Analiza tipa, strukture i učinkovitosti marketinških poruka poslanih metodom elektroničke pošte

Galić, Martina

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 558

ANALIZA TIPA, STRUKTURE I UČINKOVITOSTI MARKETINŠKIH PORUKA POSLANIH METODOM ELEKTRONIČKE POŠTE

Martina Galić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 558

ANALIZA TIPA, STRUKTURE I UČINKOVITOSTI MARKETINŠKIH PORUKA POSLANIH METODOM ELEKTRONIČKE POŠTE

Martina Galić

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 4. ožujka 2024.

DIPLOMSKI ZADATAK br. 558

Pristupnica:	**Martina Galić (0036525378)**
Studij:	Računarstvo
Profil:	Znanost o podacima
Mentor:	prof. dr. sc. Damir Pintar

Zadatak: **Analiza tipa, strukture i učinkovitosti marketinških poruka poslanih metodom elektroničke pošte**

Opis zadatka:

Elektronička pošta postala je neizostavan alat u marketinškim strategijama zbog svoje brzine, dostupnosti i mogućnosti personalizacije. No, s obzirom na veliku konkurenciju u ulaznom sandučiću korisnika, važno je razumjeti kako struktura i sadržaj ovakvih poruka utječu na njihovu učinkovitost. Boljim razumijevanjem ovakvih strategija poslovni subjekti mogu postići veći udio odgovora, dok se s korisničke strane može postići bolje i učinkovitije upravljanje ovakvim sadržajima kako bi se u konačnici zadržale samo poruke sa interesantnom i relevantnom informacijom. Zadatak rada jest prikupiti i obraditi dovoljno veliki podatkovnih skup marketinških poruka kako bi se dobio uvid u tip i strukturu informacija koje se prenose ovakvim tipom marketinga. Potrebno je provesti iscrpnu eksploratornu analizu ovakvog sadržaja te predložiti sustav klasifikacije koji bi na osnovu osobnih preferenci klasificirao primljene poruke na adekvatni način. Konačno rješenje bilo bi realizirano u obliku programske skripte koja će na osnovu primljenih podataka generirati relevantne zaključke te demonstrirati izvedenu funkcionalnost.

Rok za predaju rada: 28. lipnja 2024.

Sadržaj

Uvod

Elektronička pošta je postala esencijalni alat u marketinškim strategijama različitih poslovnih korporacija, posebno onih koje se oslanjaju na email marketing kako bi povećale prodaju proizvoda i angažirale kupce. Međutim, iste se suočavaju s izazovom u vidu konkurencije koja također šalje email poruke korisnicima. Kako bi postigle željene reakcije, marketinške poruke moraju biti temeljito osmišljene, relevantne i privlačne kako bi privukle pažnju ciljane publike.

U fokusu rada je istraživanje strukture i sadržaja marketinških poruka poslanih putem elektroničke pošte, odnosno kako ti elementi utječu na njihovu učinkovitost. Cilj je dubljim razumijevanjem strategija email marketinga omogućiti marketinškim stručnjacima pećanje stope otvaranja i angažmana korisnika s poslanim email porukama.

U skladu s tim, osnovni zadatak ovog rada je prikupiti i analizirati dovoljno velik skup podataka marketinških poruka kako bi se stekao uvid u njihovu strukturu, tipove informacija koje prenose te učinkovitost. Kroz detaljnu analizu ovakvog sadržaja, cilj je identificirati obrasce i trendove koji se javljaju u marketinškim porukama putem elektroničke pošte te nastaviti kontinuirano prikupljati podatke i pratiti promjene u strategijama slanja email-ova.

Iako je u tekstu zadatka kao način analize navedena eksploratorna analiza, s obzirom da prikupljeni skup podataka neće sadržavati metrike poput open rate ili click rate, fokus će biti na izradu alata uz pomoć kojeg korisnik sam može donijeti zaključke o kvaliteti email marketinga i provesti analizu.

Kao rezultat istraživanja, planira se razviti sustav klasifikacije koji će biti istreniran na temelju već prikupljenih poruka te će se kontinuirano usavršavati kako bi bio prilagođen novim marketinškim strategijama i trendovima. Sustav će omogućiti korporacijama da bolje ciljaju svoju publiku i povećaju učinkovitost svojih email kampanja, dok će istovremeno korisnicima pružiti relevantnije i više personalizirano iskustvo. Nakon toga slijedi razvijanje alata koji je namjenjen marketinškim stručnjacima. Uz pomoć tog alata, stručnjaci će moći analizirati preko tisuću različitih korporacija te dodavati nove željene korporacije.

1. Elektronička pošta u marketinškim strategijama

Elektronička pošta predstavlja ključni alat u marketinškim strategijama poslovnih subjekata širom svijeta. Razumijevanje različitih vrsta email marketinga omogućuje poslovnim subjektima maksimalnu iskoristivost elektroničke pošte kao alata komunikacije kako bi ostvarili svoje marketinške ciljeve.
Email marketing najčešće koriste :

- Poduzeća koja se bave e-trgovinom. Email marketing omogućuje prodaju proizvoda, vraćanje napuštenih košarica i povećanje dugotrajne vrijednosti.
- Mala poduzeća. Koriste email marketing kako bi održali svijesti o robnoj marki i izgradnju odnosa tijekom vremena.
- SaaS (Software as a service) tvrtke. Koriste email marketing primarno kako bi pretplatnici postali korisnici besplatnih probnih razdoblja, a nakon toga korisnici koji plaćaju uslugu.

1.1. Vrste marketinških poruka

Email marketing je ključan alat u digitalnom marketingu, omogućujući tvrtkama da izgrade i održavaju odnos s publikom, predstave nove proizvode i potaknu korisnike na kupnju. Da bi se email marketing iskoristio na najbolji način, važno je razumjeti različite vrste emailova i njihove specifične uloge u marketinškoj strategiji. Postoje četiri glavne vrste email marketinga: transakcijski, promotivni, newsletteri i retencijski (zadržavanje).

1. Transakcijski emailovi

Transakcijski emailovi pružaju važne informacije o specifičnim transakcijama i predstavljaju priliku za daljnje jačanje odnosa s korisnicima te unapređenje ugleda branda. Primjeri transakcijskih emailova uključuju potvrde o kupnji, informacije o isporuci, promjene lozinke, i slično. Osim osnovne informacije, transakcijski emailovi mogu sadržavati personalizirane preporuke proizvoda, zahvalnice ili ponude za dodatne kupnje.

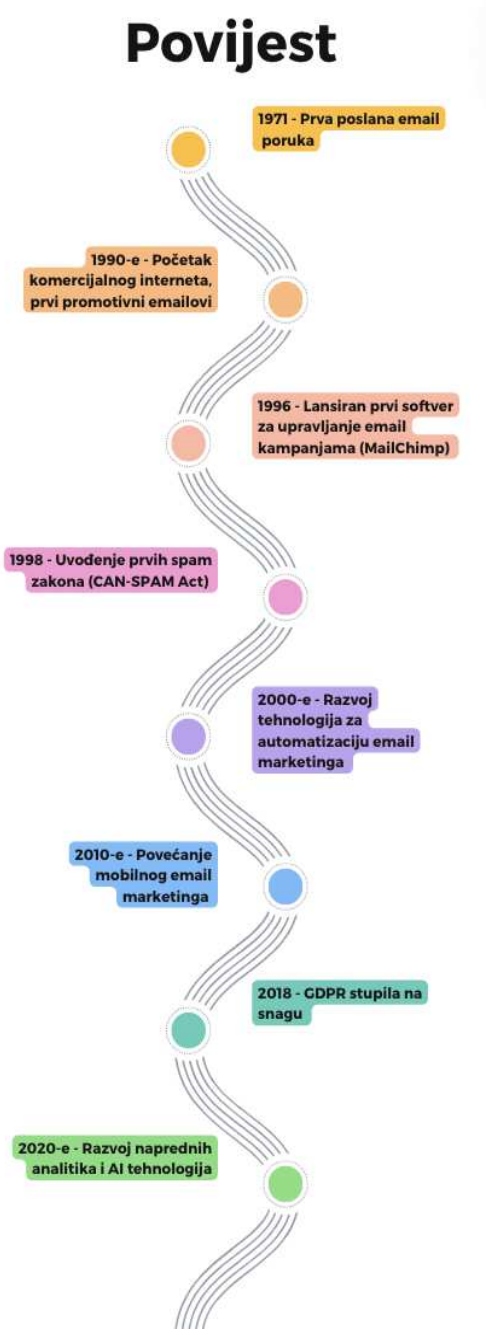
2. Promotivni emailovi

Promotivni emailovi služe za izravno poticanje prodaje i nude popuste, posebne ponude ili ekskluzivne promocije kako bi potaknuli korisnike na kupnju. Idealni su za lansiranje novih proizvoda, sezonske promocije ili ograničene ponude. Dobro osmišljen promotivni email može značajno povećati konverzije i generirati dodatni prihod za tvrtku.

3. Newsletteri

Newsletteri omogućuju redoviti kontakt s publikom te su izvrsni za izgradnju branda i ostajanje u mislima korisnika. Kroz newslettere moguće je dijeliti važne informacije, novosti, blog postove, korisne savjete ili posebne ponude kako bi se korisnici potaknuli na akciju. Kontinuirano dostavljanje relevantnih i vrijednih sadržaja pomaže u jačanju lojalnosti i angažmana korisnika.

1.2. Povijest email marketinga



Slika 1.1 Povijest email marketinga

Smatra se da je email marketing počeo kada su prvi korisnici interneta razmjenjivali poruke elektroničke pošte, no prava revolucija se zapravo dogodila s razvojem komercijalnog interneta 1990-ih godina. Detaljan razvoj email marketinga kroz godine prikazuje Slika 1.1.

U to vrijeme, korporacije su počele koristiti email za slanje promotivnih poruka svojim korisnicima i potencijalnim klijentima, otvarajući put novoj formi direktnog marketinga.

Razvoj tehnologije za automatizaciju email marketinga označio je značajnu prekretnicu u raznim industrijama. Uvođenje softvera za upravljanje email kampanjama omogućilo je poslovnim subjektima da lakše planiraju, implementiraju i prate rezultate svojih marketinških aktivnosti putem elektroničke pošte.

Softveri su kroz godine postali sofisticiraniji i više prilagođeni potrebama marketinških stručnjaka, naročito uvođenjem personalizacije, segmentacija publike i raznih analitika.

1.3. Popularni softveri za email marketing

Mailchimp

Mailchimp je jedan od najpoznatijih alata za email marketing. Omogućuje korisnicima stvaranje, slanje i praćenje email kampanja. Korisnici mogu izraditi predloške email-ova, dodavati personalizirane sadržaje i postaviti agendu za slanje. Mailchimp također omogućuje segmentaciju publike na temelju različitih kriterija poput demografskih podataka ili ponašanja korisnika.

Constant Contact

Constant Contact je još jedan popularan alat za email marketing koji nudi slične funkcionalnosti kao i Mailchimp. Omogućuje izradu personaliziranih email kampanja, segmentaciju publike i praćenje rezultata kampanja. Korisnici mogu odabrati kada žele poslati svoje emailove te prilagoditi sadržaj prema svojim marketinškim ciljevima.

HubSpot

HubSpot je sveobuhvatan alat za automatizaciju marketinga koji uključuje i alate za email marketing. Omogućuje automatizaciju slanja emailova na temelju korisničkog ponašanja, interakcija s web stranicom ili prethodnih kupovina. HubSpot također nudi napredne analitičke alate koji omogućuju detaljno praćenje performansi email kampanja.

Većina korporacija čije mail-ove pratimo i spremamo koriste navedene alate, a ponajviše Mailchimp.

2. Metodologija prikupljanja i obrade podataka

U sklopu drugog poglavlja diplomskog rada ćemo razmotriti isprobane metodologije prikupljanja podataka kao i njihove obrade. Neki od načina nisu bili mogući, odnosno automatizacija nekih postupaka nije bila moguća pa je bilo potrebno odraditi iste ručno što je otežalo cijeli proces.

2.1. Proces registracije i pretplate na newsletter korporacija

Kako bismo skupili dovoljno velik, raznolik i kvalitetan skup podataka u svrhu izrade diplomskog rada, bilo je potrebno na neki način uzrokovati slanje email-ova od strane korporacija. Korporacije marketinške email-ove šalju najčešće nakon što se korisnik registrira ili popuni formu za slanje newsletter-a. Uzmemo li u obzir da želimo raznolik i velik skup podataka, zaključujemo da trebamo obaviti kreiranje jako puno korisničkih računa, konkretno cilj je bio kreirati otprilike 1000 korisničkih računa. S obzirom na količinu posla, pokušat ćemo kreirati automatiziranu registraciju uz pomoć Selenium-a.

2.2. Automatizirana registracija uz pomoć Selenium-a

Prije nego opišemo postupak registracije uz pomoć Selenium-a, ukratko ćemo reći nešto o Selenium-u.

2.2.1. Selenium

Selenium je alat za automatizaciju web preglednika koji omogućuje programerima simuliranje korisničkih interakcija na web stranicama u svrhu testiranja ili automatizacije određenih zadataka. Selenium podržava različite programerske jezike kao što su Python, Java, C#, JavaScript, Ruby, i drugi. U sklopu rada kod je napisan u programskom jeziku Python.

Pomoću Seleniuma, moguće je pisati skripte koje upravljaju web preglednicima kao što su Google Chrome, Mozilla Firefox, Microsoft Edge, itd.

Selenium omogućuje identifikaciju i manipulaciju različitim elementima na web stranicama kao što su tekstualna polja, gumbi, padajući izbornici, slike i drugi.

Glavne svrhe Selenium-a:

- Testiranje web aplikacija:

Jedna od glavnih svrha Seleniuma je automatizacija testiranja web aplikacija. Programeri mogu pisati testne skripte koje automatski provjeravaju funkcionalnosti, izgled i performanse web aplikacija. Ovaj proces osigurava da web aplikacije rade kako treba pri svakoj promjeni u kodu, što doprinosi kvaliteti softvera.

- Automatizacija rutinskih zadataka:

Selenium se koristi za automatizaciju raznih rutinskih zadataka na webu poput automatskog popunjavanja obrazaca, slanja e-mailova, praćenja cijena proizvoda, raspoređivanja sadržaja na društvenim mrežama i mnogih drugih sličnih zadataka. Automatizacija ovih zadataka štedi vrijeme korisniku i smanjuje potrebu za ručnim radom.

- Scraping podataka:

Selenium se koristi za scraping podataka s web stranica. Moguće je pisati skripte koje automatski pretražuju web stranice, izvlače željene informacije i spremaju ih u strukturirani oblik poput CSV, Excela ili baze podataka. Ova funkcionalnost je korisna za istraživanje i analizu web sadržaja.

2.2.2. Izgradnja Selenium automatiziranog softvera za registraciju

Na samom početku izrade diplomskog rada potrebno je prikupiti dovoljno velik i raznolik skup podataka. S obzirom da posjedujemo skup web stranica raznih korporacija, potrebno je posjetiti svaku tu web stranicu i obaviti registraciju ili subscribe. Registraciju obavljamo uz pomoć već opisanog alata Selenium u web pregledniku Google. Prije izgradnje softvera za autoregistraciju, imali smo skup poveznica na web stranice korporacija te smo dodali nastavak `/account/register` na svaku poveznicu kako bi početna točka uvijek bio odjeljak za registraciju. Pri korištenju Seleniuma u web pregledniku Google potrebno je

definirati chrome opcije. U nastavku su prikazane opcije koje koristimo (Kod 2.1) te objašnjenje istih u Tablica 1.

```
chrome_options = ChromeOptions()
chrome_options.add_argument("--headless")
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")
chrome_options.add_argument("--hide-scrollbars")
chrome_options.add_argument("--mute-audio")
chrome_options.add_argument("--accept-lang=en-US")
```

Kod 2.1 Chrome opcije

OPCIJA	OPIS
`--headless`	Pokreće preglednik u headless načinu rada, što znači da se preglednik pokreće bez grafičkog sučelja. Ovo je korisno za automatizaciju u okruženjima bez GUI-a ili kada nije potrebno vidjeti preglednik tijekom izvršavanja skripti.
`--no-sandbox`	Onemogućuje sandboxing mehanizam preglednika. Sandboxing je sigurnosni mehanizam koji izolira procese preglednika kako bi se spriječili potencijalni napadi. Isključivanje sandboxinga može biti potrebno u nekim specifičnim slučajevima automatizacije kada sandboxing uzrokuje probleme.
`--disable-dev-shm-usage`	Onemogućuje korištenje /dev/shm (shared memory) u pregledniku. Korisno u sustavima s ograničenim resursima ili kada /dev/shm nije dovoljno velik za potrebe aplikacije, jer omogućuje pregledniku korištenje diska umjesto RAM-a za privremene datoteke.
`--hide-scrollbars`	Skriva scroll trake u pregledniku. Pomaže u sprječavanju ometanja tijekom snimanja snimki zaslona ili prilikom izvođenja određenih akcija gdje scroll trake nisu potrebne.
`--mute-audio`	Isključuje zvuk u pregledniku. Korisno za automatizirane testove koji uključuju medijske sadržaje, kako bi se spriječilo ometanje zvukom.
`--accept-lang=en-US`	Postavlja preferirani jezik preglednika na engleski (SAD). Ovo je važno za konzistentnost rezultata testiranja kada je potrebno osigurati da jezik sučelja i poruka bude uvijek isti.

Tablica 1 Chrome opcije za automatizaciju i njihovi opisi

Nakon iscrpnog istraživanja kojeg su sve oblika forme registracije, zaključujemo da se najčešće pojavljuju oblici forme koji se nalaze u Kod 2.2, no u kodu su korišteni i još neki drugi oblici.

```
firstname_field= driver.find_element(By.ID,"RegisterForm-FirstName")
lastname_field= driver.find_element(By.ID,"RegisterForm-LastName")
email_field=
driver.find_element(By.ID, "RegisterForm-email")
password_field=
driver.find_element(By.ID, "RegisterForm-password")
```

Kod 2.2 Primjer koda za pronalaženje elemenata obrasca registracije

No ipak najlakše je definirati uz pomoć ID-a polje u koje je potrebno unijeti podatke. Nakon unošenja podataka potrebno je iste i potvrditi.

Naravno, treba biti na oprezu jer postoje i drugi oblici forme te iz tog razloga bilježimo web stranice na koje je registracija bila uspješna, a one za koje nije bila moguća automatizirana registracija, obavljamo to ručno. Uz problem drugačijih oblika forme tu su i drugi problemi poput recaptcha-e i skočnih prozora.

2.3. Proces prikupljanja podataka

Nakon što su obavljene registracije na sve web stranice korporacija, nakon nekog vremena korporacije kreću slati marketinške email-ove na email adresu koja je ostavljena prilikom registracije. U svrhu preuzimanja email pošte koristit ćemo protokol IMAP.

2.3.1. IMAP (imaplib)

IMAP (Internet Message Access Protocol) je standardni protokol za pristupanje e-pošti na poslužitelju. Omogućuje korisnicima pristup e-poštanskom sandučiću na poslužitelju putem različitih klijentskih aplikacija, kao što su web preglednici ili e-poštanski klijenti. Za razliku od protokola POP3 (Post Office Protocol), koji preuzima e-poštu s poslužitelja i čuva je lokalno, IMAP omogućuje korisnicima upravljanje e-poštom izravno na poslužitelju.

Glavne značajke IMAP protokola:

- Centralizirano pohranjivanje - poruke ostaju pohranjene na poslužitelju čak i nakon što ih korisnik pročita ili izbriše s lokalnog uređaja.
- Višestruki uređaji - korisnici mogu pristupiti svojoj e-pošti s više uređaja, a sve promjene (npr. označavanje kao pročitano ili brisanje) sinkroniziraju se na svim uređajima.
- Mapiranje - IMAP podržava hijerarhijsku strukturu mapa, što omogućuje organizaciju e-pošte u mape i podmape prema potrebama korisnika.
- Dohvaćanje samo zaglavlja - moguće je dohvatiti samo zaglavlja e-pošte radi brzog pregleda ili pretraživanja, bez preuzimanja cijele poruke.
- Pretraživanje - IMAP omogućuje napredno pretraživanje e-pošte temeljeno na različitim kriterijima, kao što su pošiljatelj, primatelj, predmet i datum.

Nakon kratkog općenitog opisa IMAP protokola, opisat ćemo imaplib koji ćemo koristiti za čitanje i pohranu podataka.

imaplib je Python biblioteka koja omogućuje komunikaciju s IMAP (Internet Message Access Protocol) poslužiteljima korištenjem Python programa. Imaplib ima razne kriterije odabira koje poruke želimo čitati, neke od njih su pobrojani u Tablica 2.

KRITERIJ	ZNAČENJE
ALL	Sve poruke u mapi.
ANSWERED	Poruke s postavljenom oznakom odgovoreno.
DELETED	Poruke s postavljenom oznakom izbrisano.
UNSEEN	Poruke bez postavljene oznake viđeno.
SEEN	Poruke s postavljenom oznakom viđeno.
SINCE <date>	Poruke čiji je interni datum unutar ili nakon određenog datuma.

Tablica 2 Kriteriji odabira poruke

Kao kriterij čitanja poruka korišten je kriterij UNSEEN, odnosno program čita poruke koje nisu otvorene. Nakon čitanja sadržaja poruke, poruka biva označena kao SEEN tako da svaku poruku čitamo samo jedan put. U sklopu daljnjeg istraživanja na koji način korporacije šalju poruke korisnicima koji ne otvaraju njihove poruke, koristit ćemo kriterij SINCE te nećemo otvarati poruke nego ćemo samo čitati podatke koje je moguće saznati bez otvaranja mail-a. U nastavku je naveden kod uz pomoću kojeg se spajamo na IMAP korištenjem email-a user-a i lozinke (Kod 2.3).

```
imap_url = "imap.gmail.com"

my_mail = imaplib.IMAP4_SSL(imap_url)

my_mail.login(user, password)

my_mail.select("Inbox")

key = "UNSEEN"

_, data = my_mail.search(None, key)
```

Kod 2.3 Spajanje s IMAP API-em

Nakon uspostave veze, u varijabli data su pospremljeni id-evi email-ova koji imaju oznaku UNSEEN. Iteriramo po email id-evima te čitamo poruke i spremamo podatke iz poruke koji su nam bitni u bazu i na S3 bucket (Kod 2.4).

```
_, email_data = my_mail.fetch(email_id, "(RFC822)")

raw_email = email_data[0][1]

msg = email.message_from_bytes(raw_email)

date_str = msg["Date"]

subject = str(msg["Subject"])

from_field = msg.get("From")

sender_name, sender_email = parseaddr(from_field)

sender_email = parseaddr(from_field)[1]
```

Kod 2.4 Dohvat bitnih podataka

Za spremanje podataka u bazi imamo tablice competitor_email_subscribed i competitor_email_tracking. Konkretno u tablicu competitor_email_subscribed pospremamo podatke o korporaciji, dok u competitor_email_tracking tablicu spremamo podatke o samim porukama. U Tablica 3 i Tablica 4 navedeni su izgledi obje tablice u kojima vidimo pojedinosti o stupcima samih tablica.

Competitor_email_subscribed

competitor_id	created_at	profile_created_at
is_subscribed	is_signed_up	email
sender_name		

Tablica 3 Stupci tablice u kojoj su pohranjeni podaci o korporaciji

Competitor_email_tracking

id	competitor_id	created_at
email_created_at	subject_line	bucket_url
email	recipient_email	pdf_size

Tablica 4 Stupci tablice u kojoj su pohranjeni podaci o svakom email-u

Bitno je naglasiti da su u bazi spremljene pojedinosti o poruci, no na S3 spremniku se nalazi cijeli sadržaj poruke pospremljen u pdf-u. Motivacija za spremanje sadržaja poruke u obliku

pdf-a je činjenica da korisnik nakon što iz raznih analiza zaključi da određena korporacija postiže dobre rezultate, može pogledati kako točno izgleda poruka koju korporacija šalje korisniku.

2.3.2. Amazon Web Services (S3)

Amazon Simple Storage Service (Amazon S3) usluga je za pohranu objekata koja nudi skalabilnost, dostupnost podataka, sigurnost i performanse što je čini vodećom u industriji. Korisnici mogu koristiti Amazon S3 za pohranu i zaštitu bilo koje količine podataka za razne slučajeve uporabe, kao što su web stranice, mobilne aplikacije, sigurnosno kopiranje i vraćanje, arhiviranje. Postoji niz klasa pohrane, poput S3 Standard ili S3 Express One Zone za čest pristup, S3 Standard-IA ili S3 One Zone-IA za uštedu troškova pohranjivanjem podataka kojima se rijetko pristupa, S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval i S3 Glacier Deep Archive za arhiviranje podatka po najnižoj cijeni.

U Tablica 5 ispod su detaljnije objašnjenje klase pohrane.

KLASA POHRANE	OPIS	PRIMJENA
S3 Standard	Za podatke kojima se često pristupa; pruža visoku dostupnost i performanse.	Web stranice, mobilne aplikacije, dnevne poslovne operacije.
S3 Standard-IA	Za podatke kojima se rijetko pristupa, ali su potrebni brzo kada im se pristupa.	Sigurnosno kopiranje, dugoročna pohrana podataka.
S3 One Zone-IA	Slično kao S3 Standard-IA, ali pohranjeni podaci su smješteni u jednoj zoni dostupnosti, što smanjuje troškove.	Podaci koji se lako mogu ponovno stvoriti, sigurnosne kopije.
S3 Glacier Instant Retrieval	Arhivska pohrana s niskim troškovima i brzim pristupom podacima.	Dugoročno arhiviranje, pravna ili financijska dokumentacija.
S3 Glacier Flexible Retrieval	Vrlo niskih troškova za arhiviranje podataka uz fleksibilne opcije za dohvaćanje.	Arhiviranje podataka kojima se rijetko pristupa.
S3 Glacier Deep Archive	Najniža cijena za dugotrajno arhiviranje podataka koji se rijetko ili nikada ne dohvaćaju.	Dugotrajno čuvanje podataka, zakonske obveze arhiviranja.
S3 Intelligent-Tiering	Automatski premješta podatke između različitih klasa pohrane kako bi optimizirao troškove na temelju promjena u obrascima pristupa.	Podaci s nepredvidivim ili promjenjivim obrascima pristupa.

Tablica 5 Opis klasa pohrane S3

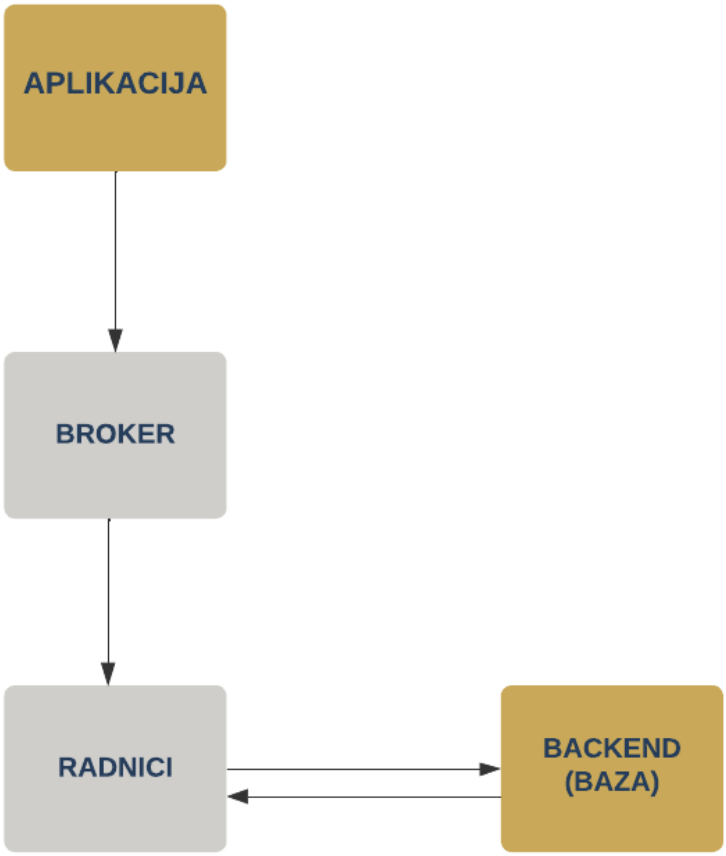
Kao što je već navedeno, pdf dokumenti sa sadržajem poruka su pohranjeni na S3 bucket-u na sljedeći način. Potrebno je prvo dohvatiti sadržaj poruke i to tekstualne i html dijelove. Nakon pretvorbe html sadržaja u pdf, dokument se pohranjuje na S3 bucket kao što je prikazano na Slika 2.5.

```
for part in msg.walk():
    if part.get_content_type() == "text/html":
        payload = part.get_payload(decode=True)
        if payload:
            html_content = html_content + payload.decode(
                part.get_content_charset() or "utf-8"
            )
html_content = f"<div>{html_content}</div>"
pdf_filename = f"{random_uuid}.pdf"

message_pdf = converter.convert(html_content)
object_name = f"competitor_messages/{competitor_id}/{pdf_filename}"
content_type = "application/pdf"
s3.upload_file(io.BytesIO(message_pdf), bucket, object_name, content_type)

logger.info(f"Uploaded file {object_name} to s3")
url = f"https://{bucket}.s3.amazonaws.com/{object_name}"
```

Slika 2.5 Kreiranje pdf datoteka i pohrana iste na S3 bucket

2.3.3. Upravljanje zadacima čitanja i spremanja

Pristizanje i čitanje poruka može potrajati, posebno ako ima puno poruka koje treba obraditi. Iz toga razloga, koristit ćemo Celery, otvorenu Python biblioteku za upravljanje zadacima u pozadini. Celery omogućuje asinkrono izvršavanje zadataka, što znači da se mogu izvoditi izvan glavne programske petlje. To je posebno korisno za dugotrajne, periodične ili ponavljajuće zadatke. Povezanost između aplikacije i Celery-a vidimo na Slika 2.6.



Slika 2.6 Arhitektura sustava koji čini aplikacija i Celery

Kada kod ovisi o vanjskim uslugama, nikada nismo sigurni kada ćemo dobiti rezultate ili čak hoćemo li ih uopće dobiti. Stoga je još jedan razlog za korištenje asinkronih redova zadataka izbjegavanje zastoja prilikom izvršavanja vanjskih API poziva.

U sklopu Tablica 6 objasniti ćemo glavne koncepte vezane uz Celery.

POJAM	OPIS
Radnici (engl. workers)	Procesi koji izvršavaju zadatke. Čekaju nove zadatke u redu čekanja i izvršavaju ih asinkrono.
Zadaci (engl. tasks)	Jedinice posla koje se izvršavaju asinkrono. Definirane su u Python kodu kao funkcije ili metode označene Celery dekoratorom.
Redovi (engl. queues)	Spremnici u kojima se pohranjuju zadaci koje će izvršiti radnici. Celery podržava više redova za organiziranje i prioritizaciju zadataka.
Periodički zadaci	Zadaci koji se izvršavaju periodično u određenim vremenskim intervalima. Korisno za redovite zadatke poput generiranja izvještaja ili čišćenja podataka.

Tablica 6 Glavni pojmovi vezani za Celery

Svaku funkciju u kodu koja želimo da je task moramo označiti Celery dekoratorom poput ovoga: `@celery_app.task(base=DBTask, bind=True)`

Celery dekorator označava da će navedena funkcija biti izvršena asinkrono, odvojeno od glavne programske petlje. `base=DBTask` označava da se koristi DBTask kao osnova za ovaj zadatak, što uključuje dodatnu funkcionalnost definiranu u DBTask klasi. `bind=True` označava da će se prvi argument funkcije vezati na instancu zadatka, što omogućuje pristup metapodacima zadatka putem `self`.

Inicijalizacija Celery aplikacije:

```
celery_app = Celery( "worker", backend = settings.REDIS_URL, broker = settings.BROKER_URL, include = "src.tasks", task_cls = Task )
```

`'worker'` predstavlja ime aplikacije, dok `backend=settings.REDIS_URL` definira URL za pohranu rezultata izvršenih zadataka, koji je u ovom slučaju postavljen na Redis bazu podataka `redis://localhost`. Celery koristi Redis za posredovanje između aplikacije i radnika, odnosno za slanje i primanje poruka između njih. `broker=settings.BROKER_URL` definira URL poslužitelja za poruke, koji se koristi za slanje i primanje poruka između aplikacije i radnika. U našem slučaju, u `BROKER_URL` je pospremljena adresa

`redis://localhost:6379.` `include="src.tasks"` definira putanju do modula koji sadrži Celery zadatke. `task_cls=Task` označava da će se koristiti klasa Task za sve stvorene zadatke.

Također, potrebno je definirati i redove. Konkretno za ovaj problem čitanja mail-ova, kreirat ćemo email-queue. Uz pomoć sljedeće naredbe, potrebno je definirati u koji red idu koji taskovi.

```
celery_app.conf.task_routes = {
   "src.tasks.competitors.*": "competitors-queue",
    "src.tasks.email_tracking.*": "email-queue",
    "src.tasks.common.*": "common-queue" }
```

Naredba za pokretanje worker-a i kreiranje redova je: `celery -A src.tasks worker -l info -Q email-queue, competitors-queue, common-queue`

Iako je velika prednost korištenja Celery-a već navedena, bitno je spomenuti i to da je uz pomoć Celery-a moguće postaviti periodično pokretanje zadataka. S obzirom da mail-ovi stižu svakodnevno i htjeli bismo biti u tijeku s novim mail-ovima, ovakva značajka Celery-a nam je potrebna. Postavili smo pokretanje čitanja mail-ova svako jutro u 6 i 7 sati po UTC-u uz pomoć naredbe u Kod 2.7.

```
schedule = crontab(minute="0", hour="7")
sender.add_periodic_task ( schedule, import_email_data.s( settings.NEW_REGISTERED_EMAIL, settings.NEW_REGISTERED_EMAIL_PASSWORD, EmailType.register),)
```

Kod 2.7 Postavljanje periodičkih zadataka

Na opisani način stvoren je skup podataka koji ćemo koristiti u nastavku za treniranje raznih modela te izradu statističkih izračuna.

2.4. Proces kreiranja skupova za treniranje i testiranje

Kroz vrijeme skupljena je velika količina podataka, no iste je potrebno označiti kako bi bilo moguće trenirati modele. Korištenjem Python-a napisat ćemo skripte koje čitaju podatke iz baze te ih označavaju. Označavanje ćemo obaviti korištenjem Open-AI API gpt-4 modela te ćemo provesti dodano ručno pregledavanje označenih podataka kako bismo spriječili pogrešno označavanje. Da bismo se spojili na Open-AI API potreban je api-key. Unutar poruke upita ćemo objasniti gpt-4 modelu na koji način treba raspodijeliti subject line-ove koje mu šaljemo.

Na opisani način kreirane su datoteke: test_labeled.csv, train_labeled.csv i unlabeled_data.csv. Train_labeled.csv i test_labeled.csv ćemo koristiti za treniranje i testiranje modela, dok je unlabeled_data.csv skup neoznačenih podataka koje će istrenirani model označiti.

U zaglavlju datoteka imamo subject_line, Sales_Discounts_Promotions, Welcome/Confirm, Other što znači da su subject line-ovi sortirani u tri navedene skupine.

3. Sustav klasifikacije poruka

Klasifikacija teksta je tehnika strojnog učenja koja tekstu otvorenog tipa dodjeljuje neku od unaprijed definiranih kategorija. Klasifikatori teksta mogu se koristiti za organiziranje, strukturiranje i kategoriziranje gotovo bilo koje vrste teksta – od dokumenata, medicinskih studija, poruka, datoteka do cijelog weba.

Na primjer, novi se članci mogu organizirati po temama, poruke za podršku po hitnosti, razgovori prema jeziku, e-mailovi prema važnosti, korisnički zahtjevi prema vrsti problema, tweetovi prema relevantnosti, dokumenti prema kategorijama, i tako dalje.

Klasifikacija teksta jedan je od temeljnih zadataka u obradi prirodnog jezika sa širokim primjenama kao što su analiza osjećaja, označavanje tema, otkrivanje neželjene pošte i otkrivanje namjere.

Procjenjuje se da je oko 80% svih informacija nestrukturirano, a tekst je jedna od najčešćih vrsta nestrukturiranih podataka. Zbog neuredne prirode teksta, analiziranje, razumijevanje, organiziranje i sortiranje tekstualnih podataka je teško i dugotrajno, tako da se većinom ne uspijeva iskoristiti u punom potencijalu.

Koristeći klasifikatore teksta, moguće je automatski strukturirati sve vrste relevantnog teksta, od e-pošte, pravnih dokumenata, društvenih medija, chatbota, anketa, na brz i isplativ način. To omogućuje, primjerice tvrtkama, da uštede vrijeme analizirajući tekstualne podatke, automatiziraju poslovne procese i donose poslovne odluke temeljene na podacima.

Neki od glavnih razloga zašto koristiti klasifikaciju teksta uz pomoć strojnog učenja:

Skalabilnost

Ručno analiziranje i organiziranje je sporo i puno manje precizno. Strojno učenje može automatski analizirati milijune anketa, komentara, e-poruka itd., i to jako jeftino, u samo nekoliko minuta. Alati za klasifikaciju teksta prilagodljivi su svim poslovnim potrebama, velikim ili malim.

Analiza u stvarnom vremenu

Postoje kritične situacije koje je potrebno identificirati što prije i odmah djelovati (npr. krize PR-a na društvenim mrežama). Klasifikacija teksta strojnog učenja može neprekidno pratiti

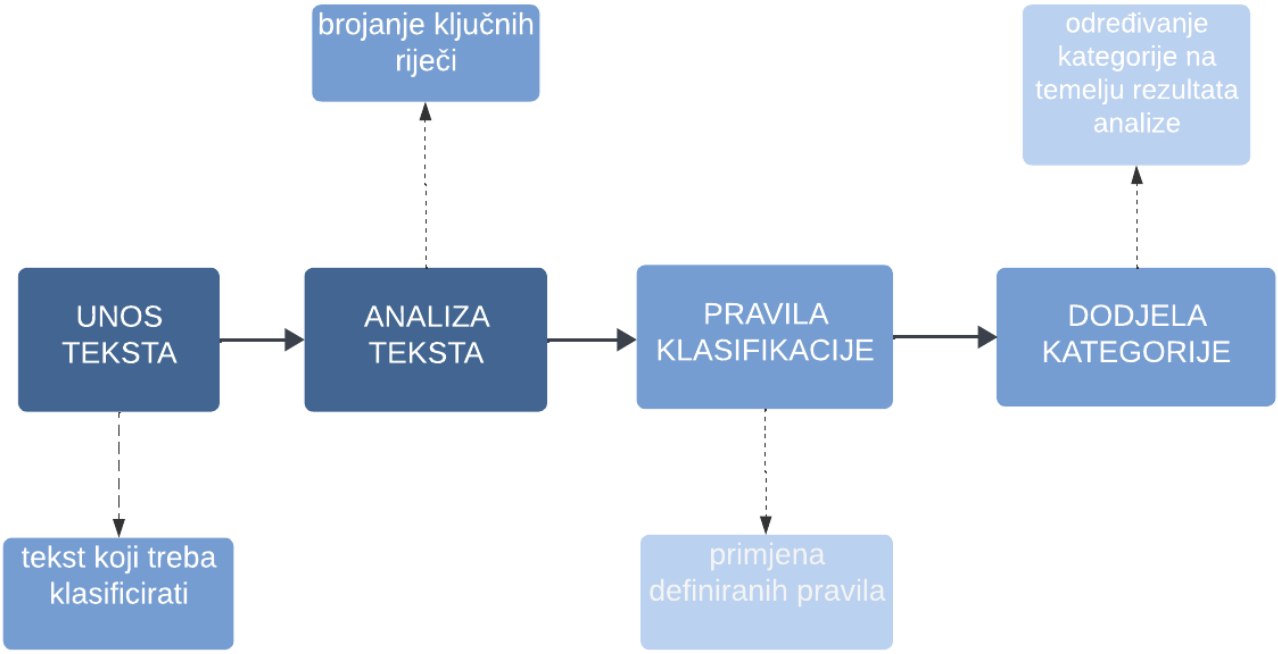
spominjanje brenda u stvarnom vremenu, tako da je moguće identificirati kritične informacije i moći odmah djelovati.

Dosljedni kriteriji

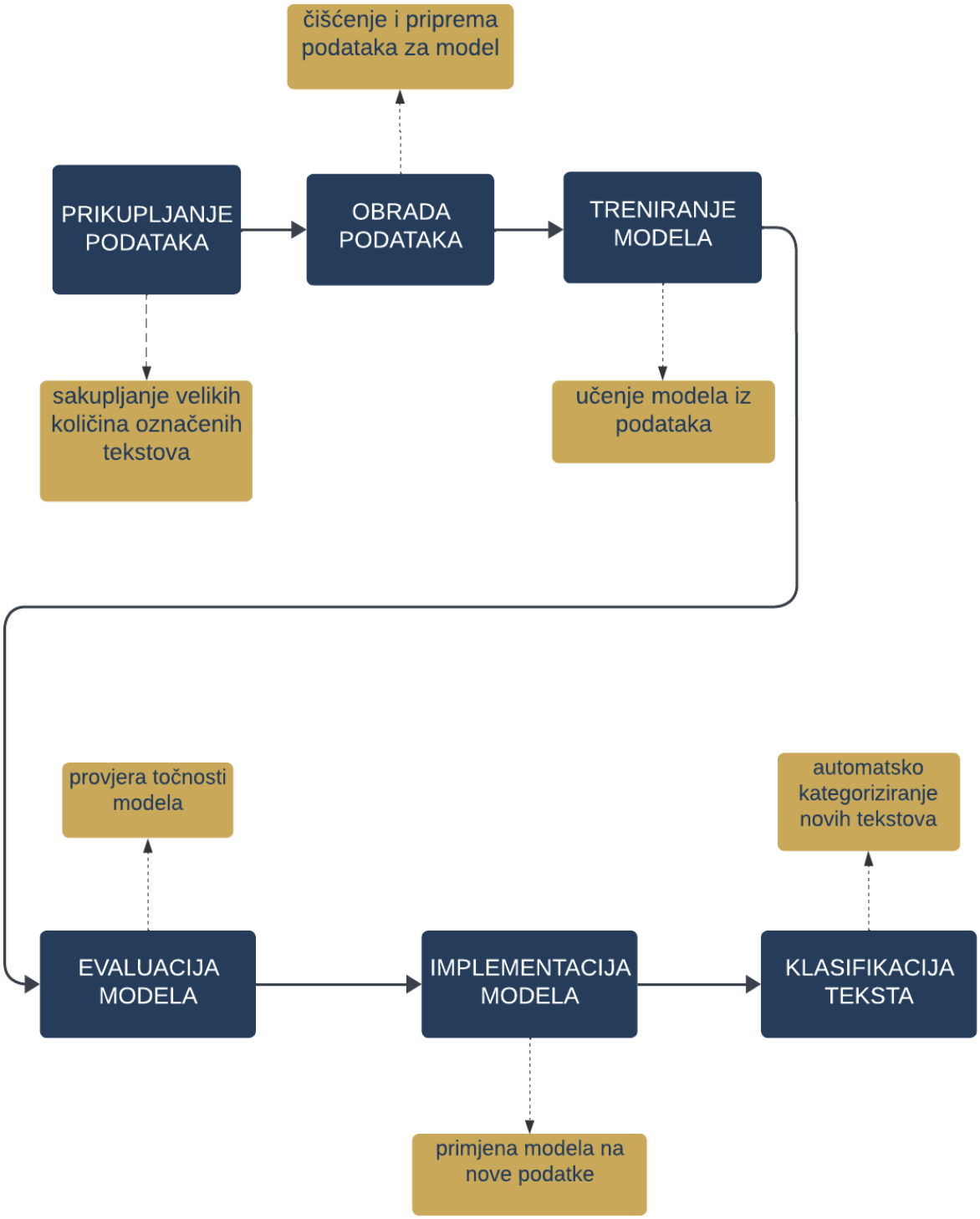
Ljudski anotatori griješe kada klasificiraju tekstualne podatke zbog smetnji, umora i dosade, a ljudska subjektivnost stvara nedosljedne kriterije. Strojno učenje, s druge strane, primjenjuje iste kriterije na sve podatke i rezultate. Jednom kada se model klasifikacije teksta pravilno uvježba, radi s nenadmašnom točnošću.

Postoje tri glavne vrste sustava za automatsku klasifikaciju teksta:

- Sustavi temeljeni na pravilima (Slika 3.1)
- Sustavi temeljeni na strojnom učenju (Slika 3.2)
- Hibridni sustavi



Slika 3.1 Sustav temeljen na pravilima



Slika 3.2 Sustav temeljen na strojnom učenju

3.1. Sustavi temeljeni na pravilima

Sustavi temeljeni na pravilima klasificiraju tekst u organizirane skupine korištenjem skupa ručno izrađenih jezičnih pravila. Pravila upućuju sustav da koristi semantički relevantne elemente teksta kako bi identificirao relevantne kategorije na temelju njegovog sadržaja. Svako pravilo sastoji se od uzorka i predviđene kategorije.

Recimo da novinske članke klasificiramo u dvije skupine: sport i politika. Prvo moramo definirati dva popisa riječi koje karakteriziraju svaku grupu (riječi koje se odnose na sport kao što su nogomet, košarka, LeBron James, itd., i riječi koje se odnose na politiku, kao što su Donald Trump, Hillary Clinton, Putin , itd.).

Kada klasificiramo novi dolazni tekst, brojimo riječi povezane sa sportom koje se pojavljuju u tekstu i riječi povezane s politikom. Ako je broj pojavljivanja riječi povezanih sa sportom veći od broja riječi povezanih s politikom, tada se tekst klasificira kao sport i obrnuto.

Na primjer, sustav temeljen na pravilima će klasificirati naslov "Kada je prva utakmica LeBrona Jamesa s Lakersima?" kao sport jer je pronašao jedan termin vezan uz sport (LeBron James) i nije pronašao pojmove povezane s politikom.

Sustavi koji se temelje na pravilima razumljivi su ljudima i mogu se poboljšati tijekom vremena, ali tu su i nedostaci. Za početak, sustavi temeljeni na pravilima zahtijevaju duboko poznavanje domene. Također oduzimaju puno vremena, budući da generiranje pravila za složen sustav može biti prilično izazovno i obično zahtijeva puno analiza i testiranja. Sustave temeljene na pravilima također je teško održavati i ne skaliraju se dobro s obzirom na to da dodavanje novih pravila može utjecati na rezultate već postojećih pravila.

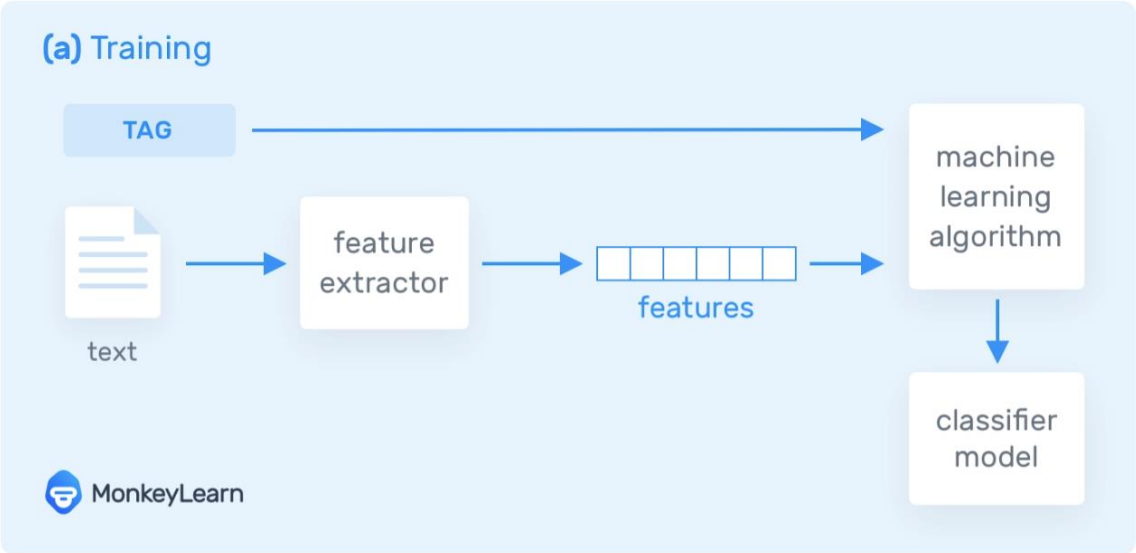
3.2. Sustavi temeljeni na strojnom učenju

Umjesto oslanjanja na ručno izrađena pravila, klasifikacija teksta strojnog učenja nastoji napraviti klasifikacije na temelju prošlih promatranja. Upotrebom unaprijed označenih primjera kao podataka za obuku, algoritmi strojnog učenja mogu naučiti različite asocijacije između dijelova teksta i da se određeni izlaz (tj. oznake) očekuje za određeni ulaz (tj. tekst). "Oznaka" je unaprijed određena klasifikacija ili kategorija u koju može spadati bilo koji tekst. U sklopu ovog rada korištene su tri oznake: Sales_Discounts_Promotions, Welcome/Confirm, Other.

Prvi korak u obučavanju NLP klasifikatora za strojno učenje je izdvajanje značajki: koristi se metoda za transformaciju svakog teksta u numeričku reprezentaciju u obliku vektora. Jedan od najčešće korištenih pristupa je bag of words, gdje vektor predstavlja učestalost riječi u unaprijed definiranom rječniku riječi.

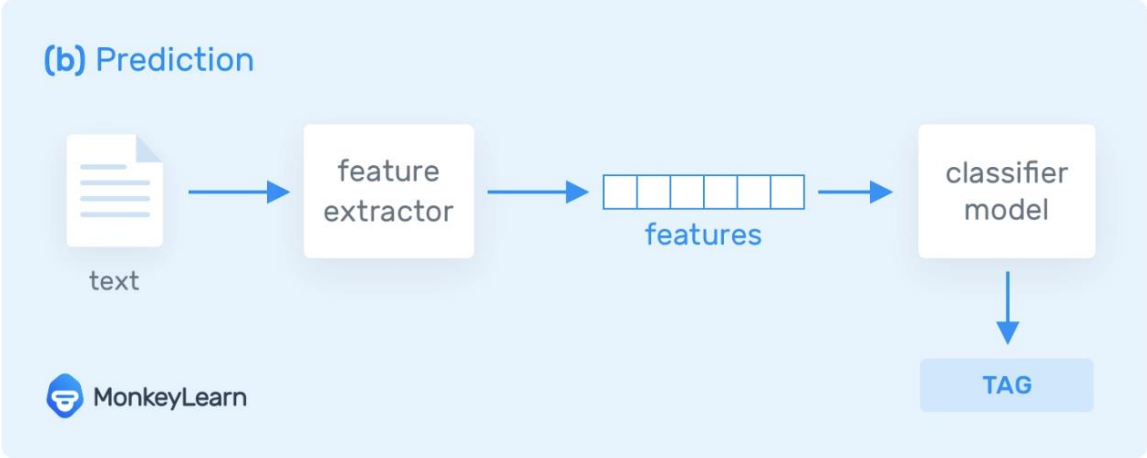
Na primjer, ako bismo definirali rječnik da sadrži sljedeće riječi {This, is, the, not, awesome, bad, basketball}, i htjeli bismo vektorizirati tekst "This is awesome", imali bi sljedeću vektorsku reprezentaciju tog teksta: (1, 1, 0, 0, 1, 0, 0).

Zatim se algoritam strojnog učenja uči podacima koji se sastoje od parova skupova značajki (vektora za svaki tekstualni primjer) i oznaka za izradu modela klasifikacije (Slika 3.3)



Slika 3.3 Treniranje sustava temeljnog na strojnom učenju

Model daje točna predviđanja nakon što se istrenira s dovoljno uzoraka (Slika 3.4). Prilikom predviđanja koristi se se isti ekstraktor značajki za pretvaranje teksta u skupove značajki, koji se mogu unijeti u model klasifikacije da bi se dobila predviđanja oznaka.



Slika 3.4 Proces stvaranja predikcije sustava temeljenog na strojnom učenju

3.2.1. Naive Bayes

Naive Bayes je jedan od najčešće korištenih algoritama u klasifikaciji teksta i analizi teksta. Posebno je popularan zbog svoje jednostavnosti i učinkovitosti, čak i kada se primjenjuje na relativno male skupove podataka. Jedna od najpoznatijih varijanti ovog algoritma je Multinomial Naive Bayes (MNB), koja je vrlo učinkovita u zadacima klasifikacije teksta.

Multinomial Naive Bayes (MNB) model omogućuje postizanje vrlo dobrih rezultata čak i kada skup podataka nije jako velik (~ nekoliko tisuća označenih uzoraka), a računalni resursi su oskudni. Ovaj model temelji se na Bayesovom teoremu, koji omogućuje računanje uvjetnih vjerojatnosti pojavljivanja dva događaja na temelju vjerojatnosti pojavljivanja svakog pojedinačnog događaja.

Osnove Bayesovog teorema

Bayesov teorem koristi se za izračunavanje posteriorne vjerojatnosti događaja na temelju prethodnih (prior) vjerojatnosti i uvjetnih vjerojatnosti. Matematika koja stoji iza toga može se izraziti kao jednadžba na Slika 3.5.

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

Slika 3.5 Bayeosovo pravilo

Gdje je:

P(c|d) - posteriorna vjerojatnost klase c s obzirom na dokument d.

$P(d|c)$ - vjerojatnost dokumenta d s obzirom na klasu c.

$P(c)$ - prior vjerojatnost klase c.

$P(d)$ - vjerojatnost dokumenta d.

Način rada Naive Bayes algoritma

Naive Bayes algoritam pretpostavlja da su značajke međusobno nezavisne unutar svake klase, što je često pojednostavljivanje u stvarnom svijetu, ali se pokazalo vrlo učinkovitim u praksi.

- **Predobrada podataka**

Dokument se najčešće pretvara u Bag of Words (BoW) reprezentaciju, što znači da se dokument izražava kao nesređena zbirka riječi, zanemarujući gramatiku i redoslijed riječi. Svaka riječ predstavlja značajku, a broj pojavljivanja te riječi u dokumentu predstavlja vrijednost te značajke.

- **Klasifikacija teksta**

Za klasifikaciju teksta, izračunavaju se posteriorne vjerojatnosti za svaku klasu pomoću Naive Bayes algoritma. Postupak je sljedeći:

- Procjena **prior vjerojatnosti** klase - procjenjuje se na temelju podataka za treniranje. Na primjer, ako se određena klasa pojavljuje u 30% dokumenata u skupu za treniranje, prior vjerojatnost za tu klasu je 0.3.
- Izračunavanje **uvjetne vjerojatnosti** za svaku riječ - uvjetne vjerojatnosti za svaku riječ u dokumentu s obzirom na svaku klasu izračunavaju se na temelju pojavljivanja riječi u dokumentima te klase u skupu za treniranje.

- Izračunavanje **vjerojatnosti dokumenta za svaku klasu** - množe se uvjetne vjerojatnosti svih riječi u dokumentu i prior vjerojatnost klase kako bi se dobila vjerojatnost dokumenta s obzirom na klasu.
- **Normalizacija** vjerojatnosti - vjerojatnosti se normaliziraju kako bi se dobile posteriorne vjerojatnosti za svaku klasu.
- **Odabir klase** - klasa s najvećom posteriornom vjerojatnošću odabire se kao predikcija za dani dokument.

Objasnimo sve navedeno na kratkom primjeru. Pretpostavimo da imamo zadan sljedeći skup za treniranje (Slika 3.6).

Review	Class
"The movie was amazing!"	Positive
"I didn't like the film."	Negative
"Great acting and an engaging plot."	Positive
"Terrible movie, I would not recommend it to anyone."	Negative

Slika 3.6 Skup za treniranje

Pretprocesiramo tekst tako da ga tokeniziramo u riječi i stvaramo rječnik: ["the", "film", "was", "amazing", "I", "didn’t", "like", "film," " great”, “acting”, “and”, “engaging”, “plot”, “terrible”, “would”, “not”, “recommend”, “it”, “to”, “anyone”].

Zatim konstruiramo BoW prikaz za svaki primjer (Slika 3.7).

Review	BoW Representation	Class
"The movie was amazing!"	[1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	Positive
"I didn't like the film."	[0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]	Negative
"Great acting and an engaging plot."	[0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0]	Positive
"Terrible movie, I would not recommend it to anyone."	[0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1]	Negative

Slika 3.7 BoW prikaz skupa za treniranje

Sada, pretpostavimo da imamo recenziju novog filma: "Acting was amazing!" Ovu recenziju možemo pretvoriti u njenu BoW reprezentaciju: [0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 , 0].

Kako bismo klasificirali ovaj primjer, izračunavamo posteriornu vjerojatnost za svaku klasu pomoću Naive Bayesa. Procjenjujemo prethodne vjerojatnosti klase na temelju podataka za treniranje i izračunavamo vjerojatnost BoW reprezentacije za svaku klasu. Na kraju, normaliziramo vjerojatnosti kako bismo dobili posteriorne vjerojatnosti.

Pretpostavimo da smo dobili vjerojatnosti kao na Slika 3.8

Class	Prior Probability	Likelihood	Posterior Probability
Positive	0.5	0.004	0.002
Negative	0.5	0.000000000002	0.000000000001

Slika 3.8 Rezultat modela nakon treniranja

Na temelju izračuna možemo zaključiti da će nova recenzija vjerojatnije pripadati pozitivnoj klasi. Stoga ga svrstavamo u pozitivnu recenziju.

U radu koristimo algoritam Naivnog Bayesa implementiran u jeziku Python za analizu teksta. Za manipulaciju podacima koristimo biblioteke numpy i pandas, dok za strojno učenje koristimo sklearn. Krajnja vizualizacija podataka je napravljena uz pomoć biblioteka seaborn i matplotlib.

Prvi korak u analizi podataka je podjela skupa podataka na skupove za treniranje i testiranje. To postižemo korištenjem funkcije `train_test_split` iz sklearn.model_selection modula, koja podatke dijeli u omjeru 8:2.

Nakon podjele podataka, pristupamo pretvaranju tekstualnih podataka u numeričku reprezentaciju. To činimo korištenjem CountVectorizer-a iz sklearn.feature_extraction.text modula. CountVectorizer transformira tekst u matricu broja pojavljivanja riječi, što je bitno za daljnju analizu i modeliranje.

Zatim, instanciramo Multinomijalni Naivni Bayes klasifikator i treniramo ga na podacima koristeći funkciju `fit`.

Nakon treniranja, slijedi evaluacija performansi klasifikatora na testnom skupu. Ovdje koristimo metode kao što su precision, recall i F1-score kako bismo procijenili koliko dobro naš model klasificira podatke. Uz pomoć naredbe `classification_report(y_test, y_pred)` vidimo kakve rezultate ima model (Slika 3.9).

	precision	recall	f1-score	support
0	0.89	0.94	0.92	109
1	0.88	0.72	0.79	32
2	0.82	0.80	0.81	56
accuracy			0.87	197
macro avg	0.86	0.82	0.84	197
weighted avg	0.87	0.87	0.87	197

Slika 3.9 Rezultati nakon treniranja Naive Bayes

U prvom slučaju, model Naivnog Bayesa postigao je ukupnu točnost od 87%, što sugerira na solidne performanse u klasifikaciji, no nastavit ćemo s optimizacijom hiperparametara.

Precision je mjera koliko je klasifikator precizan u predviđanju pozitivnih instanci, odnosno omjer stvarno pozitivnih instanci prema svim pozitivnim predikcijama.

Recall je mjera koliko je klasifikator sposoban prepoznati sve stvarno pozitivne instanci, odnosno omjer stvarno pozitivnih instanci prema ukupnom broju pozitivnih instanci u podacima.

F1-score predstavlja harmonijsku sredinu između precisiona i recalla.

Kako bismo poboljšali performanse modela, izvršavamo optimizaciju hiperparametara korištenjem GridSearchCV algoritma kao što smo već i spomenuli. GridSearchCV prolazi kroz različite kombinacije hiperparametara kako bi pronašao najbolje parametre modela koji će rezultirati najboljim performansama.

Kao rezultat optimizacije, dobivamo 1.0 za vrijednost alfe. Možemo vidjeti da su performanse modela ostale gotovo identične u odnosu na performanse prije optimizacije. Preciznost, potpunost i F1-score za svaku od klasa su slični, kao i ukupna točnost modela, koja iznosi 87%. Optimizacija hiperparametra alfa nije utjecala na performanse modela u ovom slučaju. Iako je algoritam pronašao optimalnu vrijednost alfa, rezultati su ostali vrlo

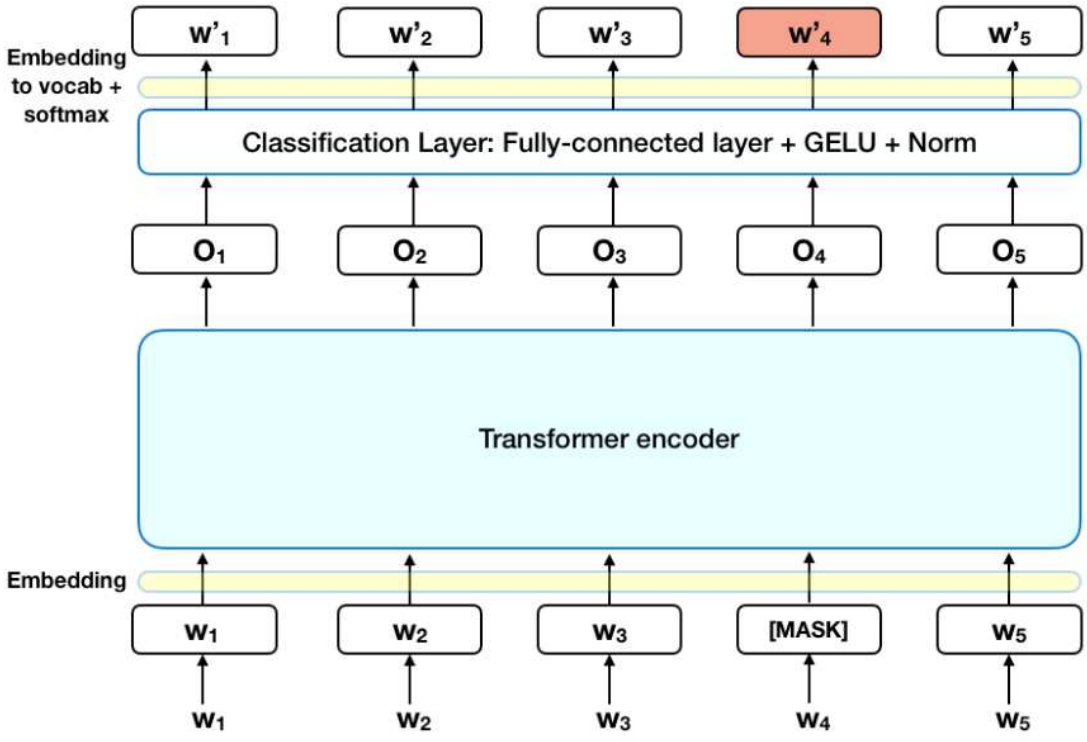
slični, što sugerira da je prethodno odabrana vrijednost alfa već bila prilično dobra za ovaj skup podataka.

3.2.2. BERT

2018. godine, istraživači iz Googlea predstavili su BERT (Bidirectional Encoder Representations from Transformers), model baziran na Transformer arhitekturi, koji je označio veliki napredak u području obrade prirodnog jezika (NLP). BERT je unaprijed obučeni jezični model sposoban za razumijevanje konteksta u jeziku, što ga čini iznimno korisnim za različite zadatke kao što su klasifikacija, prevođenje, odgovaranje na pitanja itd.

BERT se sastoji od niza Transformer enkodera. Svaki enkoder je modularna jedinica koja se sastoji od nekoliko slojeva, gdje svaki ima specifičnu ulogu u obradi ulaznog teksta. Sve navedeno vidimo na Slika 3.10.

Prije nego što ulazni tekst dospije do BERT-a, pretvara se u niz tokena. BERT koristi poseban token za početak teksta (`[CLS]`) koji se dodaje na početak svakog ulaznog primjera, te token za kraj teksta (`[SEP]`) koji se koristi za odvajanje različitih rečenica u istom ulaznom primjeru.



Slika 3.10 Arhitektura BERT modela

Prvi korak u svakom Transformer enkoderu je pretvorba svakog tokena u vektorsku reprezentaciju poznatu kao embeddings. BERT koristi tokenizaciju kako bi tekst podijelio na manje jedinice zvane tokeni. Svaki token se zatim pretvara u vektorski prikaz koji omogućuje modelu da razumije i obrađuje tekst u numeričkom formatu.

BERT koristi WordPiece tokenizaciju, metodu koja omogućuje podjelu riječi na manje dijelove poznate kao subword units ili subtokens. Na primjer, riječ "unhappiness" može se podijeliti na "un" i "##happiness", gdje "##" označava nastavak prethodnog tokena. Ova tehnika omogućuje modelu da se nosi s nepoznatim i rijetkim riječima, jer može podijeliti nepoznate riječi na poznate podtokene.

Kombinacija WordPiece tokenizacije i embeddingsa omogućuje BERT-u fleksibilnost u obradi različitih duljina riječi i rečenica. Bez obzira na duljinu riječi ili rečenica, BERT ih može obraditi jer su riječi razbijene na manje, lako razumljive dijelove. To omogućuje modelu da bolje razumije kontekst i značenje riječi u različitim situacijama.

Jedna od ključnih komponenti Transformer enkodera je self-attention sloj. Self-attention omogućuje BERT-u da istovremeno uzima u obzir sve riječi u rečenici, što je ključno za razumijevanje konteksta. Svaki token ima svoj vlastiti self-attention vektor koji se izračunava na temelju odnosa s drugim tokenima u rečenici.

Nakon self-attention sloja, izlazi se prosljeđuju kroz feed-forward mrežu. Feed-forward mreža se sastoji od nekoliko potpuno povezanih slojeva s aktivacijama ReLU (Rectified Linear Unit), a svaki sloj ima zasebne težine za transformaciju izlaza self-attention sloja.

BERT koristi više takvih Transformer enkodera koji se stapaju jedan na drugi (npr. 12 ili 24 enkodera, ovisno o verziji modela). Svaki dodatni enkoder može učiti sve složenije obrasce u tekstu, doprinoseći ukupnom razumijevanju.

Neke od prednosti BERT modela navedene su u Tablica 7.

PREDNOST	OPIS
Kontekstualno učenje	BERT koristi kontekstualno učenje koje mu omogućuje da bolje razumije značenje riječi u kontekstu rečenice. To znači da dobro primjećuje da riječi mogu imati različito značenje ovisno o okolini u kojoj se pojavljuju.
Dvosmjerna obrada	Dvosmjerni pristup omogućuje BERT-u da gleda i lijevo i desno od svake riječi u rečenici, što mu pomaže da bolje razumije cjelokupni kontekst.
Univerzalnost	BERT je unaprijed obučen na velikom skupu teksta, što znači da može biti prilagođen za različite zadatke s minimalnim finim podešavanjem.

Tablica 7 Prednosti BERT-a

U sklopu ovog rada korišten je unaprijed trenirani BERT model. Nakon učitavanja unaprijed treniranog BERT modela i njegovog tokenizer-a, koristi se model 'bert-base-uncased', a tokenizer i model se inicijaliziraju s 3 klase.

Podaci za trening učitavaju se iz CSV datoteka ,gdje se tekstualni podaci uzimaju iz stupca 'subject_line', a oznake iz ostalih. Oznake se transformiraju u jedinstvene kategorije koristeći `np.argmax`.

Podaci se zatim tokeniziraju i pripremaju za unos u BERT model. Kreira se TensorDataset koji sadrži input ID-ove, maske za pažnju i oznake, nakon čega se pravi DataLoader koji omogućava iteraciju kroz podatke u batch-evima od po 16.

Model se trenira koristeći AdamW optimizator i funkciju gubitka unakrsne entropije. Trening se odvija kroz tri epohe, pri čemu se nakon svake epohe ispisuje broj batch-eva i trenutni gubitak. Gubitak se smanjuje kroz epohe, što ukazuje na uspješan trening modela.

Nakon treninga, model se evaluira na test setu učitanom iz 'test_labeled.csv'. Test podaci se tokeniziraju na isti način kao i podaci za trening, i koriste se za evaluaciju modela. Tijekom

evaluacije, model pravi predikcije bez gradijentnog ažuriranja. Predikcije i stvarne oznake se prikupljaju kako bi se izračunala točnost modela.

Rezultati treninga pokazuju progresivno smanjenje gubitka kroz epohe, s vrijednostima gubitka od 0.3319, 0.3061, i 0.1311 za prve, druge i treće epohe. Tijekom evaluacije, stvarne oznake i predikcije su korištene za izračunavanje točnosti, koja je ispisana zajedno s klasifikacijom svakog predmeta iz test seta. Klasifikacije su mapirane na tri kategorije: 'Sales_Discounts_Promotions', 'Welcome/Confirm', i 'Other'. Nakon treniranja točnost BERT modela iznosi 85% što je ipak nešto lošije od Naive Bayes modela. Uzevši sve u obzir, ipak ćemo koristiti Naive Bayes za klasifikaciju poruka. Naive Bayes u ovom slučaju ne samo da daje točnije predikcije nego i puno brže određuje iste.

4. Izrada alata za marketinške stručnjake

Glavni cilj ovog diplomskog rada je razviti alat koji će marketinškim stručnjacima pružiti mogućnost učenja i usavršavanja kroz analizu uspješnih korporacija. Motivacija za izradu rada leži u potrebi za praktičnim rješenjima koja omogućuju temeljitu analizu i usporedbu marketinških strategija različitih korporacija.

U okviru rada, dizajnirat ćemo i implementirati alat koji korisnicima omogućava odabir više korporacija, nakon čega će im biti dostupne sveobuhvatne analize njihovih email marketing kampanja.

Pored analitičkih izvještaja, korisnicima će biti prikazani i grafički prikazi koji će omogućiti vizualnu procjenu uspješnosti odabranih korporacija. Na temelju grafova, korisnik će moći zaključiti koliko je određena korporacija uspješna u svojim marketinškim aktivnostima i predstavlja li kvalitetan primjer iz kojeg mogu učiti i primijeniti stečeno znanje u vlastitim marketinškim kampanjama.

Za razvoj backend dijela aplikacije koristit ćemo programski jezik Python, uz primjenu okvira FastAPI i SQLAlchemy za efikasno rukovanje podacima i kreiranje API-ja. S druge strane, frontend aplikacije bit će izrađen u React-u, čime ćemo osigurati modernu i responzivno korisničko sučelje. Koristit ćemo Docker za pakiranje backend-a i baze podataka, što će omogućiti jednostavniju distribuciju i skalabilnost aplikacije.

4.1. Docker

Docker je platforma za kontejnerizaciju koja omogućava kreiranje, distribuciju i pokretanje aplikacije u izoliranim okruženjima koje zovemo kontejneri. Kontejneri omogućuju pakiranje aplikacija zajedno s njihovim ovisnostima, čime se osigurava konzistentno ponašanje aplikacija neovisno o okruženju u kojem se pokreću.

Docker se sastoji od nekoliko ključnih komponenti koje zajedno omogućuju kreiranje, pokretanje i upravljanje kontejnerima.

Docker Engine je osnovna komponenta, a sastoji se od tri glavna dijela: Docker Daemon, Docker API i Docker CLI.

- Docker Daemon - sluša Docker API zahtjeve i upravlja Docker objektima (slike, kontejneri, mreže i volumeni); može komunicirati s drugim daemonima za upravljanje Docker uslugama.
- Docker API - Docker nudi REST API koji se koristi za interakciju s Docker daemon-om.
- Docker CLI - `docker` je komandna linija koju korisnici koriste za interakciju s Docker daemon-om. Docker CLI šalje API zahtjeve Docker daemon-u kako bi izvršio različite zadatke kao što su izgradnja, pokretanje i zaustavljanje kontejnera. Neke docker CLI naredbe su:
  - `docker ps` - prikaz svih aktivnih kontejnera
  - `docker build -t my-image .` - izgradnja Docker slike iz Dockerfile-a
  - `docker run -d --name my-container my-image` - pokretanje novog kontejnera iz slike
  - `docker stop my-container` - zaustavljanje kontejnera

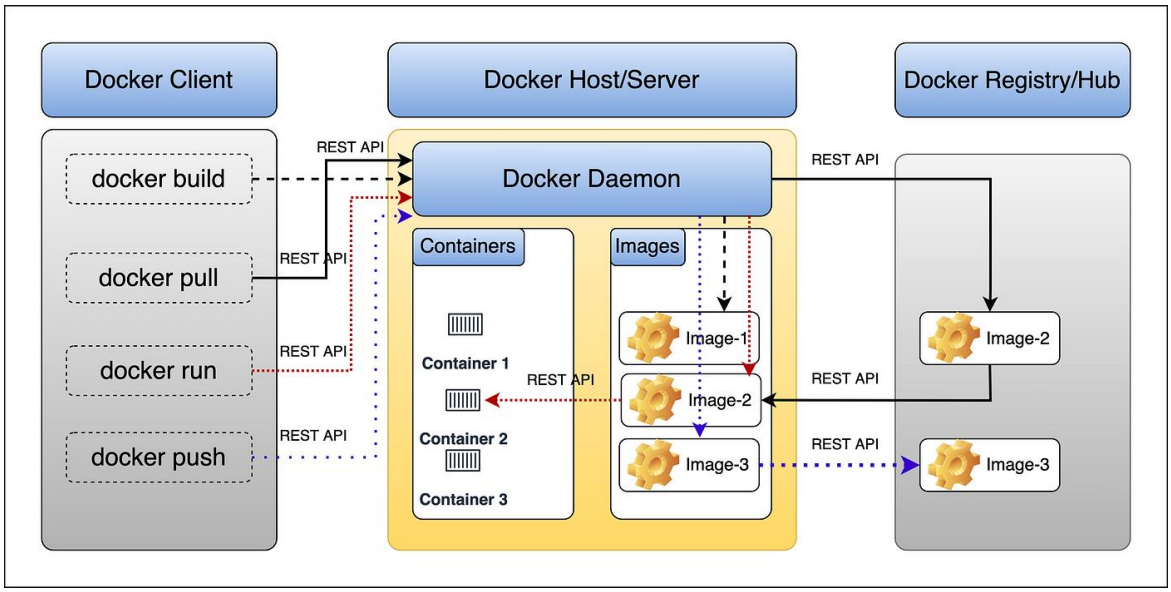
Docker Image je statični predložak koji sadrži sve što je potrebno za pokretanje aplikacije. Docker slika se sastoji od više slojeva, gdje svaki sloj predstavlja promjenu u odnosu na prethodni sloj, kao što je promjena datoteke ili instalacija nove komponente. Kada se slika mijenja, dodaje se novi sloj, dok se postojeći slojevi ne mijenjaju. To omogućuje učinkovito korištenje prostora na disku jer se slojevi dijele između slika.

Dockerfile je skripta koja sadrži niz uputa za kreiranje Docker slike. Svaka uputa u Dockerfile-u stvara novi sloj u slici. Upute uključuju specificiranje baze slike, kopiranje

datoteka, instalaciju paketa i definiranje komandi koje će se izvršiti kada se kontejner pokrene.

Docker Containers su izolirana okruženja koja se kreiraju iz Docker slika. Kontejneri dijele kernel host operativnog sustava, ali su odvojeni od njega i drugih kontejnera. Svaki kontejner ima vlastiti datotečni sustav, mrežu i procesni prostor, što omogućuje konzistentno i predvidljivo ponašanje aplikacija.

Docker Compose je alat za definiranje i pokretanje Docker aplikacija s više spremnika što je upravo naš slučaj. Koristi YAML datoteke za konfiguriranje usluga aplikacije i izvršava proces stvaranja i pokretanja svih spremnika jednom naredbom. Docker-compose CLI omogućuje korisnicima pokretanje naredbi na više spremnika odjednom; na primjer, izrada slika, skaliranje spremnika, pokretanje spremnika koji su zaustavljeni i slično. Datoteka docker-compose.yml koristi se za definiranje usluga aplikacije i uključuje različite opcije. Na primjer, opcija izgradnje definira putanju do Dockerfile-a, nadjačavanje zadanih naredbi Dockera i slično. Docker Volume olakšava neovisnu postojanost podataka, dopuštajući da podaci ostanu čak i nakon što se spremnik izbriše ili ponovno stvori.



Slika 4.1 Glavne komponente Docker-a

Docker Hub je registar na kojem su pohranjene unaprijed izrađene službene slike spremnika za mnoge alate, okruženja, baze podataka i aplikacije. Tako da postoji i službena slika Pythona, raznih baza poput PostgreSQL, MySQL, MongoDB, Redis itd. Sve opisane komponente prikazane su na Slika 4.1.

Korištenjem unaprijed napravljene slike spremnika vrlo je jednostavno kombinirati i koristiti različite alate. Na taj način ćemo kreirati i pokrenuti bazu podataka, Python aplikaciju i povezati ih zajedno putem interne mreže. Svi sustavi za upravljanje spremnicima (kao što su Docker ili Kubernetes) imaju mrežne značajke integrirane u sebe.

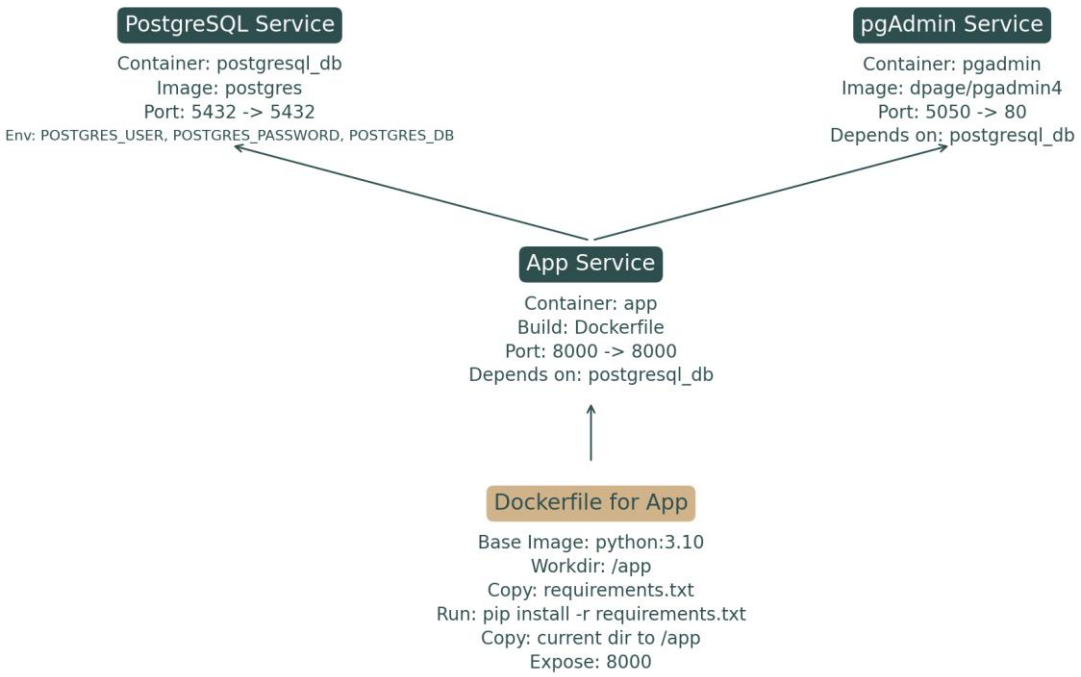
Slika spremnika obično uključuje u svoje metapodatke zadani program ili naredbu koja bi se trebala pokrenuti kada se spremnik pokrene i parametre koji se prosljeđuju tom programu. Vrlo slično onome što bi bilo da pokrećemo iz naredbenog retka. U našem slučaju je to naredba:

```
command: bash -c "uvicorn main:app --host 0.0.0.0 --port 8000 --reload"
```

Kada se spremnik pokrene, pokrenut će upravo tu naredbu koju smo predali kao parametar command. Isto tako potrebno je imati requirements.txt datoteku sa svim paketima koje želimo instalirati i koristiti. Uz korištenje naredbe RUN pip3 install -r requirements.txt unutar Dockerfile-a, instalirat će se svi paketi navedeni u requirements.txt datoteci. Bitno je napomenuti da u Dockerfile-u je dobra praksa prvo kopirati samo requirements.txt, bez koda jer kako je već navedeno, Docker i slični alati koriste unutarnju predmemoriju pri izradi slike. Ako se datoteka nije promijenila od zadnjeg puta izrade slike spremnika, tada će ponovno upotrijebiti isti sloj koji je stvoren zadnji put, umjesto ponovnog kopiranja datoteke i stvaranja novog sloja od nule.

Pri kraju Dockerfilea, kopiramo sav kod. Budući da je to ono što se najčešće mijenja, stavili smo ga na kraj, jer gotovo uvijek, ništa nakon ovog koraka neće moći koristiti predmemoriju. Na taj način smo kreirali Dockerfile te docker-compose.yml.

Kratko ćemo opisati docker-compose datoteku, odnosno Docker kontejnere (Slika 4.2).



Slika 4.2 Grafički prikaz Docker kontejnera aplikacije

Docker-compose datoteka definira konfiguraciju za pokretanje tri servisa koristeći Docker kontejnere: PostgreSQL bazu podataka, pgAdmin alat za upravljanje bazom podataka i aplikaciju. Prvi servis je PostgreSQL baza podataka koja koristi kontejner imena postgresql_db s imageom postgres. Kontejner će se uvijek restartirati ako se zaustavi, a port 5432 na hostu mapira se na port 5432 unutar kontejnera. Varijable okruženja POSTGRES_USER, POSTGRES_PASSWORD i POSTGRES_DB preuzimaju vrijednosti iz varijabli okruženja DB_USER, DB_PASSWORD i DB_NAME.

Drugi servis je pgAdmin, alat za upravljanje PostgreSQL bazom podataka. Kontejner je nazvan pgadmin i koristi image dpage/pgadmin4. Za varijable okruženja vrijedi isto kao i za slučaj kod kreiranja baze podataka. Port 5050 na hostu mapira se na port 80 unutar kontejnera, a servis pgAdmin ovisi o servisu db te će se pokrenuti nakon njega.

Treći servis je aplikacija koja koristi kontejner imena app. App kontejner se gradi koristeći Dockerfile iz istog direktorija i pokreće uvicorn server za FastAPI aplikaciju na portu 8000 s automatskim reloadom. Kod iz trenutnog direktorija mapira se na /app unutar kontejnera, a port 8000 na hostu mapira se na port 8000 unutar kontejnera. App servis ovisi o db servisu te će se pokrenuti nakon njega, a kontejner će se uvijek restartirati ako se zaustavi.

Dockerfile za aplikaciju počinje s baznom slikom python:3.10, što znači da će Docker koristiti Python verziju 3.10 kao osnovu za sliku. Radni direktorij unutar kontejnera postavlja se na /app, što znači da će sve naredne operacije biti izvršene unutar tog direktorija. Fajl requirements.txt, koji sadrži popis Python ovisnosti, kopira se u radni direktorij. Potom se koristi pip za ažuriranje instalacijskog alata i instalaciju svih ovisnosti navedenih u requirements.txt. Nakon instalacije ovisnosti, cijeli sadržaj trenutnog direktorija kopira se u radni direktorij /app unutar kontejnera. Konačno, Dockerfile specificira da će kontejner izložiti port 8000, što znači da će aplikacija unutar kontejnera biti dostupna na tom portu.

Nakon kreiranja spremnika i slika u Dockeru, prelazimo na izgradnju same aplikacije, odnosno frontend-a i backend-a.

4.2. Aplikacija - backend

U projektu su korištene različite tehnologije za izradu backend dijela aplikacije. Glavne tehnologije su Python, FastAPI i SQLAlchemy, dok je za migracije baze podataka korišten alat Alembic.

Alembic je alat za upravljanje migracijama baze podataka u Pythonu. On omogućuje praćenje i primjenu promjena u strukturi baze podataka kroz vrijeme. Glavna svrha Alembica je održavanje baze podataka na ažurnom stanju tijekom evolucije aplikacije. To znači da se kroz migracijske skripte definirane u Alembicu mogu stvoriti, promijeniti ili izbrisati tablice, stupci ili indeksi u bazi podataka. Alembic prati verzije sheme baze podataka i omogućuje lako prebacivanje između različitih verzija baze podataka.

FastAPI je moderan web framework za Python koji omogućuje brzo stvaranje API-ja. On kombinira brzinu i jednostavnost korištenja uz Python tipove podataka, što omogućuje brzo razvijanje sigurnih i skalabilnih web API-ja. FastAPI podržava sve HTTP metode (GET, POST, PUT, DELETE, itd.) i omogućuje definiranje ruta i logike rukovanja zahtjevima pomoću Python funkcija. Sadrži ugrađene alate za validaciju podataka, automatsku dokumentaciju API-ja i upravljanje ovisnostima.

SQLAlchemy je Python SQL toolkit i ORM (Object-Relational Mapping) biblioteka. Ona omogućuje korištenje Python objekata za interakciju s bazama podataka umjesto direktnog pisanja SQL upita. SQLAlchemy olakšava rad s bazama podataka kroz apstrakciju nad SQL-om, što omogućuje lakše održavanje i proširenje aplikacija. On podržava različite vrste baza podataka i omogućuje mapiranje Python objekata na tablice u bazi podataka, čime olakšava manipulaciju podacima. SQLAlchemy pruža visoku razinu fleksibilnosti i performansi, te je široko korišten u industriji za razvoj backend aplikacija.

Backend aplikacije je baziran na kreiranje podataka za grafove za frontend.

Komentirajmo prvo graf koji prikazuje brand awareness trend, odnosno način na koji smo došli do tih podataka i kako je tekla njihova obrada. Konkretno radi se o prikupljanju povijesnih podataka za ključne riječi korporacije koristeći Celery taskove.

4.2.1. Google Ads API - Brand Awareness Graf

Google Ads API (ranije poznat kao AdWords API) je izuzetno koristan alat koji omogućava pristup podacima o Google Ads kampanjama i njihovo upravljanje putem integriranih aplikacija. Pruža širok spektar funkcionalnosti, što korisnicima omogućava automatizaciju zadataka, prilagodbu i optimizaciju oglašivačkih kampanja na učinkovitiji način nego ručnim upravljanjem preko Google Ads sučelja.

Neke od ključnih mogućnosti Google Ads API-a su:

1. Upravljanje kampanjama:

- Kreiranje, ažuriranje i brisanje kampanja, oglasa, grupa oglasa i ključnih riječi.

- Podešavanje proračuna, licitacija i ciljanja korisnika za optimizaciju performansi kampanja.

2. Prikupljanje podataka i izvještavanje:

- Dohvaćanje detaljnih izvještaja o izvedbi kampanja, uključujući metrike poput impresija, klikova, CTR-a (click-through rate) i konverzija.

- Pristup povijesnim podacima za analizu trendova i donošenje informiranih odluka.

3. Ciljanje i segmentacija:

- Postavljanje ciljanih demografskih grupa, lokacija, uređaja i drugih parametara za preciznije oglašavanje.

- Upravljanje publikom i remarketing listama za bolje dosezanje relevantnih korisnika.

Za korištenje API-ja, potrebno je postaviti klijenta (GoogleAdsClient) s odgovarajućim vjerodajnicama i konfiguracijom.

4.2.2. Brand Awareness Graf

Funkcija `import_historical_metrics` je definirana kao Celery zadatak s bazom `DBTask` i koristi se za uvoz metričkih podataka za određene korporacije korištenjem GoogleAdsClient-a. Celery je distribuirani task queue sustav koji omogućava izvršavanje dugotrajnih zadataka u pozadini kao što smo već i opisali. S obzirom na velik broj korporacija u bazi, prvo je potrebno podijeliti u grupe na osnovu vrijednosti varijable

KEYWORD_IDEA_PER_REQUEST. Za svaki blok korporacija kreira se zadatak koji generira metrike te se nakon toga lančano poziva zadatak koji sprema iste. Zadatak se pokreće asinkrono s određenim vremenskim odmakom.

Funkcija `generate_historical_metrics` je definirana kao Celery zadatak s bazom `RetryTask` i koristi se za generiranje povijesnih podataka za zadane ključne riječi.

Funkcija uzima listu tupleova `competitor_keywords`, gdje svaki tuple sadrži ključnu riječ i ID korporacije. Početna ključna riječ je ime korporacije. Ključne riječi se normaliziraju pretvaranjem u mala slova i uklanjanjem vodećih/trailing whitespace znakova. Nazivi korporacije se dalje koriste za generiranje prijedloga ključnih riječi pomoću funkcije `generate_by_kw` iz modula `keyword_ideas`.

```
keywords = list({kw.lower().strip() for kw, _ in competitor_keywords})

kw_ideas = keyword_ideas.generate_by_kw(keywords)
```

Stvara se rječnik `kw_comp_dict` koji mapira svaku ključnu riječ na listu ID-ova konkurenata. Za svaku ključnu riječ iz ulazne liste, ID konkurenta se dodaje u odgovarajuću listu u rječniku.

Zatim iteriramo kroz generirane ideje ključnih riječi. Za svaku ideju provjerava je li njena ključna riječ prisutna u originalnoj listi ključnih riječi, ako jest, spremamo mjesečne volumene pretraga (`monthly_search_volumes`) za tu ključnu riječ. Za svaki period i volumen pretrage, zapisi se dodaju u dataset uz odgovarajuće ID-ove korporacije. Opisani zadatak na kraju vraća generirani dataset te sprema iste u bazu.

Nadalje ćemo objasniti na koji način su kreirane ključne riječi uz pomoć Google Ads API-a. API generira ideje za ključne riječi na temelju zadanih ključnih riječi ili URL-a stranice. Google Ads API klijent je alat koji omogućava integraciju s Google Ads platformom putem programskih sučelja, omogućavajući dohvaćanje podataka, upravljanje kampanjama i analizu ključnih riječi.

Za kreiranje klijenta korištena je funkcija load_ads_client koja na osnovu konfiguracijskih postavki predanih u rječniku kreira istog (Kod 4.3). Rječnik može izgledati ovako:

```
config_dict = {
    "developer_token": settings.GOOGLE_DEVELOPER_TOKEN,
    "client_id": settings.GOOGLE_KEY,
```

```
        "client_secret": settings.GOOGLE_SECRET_KEY,
        "use_proto_plus": settings.GOOGLE_USE_PROTO_PLUS,
        "refresh_token": refresh_token,
    }
    if login_customer_id is not None:
        config_dict["login_customer_id"] = str(login_customer_id)
    res = GoogleAdsClient.load_from_dict(config_dict)
```

Kod 4.3 Kreiranje GoogleAds klijenta

Funkcija generate_keyword_ideas je najvažnija funkcija u ovoj priči. Potreban joj je Google Ads klijent te vrijednosti za parametre kao što su customer_id, location_ids, language_id i page_url.

Korištenjem dobivenih podataka kreirani su endpoint-i koji šalju frontendu podatke za crtanje grafova i bitnih metrika. Prvo imamo endpoint koji analizira volumen pretrage ključnih riječi povezanih s brendom, što je ostvareno kroz `brand_keyword_volume`.

Podaci o volumenu pretrage se procesuiraju tako da se za svaki konkurent stvara rječnik koji mapira periode na odgovarajuće volumene pretrage. Analiza omogućuje praćenje promjena u popularnosti brenda kroz vremenska razdoblja.

Metrike uključuju trenutnu veličinu (current_size) koja predstavlja najnoviji dostupni volumen pretrage za korporaciju. Također, izračunava se mjesečni (MOM) i godišnji (YOY) postotni rast volumena pretrage u odnosu na prethodne periode. Ove metrike pružaju dublji uvid u promjene u popularnosti brenda tijekom vremena.

Uz glavne metrike, tu je i prikaz grafa za koji je potrebno pripremiti podatke. Kreiramo endpoint koji pruža uvid u trendove svijesti o brendu. Algoritam prolazi kroz svaku korporaciju kako bi analizirao volumene pretrage ključnih riječi koje su povezane s njihovim brendom. Svaki konkurent ima svoj skup podataka o volumenima pretrage, koji se bilježe u `brand_keyword_volume`.

Zatim se za svakog konkurenta stvara mapa koja preslikava različite periode na odgovarajuće volumene pretrage. Ova analiza omogućuje dublji uvid u dinamiku svijesti o brendu kroz različita vremenska razdoblja u obliku grafa.

Ove informacije pružaju korisnicima uvid u trendove svijesti o brendu odabranih korporacija tijekom vremena.

4.2.3. Email barplot graf podaci

U prethodnim poglavljima rada smo opisali načine prikupljanja i spremanja podataka iz email inbox-a u bazu koje ćemo sada u nastavku obraditi i poslati frontend-u na prikaz u obliku barplot grafova.
Kreirat ćemo endpoint koji pruža detaljne metrike za grafički prikaz elektroničke pošte, a koristit će podatke iz baze koji su prikupljeni za ovu svrhu. Endpoint vrši analizu podataka o praćenju e-pošte za svakog konkurenta kako bi se generirale metrike za grafički prikaz.

Neke od metrika:

- Frekvencija slanja e-pošte tijekom vremena - analizira se frekvencija slanja e-pošte kako bi se utvrdio prosječni broj e-pošta poslanih tjedno. Također se prate promjene u frekvenciji tijekom vremena.
- Aktivnost slanja e-pošte tijekom dana - podaci se analiziraju kako bi se odredili najaktivniji dani i dijelovi dana za slanje e-pošte što uključuje identifikaciju dana u tjednu i dijelova dana kada je korporacija najviše slala e-poštu.

Bitno je napomenuti da se za generiranje metrika čeka 14 dana prikupljanja poruka kako bismo imali podatke na osnovu kojih je moguće izračunati metrike

Na temelju analize podataka generiraju se sljedeće metrike:

- Prosječni tjedni broj e-pošta
- Najaktivniji dan za slanje e-pošte
- Najaktivniji dio dana za slanje e-pošte
- Vrijeme do dostupnih podataka
- Kontinuirani podaci za graf za sve navedene metrike

Rezultati su strukturirani kao rječnik, gdje svaki konkurent ima svoj identifikator, a pripadajuće vrijednosti su liste metrika za grafički prikaz e-pošte. Metrike pružaju korisnicima uvid u obrasce i trendove u slanju e-pošte korporacija te im omogućuju bolje razumijevanje tržišta i prilagodbu vlastitih marketinških strategija.

Bitno je još napomenuti i postojanje endpoint-a koji kreira vrijednosti za graf koji će prikazivati grupiranje poruka u unaprijed definirane skupine.

Endpoint analizira svaku e-poruku i razvrstava je prema njenom tipu. Poruke se klasificiraju u tri glavne kategorije: "Dobrodošlica/Potvrda", "Prodaja/Popusti/Promocije" i "Ostalo". Nakon što su e-mailovi razvrstani, endpoint broji koliko često je svaka vrsta e-pošte poslana. To stvara statistiku koja pokazuje distribuciju e-pošte prema vrstama.

Konačno, endpoint oblikuje rezultate analize u korisne metrike. Svaka korporacija dobiva set metrika koji prikazuje broj e-pošta po vrstama.

4.3. Aplikacija - frontend

React

React je popularna JavaScript biblioteka koja se koristi za izradu interaktivnih korisničkih sučelja. U uvodu ćemo ukratko objasniti osnovne koncepte React-a koji je korišten za izradu frontend-a aplikacije.

React je open-source JavaScript biblioteka koja je razvijena od strane Facebooka i pruža jednostavan i efikasan način za izgradnju korisničkih sučelja. Osnovna ideja Reacta je razdvajanje korisničkog sučelja na manje komponente, što olakšava upravljanje kompleksnošću aplikacija.

Komponente - centralni koncept u Reactu su komponente, koje su samostalne, modularne i ponovno upotrebljive jedinice korisničkog sučelja. Komponente se sastoje od JavaScript koda koji definira strukturu i ponašanje korisničkog sučelja.

Virtualni DOM - React koristi virtualni DOM kako bi efikasno upravljao ažuriranjem korisničkog sučelja. Umjesto direktnog manipuliranja stvarnim DOM-om, React radi s virtualnim DOM-om, što rezultira bržim ažuriranjem korisničkog sučelja i boljim izvedbom.

JSX je sintaktički šećer koji omogućava pisanje HTML-a unutar JavaScripta. JSX čini kod Reacta lakše čitljivim i lakšim za razumijevanje. Najpoznatiji primjer JSX-a su stanje (**State**) i svojstva (**Props**).

Stanje se koristi za pohranu podataka unutar komponente, dok se svojstva koriste za prenošenje podataka između komponenti. Korištenje stanja i svojstava omogućava dinamičko ponašanje korisničkog sučelja.

Ukratko, React je JavaScript biblioteka koja omogućava izradu interaktivnih korisničkih sučelja. Sa svojim konceptima, kao što su komponente, virtualni DOM i JSX, React olakšava razvoj modernih web aplikacija i unapređuje performanse korisničkog sučelja.

Nakon kratkog opisa React-a, pokazat ćemo sam izgled frontend-a aplikacije te objasniti na koji način je izrađena koja komponenta i koja je njezina svrha.

Na samom početku prilikom učitavanja stranice, dolazi dohvat podataka. Podaci sa gore opisanih endpoint-a su dobiveni uz pomoć sljedeće linije.

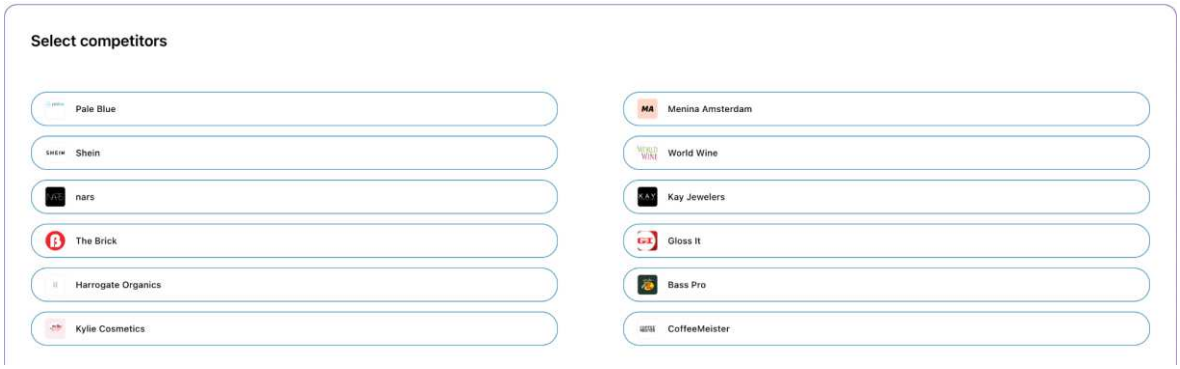
```
const response = await api.get('/email-graph-data/');
```

Prvo što dočeka korisnika nakon učitavanja podataka je popis korporacija na osnovu kojeg korisnik klikom odabire koje korporacije želi analizirati. Primjer komponente u React-u koju je moguće koristiti više puta vidimo u Kod 4.4.

```
export const CompetitorsGrid = styled.div`
  display: grid;
  grid-template-columns: 1fr 1fr;
  column-gap: 80px;
  row-gap: 12px;
  max-height: 42vh;
  overflow: scroll;
  scrollbar-width: none;
`;
```

Kod 4.4 Primjer React komponente

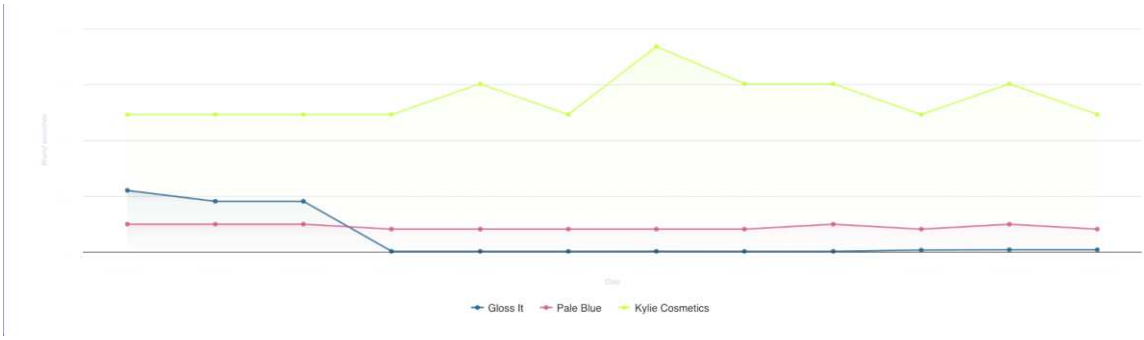
Na Slika 4.5 vidimo i prikaz te komponente. Komponenta ima grid display u stupcima, te je scrollabilna. U komponenti se nalaze druge komponente poput `RecommendedCompetitorCard` u kojoj je pohranjen naziv korporacije te njezina ikonica. Ta komponenta je i klikabilna.



Slika 4.5 Komponenta u kojoj korisnik odabire korporacije

Dakle, nakon učitavanja korporacija, korisnik ima mogućnost odabrati proizvoljan broj korporacija što će uzrokovati prikaz raznih metrika i podataka o toj korporaciji. Prva stvar koju korisnik vidi nakon odabira korporacija jeste graf na osnovu kojeg može zaključiti

koliko je kvalitetan marketing koji reproducira ta korporacija na osnovu metrike koju nazivamo brand searches. Primjer takvog grafa vidimo na Slika 4.6. Korisnik je odabrao tri korporacije te je dobio grafički prikaz kroz vrijeme o pretragama korisnika tog branda. Podaci za ovaj graf su dobivene na osnovu keyword search volumes koje smo opisali u prethodnom poglavlju prilikom opisa backend-a (podaci dobiveni uz pomoć Google Ads API-a).



Slika 4.6 Brand Awareness graf

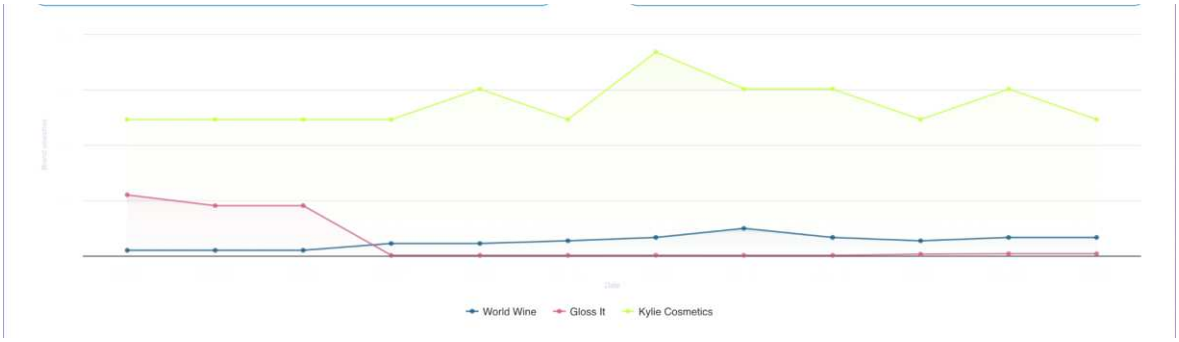
Za prikaz ovakvih grafova korištena je biblioteka Highcharts.

Highcharts je JavaScript biblioteka za vizualizaciju podataka koja omogućava kreiranje interaktivnih grafikona i dijagrama na web stranicama i aplikacijama. Biblioteka je razvijena od strane Highsoft AS. Postala je jedan od najpopularnijih alata za vizualizaciju podataka zbog svoje fleksibilnosti, široke podrške za različite vrste grafikona, te jednostavne integracije s različitim tehnologijama i frameworkovima. Podržava razne vrste grafikona uključujući linijske, stupčaste, tortne, područja, scatter plotove, mape, i mnoge druge. Uz raznolikost u izboru grafova, nudi i bogatu interaktivnost. Korisnik može zumirati, pomicati, i klikati na dijelove grafikona za dodatne informacije. Tooltipovi se automatski pojavljuju kada korisnik prelazi mišem preko grafikona, pružajući dodatne informacije o podacima. Korisnik može mijenjati izgled grafikona, boje, fontove, osi, legende, i druge elemente kako bi grafikon odgovarao dizajnu aplikacije.

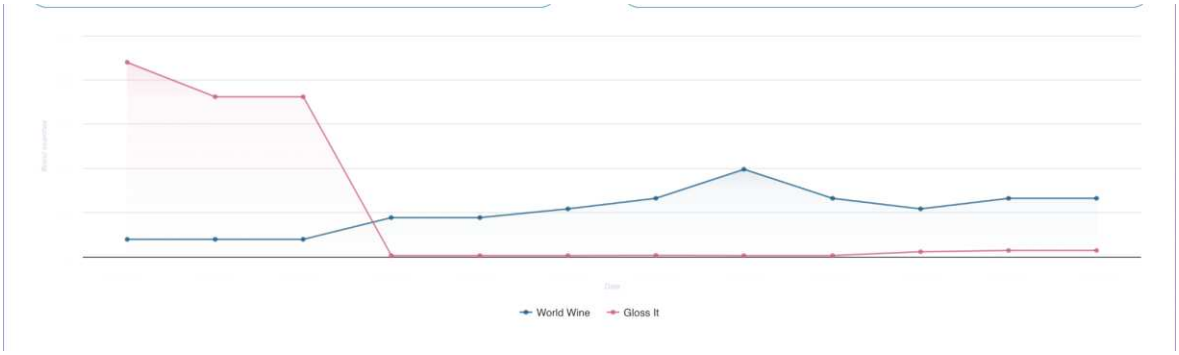
Uz to grafovi su responzivni, što znači da se automatski prilagođavaju različitim veličinama ekrana i uređajima, pružajući optimalno korisničko iskustvo na svim platformama. Važno je napomenuti da je lako rukovati velikim skupovima podataka koristeći highcharts što je jako bitno za ovaj rad. Optimizacije kao što su lazy loading i data grouping omogućavaju glatko

prikazivanje i manipulaciju s velikim količinama podataka. Može se lako integrirati s popularnim frameworkovima kao što su React, Angular, i Vue.js.

Ideja za prvi graf jeste da korisnik može odlučiti koja korporacija ima dobre marketinške poteze. Važno je primijetiti kako se highcharts dobro prilagodi veličinama u podacima. U ovom slučaju je to bitno s obzirom da u ponudi imamo ogromne korporacije, kao i puno manje. Primjetimo na promjenu skale na Slika 4.7 i Slika 4.8 nakon dodavanja korporacije Kyle Cosmetics.



Slika 4.7 Brand Awareness graf s velikom korporacijom



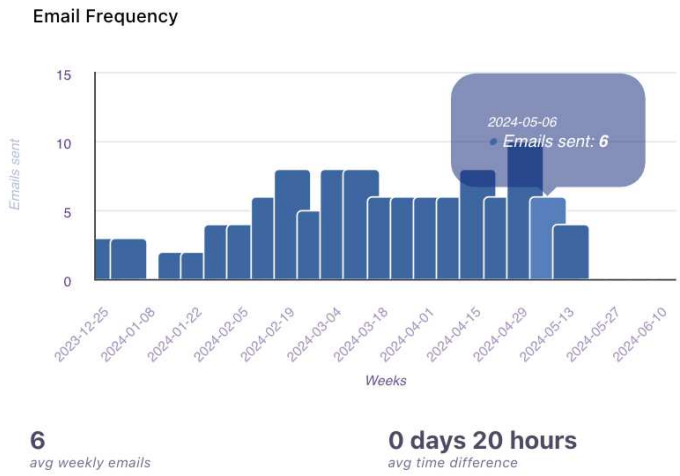
Slika 4.8 Brand Awareness graf bez velike korporacije

Uz prikaz svih korporacija na jednom grafu, korisnik dobiva i za svaku korporaciju po karticu u kojoj je korporacija detaljnije opisana. Prvi graf koji korisnik vidi, a za koji su korišteni podaci kao i za prethodni graf je brand awareness graf sa Slika 4.9.

Slika 4.9 Pojedinačni Brand Awareness graf

U podnožju grafa se ispisuje trenutno stanje svjesnosti o brandu.

Nakon toga slijedi opis email strategije korporacije. Prvi graf koji opisuje strategiju je graf o općenitoj email frekvenciji (Slika 4.10). Omogućeni su i tooltipovi na svim grafovima te pokazuju korisniku informacije koje je nešto teže iščitati iz samog grafa. Označavanjem dijela grafa, graf se zumira kao što je prikazano na slici ispod. Širina grafa je fiksna, stoga se podaci prikazuju po tjednima do unaprijed definiranog perioda, a nakon toga se prikazuju po mjesecima.



Slika 4.10 Graf frekvencije email-ova

Na isti način su izgrađena preostala dva grafa o frekvenciji poruka (Slika 4.11). Sljedeća dva grafa su bazirana na tjedne trendove i trendove povezane s dijelom dana u kojem je poruka poslana kako bi korisnik mogao zaključiti kada korporacija koju proučava smatra da je korisnik najaktivniji i najraspoloženiji za kupnju ili neku drugu željenu akciju. Također, korisnik na osnovu tih informacija može razviti strategiju kako prestići korporaciju te poslati elektroničku poruku korisniku prije suparnika. Ispod grafova se ispisuju metrike povezane s grafom kako bi korisnik mogao brzim pregledom dobiti osnovne informacije.

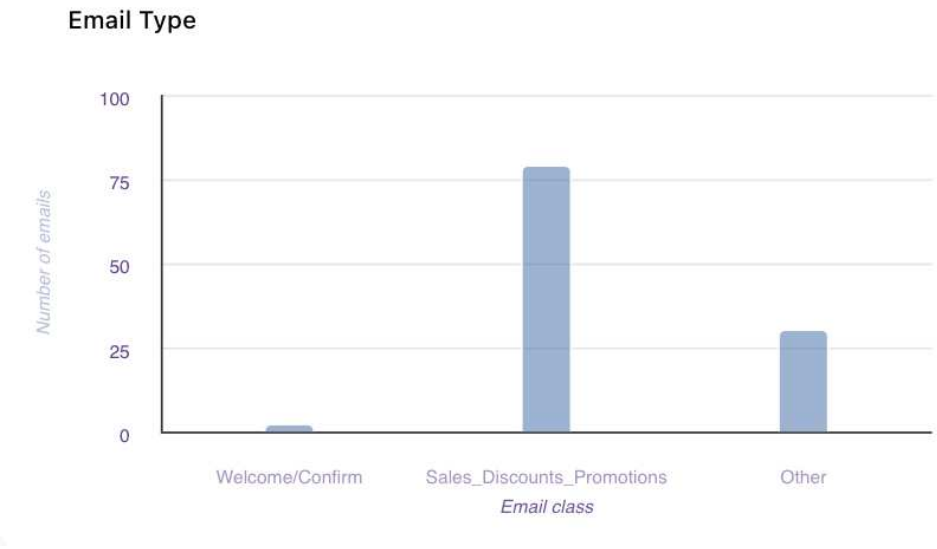
Slika 4.11 Grafovi frekvencije email-ova po danima u tjednu i satima u danu

Podaci koji su prikazani po korporaciji u jednoj kartici su:

- Historical periods - povijesni periodi slanja emailova.
- Historical number of emails - broj emailova po povijesnim periodima.
- Average weekly emails - prosječan broj emailova po tjednu.
- Average time difference - prosječno vrijeme između slanja emailova.
- Day of week (number of emails) - broj emailova po danima u tjednu.
- Top emailing day - dan s najviše poslanih emailova.
- Least active email day - dan s najmanje poslanih emailova.
- Part of day (number of emails) - broj emailova po dijelovima dana.
- Part of day with most emails - dio dana s najviše emailova.

- Part of day with least emails - dio dana s najmanje emailova.
- Hour of day with most emails - sat u danu s najviše emailova.
- Days until data - broj dana do dostupnosti podataka (ako podaci nisu odmah dostupni).

Za sam kraj, nakon svih analiza, kao finalni rezultat prikupljanja podataka, treniranja modela te na kraju označavanja poruka uz pomoć istreniranih modela, prikazujemo podjelu poruka u skupine (Slika 4.12).



Slika 4.12 Graf frekvencije skupina email-ova

5. Motivacija i ideje za daljnji rad

Razvijanje alata za marketinške stručnjake koji omogućuje klasifikaciju i analizu marketinških poruka predstavlja korak naprijed u razumijevanju i optimizaciji marketinških strategija. Međutim, postoji mnogo prostora za daljnji napredak i usavršavanje. U nastavku su opisane neke od ideja za daljnji napredak.

1. Unapređenje trenutnih modela

Jedan od glavnih smjerova za daljnji rad jest unapređenje postojećih modela za klasifikaciju poruka. Iako su BERT i Naive Bayes modeli pokazali dobre rezultate, postoji nekoliko načina kako se njihova točnost i učinkovitost mogu poboljšati:

- Fina prilagodba BERT modela: Dodatno treniranje BERT modela na specifičnim marketinškim porukama ili većim skupovima podataka može poboljšati njegovu sposobnost razumijevanja konteksta u porukama.
- Hibridni modeli: Kombinacija različitih modela (npr. spajanje BERT-a s drugim dubokim modelima ili tradicionalnim metodama strojnog učenja) može pružiti bolje performanse.

2. Implementacija naprednijih modela

Drugi smjer za daljnji rad uključuje istraživanje i implementaciju naprednijih modela koji bi mogli pružiti bolje rezultate u klasifikaciji marketinških poruka:

- Transformers arhitekture: Razmatranje drugih transformatora poput GPT, RoBERTa ili T5 koji su pokazali izvanredne rezultate u zadacima prirodne obrade jezika.
- Sekvencijsko modeliranje: Istraživanje modela kao što su LSTM ili GRU koji bi mogli bolje razumjeti sekvencijske podatke u marketinškim porukama.

3. Proširenje funkcionalnosti alata

Proširenje postojećeg alata s novim mogućnostima može značajno povećati njegovu vrijednost za marketinške stručnjake:

- Analiza ključnih riječi: Identifikacija ključnih riječi i fraza koje su najefikasnije u marketinškim kampanjama.
- Prilagodljivi dashboard: Razvoj prilagodljivog korisničkog sučelja koje omogućuje korisnicima da prilagode analize prema svojim specifičnim potrebama.

Opisana motivacija za daljnji rad je zapravo temelj za buduće istraživanje i razvoj, što može rezultirati još učinkovitijim i više prilagođenim rješenjima za marketinške stručnjake.

Zaključak

U sklopu rada istražena je uloga elektroničke pošte kao ključnog alata u marketinškim strategijama suvremenih korporacija. Analizom različitih vrsta marketinških poruka, povijesti email marketinga i popularnih softvera za email marketing, prikazane su mnogobrojne mogućnosti koje email marketing nudi.

Detaljno je predstavljena metodologija prikupljanja i obrade podataka, uključujući proces registracije i pretplate na newslettere korporacija, automatiziranu registraciju uz pomoć Selenium-a te proces prikupljanja podataka korištenjem IMAP-a i Amazon Web Services (S3). Poseban naglasak stavljen je na postupak kreiranja skupova za treniranje i testiranje, što je omogućilo razvoj preciznog sustava klasifikacije poruka.

Analizirani su sustavi klasifikacije temeljeni na pravilima i strojnom učenju, s posebnim fokusom na Naive Bayes i BERT modele. Na temelju tih analiza, izrađen je alat za marketinške stručnjake koji olakšava klasifikaciju i upravljanje marketinškim porukama.

Važno je napomenuti da, iako je navedeno u tekstu zadatka, eksploratorna analiza nije prikazana u sklopu diplomskog rada. Prilikom definiranja diplomskog zadataka nije predviđeno da skup podataka neće biti dobiven od strane korporacija nego ga mi prikupljamo, stoga nemamo metrike poput open rate, click rate i slično. Iz tog razloga nemamo parametar koji pokazuje koliko je koja poruka kvalitetna pa eksploratorna analiza ne bi imala smisla.

Motivaciju za daljnji rad predstavljaju dodatna istraživanja u području automatizacije prikupljanja podataka, poboljšanja algoritama za strojno učenje te razvoj novih alata koji će omogućiti još učinkovitije upravljanje marketinškim kampanjama.

Literatura

[1] Chelsey Church, *The History of Email Marketing*, Poveznica: https://www.brafton.com/blog/email-marketing/the-history-of-email-marketing/; pristupljeno 4. veljače 2024.

[2] Jonas Fischer, *7 types of emails that'll keep people coming back for more*, Poveznica: https://www.mailerlite.com/blog/types-of-emails/ ; pristupljeno 24. veljače 2024.

[3] Harish Rajora, *What is Selenium WebDriver?*, Poveznica: https://testgrid.io/blog/selenium-webdriver/ ; pristupljeno 14. veljače 2024.

[4] Ankur Dhuriya, *Understanding Celery Workers: Concurrency, Prefetching, and Heartbeats*, Poveznica: https://ankurdhuriya.medium.com/understanding-celery-workers-concurrency-prefetching-and-heartbeats-85707f28c506 ; pristupljeno 9. veljače 2024.

[5] *Text Classification: What it is And Why it Matters*, Poveznica: https://monkeylearn.com/text-classification/ ; pristupljeno 24. svibnja 2024.

[6] Patrizia Orza, *Text Classifiers in Machine Learning: A Practical Guide*, Poveznica: https://levity.ai/blog/text-classifiers-in-machine-learning-a-practical-guide ; pristupljeno 15. svibnja 2024.

[7] Arman Tunga, *Lead Calculation with Rule-Based Classification: Step by Step Guide*, Poveznica: https://medium.com/@armantunga/lead-calculation-with-rule-based-classification-step-by-step-guide-8f341b49a59 ; pristupljeno 8. ožujka 2024.

Sažetak

Naslov:

Analiza tipa, strukture i učinkovitosti marketinških poruka poslanih metodom elektroničke pošte

Sažetak:

U sklopu ovog rada istražena je uloga elektroničke pošte kao ključnog alata u marketinškim strategijama suvremenih korporacija. Analizirane su različite vrste marketinških poruka, povijest email marketinga i popularni softveri za email marketing, čime su prikazane brojne mogućnosti koje email marketing nudi. Detaljno je opisana metodologija prikupljanja i obrade podataka, uključujući automatiziranu registraciju uz pomoć Selenium-a te prikupljanje podataka korištenjem IMAP-a i Amazon Web Services (S3). Poseban naglasak stavljen je na kreiranje skupova za treniranje i testiranje, što je omogućilo razvoj preciznog sustava klasifikacije poruka. Analizirani su sustavi temeljeni na pravilima i strojnom učenju, s fokusom na Naive Bayes i BERT modele. Na temelju tih analiza izrađen je alat koji olakšava klasifikaciju i upravljanje marketinškim porukama.

Ključne riječi:

email marketing, marketinške strategije, automatizacija, strojno učenje, BERT, Naive Bayes

Summary

Title:

Analysis of the Type, Structure, and Effectiveness of Marketing Messages Sent via Email

Abstract:

This study explores the role of email as a key tool in the marketing strategies of modern corporations. Various types of marketing messages, the history of email marketing, and popular email marketing software are analyzed, highlighting the numerous possibilities offered by email marketing. The methodology for data collection and processing is detailed, including automated registration using Selenium and data collection using IMAP and Amazon Web Services (S3). Special emphasis is placed on creating training and testing datasets, which enabled the development of an accurate message classification system. Classification systems based on rules and machine learning, with a focus on Naive Bayes and BERT models, are analyzed. Based on these analyses, a tool was developed to facilitate the classification and management of marketing messages.

Keywords:

email marketing, marketing strategies, automation, machine learning, BERT, Naive Bayes