**Mailden Veri Çekme Projesi**

Allah’a hamd, Resulüne salat ve selam olsun. Selamun aleykum ve Rahmetullahi ve Berakatuhu.

**Özet:** Çağın ilerlemesiyle Allah’ın bizlere ikram ettiği nimetler artmaktadır. İnsanlar arasındaki iletişimin hızlanması ve yakınlaşmasıyla birlikte bizlere yeni hizmet alanları da açılmaktadır. Bu alanlardan biri de insanların mail olarak gönderdikleri soruların takip edilmesi, düzenlenmesi ve ümmet için faydalı hale getirilmesidir. Allah’ın izniyle projemiz bunu hedeflemektedir. Genel çerçevede projemiz gmailden topladığı verileri ham, temizlenmiş ve spam olarak ayırmaktadır. Ham veriler temizlenerek hem anonimleştirilmekte hem de soru – cevap çiftleri oluşturulmaktadır. Ayrıntılar için aşağıdaki adımları takip edebilirsiniz.

**Abstract**: As the age progresses, the blessings that Allah has bestowed upon us are increasing. As communication between people accelerates and becomes closer, new areas of service are also opening up for us. One of these areas is to track, organize and make useful for the Ummah the questions that people send via e-mail. With Allah’s permission, our project aims to do this. In general, our project separates the data it collects from gmail as raw, cleaned and spam. Raw data is cleaned and both anonymized and question-answer pairs are created. You can follow the steps below for details.

**Projedeki adımlar:**

1. **Araştırma**: Var olan çalışmalara göz atıldı. Bulunan Github çalışmaları incelendi, denendi.
2. **Veri oluşturulması:** Fake mailler üzerinde deneme amaçlı soru cevap mailleri oluşturuldu. İkinci adım olarak daha karmaşık veri yapısı tespiti için kendi şahsi maillerimiz üzerinde denendi.

* **Veri tipleri:**

Soru

Soru – cevap

Soru – soru – cevap

Soru – cevap – soru – cevap

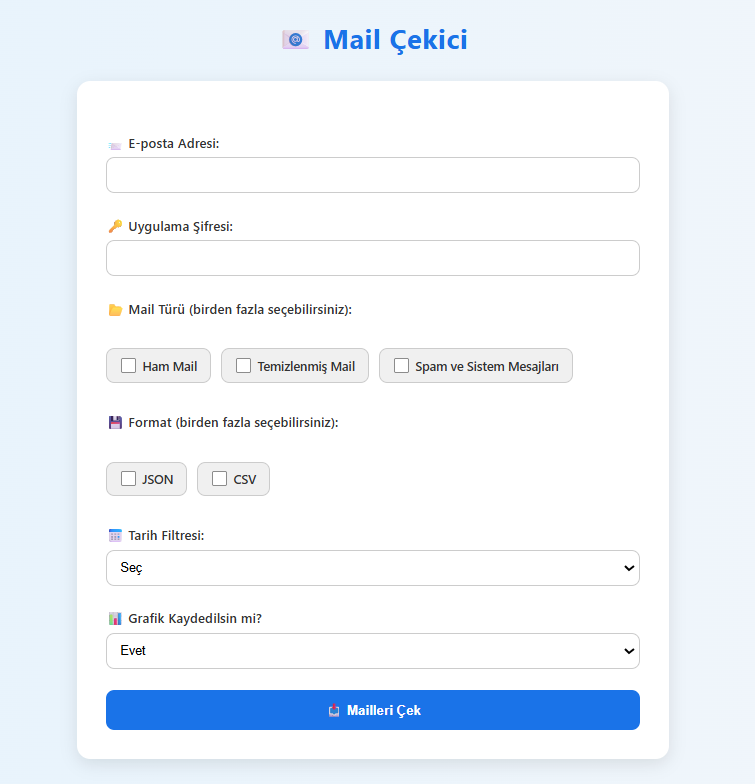
1. **Kodların oluşturulması**: İlk adım olarak yeni bir env oluşturuldu. Kodlar yz desteğiyle oluşturuldu ve gerekli değişiklikler tespit edilip kimi manuel kimi de gpt desteğiyle düzeltildi.
2. **Verinin çekilmesi:**

* Gmail sunucusuna güvenli şekilde bağlanıyoruz.
* Kullanıcı adı ve şifre ile giriş yapıyoruz.
* Gelen kutusunu açılıyor.
* Gelen kutusundaki tüm maillerin numaralarını alınıyor.
* Her mailin tam içeriği sırayla indiriliyor.
* İndirilen ham mail, Python koduyla okunabilir hale getiriliyor.
* Böylece mailin konusu, tarihi ve içeriği gibi bilgilere ulaşabiliyoruz.
* Ham mail, mail gönderen kişi ve mail ile ilgili tüm ham verilerin çekildiği kısımdır. İçeriğinde şu bilgiler bulunur:
* msg\_id: Mailin benzersiz kimliği
* in\_reply\_to: Yanıtlanan mailin kimliği
* references: Mail zincirindeki önceki mesajlar
* konu: Mailin konusu
* tarih: Gönderim zamanı
* base\_questions: Sorulan soru
* full\_answer: Sorunun cevabı

1. **Verinin temizlenmesi:**

* [image:...] etiketleri, HTML kodları ve fazla boşluklar kaldırılır.
* Alıntı satırları silinir: "On ... wrote:" gibi önceki mesaj alıntıları metinden çıkarılır.
* Sistem mailleri filtrelenir: "Güvenlik uyarısı", "İki Adımlı Doğrulama" gibi konulara sahip sistem mesajları atlanır.
* Google kaynaklı mailler hariç tutulur: Gönderen bilgisi "google" içeren mailler temizlenmiş veride yer almaz.
* Zincirli soru-cevaplar eşleştirilir: İlk soruya bağlı ikinci soru ve cevap aynı zincirde gruplanır.
* Metin içerisindeki linkler toplanır: Sorular ve cevaplardan çıkarılan tüm linkler ayrı bir sütunda listelenir.
* Ham mailden alınan verileri temizlendikten sonra bu veri setinde beş sütun bulunmaktadır:
* base\_question: İlk sorunun metni
* full\_answer: İlk sorunun cevabı
* soru\_2: Zincirdeki ikinci soru
* cevap\_2: Zincirdeki ikinci cevabın metni
* link: base\_question ve full\_answer içindeki linkler bu sütunda toplanır

Ayrıca içeriklerde yer alan gereksiz bölümler (örneğin “Şu tarih de şu kişi yazdı” gibi ifadeler), spam mesajlar ve sistem mesajları temizlenerek veri kalitesi artırılır.

1. **Arayüz oluşturulması:**

Backend ve Frontend kısımlarının son kodlara göre oluşturulup düzenlenmesi yapıldı.

Gmaile arayüzden giriş yaptıktan sonra; maillerinizi hangi tür, format ve tarih aralığında kaydetmek istediğini seçiyorsunuz. Bunları görselleştirme seçeneğinizde ekli. Tarihte seçim yapılmadığı zaman tüm mailleri ele almaktadır.

1. **Sonuç olarak:** Çalışmamız tamamlandı. Gmail adresimize uygulama şifremizle giriş yaparak oradaki temiz maillerimize erişim sağlayabiliyoruz. Bizlere tamamlamayı nasip eden Allah’a hamdolsun.

## **Yaşadığımız sıkıntılar:**

1. Hotmail hesabından veri indirmek isteyince otomatik veri indirme kısmı için 4 saat süre verdi.
2. İndirilen veride tarih kaynaklı ilerlediği için soru cevap eşlerinde eşleşme problemi yaşandı.
3. [GitHub - nsv777/get-data-from-gmail: Extracts data from gmail emails](https://github.com/nsv777/get-data-from-gmail) adımları çok uğraştırdı. Adımların açıklaması çok net değil.
4. credentials.json dosyasını alabilmek için Client ID oluşturmamız gerekti ve bu sırada hata verdi. OAuth onay ekranı tamamlanmamış dedi.
5. OAuth istemci kimliği oluşturmaya çalışırken web tercihi yaptık. Bunun için web adresine ihtiyaç oldu.
6. Çok düzenleme gerektirdi ve vakit aldı.
7. <https://github.com/nsv777/get-data-from-gmail> boş döndürdü. Read Me hiç açıklayıcı değil. Karışık yazılmış.
8. <https://github.com/akcarsten/archive_email> boş döndürdü. Read Me hiç açıklayıcı değil. Karışık yazılmış.
9. Mailden çekilen veriler düzenlenirken kodları Chatgpt’ye yazdırdık. Verinin ufak bir kısmını paylaşarak buradan json formatında soru cevap çiftleri oluştur, dedik. Soru ve cevabı istediğimiz şekilde ayırmadı. Gereksiz bilgileri ekledi. Başta veriden olmayan kendi soru cevap çiftleri oluşturdu.
10. Temel soru cevap mailleri dışında reklam ve spam mailleri de ham verimizde mevcuttu. Bunları silince cevapsız sorular da siliniyordu.
11. Farklı bir mail adresi için deneyince klasör adında problem oldu. Klasör adı yeni mail adresinin klasör adına göre değiştirildi. Bu sefer de Türkçe karakter olduğu için hata verdi.
12. Türkçe karakter seçimi yapılsa da klasör adlarıyla tam bir eşleşme olmadı.
13. Kendi mailimizle deneyince 1365 mailden sadece 281 tanesini temizlenmiş olarak ekledi. Ayrıca gönderilmiş mailleri yine dahil etmedi. Koddaki bir adımdan dolayı onları siliyor galiba. Daha bunun sebebini bulamadık.
14. Her bir denemede ufak hatalarla karşılaştık. 7.Denemedeki hata soru olmadan cevap olanları silip temizlenmiş veriye dahil etmiyordu.
15. Kendi maillerimizde uygun soru cevap maillerinin bulunmaması.
16. Spam maillerini güzel bir şekilde temizlemiyor.
17. Fake mailden veri çekerken sebepsiz yere problem yaşadık. (Şifre değişmiş gözüküyordu.)
18. Arayüz oluşturma (frontend) de bazı normal sorularımızı da spam olarak algıladı. Nedenini bilmiyorum.
19. Veri de spam olarak algılanmayan google mesajları vardı.
20. Tarih filtresi koyduğumda verileri boş dönüyordu.

Kullanıcıdan alınan tarih (datetime nesnesi) zaman dilimi bilgisi olmadan (offset-naive) oluşturuluyordu. E-postalardan gelen tarih ise zaman dilimi bilgisiyle (offset-aware) geliyordu. Python, offset-naive ile offset-aware datetime nesnelerini karşılaştıramaz, bu yüzden hata alınıyordu.

1. Ama bu eklentiden sonra hiçbir mail gelmez oldu. Meğer Python, “zaman bilgisini içeren tarih” ile “sade tarih” karşılaştırmasını kabul etmiyormuş. Bu yüzden her e-posta hata verip işlenmeden geçilmiş.
2. Mailleri de zincir yaparken ikinci sorulardan ve cevaplardan sonra önceki verilen yanıtlar ve şu kişiden gönderildi gibi kısımlar alınıyordu.
3. requirements.txt dosyasına kütüphaneleri yüklerken dosyalarımdan aldığı için @ işaretini koyuyordu. Bu da farklı bilgisayarlarda hata verebilir.

## **Nasıl çözdük:**

1. Hotmail yerine gmail kullanımı daha hızlı indirmeye yardımcı oldu.
2. Eşleştirme seçenekleri değiştirildi.
3. Adımları anlayabilmek adına chatgpt’den yardım aldık.
4. Client ID için, OAuth onay ekranını yapılandırdık:
   * Sol menüde **"OAuth onay ekranı"** seçeneğine git.
   * Kullanıcı türü olarak **"Harici"** seçin.
   * Uygulama adını yazın (örneğin: Gmail Extractor).
   * E-posta adresinizi yazın.
   * Diğer alanlar boş kalabilir. Kaydet ve devam et.
   * Test kullanıcıları kısmına kendi Gmail adresinizi ekleyin.
   * Devam edip onay ekranını tamamlayın.
5. Önceki oluşturduğumuzu silip aşağıdaki adımlara göre yeniden oluşturduk.
   * Geri dön Kimlik Bilgileri sayfasına.
   * Tekrar "OAuth istemci kimliği" oluştur'a basın.
   * **Uygulama türü: Masaüstü Uygulama (Desktop App)** seçin.
   * Ad verin ve oluştur’a basın.
6. Devam edilmedi. Başka seçeneklere geçildi.
7. Uğraştık ama olmadı. Başka yol denendi.
8. Uğraştık ama olmadı. Başka yol denendi.

* 7 . ve 8. madde de olmadığı için Chatgbt ile beraber kendi kodumuzu yazdık. Resmi sitelerde ve diğer kısımlardan bakıldı.
  + import imaplib
  + import email
  + from email.header import decode\_header
  + from email.utils import parseaddr
  + import getpass
  + import json
  + def temizle(metin):
  + if not metin:
  + return ""
  + if isinstance(metin, bytes):
  + metin = metin.decode(errors="ignore")
  + return metin.replace("\\r", "").replace("\\n", " ").strip()
  + # Giriş bilgileri
  + kullanici = input("E-posta adresiniz: ")
  + sifre = getpass.getpass("Uygulama şifresi: ")
  + …
  + msgid\_to\_mail[in\_reply\_to]["Cevap"] = temizle(icerik)
  + # JSON dosyası oluştur
  + sonuclar = list(msgid\_to\_mail.values())
  + with open("epostalar.json", "w", encoding="utf-8") as dosya:
  + json.dump(sonuclar, dosya, ensure\_ascii=False, indent=2)
  + print("epostalar.json dosyası oluşturuldu.")
  + imap.logout()

bu kısımda da ilk önce <https://myaccount.google.com/security> linkinden 2 adımlı doğrulama kısmını açmamız gerekiyor.

Daha sonra <https://myaccount.google.com/apppasswords> bu linkten hesabımızı şifremizi girerek açtıktan sonra Uygulama adı kısmına Mail yazıyoruz ve bize verdiği şifreyi bir kenara kaydediyoruz. Kaydetmezsek şifreyi bir daha göstermiyor.Kodda istenilen uygulama şifresine bu şifreyi giriyoruz.

1. Promptumuzda düzeltmeler yaptık. İçerik kolonundakiyle soru kolonu, benim\_cevabım kolonuyla da cevap kolonu oluştur. Burada yer alan linkleri de ayrı bir kolon olarak link adıyla ekle ve önceki bulunduğu kolondan sil, dedik.
2. Bunları ayırmak için sorulardan cevabı olanları ekle, cevabı ve linki olmayanları ekle, cevabı olmayıp linki olanları atla yönlendirmeleriyle aşağıdaki kodu elde ettik.

# 🚫 Filtreleme mantığı

if len(raw\_cevap) < 1:

if linkler and len(linkler) > 10:

continue # çok fazla linkli ama boş cevap → atla

# ✅ Geçerli kayıtları ekle

if soru and (cevap or not linkler):

yeni\_kayit = {

"soru": soru,

"cevap": cevap

}

if linkler:

yeni\_kayit["link"] = linkler

temiz\_veri.append(yeni\_kayit)

1. Türkçe karakter problemi çözümü için:

turkce\_klasor = "Gönderilmiş Postalar" imap\_uyumlu\_klasor = encode\_utf7(turkce\_klasor) imap.select(f'"[Gmail]/{imap\_uyumlu\_klasor}"') denendi.

1. Aşağıdaki kodla maildeki klasör adlarını listedik ve devamındaki kodla düzenledik.

import imaplib import getpass

#Giriş yap

kullanici = input("E-posta adresiniz: ") sifre = getpass.getpass("Uygulama şifresi: ") imap = imaplib.IMAP4\_SSL("[imap.gmail.com](http://imap.gmail.com/)") imap.login(kullanici, sifre)

#Klasörleri listele

status, klasorler = imap.list() for k in klasorler: print(k.decode())

imap.logout()

#gönderilmiş klasör seçimi

status, \_ = imap.select('"[Gmail]/G&APY-nderilmi&AV8- Postalar"') if status != "OK": print("Gönderilmiş klasör seçilemedi.")

1. Kodumuzu değiştirdik. Çerçevesini daha net çizip en baştan kod oluşturduk. Gelen kutusu soru, giden kutusu cevap olarak kaydedilsin. Soru cevap çiftleri eşlensin. Spamlar silinsin dedik. Güzel bir ham veri elde ettik.
2. Cevap olarak yollanan ama sorusu olmayanların sorusunu null olarak al dedik.
3. **Aktif kullanılan bir mail adresine ihtiyacımız oldu.**
4. Github’daki kodlardan spam temizleme kısmını kendi kodlarımızla birleştirdik.
5. Yeni mail alıp yeni soru cevap verisi oluşturduk.
6. Aynı iki tane soru vardı ondan kaynaklı olabilir.
7. Bu google mesajlarının konu kısımları zaten belirli oluyor bu konu kısımları olanları spam olarak aldık. Ayrıca da kimden kısmında google geçenleri aldık. Temizlemiş veriden sildik. Spam mailleri kısmına kaydettik.
8. İlk önce sayac tarzı bir şey konuldu, terminalde çalıştırdı ve her mailin tarihi hangi yapıda gösterdi. Kullanıcıdan alınan tarihlerle e-postalardaki tarihler aynı formata getirildi. Bu sayede tarih karşılaştırmaları doğru şekilde yapıldı. Böylece tarih filtresi düzgün çalıştı, istenilen aralıkta olan e-postalar çekilip kaydedildi.

(Ek bilgi: Datetime kütüphanesi içindeki timezone Gmail’den gelen Date header’ı genellikle +0300 (Türkiye Saati) gibi offset-aware olarak geliyor. Bunu kullandım.)

1. Hem formdan gelen tarihi, hem e-posta tarihini aynı şekilde sade hâle (zaman dilimi olmadan) getirdik. Böylece Python artık karşılaştırmayı kabul etti ve mailler sorunsuz gelmeye başladı.
2. **Önce** dosyadaki yapıları orijinal haliyle sözlüğe eklendi. Sonra o kısmın başladığı ortak ifade içeriklerin içinde aranıp bulunduysa: O ifadeden **sonrası silindi**, Ve **aynı içeriklerin üzerine yazıldı**.
3. grep -v "@" requirements.txt > requirements.txt bu komut ile çözüldü.