analizi Başla) düğmesine basılarak başlatılır. Takipçi listesi sayfası tasarlandıktan sonra aşağıdaki (Şekil 3.4)'de gösterildiği gibi görünmektedir:

Şekil 3.4. Takipçi listesi sayfası

Kazıma işlemine tabi tutulan veriler iki türde bulunmaktadır:

- Her gönderiye ait bilgileri içeren veriler (Post Metadata) ve bu veriler bir (JSON) türünde dosyası biçimindedir.
- Fotoğraflara ait veriler (JPG) türündedir.

JSON dosyası: "JavaScript Object Notation"nin kısaltmasıdır. Kullanıcının kolaylıkla okuyup yazabileceği basit bir formattır. Verilerin temsil edilmesi ve farklı yazılım

sistemleri arasında paylaşılması için kullanılır çünkü kolayca programlar tarafından analiz edilebilmektedir. Aşağıdaki (Şekil 3.5.)'de JSON simgesi görünmektedir:

Şekil 3.5. JSON simgesi

Bir programlama dili değildir, verilerin temsili ve farklı programlama dilleri arasında veri alışverişini kolaylaştırmak amacıyla, farklı programlama dilleri arasında üzerinde anlaşmaya varılmış bir yöntemdir. JSON formatında temsil edilen veriler geçici bir veri tabanı olarak kabul edilebilmektedir. XML gibi diğer veri değişim formatlarına göre daha az depolama alanı gerektirmektedir. JSON dosyası, iki ana bölümden oluşan biçimlendirilmiş metin dizelerinden oluşur: Anahtarlar (Keys) ve Değerler (Values). Anahtar, veri değerinin benzersiz bir adını temsil etmektedir ve genellikle tırnak işaretleri arasına yerleştirilmektedir. Öte yandan, değer verileri temsil etmektedir ve metinler, sayılar, diziler vb. gibi birden fazla veri türünü temsil edebilmektedir. JSON, sunucu ve istemci (Client-Server) arasında veri alışverişi için yaygın olarak kullanılmaktadır. Çoğu programlama dilinde desteklenmektedir, burada JSON formatında yazılmış herhangi, bir metin bir nesneye (Object) dönüştürülür ve tersi de geçerlidir.

Instagram'dan kazılan JSON dosyaları, gönderiler ve profil hakkında meta veriler (Post Metadata) içerir.

Meta veriler (Metadata): diğer verilerin özelliklerini tanımlayan verilere atıfta bulunan genel bir terimdir. Geniş bir dizi bağlamda kullanılır, bu da medya dosyaları (fotoğraflar, videolar, metin belgeleri vb.) ve ayrıca yazılımlar, veri tabanları vb. Başlık, tarih, yazar, anahtar kelimeler, açıklama, haklar vb. gibi bilgileri içerir.

Post Metadata: Bu, belirli bir Metadata türüdür. Belirli bir internet platformunda (bloglar veya sosyal medya gibi) bir paylaşımla ilişkili bilgilere atıfta bulunur. Bu bilgiler genellikle (yayın tarihi ve saati, kullanıcı bilgileri, yer etiketleri (Tags), etkileşim bilgileri, URL'ler ve kaynaklar, konum bilgileri (enlem ve boylam) ve diğerleri) içerir.

EXIF: Dijital fotoğraflar için kullanılan belirli bir Metadata türüdür. Dijital kameralarla çekilen dijital fotoğraflar genellikle, fotoğrafın nasıl çekildiği hakkında çeşitli verileri açıklayan EXIF bilgileri içerir. Bu bilgiler genellikle (fotoğrafın çekildiği zaman damgası ve tarih, kamera ayarları gibi diyafram ve enstantane hızı, mevcutsa coğrafi etiketler (GPS) ve fotoğrafla ilgili diğer ayrıntılar) içerir.

Çeşitli türlerde meta veriler bulunmaktadır, ancak bu çalışma, yayınlarla (Post Metadata) ilgili meta verilere odaklanmaktadır. Instagram'dan kazılan JSON dosyası, bu yayınlarla ilgili birçok veriyi içerir ve burada önemli olan konum (location) bilgileridir.

Veri kazıma işlemi başladığında, (instaloader) kütüphanesi, verilerin kazınacağı kullanıcının adıyla aynı adı taşıyan bir klasör oluşturur. (dirname_pattern) parametresi, dosyaların kaydedileceği klasörün adını belirlemek için kullanılmaktadır. Bu parametrenin varsayılan değeri {target} 'dir, bu da dosyalar yüklendiğinde kullanıcının (hedef) adında bir klasör oluşturulacağı anlamına gelmektedir. Bunun amacı, her kullanıcının dosyalarını, adını taşıyan özel bir klasöre izole etmektir. Bu parametre aynı zamanda klasörü bu projenin bulunduğu yerde varsayılan olarak kaydeder.

```
self. dirname_pattern = dirname_pattern or "{target}"
```

Gerekli olduğunda, klasör adı ("{target}") değiştirilerek belirlenen adla değiştirilebilmektedir.

(Instaloader) kütüphanesi kullanılarak kazılan her gönderi için tüm dosyalar aynı adı taşımaktadır, yani, (JSON) dosyaları, fotoğraf dosyaları (JPG) ve diğerleri aynı adı alır, bu (filename_pattern) parametresi kullanılarak gerçekleştirilir. Her türlü dosya, bu gönderinin yayın tarihi ile adlandırılır ve aynı gönderi için ortak dosyaların adlandırılması bu çalışmayı kolaylaştırır. Eğer bir gönderinin (Post Metadata) JSON dosyasında bu gönderinin konum bilgilerini içermiyorsa, kullanıcının yayınladığı

bilgilerde sıklıkla paylaştığı, (yer etiketi) olarak bilinen, aynı dosya adına (JSON) sahip fotoğraflar, aynı ada sahip fotoğraf bu gönderiye ait olduğu için ayrıştırılmıştır.

```
self. filename_pattern = filename_pattern or "{date_utc}_UTC"
```

Klasör adıyla ilgilenirken olduğu gibi, kazılan dosyaların adını da değiştirebilir ve onlara varsayılan adın dışında bir ad verebilir. Bunun için ("{date_utc}_UTC")'yi değiştirerek ve istenilen adı vermek için onu dönüştürebilir.

Bu çalışma sadece görüntü analizine dayanmaktadır, videoları analiz etmemektedir, bu nedenle videoların kazınması (download_videos = False) ile devre dışı bırakılmıştır. Ancak bazı kullanıcılar, videoların da olduğu gönderilerde konum bilgilerini (yer etiketi) paylaşmaktadır, ayrıca, kazınan gönderilerin (JSON) dosyasındaki meta veriler konum bilgisi (yer etiketi) içerir, (download_videos = False) kullanımı, videoların ve bu gönderilerin meta verilerinin kazınmasını devre dışı bırakır, dolayısıyla, bu çalışmada sadece videoların kazılması devre dışı bırakılmış ve videolar olan gönderilerin meta verilerinin kazılması etkinleştirilmiştir, bu da bu çalışmanın sonuçlarının doğruluğunu arttırmaktadır.

Kazıma işlemi tamamlandıktan sonra, hedef kullanıcının adını taşıyan klasör, her gönderi için aynı adı taşıyan iki tür dosya (JSON, JPG) içermektedir ve ayrıca video gönderilerinin meta verileri için (JSON) dosyaları da içermektedir.

Ardından, bu klasörün ve içerdiği dosyaların, özel bir merkezi dış veri tabanına yüklenme işlemi başlar, bu çalışmada veri tabanı olarak (Firebase) kullanılmıştır.

3.2.3. Firebase platformu

Bu platform, web uygulamaları, akıllı telefon uygulamaları ve oyun uygulamaları için çeşitli hizmetler sunan bir geliştirme platformudur, Google şirketi tarafından piyasaya sürülmüştür. Bu platform, geliştiricilerin altyapıyı doğrudan yönetmeye gerek kalmadan uygulamalarını hızlı ve verimli bir şekilde oluşturmalarına ve çalıştırmalarına yardımcı olan çok çeşitli hizmetler ve araçlar sağlayarak uygulama geliştirme sürecini kolaylaştırmayı amaçlamaktadır. Ayrıca, Backend kod yazma ihtiyacını azaltır, çünkü bulut hizmetleri doğrudan SDK'lar aracılığıyla ele alınır. Ayrıca, uygulamaların kolayca yönetilmesine izin veren bir kullanıcı arayüzü (Console) bulunmaktadır, Firebase Documentation. (2024). Aşağıdaki (Şekil 3.6.)'de Firebase simgesi görünmektedir:

Şekil 3.6. Firebase simgesi

Firebase tarafından sunulan hizmetler şunları içerir:

1. Gerçek Zamanlı Veri tabanı (Realtime Database): uygulamaların verilere doğrudan erişmesine ve kullanıcılar arasında gerçek zamanlı olarak senkronize etmesine olanak tanıyan bir NoSQL veri tabanıdır.
2. Depolama (Storage): Geliştiriciler, fotoğraf ve video gibi dosyaları kolayca depolamak ve yönetmek için Firebase'i kullanabilmektedir.
3. Kimlik Doğrulama (Authentication): Firebase, kullanıcı hesaplarını yönetmek ve giriş yapmak için kimlik doğrulama hizmetleri sunar. Bu, e-posta, Google hesapları, Twitter ve diğerleri gibi çeşitli türlerde olabilir. Uygulamalarda güvenli ve kolay kimlik doğrulama yöntemleri sağlamaktadır.
4. Anlık Mesajlaşma (FCM): Bu hizmet, geliştiricilere mobil cihazlar ve web üzerinden kullanıcılara metin mesajları ve anlık bildirimler gönderme olanağı sağlamaktadır.
5. Barındırma (Hosting): Firebase, uygulamalar için bir web barındırma hizmeti sunarak web uygulamalarının hızlı ve kolay bir şekilde dağıtılmasını sağlamaktadır.
6. Analitik (Analytics): Firebase, uygulama kullanımı ve performansı hakkında ayrıntılı analizler sağlayarak geliştiricilerin kullanıcı davranışlarını anlamalarına ve kullanıcı deneyimini geliştirmelerine yardımcı olmaktadır.

Firebase, geliştiricilerin uygulamalarını kolaylıkla ve etkin bir şekilde oluşturup çalıştırmalarına yardımcı olmak için birçok diğer hizmet sunmaktadır.

3.2.3.1. Firebase'i python ile bağlama

Veri tabanına yüklenecek klasör, iki tür dosya (JSON, JPG) içerir, bu nedenle bu çalışma (Firebase Storage) hizmetine dayanmaktadır. Google tarafından sağlanan ve Firebase platformunun bir parçası olan bulut tabanlı bir veri depolama hizmetidir. Bu hizmet, bulutta fotoğraflar, videolar ve belgeler gibi dosyaların depolanması ve yönetilmesi için kullanılır. Kullanımı kolay bir Uygulama Programlama Arayüzü (API) sağlar, bu sayede geliştiriciler mobil uygulamalar, web uygulamaları ve sunuculardan dosyaları yükleyebilir ve indirebilir.

(Firebase Storage) ile (Python) projesini bağlamak için, (Firebase) ve (Firebase Storage)'a erişmek üzere (firebase_admin) ve (pyrebase4) kütüphaneleri kullanılabilir. Bu kütüphaneler, (Python) aracılığıyla (Firebase) ile kolayca etkileşim kurmayı sağlamaktadır. (SBDeveloper, 2023).

Bağlama işlemi aşağıdaki adımlara göre gerçekleştirilir:

1. Firebase Projesi Oluşturma:

(Firebase) resmî sitesine "https://console.firebase.google.com/" bağlantısı üzerinden gidildiğinde, yeni bir hesap oluşturulur veya daha önce var olan bir hesaba giriş yapılır. Ardından, "**Add Project**" butonuna basarak yeni bir proje oluşturulur veya daha önce var olan bir proje kullanılır. Bu projeye bir adın verildiği bir sayfa belirir, ardından "**Continue**" butonuna basılır, bundan sonra, bu projede '**Google Analytics**'i etkinleştirmek için yeni bir sayfa belirir, ardından '**Continue**' butonuna basılmalıdır, sonra, Firebase Analytics'in yapılandırılması ve ayarlanmasıyla ilgili bir sayfa belirir, bu sayfada '**Default Account for Firebase**' seçilir ve ardından bu projenin hazır hale gelmesi için '**Create project**' butonuna basılmalıdır. Proje kurulumu tamamlandıktan sonra görünen sayfada '**Continue**' butonuna basıldığında, bu projenin ana kontrol paneli görüntülenmektedir.

2. Firebase projesini web uygulamasına ekleme:

Projenin ana sayfasında, web'e özgü (</>) sembolünü taşıyan düğme seçilmektedir. Bu çalışma bir web uygulaması oluşturmayı hedeflediğinden, Firebase projesinin web uygulamasıyla bağlanması gerekmektedir. Sonra, web uygulamasına bir adın verildiği bir sayfa görüntülenir (herhangi bir geçerli ad, hatta Firebase projesinin adıyla aynı olsa bile), ardından '**Register app**' düğmesine basılmalıdır. Sonra, projede Firebase SDK paketlerini yüklemek ve yönetmek için '**use npm**' seçilir, ardından "continue to the console" düğmesine basılır. Böylece, Firebase, web uygulamasına eklenmiş olur.

3. Kimlik doğrulama dosyasını (Service Account) kullanarak kimlik doğrulama:

Firebase projesi web uygulamasına eklendikten sonra, ana sayfada kimlik doğrulama ve kullanıcı yönetimi konusunda sorumlu olan '**Authentication**' ürünü seçilmektedir. Daha sonra, Proje Ayarları (Project Settings) bölümünde, sayfanın altındaki Genel (General) bölümünde, (config, CDN, npm) içeren SDK ayar ve yapılandırma bölümünde, bu uygulamaya özgü anahtarları ve özel anahtarları içeren Firebase yapılandırma nesnesini içeren (config) seçeneği belirlenir ('apiKey', 'authDomain', 'projectId', 'storageBucket', 'messagingSenderId', 'appId', 'measurementId'). Bunlar kopyalanır ve değişken parantezleri içindeki koda eklenir: config = {}.

Proje ayarları sayfasının bölümünde, '**Service accounts**' seçeneği seçilir, bu, bir dizi programlama diline (Node.js, Java, Python, Go) yönelik SDK'nın yapılandırmasını içerir. Bu çalışmada, Python programlama dili seçilmiştir çünkü bu dili kullanmaktadır. Daha sonra, (Generate new Private Key) düğmesine basılmalı ve ardından (Generate Key) seçilmelidir. Bu işlem, (JSON) formatında, programın proje veri tabanıyla bağlantısını sağlayan bilgileri içeren bir dosya oluşturulur. Bu dosya, projenin dosyalarına eklenir ve programlama kodunda, config = {} değişkeni parantezleri içinde, bu dosya ('serviceAccount') aracılığıyla çağrılır.

Firebase projesinin ana sayfasında, '**Build**' bölümünde '**Realtime Database**' seçilir. bu sayfada, '**Create Database**' düğmesine basılmaktadır, ardından '**Next**' düğmesine ve daha sonra '**Start in Test mode**' seçeneği belirlenmektedir ve '**Enable**' düğmesine basılmaktadır. Daha sonra, bu uygulamaya özel bir URL içeren bir sayfa görünmektedir. Bu URL kopyalanmaktadır ve programlama koduna, config = {} değişkeni parantezleri içinde, ('databaseURL') aracılığıyla eklenmektedir.

Uygulama bağlantısı, programlama koduna eklenmektedir. Daha sonra, (Firebase) projesinin ana sayfasından '**Build**' ve ardından '**Storage**' seçilmektedir, bu da veri tabanında depolamanın etkinleştirilmesi için (Firebase Storage) yapılır. '**Storage**' seçildiğinde açılan sayfada '**Get started**' düğmesine basılmaktadır, ardından '**start in production mode**' seçilmekte ve '**Next**' düğmesine ve sonra '**Done**' düğmesine basılmaktadır. Bu şekilde veri tabanında depolama etkinleştirilmektedir.

Bu adımlardan sonra, (Firebase Storage) Python projesi ile bağlanmaktadır. Dosyaları yükleme, indirme ve yönetme gibi işlemleri gerçekleştirmek, sunulan Uygulama

Programlama Arayüzü (API) üzerinden kolaydır. Aşağıda (Şekil 3.7) gösterildiği üzere, Firebase yapılandırması görülmektedir:

```
import pyrebase
from firebase_admin import credentials, storage, initialize_app
config = {
    'apiKey': "********************",
    'authDomain': "***************",
    'projectId': "****************",
    'storageBucket': "************",
    'messagingSenderId': "********",
    'appId': "*********************",
    'measurementId': "************",
    'serviceAccount': "************",
    'databaseURL': "**************"
}
```

Şekil 3.7. Firebase yapılandırması

3.2.3.2. Dosyaları (firebase storage)'a yükleme

Bu işlem iki aşamaya bölünebilir:

Dosyaların yükleme işleminin yapılandırması

Proje Firebase ile bağlantıldıktan sonra, kodda yapılandırması yapılmalıdır. Yapılandırma aşağıdaki adımlara göre gerçekleştirilir:

1. 'initialize_app' fonksiyonu, uygulamayı Firebase hizmetlerine bağlanmak ve onlarla etkileşimde bulunmak üzere yapılandırmak için kullanılmaktadır. Kütüphanenin 'pyrebase' bir parametre olan 'config' değişkenini alan bir fonksiyonudur. Bu parametre, (Firebase) uygulamasıyla bağlantı kurmak için gerekli yapılandırma bilgilerini içerir, örneğin (API) anahtarı, proje kimliği ve daha önce eklenmiş olan (Firebase) projesiyle bağlantı kurmak için gerekli diğer bilgiler. 'initialize_app(config)' çağrıldığında, uygulama, sağlanan yapılandırma bilgilerine göre yapılandırılır.

firebase = pyrebase. initialize_app(config)

2. Ardından, 'credentials.Certificate' aracılığıyla 'Certificate' sınıfından bir nesne oluşturulur. Bu, 'firebase_admin' kütüphanesinin bir parçasıdır ve bir 'Certificate' kimlik doğrulama belgesi dosyasını, bu sınıfın bir parametresi olarak geçirilen ve daha önce proje dosyalarına eklenmiş olan bir (JSON) dosyasından yüklemek için

kullanılır. Bu, (Firebase) projesi hizmetlerine bağlanmak için kullanılan özel Hizmet Anahtarını (Service Key) içerir.

```
cred = credentials.Certificate("Servers//firebase_storage_server.json")
```

3. Ardından, 'initialize_app' fonksiyonu çağrılır ve bu, 'firebase_admin' kütüphanesinin bir parçası olduğu için önceki fonksiyondan farklıdır ve 'pyrebase' kütüphanesinde bulunan aynı fonksiyon değildir. Genellikle iki parametre alır: İlki, önceki adımda (JSON) dosyasından yüklenen ve 'cred' değişkeni olan Kimlik Doğrulama 'Credentials' bilgilerini temsil eder. İkincisi, (**Options**) uygulamanın yapılandırılması için ek seçenekleri belirlemek için kullanılır. Mevcut bağlamda, bunlar, uygulamanın ilişkilendirildiği depolama kovası (**Bucket**) adını geçirmek için kullanılır ve bu da "sosyal-medya-lokasyon-alalizi.appspot.com" olarak belirlenmiştir.

```
initialize_app (cred, {"storageBucket": "sosyal-medya-lokasyon-
alalizi.appspot.com"})
```

Klasör dosyalarını yüklenme işleminin başlatılması

Yapılandırma işlemi tamamlandıktan sonra dosya klasörünün yüklenme işlemi başlar. Çünkü bu çalışma, hedef kullanıcının hesabından kazılan dosyaları yüklemek üzerine çalıştığından ve bu dosyaların hedef kullanıcının adını taşıyan bir klasör olduğundan ve iki tür dosya içerdiğinden (JSON, JPG), klasör ve içindeki dosyalar aynı isimle yüklenir ve kalır. Bu şekilde, her kullanıcının dosyaları kendi klasör adı altında belirginleşir, karıştırılmaz ve her biri ayrı ayrı ele alınır, bu da onlara erişimi ve yönetimini kolaylaştırır.

Bu işlem, birkaç adımda gerçekleştirilir:

1. Kaynak ve hedef klasörün belirlenmesi: Kaynak klasörün yolunu belirlemek için yüklenecek dosyaları içeren klasörün konumu belirlenir. Bu klasör, uygulamanın bulunduğu yerde olmalıdır ve değişkene (**source_folder_path**) atanır. Ayrıca, dosyaların depolanacağı (Firebase Storage) içindeki hedef klasörün yolunu da belirlemek gerekmektedir. Depolamada kullanıcı adıyla aynı klasör adı kullanılarak özel bir klasör oluşturulur ve bu da değişkene (**destination_folder_path**) atanır.

2. Dosya Listesinin Alınması: Kaynak klasör olan (**source_folder_path**) içindeki tüm dosya ve klasör isimlerinin bir listesini almak için 'os.listdir' fonksiyonu

kullanılır, ardından fonksiyon 'os.path.join', bu klasörün içindeki her dosya için tam bir yol oluşturur, yani klasör adını ve ardından dosya adını, ardından 'os.path.isfile' fonksiyonu, 'os.path.join' tarafından oluşturulan yolun bir dosyayı mı yoksa başka bir şeyi mi gösterdiğini kontrol eder. Eğer öyleyse, 'True' döndürür. Eğer değilse (yani yol bir klasöre işaret ediyor veya yol mevcut değilse), 'False' döndürür. Böylelikle bu liste filtrelenir ve kaynak klasörde bulunan tüm dosyaların bir listesi elde edilir.

3. Firebase Storage'a bağlanın: Firebase Storage hizmetine bağlantı, 'storage.bucket' kullanılarak depolama kapsayıcısı (bucket) için bir nesne oluşturularak yapılandırılmaktadır.

4. Dosyaların Yüklenmesi: Kaynak klasördeki her dosya, 'for' döngüsü kullanılarak (Firebase Storage) 'a yüklenir. Adım 2'de elde edilen her dosya adı üzerinde döngü tekrarlanır. Her tekrarda, mevcut (**file_path**) dosyasının tam yolu ve (Firebase Storage) (**destination_path**) içindeki hedef yolu belirlenir. Ardından, 'blob = storage.bucket().blob(destination_path)' kullanılarak depolama kovasında bir 'blob' nesnesi oluşturulur ve dosya 'blob.upload_from_filename(file_path)' kullanılarak bu nesneye yüklenir. Bu işlem, tüm dosyalar yüklenene kadar devam eder.

Böylelikle, tüm dosyalar (Firebase Storage) 'a yüklenir ve dosyaların yüklendiği klasörün aynı adını taşır, yani hedef kullanıcının adıyla aynıdır. Aşağıda (Şekil 3.8) gösterildiği üzere, kodun Firebase Storage'a dosya yüklemesi görülmektedir:

```
source_folder_path = f"./{target_username_data}"
destination_folder_path = target_username_data
files = [f for f in os.listdir(source_folder_path)
         if os.path.isfile(os.path.join(source_folder_path, f))]
bucket = storage.bucket()
for file_name in files:
    file_path = os.path.join(source_folder_path, file_name)
    destination_path = f"{destination_folder_path}/{file_name}"
    blob = storage.bucket().blob(destination_path)
    blob.upload_from_filename(file_path)
```

Şekil 3.8. Kod firebase storageye dosyalarını yüklenmesi

Bu yaklaşım, verilerin bulut altyapısına etkili bir şekilde taşınmasını ve güvenli ve düzenli bir şekilde depolanmasını sağlar. Bu, veri yönetimini ve web ve mobil uygulamalarda veriyle etkili bir şekilde çalışmayı kolaylaştırır.

3.2.4. Lokasyon verilerinin analizi

Klasör ve içerdiği dosyalar veri tabanına (Firebase Storage) yüklendikten sonra, lokasyon verilerinin analiz işlemi başlar. Bu çalışma, analiz sürecinde üç yönteme dayanmaktadır ve bunlar ardışık yöntemlerdir, yani birinci yöntemden sonuç alınamadığında ikinci yönteme geçilir, ikinci yöntemden sonuç alınamadığında üçüncü yönteme geçilir ve üçüncü yöntemden sonuç alınamadığında bu, lokasyona ait herhangi bir bilginin bulunmadığı anlamına gelir.

Analiz işlemine başlamadan önce, daha önce yüklenmiş dosyaları çağırmak için veri tabanı (Firebase Storage) ile bağlantı kurulmalıdır. Dosyaların geri çağrılması işlemi, dosyaların yüklenmesi işleminde daha önce kullanılan aynı hizmet hesabını (Service Account) kullanır.

Dosyaları çağırmak için, (Firebase Storage) içinde saklanan klasörün depolama kabı (**bucket**) yolunun belirlenmesi gerekmektedir. Bu, daha önce (**destination_folder_path**) değişkeninde saklanan bu klasör yolunun yeni bir değişken olan (**folder_path**) değişkenine atanmasıyla gerçekleştirilir. Ardından, belirtilen klasörün içindeki dosyaların bir listesini geri almak için, 'list_blobs' fonksiyonu kullanılır. Bu fonksiyon, (GCS) kütüphanesindeki (**bucket**) nesnesine aittir ve depolama kabındaki tüm (**blobs**) ile bir liste döndürür. Bu fonksiyon, 'prefix' aracılığıyla kendi parametresi içinde belirtilen yolu alır, bu da (**bucket**) içinde aranacak yolu belirlemek için kullanılır.

folder_path = destination_folder_path

blobs = bucket . list_blobs (prefix = folder_path)

3.2.4.1. JSON dosyalarından lokasyon verilerinin çıkarılması

Bu çalışmada kullanılan ilk yöntemdir, en kolay ve hızlı yöntem olarak kabul edilmektedir. Bu yöntem, gönderinin sahibi tarafından paylaşılan lokasyon bilgilerine dayanmaktadır. (Instagram) kullanıcısı kişisel hesabında bir gönderi paylaştığında, aynı zamanda bu gönderinin konumunu da ekleyebilir, bu özellik (yer etiketi) olarak

bilinir. (Instagram)'ın sunduğu bu özellik, paylaşılan konumun enlem ve boylam bilgilerini içerir.

(Instaloader) kütüphanesi her bir gönderinin meta verilerini (Metadata) kazıdığında, enlem ve boylam bilgileri (yer etiketi) kazınan veriler arasında olunmaktadır.

Gönderi enlem ve boylam bilgilerini içeriyorsa, bu bilgiler çıkarılır ve ikinci veya üçüncü yönteme gerek kalmaz. Bu bilgilerin çıkarılabilmesi için, öncelikle daha önce yüklenmiş olan (Firebase Storage) veri tabanındaki (JSON) dosyaları çağrılmaktadır.

(Firebase Storage) veri tabanından yalnızca (JSON) dosyalarının çağrılması işlemi, 'for' döngüsü kullanılarak (**blobs**) içindeki tüm dosyaların üzerinden geçilerek gerçekleştirilir. İlk olarak, genellikle dosyanın klasör içindeki yolunu belirten 'blob.name' fonksiyonu kullanılarak dosya isimlerinin bir listesi çıkarılır. Klasör adı (**folder_path**), klasör adından sonra gelen '/' karakteri ile birlikte silinir, yani sadece klasör içindeki dosya adı kalır.

Daha sonra, 'endswith' fonksiyonu kullanılır ve parametresine '.json' uzantısı verilir, böylece sadece (JSON) türünde olan dosya isimlerini çıkarır. Ardından, dosya türü (JSON) olduğunda uygulanması gereken komutlar eklenmektedir.

(JSON) türündeki ilk dosyaya ulaşıldığında, bu dosyanın içeriği 'download_as_text' fonksiyonu kullanılarak metin olarak indirilir ve işlemeyi kolaylaştırmak için özel bir değişkene (**file_content**) atanmaktadır. Ardından, (JSON) dosyasından indirilen metin içeriği, 'json.loads' fonksiyonu kullanılarak analiz edilir. Bu fonksiyon, metni Python'da işlenebilen bir JSON nesnesine dönüştürmektedir.

JSON nesnesi, birbirine bağlı birçok nesneyi içerir, bu da gönderinin (Post Metadata) meta verilerinden kazınan verilere göre belirlenir. Ancak bu çalışma, yalnızca 'lat' ve 'lng' anahtarlarıyla odaklanmaktadır, bunlar enlem ve boylam bilgilerini içeren anahtarlardır ve JSON nesnesindeki nesnelerin belirtilen sırasında bulunur: 'data', sonra 'node', sonra 'iphone_struct', ve en son 'lat' ve 'lng' anahtarları bulunmaktadır. JSON nesnesi içindeki bu anahtarlara erişmek için 'get' fonksiyonu kullanılmaktadır. Aşağıda (Şekil 3.9) gösterildiği üzere, kodun JSON dosyalarından lokasyon verilerinin çıkarılması görülmektedir.

```
for blob in blobs:
    file_name = blob.name[len(folder_path) + 1:]
    if file_name.endswith('.json'):
        file_content = blob.download_as_text(encoding='latin-1')
        data = json.loads(file_content)
        lat = data.get(node, {}).get('iphone_struct', {}).get('lat')
        lng = data.get(node, {}).get('iphone_struct', {}).get('lng')
```

Şekil 3.9. JSON dosyalarından lokasyon verilerinin çıkarılması

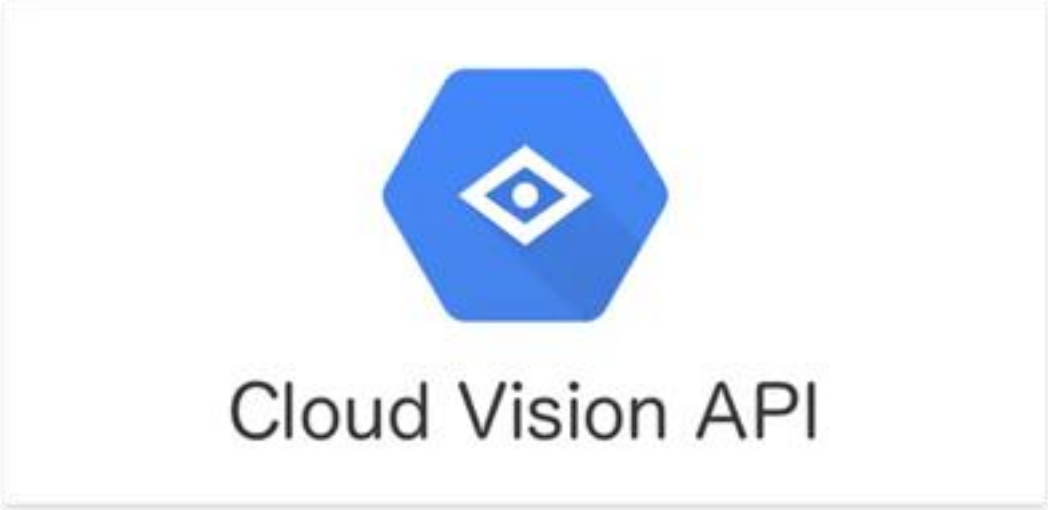
'lat' ve 'lng' anahtarları için bilgi bulunduğunda, bunlar her anahtar için özel değişkenlere kaydedilir. Daha sonra, ikinci veya üçüncü analiz yöntemini kullanmaya gerek kalmadan bir sonraki gönderi için JSON dosyasına geçilmektedir. Eğer bu gönderinin meta verilerinde (Metadata) 'lat' ve 'lng' anahtarlarına ait herhangi bir bilgi bulunamazsa, bir sonraki gönderiye geçmeden önce analizin ikinci yöntemine geçilmektedir.

3.2.4.2. "Google cloud vision" aracını kullanarak lokasyon verilerinin çıkarılması

Bu çalışmanın konum verilerinin analizinde dayandığı ikinci yöntemdir. Eğer yayına ait tanımlayıcı veri dosyası (JSON) konumla ilgili herhangi bir veri (Enlem, Boylam) içermiyorsa, bu yönteme kullanılmaktadır.

"Google cloud vision" aracı

Görüntüleri ve görsel medyayı analiz etmek ve anlamak için bir Uygulama Programlama Arayüzü (API) sağlayan güçlü bir araçtır. Google Cloud Platform tarafından geliştirilmiştir ve geliştiricilere uygulamalarına kolayca görüş algılama özelliklerini entegre etme olanağı sağlar. Görüntülerin analiz edilmesi, işlenmesi ve bilgi çıkarılması için kullanılmaktadır. Aşağıdaki (Şekil 3.10.)'de Vision API simgesi görünmektedir:



Şekil 3.10. Vision API simgesi

Makine öğrenmesi ve gelişmiş yapay zekâ teknolojilerine dayanarak, fotoğrafları analiz etmek ve onlardan bilgi çıkarmak için çeşitli ve güçlü bir hizmetler ve araçlar seti sunar. (Google Developers, 2023). Bunlar arasında:

1. Nesne Yerelleştirme (Object Localization): Bu hizmet, fotoğraftaki farklı nesnelerin konumlarını belirler ve onları ayırt etmek için etrafına bir çerçeve çizer. Bu özellik, fotoğraflardaki nesneleri tanıma ve konumlarını doğru bir şekilde belirleme gibi uygulamalarda kullanılabilir, örneğin, fotoğraftaki hayvanların, insanların, otomobillerin ve mobilyaların konumunu belirleme.
2. Yüz Tanıma (Face Detection): Bu özellik, bir fotoğraftaki kişilerin yüzlerini tanımlar ve konumlarını ve özelliklerini, örneğin gözleri, burnu ve ağzı belirler. Bu özellik, yüz tanıma ve fotoğraftaki kişi sayısını belirleme gibi uygulamalarda kullanılabilir, ayrıca yüzün yönü (öne veya yana dönük), bir gülümseme veya gözlük olma olasılığı gibi diğer bilgileri de belirleyebilir.
3. Yer İşareti Algılama (Landmark Detection): Bu özellik, fotoğraflardaki ünlü anıtları, tarihi binaları, köprüleri, doğal anıtları ve ünlü müzeleri tespit eder ve fotoğraftaki anıtın adını döndürür ve onu ayırt etmek için etrafına bir çerçeve çizer. Bu özellik, turizm uygulamalarında ve fotoğraflardaki ünlü anıtları belirlemede kullanılabilir, ayrıca bu anıtın haritadaki enlem ve boylam bilgilerini de döndürür.
4. Logo Tespiti (Logo Detection): Bu özellik, bir fotoğraftaki şirket logolarını veya ticari markaları tespit etmek için kullanılır ve ticari markanın adına ve fotoğraftaki

konumuna dair bir metin açıklaması döndürür. Bu özellik, pazarlama analizi ve ticari marka kullanımının izlenmesi gibi uygulamalarda kullanılabilir.

5. Etiket Algılama (Label Detection): Bu özellik, bir fotoğraftaki çeşitli nesneleri belirler, sınıflandırır ve etiketler, örneğin arabalar, ağaçlar veya hayvanlar. Bu özellik, fotoğrafların sınıflandırılması ve düzenlenmesi gibi uygulamalarda kullanılabilir, ayrıca bu özellik, her etiket için bir metin açıklaması ve sınıflandırma işlemi için güven derecesi de sağlamaktadır.
6. Metin Algılama (Text Detection): Bu özellik, bir fotoğraftaki metinleri çıkarır ve düzenlenebilir bir metne dönüştürür. Bu özellik, OCR (Optik Karakter Tanıma) uygulamalarında ve fotoğraflardan metin çıkarmada kullanılabilir, ayrıca el yazısıyla yazılmış metinleri algılayabilir ve bu özellik birçok dilin desteklenmesini sağlamaktadır.
7. Belge Metni Algılama (Document Text Detection): Fotoğraflardan metin algılamaya benzerdir, ancak taranmış belge dosyaları, örneğin PDF ve TIFF veya büyük miktarda metin içeren fotoğraflar için optimize edilmiştir. Bu özellik, metni çıkarır ve düzenli ve hassas bir şekilde analiz ederken belge biçimini korumaktadır.
8. Güvenli Arama Tespiti (Safe Search Detection): Bu özellik, bir fotoğrafın içeriğinin güvenli arama standartlarına ne kadar uygun olduğunu belirlemek için kullanılır, bu da fotoğraftaki uygunsuz veya istenmeyen içeriğin filtrelenmesine olanak sağlamaktadır.
9. Fotoğraf Özellikleri (Image Properties): Bu özellik, bir fotoğrafın çeşitli özelliklerini, örneğin baskın renkler, aydınlatma, kontrast ve genel yön gibi, analiz eder, bu da fotoğrafın genel içeriğinin anlaşılmasına yardımcı olur.
10. Kırpma İpuçları (Crop Hints): Bu özellik, bir fotoğrafın ideal olarak kırpılabilecek bölgelerini önermek için ipuçları sunar, bu da kompozisyonunu iyileştirmeye veya içindeki önemli içeriğe odaklanmaya yardımcı olur.
11. Web Algılama (Web Detection): Bu özellik, verilen fotoğrafla ilişkilendirilmiş veya benzer fotoğrafları web'de bulmak için kullanılır, böylece fotoğraftaki içerik hakkında daha fazla bilgi sağlamaktadır.

12. Ürün Arama (Product Search): Bu özellik, internet üzerinde satışa sunulan ve fotoğraftaki ürünlere benzer ürünler hakkında bilgi aramak ve sağlamak için kullanılır.

Daha önce belirtilen özelliklere ek olarak, Vision API, çeşitli uygulamaların özel ihtiyaçlarına daha ayrıntılı ve özelleştirilmiş bir şekilde fotoğrafları analiz etme ve işleme konusunda yardımcı olabilecek bir dizi diğer hizmet ve özelliği sunabilir. Bu, farklı uygulamalar ve endüstrilerde geniş bir kullanım yelpazesinde faydalı bir araç haline getirir. Ancak, bu aracın sunduğu özelliklerin setinin, ürün ve hizmet gelişmelerine ve zaman zaman yeni işlevler sunmak veya performansı iyileştirmek için yayınlanan yeni güncellemelere bağlı olarak değişebileceğine dikkat etmek önemlidir. Bu nedenle, her zaman resmi hizmet belgelerini kontrol ederek mevcut özelliklerin güncellenmiş bir listesini almak daha iyidir.

Bu araç diğer araçlar gibi birçok özelliğe sahip olmasına rağmen, çalışırken bazı potansiyel kısıtlamalarla karşılaşabilir, bu da performansını etkiler. Bunlardan bazıları şunlardır:

1. Yüksek kaliteli eğitim verilerine ihtiyaç: Makine öğrenme araçlarının etkili bir şekilde çalışabilmesi için, eğitimde kullanılan verilerin doğru, çeşitli ve kapsamlı olması gerekir. Veriler yetersizse, sonuçlar doğru olmayabilir.
2. İnternete Bağımlılık: 'Google Cloud Vision API' aracı, fotoğrafları işlemek ve analiz etmek için internet bağlantısına bağımlıdır, bu da performansın internet bağlantısının kalitesi ve stabilitesi tarafından etkilenebileceği anlamına gelir.
3. Teknik Kısıtlamalar: Aracın teknik kısıtlamaları, fotoğraf boyutu, dosya türü ve çözünürlüğü içerir. İşlenebilecek fotoğraf boyutu için sınırlamalar olabilir ve araç, tüm dosya türlerini desteklemeyebilir, bu da analizden önce dosyaların uyumlu formatlara dönüştürülmesini gerektirebilir.
4. Maliyet: Maliyet, talep sayısına ve işlenen veri boyutuna bağlıdır. Büyük miktarda veri işleyen şirketler için maliyet önemli bir faktör olabilir, çünkü kullanım arttıkça maliyetler önemli ölçüde artabilir.
5. Sınırlamalar: 'Google Cloud Vision API', belirli bir zaman diliminde gönderilebilecek talep sayısına sınırlamalar getirir ve belirli özelliklerin kullanımı için sınırlamalar olabilir, bu da geliştiricilerin bu sınırlamaları aşmamak için aracın kullanımını izlemelerini gerektirir.

6. Metin Tanıma için Dil Desteği: Araç, tüm dilleri eşit olarak desteklemeyebilir ve bazı dillerde metin tanıma doğruluğu, İngilizceye kıyasla daha düşük olabilir, bu da geliştiricilerin doğruluğu artırmak için dil ipuçları sunmalarını gerektirir.

Bu kısıtlamalar, bir aracın uygulamalara entegre edilmeden önce kapsamlı bir şekilde değerlendirilmesinin önemini ve hem olanakları hem de onunla ilişkili zorlukları göz önünde bulundurma gerekliliğini vurgulamaktadır.

Google cloud vision api'yi python ile kullanma

Bu araç, fotoğrafların görsel analizine odaklanmaktadır, bu nedenle daha önce Instagram'dan kazılan fotoğrafları analiz etmek ve mevcut olduğunda fotoğraflardan enlem ve boylam bilgilerini çıkarmak için kullanılabilir. Bu çalışmayla uyumlu olan özellik, Yer İşareti Algılama (Landmark Detection) özelliğidir. (Google Developers, 2023). (NeuralNine , 2023). Bu özelliği bir Python projesinde kullanabilmek için aşağıdaki adımları izlemek gerekir:

1. Google Cloud Platform'da bir hesap oluşturma: Resmi siteye "https://console.cloud.google.com/" adresine gidilir, yeni bir hesap oluşturulur veya mevcut bir hesaba giriş yapılır. Daha sonra, "**NEW PROJECT**" 'e tıklanarak yeni bir proje oluşturulur veya mevcut bir proje kullanılır. Yeni bir proje oluşturmak için tıklandığında görünen yeni sayfada, bu projeye bir isim verilmesi gerekmektedir. "**Location**" adlı diyalog kutusu olduğu gibi bırakılır. Daha sonra, "**CREATE**" düğmesine tıklanır. Yeni proje oluşturulur ve projeyi seçmek için oluşturulan projeyi seçmek üzere yeni bir sayfa görünür.
2. Google Cloud Vision API'nin etkinleştirilmesi: Proje seçildikten sonra yeni bir sayfa görünmektedir. Bu sayfadaki "Search" bölümünde, arama kutusuna "**Vision API**" yazılır. Arama sonuçları göründüğünde, "**Cloud Vision API**" seçilir. "Cloud Vision API" aracına ait sayfa görünür ve buradan "**ENABLE**" düğmesine tıklanarak "**Cloud Vision API**" etkinleştirilir. Daha sonra, kimlik doğrulama bilgileri oluşturulmalıdır; çünkü bu API'yi kullanmak için, bu API'ye anahtar kullanılarak oturum açılması gerekmektedir. Bu nedenle, sayfanın köşesindeki "**CREATE CREDENTIALS**" düğmesine tıklanır ve kimlik türünü belirlemek için yeni bir sayfa görünür. Öncelikle API seçilir, bu "**Cloud Vision API**" olmalıdır, daha sonra erişilecek veriler belirlenir ve bu "**Application data**" olmalıdır. Genellikle "**Google Cloud API**" 'ye bir hizmet hesabı kullanılarak sunucudan erişilir, bu nedenle "**Application data**" 'nın seçilmesi hizmet

hesabının oluşturulmasını sağlar. Daha sonra, "**NEXT**" düğmesine tıklanarak devam edilir. Daha sonra, hizmet hesabını oluşturmakla sorumlu yeni bir sayfa görünür, burada "**Service account name**" diyalog kutusundan hizmet hesabına bir isim verilir, herhangi bir isim mümkün ve doğru olabilir. İkinci diyalog kutusu olan "**Service account ID**" için, hizmet hesabına bir isim verildiğinde otomatik olarak bir değer verilir. Hizmet hesabının açıklaması olan "**Service account description**" için, herhangi bir şey eklenmeden boş bırakılabilir. Daha sonra, "**CREATE AND CONTINUE**" düğmesine tıklanır. Daha sonra, geri kalanı olduğu gibi bırakılır, yani varsayılan değerlerle ve sonra "**DONE**" düğmesine tıklanır, bu da hizmet hesabının oluşturulmasını sağlar.

3. Kimlik Doğrulama Bilgileri Dosyasının (API Anahtarı) Oluşturulması: Hizmet hesabı oluşturulduktan sonra görünen sayfada, sayfanın sol tarafındaki menüye gidilir ve ardından "**Credentials**" seçeneğine geçilir. "**Credentials**" seçildiğinde görünen sayfanın altında, "**Email**" bölümünde, anahtar oluşturmak için bu bölümdeki bağlantıya tıklanır. Bağlantıya tıkladığında görünen sayfada, "**KEYS**" bölümüne gidilir, ardından "**ADD KEY**" açılır menüsüne tıklanır, ardından "**Create new key**" seçilir, ardından indirilip kullanılması istenen anahtar formatı olan JSON seçilir, ardından "**CREATE**" düğmesine tıklanarak JSON türünde bir dosya oluşturulur ve indirilir. Bu dosya, oluşturulan projenin kimlik doğrulama anahtarını içerir ve "**Google Vision API**" 'ye oturum açmak için kullanılır. Daha sonra, bu dosyanın proje dosyalarına eklenmesi gerekmektedir.

Google cloud vision api'yi kodda yapılandırma

(Google Cloud Vision API) ile ilgili ayarların tamamlanmasının ve kimlik doğrulama dosyasının (JSON) proje dosyalarına eklenmesinin ardından, kodda (Google Cloud Vision) aracının kullanımını yapılandırma süreci başlar. (NeuralNine , 2023). Bu, aşağıdaki adımlara göre gerçekleştirilir:

1. (Google Cloud Vision) Kütüphanesi: (Google Cloud Client Libraries)'den biridir ve (Google Cloud Vision) hizmetini kullanmak için bir uygulama programlama arayüzü (API) sağlar. Bu kütüphane, Python dilinde, yüzleri, yerleri, metinleri ve fotoğraflardaki diğer öğeleri algılama gibi yapay zekâ tekniklerini kullanarak fotoğrafları analiz etme ve bilgileri çıkarma yeteneği sağlayan (Google Cloud Vision API) hizmetiyle etkileşim için kullanılır.

'google-cloud-vision' kütüphanesi, Python programlama kodundan (Google Cloud Vision API) ile etkileşime geçmek için basit ve kullanımı kolay bir arayüz sağlamaktadır. Bu kütüphane, fotoğrafları analiz etme ve onlardan veri çıkarma sürecini kolaylaştıran bir dizi fonksiyon ve araç içermektedir. Bu, gelişmiş fotoğraf işleme ve tanıma teknolojilerini kullanan uygulamaların geliştirilmesini kolaylaştırmaktadır.

Kütüphane, 'pip install google-cloud-vision' komutunu yazarak Komut İstemi (CMD) üzerinden kurulmaktadır ve bu kütüphane ile ilgili her şey programlama projesine kurulacaktır.

Kurulum tamamlandıktan sonra, bu kütüphaneyi programlama kodunda 'from google.cloud import Vision' kullanarak çağırmak gerekmektedir.

2. Çevresel değişkeni yapılandırma: 'google-cloud-vision' kütüphanesinin programlama kodunda yapılandırılmasının tamamlanmasının ardından, 'GOOGLE_APPLICATION_CREDENTIALS' çevresel değişkeninin değerini, önceden projenin dosyalarına eklenmiş olan (Google Cloud Vision)'a ait JSON kimlik doğrulama dosyasını içeren yola ayarlanmalıdır. Google Cloud, bu değişkeni projenin kimlik doğrulama sertifikası dosyasını bulmak ve (Google Cloud) hizmetlerine bağlanırken kullanıcının kimliğini doğrulamak için kullanmaktadır. Bu bağlamda, çevresel değişkeninin (Environment variables) değerini ayarlamak için 'os.environ' kullanılmaktadır.

'os.environ': Python'daki 'os' kütüphanesinin bir parçasıdır ve işletim sistemi ve yerel ortamla etkileşim için bir arayüz sağlar. 'os.environ', işletim sistemindeki çevre değişkenlerini temsil eder. Bu değişkenler, kullanıcının belirlediği çevre değişkenleri de dahil olmak üzere, mevcut Python işleminin ortamına erişim sağlar. Python programı başlatıldığında, bu program için yerel bir ortam oluşturulur. Bu ortam, programın davranışını etkileyebilecek yolları, değişkenleri ve diğer ayarları içerir.

'os.environ' içindeki bir değişkene değer atanırken, bu, mevcut program ortamının değiştirildiği anlamına gelmektedir.

os.environ['GOOGLE_APPLICATION_CREDENTIALS'] =
'Servers//google_codelabs_cloud_vision_api_server.json'

Bu bağlamda, 'GOOGLE_APPLICATION_CREDENTIALS' değişkeninin değeri, 'os.environ' içinde, kimlik doğrulama anahtar dosyasının yoluna ayarlanmıştır. Bu, bu değişkeni kullanan herhangi bir kodun, bu yolu bulup (Google Cloud) hizmetine bağlanma konfigürasyonunun bir parçası olarak kullanabileceği anlamına gelmektedir. Bu değer bir kez ayarlandığında, 'google-cloud-vision' kütüphanesi veya projede kullanılan araç, onu bulabilir ve (Google Cloud) hizmetine bağlanırken kimlik doğrulama için kullanabilir.

3. Yer işareti keşfi: Çevresel değişkeninin yapılandırmasının ardından, 'google_cloud_vision' fonksiyonu oluşturulur, bu fonksiyon, fotoğrafta tespit edilen yer işaretinin enlem ve boylam bilgilerini çıkarmaktan sorumludur, bu fonksiyonun oluşturulmasında aşağıdaki adımlar izlenir:

- (Google Cloud Vision) Hizmeti İçin Bir İstemci Oluşturma: Bu, (Google Cloud Vision API) hizmeti ile bağlantı kurmanın ilk ve temel adımıdır ve fotoğrafların analiz sürecinin ve bu hizmetin sunduğu hizmetlerin kullanılmasının başlangıcını temsil eder. (Google Cloud Vision API) hizmeti için bir istemci, 'ImageAnnotatorClient' sınıfı kullanılarak oluşturulur. Bu istemci, hizmetle etkileşime girer ve programın (API) ile iletişim kurmasına ve analiz için fotoğrafların gönderilmesine izin verir. 'vision' ise, (Google Cloud Vision API) ile etkileşim için sınıflar ve fonksiyonlar içeren modüldür. 'ImageAnnotatorClient' çağrıldığında, fotoğraflarla çalışmak ve onları (Google Cloud Vision) hizmeti kullanarak analiz etmek için hazır bir istemci oluşturulur.

```
client = vision.ImageAnnotatorClient()
```

Bu istemci, uygulama ile (Google Cloud Vision API) hizmeti arasındaki arayüzdür, görüntüleri hizmete gönderir, analiz eder ve sonuç olarak gelen yanıtı işler.

- Fotoğrafın okunması: Bu adımda, analiz edilecek fotoğrafı içeren dosya açılır ve içeriği ikili veri (binary data) olarak okunur. Bu, fotoğrafın etkili ve hassas bir şekilde anlaşılmasını ve işlenmesini kolaylaştırır ve bu, fotoğrafların analiz edilmesine ve bilgilerin onlardan daha verimli bir şekilde çıkarılmasına katkıda bulunur. Daha sonra, okunan içerik kullanılarak bir 'Image' nesnesi oluşturulur, bu nesne, fotoğrafın (Google Cloud Vision API) tarafından analiz için hazırlanmasını temsil eder. 'Image' nesnesi, fotoğrafın içeriğini 'vision.Image()' parametresi olarak

kullanarak oluşturulur ve bu nesne, fotoğrafı (Google Cloud Vision API) önünde temsil etmek için kullanılır.

```
with open (file_path, "rb") as image_file:
    content = image_file.read()
    image = vision.Image (content = content)
```

- Yer İşareti Algılama (Landmark Detection) Hizmeti: 'landmark_detection' fonksiyonu kullanılmaktadır. (Google Cloud Vision API) içerisinde mevcut olan işlevlerden biri olan ve kendisine sağlanan fotoğrafı analiz ederek içindeki yer işaretlerini (bilinen yerleri) tespit eder. 'Image' nesnesi, önceden oluşturulan istemci (client) aracılığıyla bu fonksiyona parametre olarak geçirilir.

```
response = client.landmark_detection (image = image)
```

(Google Cloud Vision) hizmeti, fotoğrafı analiz etmeyi ve özellikleri algılamayı tamamladıktan sonra, sonuçları (**response**) değişkenine döndürür. Bu değişken, fotoğrafta algılanan özellikler hakkında bilgi içerir, örneğin özelliklerin adları, coğrafi konumları ve diğer detaylar.

- Sonucun İşlenmesi: (Google Cloud Vision API) hizmeti, fotoğraftaki belirgin yer işaretlerini tespit ettikten sonra, bu bilgiler '**landmarks**' değişkeninde saklanmaktadır. Eğer fotoğrafta tespit edilen yer işaretleri varsa, 'landmarks' yer işareti listesine erişerek ilk yer işaretini ve coğrafi konumunu çıkarır. İlk yer işareti, 'landmarks[0]' kullanılarak 'landmark' listesine atanır. Yer işaretini elde ettikten sonra, 'locations' fonksiyonuna erişerek yer işaretinin coğrafi konumunu (Enlem, Boylam) çıkarır. Bu fonksiyon, yer işaretinin konumunu temsil eder. Daha sonra, bu konumun coğrafi koordinatları çıkarılır.

```
landmarks = response.landmark_annotations
    if landmarks:
        landmark = landmarks[0]
        locations = landmark.locations
        lat_lng = locations[0].lat_lng
```

```
lat = lat_lng.latitude

lng = lat_lng.longitude

return lat, lng
```

'lat_lng.latitude' konumun Enlem (Latitude) elde etmek için kullanılmaktadır, 'lat_lng.longitude' ise konumun Boylam (Longitude) elde etmek için kullanılmaktadır. Konumun coğrafi koordinatları çıkarıldıktan sonra, bunlar fonksiyonun sonucu olarak döndürülmektedir. Aşağıda (Şekil 3.11) gösterildiği gibi, tamamlanmış google_cloud_vision fonksiyonu görülmektedir:

```python
def google_cloud_vision(file_path):
    client = vision.ImageAnnotatorClient()
    with open(file_path, "rb") as image_file:
        content = image_file.read()
        image = vision.Image(content=content)
        response = client.landmark_detection(image=image)
        landmarks = response.landmark_annotations
        if landmarks:
            landmark = landmarks[0]
            locations = landmark.locations
            lat_lng = locations[0].lat_lng
            lat = lat_lng.latitude
            lng = lat_lng.longitude
            return lat, lng
        elif response.error.message:
            raise Exception(
                "{}\nFor more info on error messages, check: "
                "https://cloud.google.com/apis/design/errors".format(response.error.message)
            )
```

Şekil 3.11. google_cloud_vision fonksiyonu

İkinci yöntem kullanılarak fotoğraflardan (JPG) lokasyon verilerinin çıkarılması:

Bu çalışmadaki ilk yöntem, kullanıcı tarafından paylaşılan durumda konum bilgilerine (yer etiketi) dayanmaktadır, bu bilgiler çıkarılır ve Metadata bilgilerini içeren bir JSON dosyasına kaydedilir. Ancak, paylaşılmaması durumunda, aynı gönderiye eklenen fotoğrafın görsel analizi yoluyla konum bilgisinin çıkarılması için ikinci yönteme (Google Cloud Vision) kullanılmaktadır.

Meta veriler (Metadata) kazınarak elde edildiğinde, bu verileri içeren JSON dosyası, ait olduğu gönderinin yayınlanma tarihi ile aynı ada sahip olmaktadır. Ayrıca, aynı gönderiye ait olan fotoğraflar de aynı ada sahip taşımaktadır. Yani, aynı gönderiye ait olan fotoğraf dosyaları ve JSON dosyaları, sadece uzantıları farklı olacak şekilde aynı

ada taşımaktadır. Her bir gönderinin bilgileri ortak bir ad altında ayrılabilmesi için yapılmaktadır.

Meta verilerin (Metadata) herhangi bir konum bilgisi (yer etiketi) içermediğinden emin olunduğunda, gönderinin JSON dosyasıyla aynı ada sahip ancak (JPG) uzantılı olan fotoğraf çağrılır. Bu fotoğraf, önceden oluşturulmuş olan (google_cloud_vision) fonksiyonuna geçirilir ve bu fotoğrafta belirgin yer işaretleri tespit edildiğinde enlem ve boylam bilgilerini çıkarmak için analiz edilmektedir.

Bazen Instagram kullanıcıları aynı gönderi birden fazla fotoğraf paylaşır, bu yüzden bu fotoğraflardan biri ünlü bir yer işareti içerebilir, ancak tüm fotoğraflar belirgin bir yer işareti içermez. Bu nedenle, bu çalışma, aynı gönderi ait birden fazla fotoğraf olduğunda ve bunlar (Instaloader) kütüphanesi tarafından aynı gönderi ait tüm fotoğraflara sayısal bir sıralama eklenerek ayırt edildiğinde, aynı JSON dosyasının adını sahip ve sayısal bir sıralamaya sahip olan tüm fotoğrafları çağırır ve bunları (google_cloud_vision) fonksiyonuna geçirir ve bu fotoğraflarda tespit edilen yer işaretlerinin enlem ve boylam bilgilerini çıkarmak için analiz etmektedir. Aşağıda (Şekil 3.12) gösterildiği gibi, kodun fotoğraflardan lokasyon verilerini çıkardığı görülmektedir:

```
if lat is None and lng is None:
    jpg_file_name = file_name.replace('.json', '.jpg')
    similar_images = [f for f in files if f.startswith(jpg_file_name[:-5]) and f.endswith('.jpg')]
    if similar_images:
        for similar_image in similar_images:
            lat, lng = google_cloud_vision(os.path.join(source_folder_path, similar_image))
```

Şekil 3.12. Fotoğraflardan lokasyon verilerini çıkarma

Enlem ve boylam bilgileri bulunduğunda, önceden tanımlanmış olan 'lat' ve 'lng' değişkenlerine kaydedilir. Program çalışırken bulunan tüm enlem ve boylam bilgileri saklanır. Ardından, üçüncü analiz yöntemini kullanmaya gerek kalmadan bir sonraki gönderiye geçilir. Ancak, bu gönderide herhangi bir enlem ve boylam bilgisi bulunamazsa, bir sonraki gönderiye geçmeden önce üçüncü analiz yöntemine geçilecektir.

3.2.4.3. (OpenCV, TensorFlow) kütüphanelerine dayanarak lokasyon verilerinin çıkarılması

Bu çalışmanın lokasyon verilerini analiz etmek için dayandığı üçüncü yöntemdir. Eğer gönderinin meta veriler (Metadata) dosyası (JSON) herhangi bir lokasyon bilgisi içermiyorsa ve ayrıca (Google Cloud Vision) aracının aynı gönderine ait olan fotoğraf dosyalarından enlem ve boylam bilgilerini tespit etme yeteneği yoksa, bu yönteme kullanılmaktadır.

Bu yöntem, enlem ve boylam bilgilerini çıkarmak için fotoğrafların görsel analizine dayanmaktadır, yani ikinci yönteme benzer. Bu yöntemde, ikinci yöntemde olduğu gibi, bilgi çıkarımında önceden eğitilmiş bir makine öğrenme modeline de dayanılmaktadır. Her yöntemde kullanılan modellerin farklılığından dolayı, bu çalışmada üçüncü bir yöntem olarak kullanılmıştır. Bu yöntem, fotoğrafta görünen ünlü yer işaretlerini içeren modelleri içerebilirken, ikinci yöntem bunu içermeyebilir. Yöntemlerin sıralaması söz konusu olduğunda, ikinci yöntem (Google Cloud Vision aracı) daha doğru olan modellere sahip olduğu için üçüncü yöntemden (OpenCV ve TensorFlow kütüphaneleri) önce yerleştirilmiştir.

Bu yöntem, çalışma mekanizmasında Python'daki hazır kütüphane ve fonksiyonlar setine dayanmaktadır. İlk olarak, (OpenCV) kütüphanesi, fotoğrafları okur ve (TensorFlow) ve (TensorFlow Hub) kütüphanesinde kullanılan modele uygun bir formata dönüştürür. Bu kütüphaneler, fotoğraflardaki ünlü özellikleri belirlemek için kullanılır. Son olarak, (Geopy) kütüphanesi, ünlü yer işaretleriyle ilgili coğrafi bilgileri (enlem ve boylam) elde etmek için kullanılır.

Bu yöntem, fotoğrafların analiz edilmesinde ve varsa bunlardan coğrafi bilgilerin (boylam ve enlem) çıkarılmasında 3 adımı izlemektedir:

1. Fotoğrafların yapılandırması: Bu adımda görüntüler, fotoğraflardaki ünlü yer işaretlerini tahmin etme ve coğrafi bilgileri (enlem ve boylam) çıkarma sürecinde kullanılan eğitim modelinde işlenmeye hazır olacak şekilde yapılandırılır. Fotoğrafları biçimlendirme sürecinde (OpenCV) kütüphanesi kullanılmaktadır.

(OpenCV) Kütüphanesi: Bilgisayarla görme ve fotoğraf işleme alanında kullanılan en önemli kütüphanelerden biri olarak kabul edilmektedir. Araştırmacıların ve geliştiricilerin karmaşık fotoğraf ve video işleme görevlerini yüksek verimlilikle gerçekleştirmeleri için entegre ve kullanımı kolay bir ortam sağlamak üzere geliştirilmiştir. (OpenCV) çok dilli bir kütüphanedir ancak Python'a olan desteği,

Python'un kullanım kolaylığı ve genişletilebilirliği nedeniyle onu daha popüler hale getirmiştir.

(OpenCV) kütüphanesi, fotoğrafları okuma ve görüntüleme gibi temel işlemlerden, yüz tanıma ve nesne izleme gibi gelişmiş işlemlere kadar bilgisayarlı görmenin çeşitli yönlerini kapsayan geniş bir fonksiyon yelpazesi içerir. Kütüphane, nesne yönelimli programlama kavramlarına dayanmaktadır ve geliştiricilerin karmaşık uygulamaları yapılandırılmış ve bakımı kolay bir şekilde oluşturmasını kolaylaştırmaktadır. Aşağıda (Şekil 3.13) görüldüğü üzere, OpenCV simgesi görünmektedir:

Şekil 3.13. OpenCV simgesi

Bu yöntemde, (OpenCV) kütüphanesi, ünlü yer işaretlerini belirlemek için fotoğrafların önceden eğitilmiş modele geçirilmeden önce işlenmesinde önemli roller üstlenmektedir. (KNOWLEDGEDOCTOR, 2024). Bu roller şunlardır:

- Fotoğrafların Okunması: (OpenCV) kütüphanesi, belirtilen yoldaki fotoğrafı 'cv2.imread()' fonksiyonunu kullanarak açar. Bu fonksiyon, fotoğraftan görsel verileri okur ve bunları çok boyutlu bir matris biçiminde saklar.

Görsel Veriler: Bunlar, bir fotoğrafın taşıdığı ve fotoğraftaki nesnelerin görsel görünümüyle ilgili bilgilerdir. Bu veriler, renkler, şekiller, desenler, kontrastlar, yapılar, desenler vb. gibi bir dizi görsel öğeyi içerir. Görsel bağlamda, bu veriler, bir dizi nokta (piksel) olarak temsil edilir ve her nokta, rengi, parlaklığı, şeffaflığı, yakınlığı, uzaklığı veya başka bir ilgili özelliği temsil eden bir değeri taşır. Dijital fotoğraflarda, görsel veriler, bir fotoğrafın her pikselinin rengini temsil eden sayısal değerler olarak temsil edilir. Örneğin, eğer fotoğraf renkliyse, görsel veriler, üç kanalın kırmızı, yeşil ve mavi (RGB) değerlerini içerebilir. Ancak, eğer fotoğraf siyah beyazsa, görsel veriler sadece parlaklık değerlerinden oluşabilir.

- Fotoğrafın biçiminin dönüştürülmesi: Fotoğrafın okunmasının ardından, (OpenCV) kütüphanesi 'cv2.cvtColor()' fonksiyonunu kullanarak fotoğraf biçimini BGR'den RGB'ye dönüştürür. BGR'den RGB'ye fotoğrafın biçiminin dönüştürülmesi, BGR formatında mavi kanalın ilk, ardından yeşil ve sonra kırmızı kanalın saklandığı, RGB formatında ise kırmızı kanalın ilk, ardından yeşil ve sonra mavi kanalın saklandığı anlamına gelir. Bu, fotoğrafı görüntülerken, doğru renkleri elde etmek ve fotoğrafın, modelin eğitildiği görüntü biçimiyle uyumlu olmasını sağlamak için kanalların sırasının bu iki biçim arasında değiştirilmesi gerektiği anlamına gelir.
- Fotoğrafın Boyutunun Değiştirilmesi: RGB formatına dönüştürüldükten sonra, (OpenCV) kütüphanesi 'cv2.resize()' fonksiyonunu kullanarak görüntünün boyutunu istenen boyutlara değiştirir. Bu, görüntünün, modelin eğitildiği görüntülerin boyutuyla uyumlu olmasını sağlamak için yapılır.

```
img = cv2.imread(file_path)

RGBimg = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

RGBimg = cv2.resize(RGBimg, (321, 321))
```

Bu adımlar, görüntülerin, içlerindeki ünlü yer işaretlerini belirlenmesi için önceden eğitilmiş modele geçirilmeden önce uygun bir şekilde biçimlendirmeye yardımcı olur.

2. Modelin yüklenmesi ve sınıflandırma işleminin başlatılması:

Model: Önceden eğitilmiş bir makine öğrenme modelidir ve daha önce analiz edilmiş ve sınıflandırılmış geniş bir fotoğraf kümesinden öğrenilen bilgileri içerir. Yeni bir fotoğraf modeline gönderildiğinde, model bu bilgileri kullanır, fotoğraftaki desenleri ve özellikleri belirler ve eğitimine dayanarak onu sınıflandırır.

Sınıflandırma işlemi: önceden eğitilmiş modelin, modele geçirilen fotoğraflardaki nesnelerin hangi kategoriye veya ünlü yer işaretlerine ait olduğunu belirlediği bir işlemdir. Model, fotoğraflardan görsel özellikleri çıkarmak için derin sinir ağlarını kullanır. Bu özellikler, şekiller, renkler, doku ve diğer özellikler gibi özellikleri içerir ve yer işaretlerinin belirlenmesine yardımcı olur. Özellikler çıkarıldıktan sonra, model bunları eğitim verileriyle karşılaştırır. Makine öğrenme algoritmaları, çıkarılan özelliklere dayanarak fotoğrafları belirli kategorilere sınıflandırır. Sınıflandırma sonuçlarına dayanarak, fotoğraflardaki en olası yer işaretleri belirlenir, çıkarılan

özellikler ünlü yer işaretleri veri tabanıyla eşleştirilir. Model, fotoğrafın sınıflandırıldığı yer işaretlerinin veya kategorinin adını döndürür, bu daha sonra onunla ilişkili coğrafi verilerin çıkarılması için kullanılabilir.

Bu adım, öncelikle sınıflandırmada kullanılacak olan modelin yüklenmesiyle başlar ve (TensorFlow) kütüphanesi bunu gerçekleştirir.

(TensorFlow) Kütüphanesi: Makine öğrenimi ve yapay zekâ uygulamaları için en ünlü açık kaynak kütüphanelerinden biridir. (Google Brain) ekibi tarafından, sınıflandırma gibi alanlarda derin modelleri oluşturmak ve eğitmek için güçlü ve esnek bir yapı sağlamak üzere geliştirilmiştir. Sınıflandırma işleminin belkemiği olup, fotoğraflardaki özellikleri tanıyan makine öğrenmesi modelini yüklemek ve çalıştırmak için kullanılır.

(TensorFlow) kütüphanesi, modelin çalışma ortamını yapılandırmak için kullanılır, böylece fotoğrafları almak ve sınıflandırmayı gerçekleştirmek üzere hazır hale gelir.

```
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

tf.get_logger().setLevel('ERROR')
```

(TensorFlow), yürütme sırasında ortaya çıkabilecek gereksiz mesajları ve uyarıları azaltacak şekilde yapılandırılmıştır.

(TensorFlow) kütüphanesi, sınıflandırma modelini yüklemek için kullanılır. Önceden eğitilmiş modelin URL'sini belirleyerek ve (TensorFlow Hub) kütüphanesinde bulunan modeli yükleyerek gerçekleştirilir. Bu model, görüntüleri analiz etmek ve içlerindeki özellikleri belirlemek için kullanılır.

(TensorFlow Hub) Kütüphanesi: Makine öğrenmesi uygulamalarında doğrudan kullanılmak üzere önceden eğitilmiş bir dizi derin model sağlayan bir kütüphanedir ve sıfırdan eğitim yapmaya gerek kalmadan sınıflandırma için kullanılabilir. (TensorFlow Hub) kütüphanesinde bulunan modeller, sınıflandırma, teşhis, çeviri, ses analizi vb. gibi çeşitli alanları kapsar, bu da geliştiricilere uygulamaları için uygun modeli seçme olanağı sağlar. (TensorFlow Hub) kütüphanesi, fotoğraflardaki ünlü yer işaretlerinin tanınması da dahil olmak üzere çeşitli amaçlar için kullanılabilecek eğitilmiş modeller için bir depo olarak hizmet verir.

```
TF_MODEL_URL =
'https://tfhub.dev/google/on_device_vision/classifier/landmarks_classifier_asia_V1/1
'

LABEL_MAP_URL =

'https://www.gstatic.com/aihub/tfhub/labelmaps/landmarks_classifier_asia_V1_label
_map.csv'

   IMAGE_SHAPE = (321, 321)

   df = pd.read_csv(LABEL_MAP_URL)

   classifer = tf.keras.Sequential([hub.KerasLayer(

      TF_MODEL_URL,

      input_shape =IMAGE_SHAPE + (3,),

      output_key ="predictions:logits"

   )])

   label_map = dict ( zip (df.id, df.name))
```

(TensorFlow) kütüphanesi, '**TF_MODEL_URL**' değişkeninde bulunan önceden eğitilmiş derin modeli yükler. Model yüklenmesinden sonra, 'tf.keras.Sequential' kullanılarak bir (TensorFlow) modeli oluşturulur. Bu model, fotoğrafların sınıflandırılması ve içlerindeki ünlü yer işaretlerini belirlenmesi için kullanılır. (TensorFlow Hub) kütüphanesi, '**LABEL_MAP_URL**' değişkeninde bulunan CSV dosyasındaki ünlü özellik atamalarını yükler. Bu dosya, mevcut sınıflandırmalar ve bunlarla ilişkili yer işaretleri hakkında bilgi içerir. Özellik atamaları yüklendikten sonra, özellik kimliği ile adı arasında bir bağlantı sağlayan bir sözlük oluşturulur ve '**label_map**' değişkeninde saklanır.

Sonrasında, 'classifyimg' fonksiyonuna başvurularak giriş fotoğrafının modelin eğitildiği verilere uygun belirli bir kategoriye sınıflandırılması işlemi gerçekleştirilmektedir. Bu fonksiyon, öncelikle fotoğrafı modelin işleyebileceği bir biçime dönüştürür. Bu, 'RGBimg' giriş fotoğrafının bir 'numpy' dizisine dönüştürülmesi ve piksel değerlerinin 255'e bölünmesi anlamına gelir. Bu adımın amacı, piksel değerlerinin aralığını 0 ile 1 arasında bir aralığa dönüştürmektir, çünkü

yüklenen modelin 0 ile 1 arasındaki değerlerle çalışmasını beklenir, bu da eğitim sürecini hızlandırır ve performansı artırır. Bundan sonra, 'RGBimg' dizisi, (1, 321, 321, 3) şeklinde yeniden şekillendirilir, burada 1 fotoğrafların sayısını (tek bir toplu işlem), (321*321) görüntünün boyutunu ve 3 RGB renk kanallarını temsil eder. Bu yeniden şekillendirme, modelin beklediği veri biçimiyle uyumluluk için gereklidir.

'numpy' matrisi: Python'daki 'numpy' kütüphanesinde temel bir veri yapısıdır ve çok boyutlu verileri temsil etmek için kullanılır. 'numpy' matrisi, veri bilimi, görüntü işleme ve genel olarak bilimsel hesaplama için temel kabul edilir ve matematiksel işlemleri gerçekleştirme ve verilerle manipülasyon yapma konusunda güçlü araçlar sunmaktadır.

Sonrasında, 'classifer' modeli, işlenmiş '**RGBimg**' fotoğrafının hangi kategoriye ait olduğunu tahmin etmek için kullanılır. 'verbose = 0' parametresi, tahmin işlemi sırasında herhangi bir ayrıntılı mesajın yazdırılmasını önlemek için kullanılabilir. Tahminler '**prediction**' değişkeninde saklanır, ardından model, fotoğrafın ait olduğu kategorinin en yüksek olasılığını belirlemek için kullanılır ve bu kategorinin adını sonuç olarak döndürür. 'np.argmax' fonksiyonu, '**prediction**' tahmin matrisindeki en yüksek değerin indeksini bulmak için kullanılır, bu da modelin fotoğrafın ait olduğunu tahmin ettiği kategoriyi temsil eder. Ardından bu indeks, kategori indekslerini isimleriyle (önceden hazırlanmış) bağlayan '**label_map**' adlı bir sözlükte arama yapmak için kullanılır ve beklenen kategori adını döndürür. Bu, fotoğraftaki özelliklerin veya nesnelerin otomatik ve doğru bir şekilde belirlenmesine olanak sağlamaktadır.

```
def classifyimg (RGBimg):
    RGBimg = np. array (RGBimg) / 255
    RGBimg = np. reshape (RGBimg, (1, 321, 321, 3))
    prediction = classifer. predict (RGBimg, verbose = 0)
    return label_map [np. argmax(prediction)]
```

(TensorFlow Hub) kütüphanesi, (TensorFlow) kütüphanesinin yapısıyla sorunsuz bir şekilde entegre olacak şekilde tasarlanmıştır, bu da onu önceden eğitilmiş derin modelleri sağlamak ve uygulamalarda kullanmak için ideal bir (TensorFlow)

kütüphanesi tamamlayıcısı yapar. (TensorFlow) kütüphanesi, yapay sinir modellerini oluşturmak ve yüklemek için kullanılırken, (TensorFlow Hub) kütüphanesi, önceden eğitilmiş modelleri sağlamak ve makine öğrenmesi uygulamalarında kolayca kullanmak için bir aracı olarak kullanılır.

(TensorFlow Hub) kütüphanesi, (TensorFlow) kütüphanesinin yapısıyla sorunsuz bir şekilde entegre olacak şekilde tasarlanmıştır, bu da onu önceden eğitilmiş derin modelleri sağlamak ve uygulamalarda kullanmak için ideal bir (TensorFlow) kütüphanesi tamamlayıcısı yapar. (TensorFlow) kütüphanesi, yapay sinir modellerini oluşturmak ve yüklemek için kullanılırken, (TensorFlow Hub) kütüphanesi, önceden eğitilmiş modelleri sağlamak ve makine öğrenmesi uygulamalarında kolayca kullanmak için bir aracı olarak kullanılır.

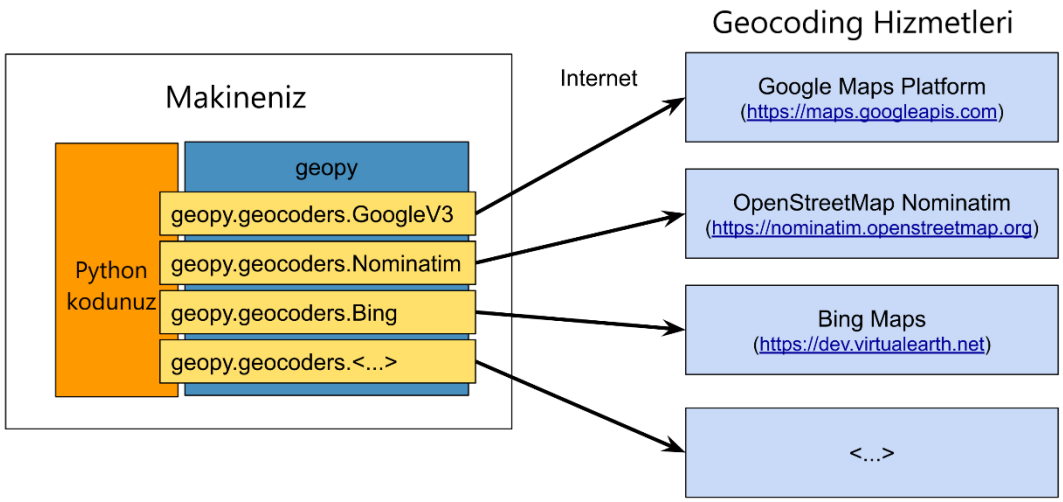
3. Coğrafi Bilgilerin Çıkarılması: Fotoğraftaki ünlü yer işaretlerini belirlendikten ve sınıflandırıldıktan sonra, fotoğraflarda keşfedilen ünlü yer işaretleriyle ilişkili coğrafi bilgileri (enlem ve boylam) elde etmek için (Geopy) kütüphanesi kullanılır. (Geopy) kütüphanesinden bir nesne oluşturulur ve (OpenStreetMap) hizmetlerindeki (Nominatim) aracını kullanarak özelliklerin adına göre arama yapmak ve onlara bağlı enlem ve boylamı elde etmek için kullanılır.

(Nominatim) aracı: Coğrafi hizmetleri sunan ve adresleri coğrafi koordinatlara dönüştüren (Coğrafi Kodlama) bir araçtır, ayrıca herhangi bir coğrafi konum için bir adres bulabilir (Ters Coğrafi Kodlama). (Nominatim) aracı, (OpenStreetMap) projesinin bir parçasıdır.

(OpenStreetMap): Dünya çapında bir proje olup, açık kaynaklı bir harita oluşturulmasıdır. Coğrafi veriler, dünya çapında gönüllüler tarafından toplanmaktadır. Kullanıcılara haritalara ve coğrafi verilere ücretsiz ve açık kaynaklı bir şekilde erişim sağlar, bu da geliştiricilerin bu verilere dayalı yenilikçi uygulamalar ve hizmetler oluşturmasına olanak sağlar.

Dolayısıyla, (Geopy) kütüphanesinden bir nesne oluşturulduğunda ve (Nominatim) aracı kullanıldığında, bu, konumları bulmak ve coğrafi koordinatlarını almak için (OpenStreetMap)'e ait (Coğrafi Kodlama) hizmetinin kullanılması anlamına gelir.

(Geopy) Kütüphanesi: Python'da coğrafi işlemler için uzmanlaşmış bir kütüphanedir ve çeşitli coğrafi işlemleri kolay ve etkili bir şekilde gerçekleştirmek için bir Uygulama Programlama Arayüzü (API) sağlar. Adresleri coğrafi koordinatlara dönüştürme (Coğrafi Kodlama), adres veya koordinatlarla konumları bulma (Ters Coğrafi Kodlama), koordinat noktaları arasındaki mesafeleri hesaplama gibi coğrafi verilere bağlı işlevler gibi. (Geopy) Kütüphanesi, Coğrafi Bilgi Sistemleri (GIS), coğrafi veri analizi, harita uygulamaları geliştirme ve coğrafi verilerle çalışan diğer birçok uygulama gibi alanlarda çalışan geliştiriciler için güçlü bir araçtır. Ayrıca, (Google Maps) ve (OpenStreetMap) gibi popüler harita hizmetlerini de dahil olmak üzere birçok coğrafi veri kaynağını destekler. Geliştiriciler, uygulamalarının ihtiyaçlarını karşılamak için uygun kaynağı seçebilirler. Aşağıda (Şekil 3.14) gösterildiği gibi, Geocoding hizmetlerinin (Geopy) kütüphanede görülmektedir:



Şekil 3.14. Geocoding hizmetleri

'classifyimg' fonksiyonu, fotoğraflardaki yer işaretlerinin tahminini almak için kullanılır ve ardından (Nominatim), ünlü yer işaretlerinin adını coğrafi bilgilere dönüştürmek ve onlara bağlı enlem ve boylamı almak için 'geolocator.geocode' fonksiyonunu kullanarak yer işaretlerini aramak için kullanılır.

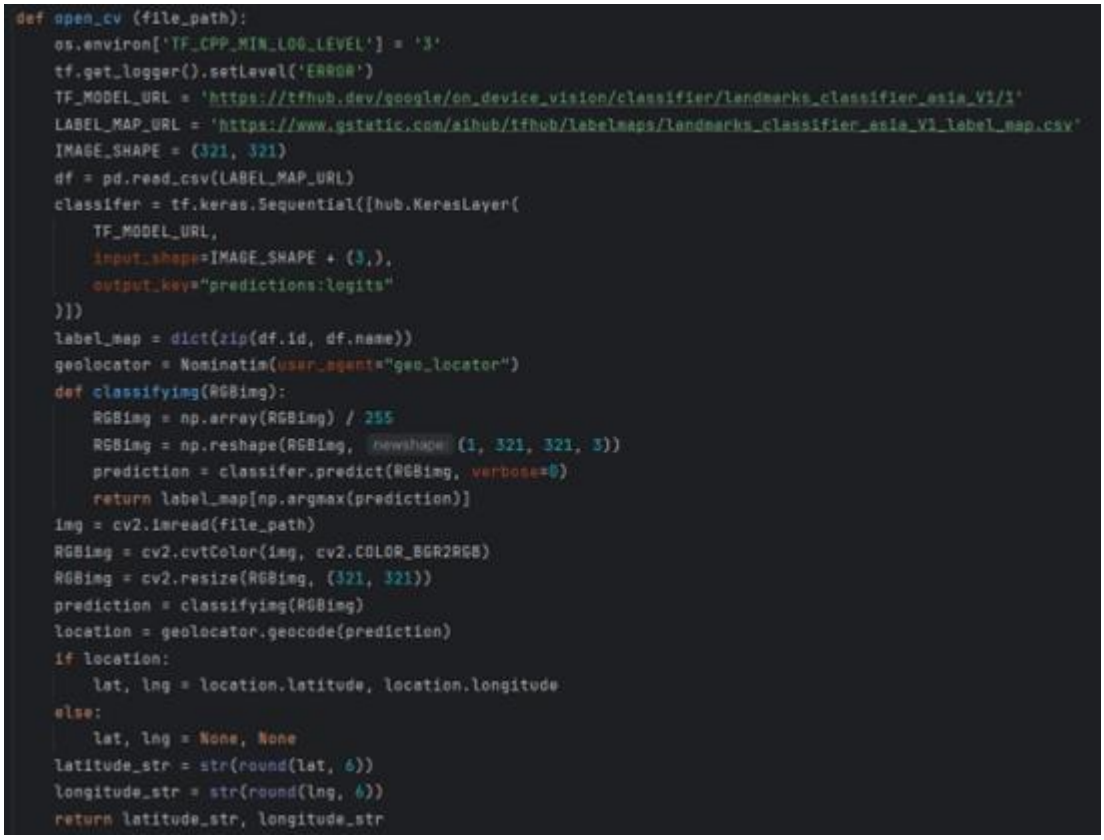
```
location = geolocator.geocode(prediction)

if location:

    lat, lng = location. latitude, location. longitude
```

```
else:
    lat, lng = None, None
latitude_str = str (round (lat, 6))
longitude_str = str (round (lng, 6))
```

Bu fonksiyonun çalışmasının bir sonucu olarak, enlem ve boylam ondalık sayılardan metne dönüştürüldükten sonra metin olarak döndürülür. Bu yöntemde izlenen programlama adımları, (**open_cv**) adını taşıyan bir fonksiyonda toplanır. Aşağıda (Şekil 3.15) gösterildiği gibi, tamamlanmış open_cv fonksiyonu görülmektedir:



Şekil 3.15. open_cv fonksiyonu

Üçüncü yöntem kullanılarak resimlerden (JPG) lokasyon verilerinin çıkarılması

İkinci yöntemde (Google Cloud Vision) olan (Metadata)'nın JSON dosyasıyla aynı adı taşıyan fotoğrafı çağrıldıktan sonra, (Google Cloud Vision) aracı aynı gönderiyi takip eden fotoğrafı analiz eder. Bu araç, fotoğraftaki coğrafi bilgileri çıkaramazsa, üçüncü yönteme (OpenCV, TensorFlow kütüphanelerinin kullanılması) geçilir. (Google Cloud Vision) aracının coğrafi bilgileri çıkaramadığı aynı fotoğraf, üçüncü yöntemin

çalışma mekanizmasını temsil eden (**open_cv**) fonksiyonuna geçirilir. Bu, coğrafi bilgilerin (Latitude, Longitude) bulunamaması durumunda, (**open_cv**) fonksiyonunun çağrılması ve (Google Cloud Vision) aracıyla coğrafi bilgilerin bulunamadığı aynı fotoğrafın geçirilmesi anlamına gelen bir koşul cümlesi eklenerek gerçekleştirilir.

(**open_cv**) fonksiyonu, ona geçirilen fotoğrafı analiz eder. Eğer bu fotoğrafta ünlü yer işaretleri bulunmuşsa, bu ünlü yer işaretlerine ait coğrafi bilgiler (enlem ve boylam) çıkarılır ve bu bilgiler daha önce tanımlanmış olan 'lat' ve 'lng' değişkenlerine eklenir. Program çalışırken çıkarılan tüm coğrafi bilgiler (enlem ve boylam) kaydedilir. Aşağıda (Şekil 3.16) gösterildiği gibi, kodun fotoğraflardan kalan lokasyon verilerini çıkardığı görülmektedir:

```
lat = data.get(node, {}).get('iphone_struct', {}).get('lat')
lng = data.get(node, {}).get('iphone_struct', {}).get('lng')
if lat is None and lng is None:
    jpg_file_name = file_name.replace('.json', '.jpg')
    similar_images = [f for f in files if f.startswith(jpg_file_name[:-5]) and f.endswith('.jpg')]
    if similar_images:
        for similar_image in similar_images:
            lat, lng = google_cloud_vision(os.path.join(source_folder_path, similar_image))
            if lat is None and lng is None:
                lat, lng = open_cv(os.path.join(source_folder_path, jpg_file_name))
```

Şekil 3.16. Fotoğraflardan lokasyon verilerini çıkarma

Eğer üçüncü yöntemde de herhangi bir coğrafi bilgi bulunamazsa, bu gönderinin, kullanıcının paylaştığı (yer etiketi) veya gönderinin fotoğraflarının görsel analizi aracılığıyla herhangi bir konum bilgisi içermediği anlamına gelir. Daha sonra, kullanıcının paylaştığı gönderilerden bir sonrakine geçilir ve her gönderi, yani üç yöntem üzerinden geçerek coğrafi bilgilerin çıkarılması için aynı işleme tabi tutulur.

Bu işlem, tüm gönderiler tamamlanana kadar devam eder. Bundan sonra, çıkarılan coğrafi bilgiler (enlem ve boylam) düzenlenir ve "Sonuçların analizi" adlı yeni bir özel sayfada görüntülenir.

3.2.5. Sonuçların Analizi Sayfası

Coğrafi konum bilgilerinin (enlem ve boylam) üçüncü taraf yöntemlerle gönderilerinden çıkarılması işlemi tamamlandıktan sonra, analiz süreci başlar. İlk olarak, enlem ve boylam bilgileri, (Geopy) kütüphanesine ait olan 'geocoder.osm' fonksiyonu kullanılarak adreslere dönüştürülür. Bu işlem, 'reverse' kullanılarak ters

coğrafi kodlama olarak adlandırılır. Çıkarılma işlemi sırasında, enlem ve boylam bilgileri koordinatlar olarak adlandırılan ondalık sayılar şeklinde bulunur ve bir haritadaki konumların bilgilerini temsil eder. Bu çalışmanın amacına ulaşmak için, bu koordinatların adreslere dönüştürülmesi gerekmektedir, böylece kullanıcının bulunabileceği ülke ve şehir belirlenebilir, bu da en çok tekrar eden şehirdir.

Program çalışırken çıkarılan tüm enlem ve boylam bilgilerini içeren 'lat' ve 'lng' değişkenleri çağrıldığında, bu değişkenler 'geocoder' fonksiyonuna geçirilir ve adreslere dönüştürülür. 'geocoder' fonksiyonu, bir ülkenin adından sokak adına veya binanın numarasına kadar herhangi bir koordinatın ayrıntılı adres bilgilerini elde edebilir, (raw.get('address')) sorgusu ile koordinatların bağlı olduğu adresle ilgili tüm bilgilere erişimi sağlayabilir.

Bu çalışma, sadece ülke adı ve şehir adı bilgilerini elde etmeye odaklanmaktadır. Ülke adı için, ülke adını döndüren (get('country')) sorgusu ile elde edilir. Şehir adı için ise, her ülkenin izlediği idari bölümlerin farklı olması nedeniyle elde etme yöntemi değişir. Örneğin, Amerika Birleşik Devletleri'nde, eyalet (province) olarak adlandırılır ve (get('province')) sorgusu ile elde edilir. Türkiye gibi diğer ülkelerde ise, il (state) olarak adlandırılır ve (get('state')) sorgusu ile elde edilir. Bu nedenle, şehir adını elde etmek için bu sorguların kullanılması gerekmektedir. Eğer ülkenin idari bölümleri eyalet (province) olarak kullanılıyorsa, bu sorgulanır ve çıkarılan şehir adı özel bir değişkene (**province**) saklanır. Eğer ülkenin idari bölümleri il (state) olarak kullanılıyorsa, bu sorgulanır ve özel bir değişkene (**state**) saklanır. Aşağıda (Şekil 3.17) gösterildiği gibi, kod coğrafi koordinatları adreslere dönüştürmektedir:

```
if lat is not None and lng is not None:
    reverse_geocoder = geocoder.osm( location: [lat, lng], method='reverse', timeout=10.0)
    country = reverse_geocoder.raw.get('address').get('country')
    country = translator.translate(country, dest='tr').text
    countryList.append(country)
    province = reverse_geocoder.raw.get('address').get('province')
    state = reverse_geocoder.raw.get('address').get('state')
    saved_data.append((lat, lng, country, province, state))
    if province:
        original_province.append(province)
        province = translator.translate(province, dest='tr').text
        provinceList.append(province)
    if state:
        original_states.append(state)
        state = translator.translate(state, dest='tr').text
        provinceList.append(state)
```

Şekil 3.17. Koordinatların adreslere dönüştürülmesi

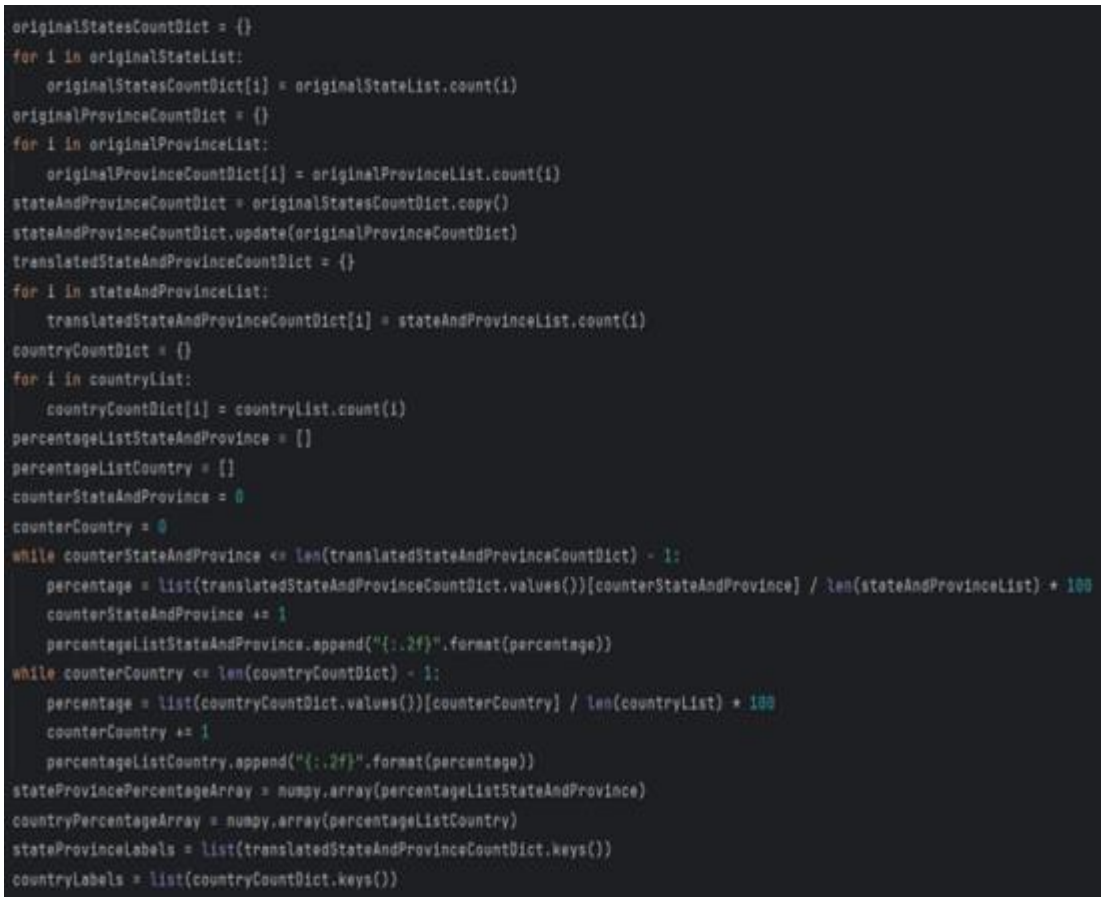
Bu çalışma, sonuçların görüntülenmesinde Türkçe dilini kullanmaktadır, bu nedenle çıkarılan ülke ve şehir adı, Python programlama dilinde anlık çeviri işlemlerinden sorumlu olan (googletrans) kütüphanesinin 'translator' fonksiyonu kullanılarak Türkçeye çevrilir. Elde edilen veriler, harita üzerinde görüntülendiğinde yararlanılabilmesi için koordinatlar ('lat' ve 'lng') ile birlikte mevcut olan ülke, il ve eyalet isimlerini içeren '**saved_data**' adında bir listede saklanır. Bu verilerin harita üzerinde görüntülenme işlemi sırasında tekrar tekrar veri çıkarma işlemine ihtiyaç duyulmaz, bu da zamandan ve emekten tasarruf sağlayarak harita üzerinde veri görüntüleme işlemini hızlandırır.

Daha sonra, her biri için sözlükler (dictionary) oluşturulur; ülke (country), eyalet (province) ve il (state). Bunun amacı, her ismin (ülke, il, eyalet) kaç kez göründüğünü belirlemektir. İsim, sözlükteki Anahtar (Key) olarak atanır ve görünme sayısı bu Anahtarın Değeri (Value) olarak atanır. Bu işlem her biri için ayrı ayrı yapılır, yani ülke için özel bir sözlük (**countryCountDict**) oluşturulur ve eyalet (**originalProvinceCountDict**) ve il (**originalStatesCountDict**) için de aynısı yapılır. Şehir adının Değeri (Value), bu şehrin (**stateAndProvinceList**) listesinde kaç kez göründüğüdür ve ülke adının Değeri (Value), bu ülkenin (**countryList**) listesinde kaç kez göründüğüdür. Bunun amacı, en çok tekrar eden ismi belirlemektir çünkü bu isim, en çok ziyaret edilen ülke ve şehir adını belirler.

İl ve eyalet adı için, her ikisi de çeviri öncesi kendi listesinde saklanır, eyalet için (**originalProvinceList**) ve il için (**originalStateList**). Çeviriden sonra, her ikisi de başka birleşik bir liste olan (**stateAndProvinceList**) listesinde saklanır. Bunun amacı,

bu verilerin haritada gösterildiğinde, haritanın çeviri sonrası il veya şehir adını bulamamasıdır. Bu yüzden, onlar çeviri öncesi özel listelerde saklanır, daha sonra en çok tekrarlanan şehri belirlemek ve haritada ona özel bir işaret koymak için birleşik bir sözlük olan (**stateAndProvinceCountDict**) içinde birleştirilirler. Grafikte gösterilmeleri söz konusu olduğunda, Türkçe olarak gösterilebilirler, bu yüzden çıkarma işlemi başladığından beri tek bir liste olan (**stateAndProvinceList**) listesinde toplanırlar.

Daha sonra, her bir ilçe ve eyalet adı için (**translatedStateAndProvinceCountDict**) sözlüğünde ve ayrıca her bir ülke adı için (**countryCountDict**) sözlüğünde yüzde oranı hesaplanır, bu, her bir ülke veya şehir adının yanında bu oranların grafikte gösterilmesi için yapılır. Bu oranlar, ülkeler için (**percentageListCountry**) listesinde ve şehirler için (**percentageListStateAndProvince**) listesinde olmak üzere her biri için özel listelere kaydedilir. Daha sonra bu listeler, 'numpy' kütüphanesi kullanılarak dizilere dönüştürülür, yani bu veriler, onlarla kolayca işlem yapma ve analiz etme yeteneği sağlayan özel bir veri yapısında temsil edilir. Bu oranlar, sözlüklerdeki şehir ve ülke adları ile aynı sıradadır, yani ilk oran, sözlükteki ilk Anahtarın adına karşılık gelir ve böyle devam eder. İsimler, (**countryCountDict**) ve (**translatedStateAndProvinceCountDict**) sözlüklerinden çıkarılır ve ülke isimleri (**countryLabels**) ve şehir isimleri (**stateProvinceLabels**) listelerine dönüştürülür ve bu, grafikteki Etiketler (Labels) belirlenirken kullanılır. Burada, sözlükteki her Anahtar, listedeki bir öğeyi temsil eder ve dolayısıyla listedeki her öğe, grafikte bir Etiket (Label) temsil eder. Aşağıda (Şekil 3.18) gösterildiği gibi, kod dönüştürülen her adresin yüzdesini hesaplamaktadır:



Şekil 3.18. Adreslerin yüzdesini hesaplama

Grafik çizmeye ve analiz sonuçlarının sayfasını göstermeye başlamadan önce, (**countryCountDict**) ve (**translatedStateAndProvinceCountDict**) sözlüklerinde verilerin mevcut olup olmadığı kontrol edilmelidir:

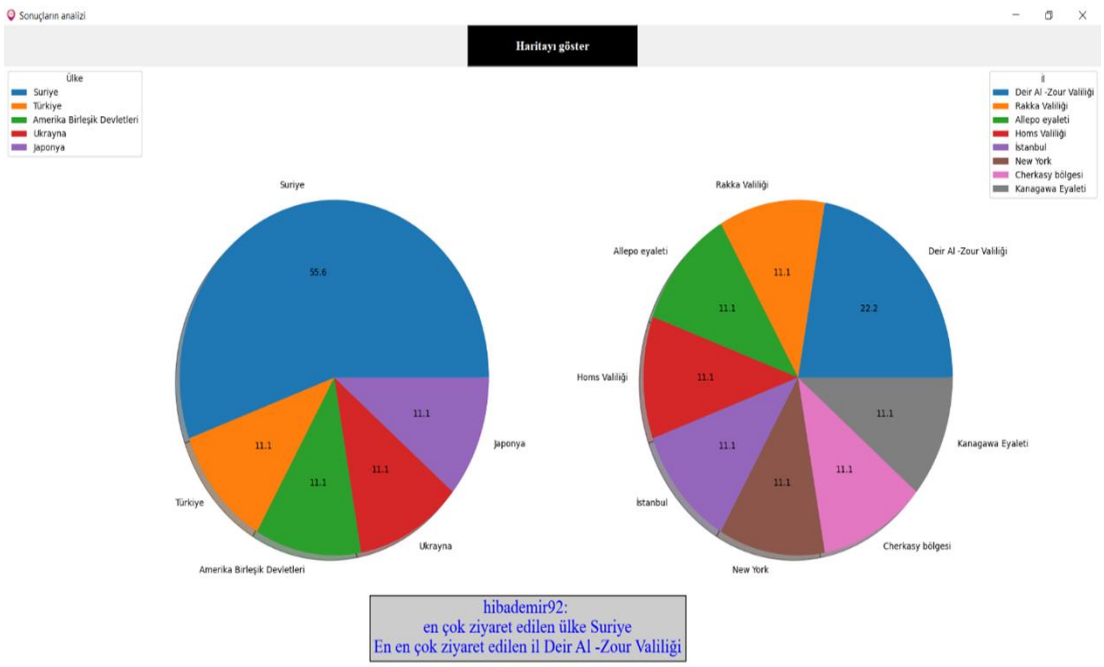
3.2.5.1. Veri mevcut olduğunda

Eğer sözlükler veri içeriyorsa, 'tkinter' kütüphanesi kullanılarak yeni bir pencere oluşturulur. Veri analizinin ve sonuçların görsel olarak gösterildiği bir penceredir. Bu pencerenin üst bölümünün ortasında, '**Haritayı göster**' adlı bir düğme bulunur, bu düğmeye tıklanıldığında, '**google_maps**' fonksiyonu çalıştırılır, bu fonksiyon, haritanın kendi özel İş Parçacığı (Thread) üzerinde gösterilmesinden ve keşfedilen yerlere pinlerin yerleştirilmesinden sorumludur.

Sonrasında, ülkelerdeki ve şehirlerdeki katılım oranlarını gösteren bir grafik oluşturulur. 'Pie' fonksiyonu, daire grafiklerini oluşturmak için kullanılır. Bu, Python'daki grafik çizme ve veri görselleştirme için olan (matplotlib) kütüphanesinin bir fonksiyonudur. Bu fonksiyon, bir değer listesi (verileri temsil eder) ve bir etiket

listesi (kategorileri temsil eder) alır ve değerlerin farklı kategoriler arasında nasıl dağıldığını gösteren bir daire grafiği çizer. Daire grafikler, (matplotlib) kütüphanesinin 'subplots' fonksiyonu kullanılarak şekilde dağıtılır. Bu fonksiyon, satır ve sütun sayısını alır ve bir eksenler ağı oluşturur. Bu pencerede, bir satır ve iki sütun içeren bir şekil oluşturulur, bu nedenle iki eksen oluşturulur, grafik boyutu 'figsize' parametresi ile belirlenir. Pencere, her biri bir ülke adı ile yüzde oranını ve diğeri bir şehir adı ile yüzde oranını gösteren iki daire grafiği içerir. Bu grafikler, ziyaretlerin farklı ülkeler ve şehirler arasında nasıl dağıldığını görsel olarak göstermektedir.

Daha sonra bu daire grafikleri pencerede gösterilir. Bunun için (matplotlib) kütüphanesi ile (tkinter) arasında bir arayüz olan 'FigureCanvasTkAgg' kullanılmaktadır. Bu arayüz, pencerede oluşturulan grafiklerin gösterilmesi için kullanılır. Ardından her bir grafik için bir açıklama kutusu eklenir. Bu açıklama kutusu, bu grafiği açıklayan ülkelerin isimlerini belirtmek için birinci eksen üzerine eklenir ve sayfanın en üst sol kısmında 'upper left' konumlandırılır. İkinci eksende ise bu grafiği açıklayan şehirlerin isimlerini belirtmek için bir açıklama kutusu eklenir ve sayfanın en üst sağ kısmında 'upper right' konumlandırılır. Aşağıda (Şekil 3.19) gösterildiği gibi, bir sonuç analiz sayfası örneği görülmektedir:



Şekil 3.19. Sonuç analiz sayfası örneği

Bu pencerenin alt kısmında, ortada, hedef kullanıcının adını, en çok tekrar eden ülkeyi (**maxCountry**) ve en çok tekrar eden şehri (**maxTranslatedStateAndProvince**) içeren bir kutu eklenir. Bu kutu, kullanıcının analiz işleminin sonucunu öğrenmesi için sunulan bir rapor olarak kabul edilir.

Kullanıcı, programın çalışması sırasında çıkarılan koordinatların haritada noktalar olarak gösterilmesini istiyorsa, bu pencerenin üst kısmındaki '**Haritayı göster**' düğmesine tıklayarak bunu yapabilir.

Harita görüntüleme

Verilerin analiz edilmesi ve sonuç sayfasında gösterilmesinin ardından, sonuçların haritada gösterilmesi, bu sonuçları daha iyi anlamak için ikinci bir ektir. Haritada gösterilen sonuçlar, programın çalışması sırasında çıkarılan enlem ve boylam koordinatlarıdır. Bu koordinatlar, haritada hangi konumda olduklarını ve hangi şehirde ve hangi ülkeye ait olduklarını belirlemeye yardımcı olan işaretler (markers) olarak haritada gösterilir.

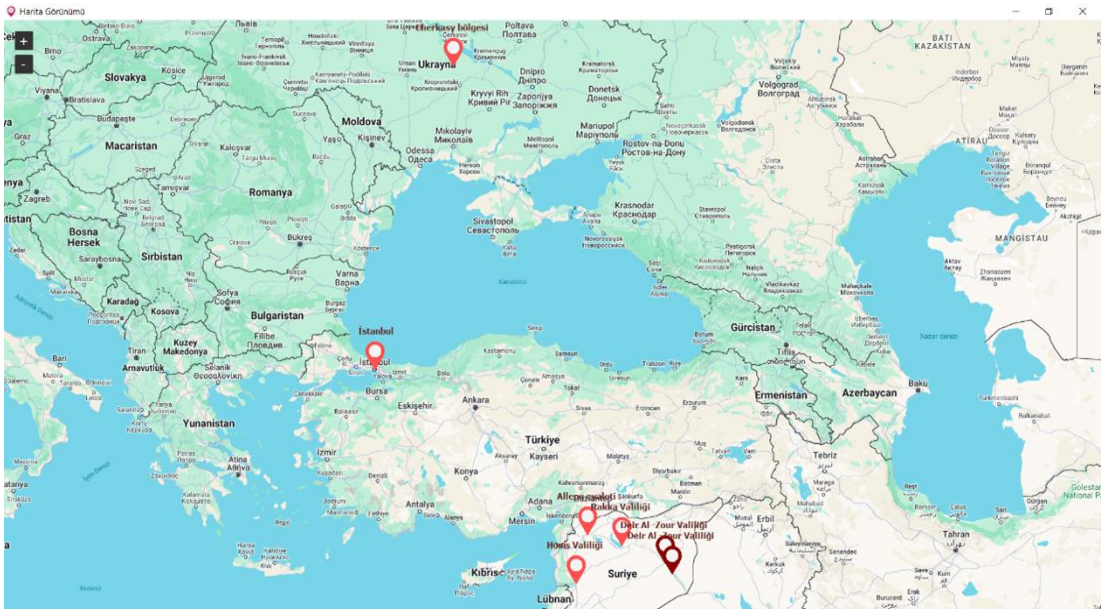
Haritanın içinde gösterileceği pencere, (Toplevel) kullanılarak oluşturulur ve bu pencerenin boyutları, adı ve simgesi belirlenir. Daha sonra 'tkintermapview.TkinterMapView' kullanılarak harita öğesi oluşturulur. Bu, (tkinter) ortamında haritaların oluşturulması ve görüntülenmesi için bir Uygulama Programlama Arayüzü (API) sağlar. Bu sınıfa birçok değer iletilir, bunlar arasında haritanın yerleştirileceği pencere (Toplevel), harita boyutu ve diğerleri bulunmaktadır.

Daha sonra, en çok öğeye sahip olan eyalet veya il (**maxStateAndProvince**), yani tekrar sayısı belirlenir. Onları diğerlerinden farklı renklerle ayırt etmek ve kullanıcının en çok ziyaret edilen şehri ve haritadaki konumunu kolayca belirlemesini sağlamak için yapılır. Haritada, çıkarılan ve (**saved_data**) içinde saklanan verilerdeki her konum koordinatı için işaretler atanır ve en çok tekrar eden şehir işareti, diğer işaretlerden farklı bir renkte olur ve her işaretin hangi şehirde olduğunu belirlemek için, bu koordinatlara ait şehir ismi, Türkçeye çevrildikten sonra işaretin yanına eklenir. Daha sonra, harita sunucusu, haritaları internet üzerinden harita sunucularından almayı sağlayan (Tile Map Server) protokolü kullanılarak 'set_tile_server' ile hazırlanır. Bu protokol, haritayı (Tiles) adı verilen küçük parçalara böler ve her parça, haritanın belirli bir bölümüne ait verileri taşır. Kullanıcı belirli bir fayansı görüntülemek istediğinde, sadece ekranda görünen bölge için bu (Tiles) yüksek çözünürlükte

indirilir, bu da haritanın görüntülenmesi ve yüklenmesi için hız ve verimlilik sağlar. Haritaları getirmek için kullanılan sunucunun adresi eklenir, bu adres, coğrafi verilerin alındığı kaynağı temsil eder ve bu programda kullanılan sunucu, (Google Maps) hizmetlerinin bir parçası olan Google'a aittir. Bu sunucunun adresinde, ülkelerin ve şehirlerin adlarının hangi dilde gösterileceği 'hl=tr' parametresi ile belirlenebilir, bu durumda bu haritada isimler Türkçe olarak gösterilir. Ayrıca, 'max_zoom' ile indirilebilecek maksimum yakınlaştırma seviyesi belirlenebilir. Aşağıda (Şekil 3.20) gösterildiği gibi, tamamlanmış google_maps fonksiyonu görülmektedir:

Şekil 3.20. google_maps fonksiyonu

Harita, en çok tekrar eden şehrin yerinde otomatik olarak ayarlanır, 'set_position' fonksiyonu kullanılarak yapılır. Bu adımları doğru bir şekilde tamamladıktan sonra, harita aşağıda (Şekil 3.21)'deki gibi görülmektedir:



Şekil 3.21. Harita görünümüne örneği

3.2.5.2. Veri mevcut olmadığında

Eğer sözlükler boşsa, kullanıcının paylaştığı gönderilerin lokasyon bilgilerini içermediğini veya hiç gönderi olmadığını belirten bir mesajı gösteren yeni bir pencere oluşturulur. Bu durum, analiz edilen hesabın hiç gönderisi olmaması veya kullanıcının paylaştığı gönderilerin lokasyon verilerini içermemesi durumunda gerçekleşir. Pencere aşağıda (Şekil 3.22)'deki gibi görülmektedir:

Şekil 3.22. Bilgi bulunamadı penceresi

3.2.6. Çalışmanın tam iş akışı diyagramı

Böylece, bu çalışma sona erer ve tüm hassasiyet ve dikkatle gerçekleştirilir. Bu çalışma sırasında hazırlanan ve uygulanan program, etkinliğini ve hedeflenen amaçları gerçekleştirme yeteneğini kanıtlamıştır. Bu çalışmanın, bu alanda daha derinlemesine

araştırma yapmak isteyen diğer araştırmacılar için yol aydınlattığı umulmaktadır. Bu çalışmanın başarısına katkıda bulunan herkese minnettarız ve gelecekte araştırmaya ve öğrenmeye devam etmeyi dört gözle bekliyoruz. Aşağıda (Şekil 3.23), projenin tam iş akışı diyagramı bulunmaktadır:

Şekil 3.23. Proje diyagramı

4. SONUÇ VE ÖNERİLER

Bilindiği gibi, teknoloji bugün hızla ve sürekli olarak gelişmektedir. Bilgiler, web sayfalarında ve sosyal medya ağlarında büyük miktarlarda yayılmaya başlamıştır. Ayrıca, Google, Bing, Yandex ve diğerleri gibi özel arama motorları veya Facebook, Twitter, Instagram ve diğer sosyal medya araçları kullanılarak tüm bilgilerin kolayca aranabildiği de bilinmektedir.

Bununla birlikte, özel arama motorları veya sosyal medya uygulamaları tarafından sunulan bilgiler bazen abartılı olabilmektedir, bu yüzden istenen bilgilere kolayca ulaşmak zordur. Örneğin, bir kişi hakkında arama yaptığında, aynı ismi taşıyan birden fazla kişi veya aynı kişi hakkında konuşan birden fazla kişi sonuç olarak çıkabilir.

Sosyal medya platformları, Facebook, Twitter ve Instagram gibi, bilgilerin yayılması ve aranması için en iyi yollardan biri olarak kabul edilmektedir, çünkü bunlar sadece bilgiyi sağlamakla kalmaz, aynı zamanda kullanıcılara sosyal medya aracılığıyla bilgi sağlamaktadır.

Bununla birlikte, olumsuz yönler de bulunmaktadır, sosyal medya araçları kullanılarak gönderilen bilgilerin aranması veya gerçek zamanlı olarak kolayca erişilmesi zordur. Bu sorunlar ve bilgiye erişim konusunda çözüm bulma çabası, bu çalışmada ele alınan sosyal medya taraması (Instagram) da dahil olmak üzere web taraması olarak adlandırılan şeyin ortaya çıkmasına neden olmuştur.

Büyük arama şirketleri, kendi özel robot programlarını (örümcekler veya tarayıcılar) hazırladığında, bu robotlar web sayfalarına ve çeşitli sosyal medya araçlarına tarama yapar ve içerideki veri içeriğini indeksler, böylece kullanıcının aradığı kesin bilgilere kolayca geçiş yapmasını sağlamaktadır.

Araştırma şirketleri, programcıların kendi programlarını oluştururken faydalanabilmeleri için bu robotların hizmetlerinden faydalanmalarını sağlamıştır. Bu çalışma, ilgili uygulamayı oluşturmak için sosyal medyayı temel aldığından, bu hizmetlerden de faydalanmaktadır.

Bu çalışma, Instagram uygulaması kullanıcılarının karşılaştığı bir sorunu bu hizmetleri kullanarak çözmeyi amaçlamaktadır. Buradaki sorun, Instagram'daki arkadaşların bulundukları veya gittikleri yerlere göre kendi gönderilerini paylaşmalarıdır. Bu gönderiler farklı yerlerden geldiğinde, kullanıcılar bu arkadaşın bulunduğu yeri tahmin etmede zorluk çekmektedir.

Doğal olarak, kullanıcılar gönderilerinin çoğunu bulundukları yerlerden paylaşmaktadırlar. Ancak, çoğu zaman, gönderinin paylaşıldığı yerin adı belirtilmemektedir.

Bu çalışma, robotları bu arkadaşın sayfasına yönlendirir ve ardından bu gönderileri, Python'daki (instaloader) gibi bu amaç için hazırlanmış kütüphaneleri kullanarak getirir, bu da bu çalışmada kullanılan kütüphanedir. Bazı araçlar ve programlama yöntemleri kullanılarak, bu gönderiler analiz edilmektedir.

Öncelikle, kullanıcının gönderilerinde paylaştığı lokasyon bilgilerine (yer etiketi) dayanılmıştır ve bu bilgiler, her gönderi için meta verileri (Post Metadata) kazıyarak (instaloader) kütüphanesi tarafından kazınmıştır. Kullanıcı tarafından paylaşılmaması durumunda, aynı gönderiyle ilişkili fotoğraflar, (Google cloud vision) aracını kullanarak analiz edilir. Eğer (Google cloud vision) aracı lokasyonla ilgili bilgiler bulamazsa, aynı fotoğraflar (OpenCV, TensorFlow) kütüphaneleri kullanılarak analiz edilmektedir.

Bundan sonra, elde edilen sonuçlar analiz edilir ve bir rapor şeklinde kullanıcıya sunulur. Ardından, bu sonuçlar haritalarla eşleştirilir, her gönderinin paylaşıldığı yer belirlenir ve bu arkadaşın bulunabileceği yer tahmin edilmektedir.

Bu çalışma, meta verilerden veya fotoğrafların analizinden elde edilen GPS konum bilgilerine (enlem ve boylam bilgileri) dayanarak analizler yapar ve sonuçları çıkarır. Bildiğimiz kadarıyla, bu, Instagram verilerinde bu tür bir analiz yapan ve üç yönteme dayanarak coğrafi konum bilgilerini çıkaran ilk çalışmadır.

Bu çalışma, her gönderinin konumunun nasıl belirleneceğini ve dikkate alınması gereken özelliklerin neler olduğunu açıklamaktadır. Ayrıca, bir robotun önce kullanıcının sayfasına nasıl tarama yaptığı, ardından gönderilerine tarama yaptığı ve bunları birer birer nasıl kazıdığı göstermektedir.

Gelecekte, araştırmalarımızı geliştirmeyi ve Facebook ve Twitter gibi diğer sosyal medya hesaplarındaki analiz ve keşifleri geliştirmeyi hedefliyoruz. Ardından, kazımayı ve analizi, arkadaşların günlük yaşamlarında paylaştıkları hikayeleri, öne çıkan gönderileri ve videoları içerecek şekilde genişletiyoruz. Bu, kullanıcıların yerini daha doğru bir şekilde tahmin etmeye yardımcı olacak en fazla bilgiye ulaşma yeteneğini artıracaktır.

KAYNAKLAR

Chau, D., Pandit, S., Wang, S., Faloutsos, C. (2007). Parallel crawling for online social networks. In Proceedings of the 16th international conference on World Wide Web, 1283-1284. https://doi.org/10.1145/1242572.1242809

Chen, Z., Pokharel, B., Li, B., Lim, S. (2020). Location extraction from twitter messages using a bidirectional long short-term memory neural network with conditional random field model. GISTAM, 45–50. http://dx.doi.org/10.1007/978-3-030-76374-9_2

Chong, W. H., ve Lim, E. P. (2017). Tweet Geolocation: Leveraging Location, User and Peer Signals. CIKM, 1279–1288. https://doi.org/10.1145/3132847.3132906

Chong, W. H., ve Lim, E. P. (2019). Fine-grained geolocation of tweets in temporal proximity. ACM TOIS, 37(2), 1–33. https://doi.org/10.1145/3291059

Dai, R., Luo, J., Luo, X., Mo, L., Ma, W., Zhou, F. (2023). Multi-modal Representation Learning for Social Post Location Inference. IEEE. https://doi.org/10.1109/ICC45041.2023.10279649

Dutt, F., ve Das, S. (2021). Fine-grained geolocation prediction of tweets with human machine collaboration. arXiv. https://doi.org/10.48550/arXiv.2106.13411

Ferrara, E., Meo, P. D., Fiumara, G., Baumgartner, R. (2012). Web Data Extraction, Applications and Techniques: A Survey. ResearchGate, 1-41. http://dx.doi.org/10.1016/j.knosys.2014.07.007

Firebase Documentation. (2024, 17 Mart). Bulut Firestore. Firebase. https://firebase.google.com/docs/ adresinden 11 Aralık 2023 tarihinde alınmıştır.

Gilani, Z., Farahbakhsh, R., Tyson, G., Crowcroft, J. (2019). A Large-scale Behavioural Analysis of Bots and Humans on Twitter. ACM Transactions on the Web, 13(7), 1–23. http://dx.doi.org/10.1145/3298789

Google Developers. (2023, 26 Kasım). Google Cloud Vision API Python ile Resim Tanıma. Google Developers Codelabs. https://codelabs.developers.google.com/codelabs/cloud-vision-api-python#0 adresinden 26 Kasım 2023 tarihinde alınmıştır.

Instaloader. (2023, 9 Ekim). Instaloader Documentation. GitHub Pages. https://instaloader.github.io/ adresinden 9 Ekim 2023 tarihinde alınmıştır.

Ishida, K. (2015). Estimation of User Location and Local Topics Based on Geo-tagged Text Data on Social Media. IEEE. 15-17. https://doi.org/10.1109/IIAI-AAI.2015.203

KNOWLEDGEDOCTOR. (2024, 17 Mart). Google Lens Clone in Python. YouTube. https://www.youtube.com/watch?v=Q_7UUT-EWHQ&ab_channel=KNOWLEDGEDOCTOR adresinden 17 Mart 2024 tarihinde alınmıştır.

Khan, A., Zhang, H., Boudjellal, N., Ahmad, A., Khan, M. (2023). LocBERT: Improving Social Media User Location Prediction using Fine-Tuned BERT. BOOK CHAPTER published 2023 in Database and Expert Systems Applications - DEXA 2023 Workshops (34th ed.). https://doi.org/10.1007/978-3-031-39689-2_3

Lau, J. H., Chi, L., Tran, K. N., Cohn, T. (2017). End-to-end network for Twitter geolocation prediction and hashing. ACL, 744–753. https://doi.org/10.48550/arXiv.1710.04802

Lamsal, R., Harwood, A., Read, M.R. (2022). Where did you tweet from? Inferring the origin locations of tweets based on contextual information. IEEE. https://doi.org/10.1109/BigData55660.2022.10020460

Li, X., Larson, M., Hanjalic, A. (2017) Geo-distinctive visual element matching for location estimation of images. IEEE, 20(5),1179–1194. https://doi.org/10.1109/TMM.2017.2763323

NeuralNine. (2023, 29 Kasım). Google Cloud Vision API For Image Annotation in Python. YouTube. https://www.youtube.com/watch?v=1EBhUDAlrYU&t=2s&ab_channel=NeuralNine adresinden 29 Kasım 2023 tarihinde alınmıştır.

SBDeveloper. (2023, 4 Eylül). How To Upload Images To Firebase Storage And Download From Firebase Using Python. YouTube. https://www.youtube.com/watch?v=f388UfOoF4g&ab_channel=SBDeveloper adresinden 4 Eylül 2023 tarihinde alınmıştır.

Tzovelekis, K., Kanakaris, V., Bandekas, D. V. (2018). Geo-Location Twitter And Instagram Based On OSINT Techniques: A Case Study. International Journal of Advanced Research, 780-791. http://dx.doi.org/10.21474/IJAR01/6283

Wang, Z., Guo, Y., Zheng, S., Xu, W., Liu, L., Liu, Z., Cui, X. (2018). Users' location analysis based on Chinese mobile social media. Wiley, 1-8. https://doi.org/10.1002/cpe.4669

Williams, E. (2016). GeoContext: Discovering geographical topics from social media. IEEE. 1342-1345. https://doi.org/10.1109/ASONAM.2016.7752411

Xu, D., Cui, P., Zhu, W., Yang, S. (2014). Graph-Based Residence Location Inference for Social Media Users. IEEE, 21(4) ,76-83. https://doi.org/10.1109/MMUL.2014.62

Xu, D., ve Yang, S. (2014). Location Prediction in Social Media Based on Contents and Graphs. IEEE. 1177-1181. https://doi.org/10.1109/CSNT.2014.239

Ye, S., Lang, J., Wu, F. (2010). Crawling Online Social Graphs. IEEE, 1–8. https://doi.org/10.1109/APWeb.2010.10

Zhou, F., Wang, T., Zhong, T., Trajcevski, G. (2022). Identifying user geolocation with Hierarchical Graph Neural Networks and explainable fusion. Information Fusion, 81(9), 1–13. http://dx.doi.org/10.1016/j.inffus.2021.11.004

ÖZGEÇMİŞ

Ad-Soyad : Yahya ALALI

ÖĞRENİM DURUMU:

- **Lisans** **:** 2022, Kocaeli Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği
- **Yükseklisans** **:** 2024, Sakarya Üniversite, Bilgisayar Mühendisliği PR, Tezli

MESLEKİ DENEYİM VE ÖDÜLLER:

- 2021-2022 yılları arasında Stajyer - Bankacılık / Bilgi Güvenliği. Ziraat Katılım Bankası - İstanbul/ TÜRKİYE.
- 2022-2023 yılları arasında Bilgisayar Mühendisliği / Web Tasarımcısı. PARS AR-GE - Kocaeli/ TÜRKİYE.

TEZDEN TÜRETİLEN ESERLER:

- ALALI .Y,12. Uluslararası Marmara Fen Bilimleri Kongresi, 31 May-1 Jun 2024, Kocaeli, Türkay. Sosyal Medya Lokasyon Analizi.