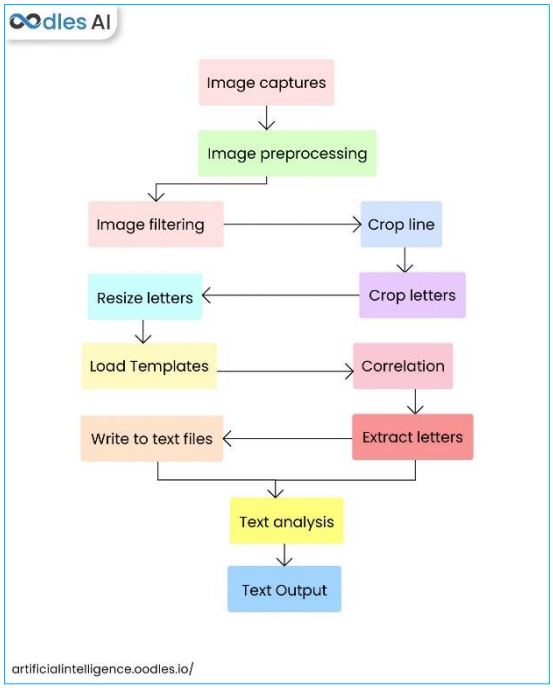
Osnovna šola Ljubečna

OPTIČNO BRANJE IN PRETVORBA BESEDILA IZ SLIK ZA POMOČ SLABOVIDNIM IN DISLEKTIKOM

RAZISKOVALNA NALOGA



Področje: Računalništvo

Avtorica:

Noemi Luiza Flis, 9. a

Mentorica:

Marjeta Gradišnik Mirt, predmetna učiteljica

Mestna občina Celje, Mladi za Celje

Celje, marec 2024

Osnovna šola Ljubečna

OPTIČNO BRANJE IN PRETVORBA BESEDILA IZ SLIK ZA POMOČ SLABOVIDNIM IN DISLEKTIKOM

RAZISKOVALNA NALOGA

Področje: Računalništvo

Avtorica:
Noemi Luiza Flis, 9. a

Mentorica:
Marjeta Gradišnik Mirt,
predmetna učiteljica

Jezikovni pregled:
Mateja Samastur,
prof. slovenščine

Mestna občina Celje, Mladi za Celje

Celje, marec 2024

ZAHVALA

Zahvaljujem se mentorici raziskovalne naloge, gospe Marjeti Gradišnik Mirt, za vso pomoč in usmerjanje pri oblikovanju raziskovalne naloge ter trud, ki ga je vložila v uspešen potek raziskovalnega dela. Zahvaljujem se gospe Mateji Samastur za lektoriranje moje raziskovalne naloge ter gospodu Žigu Čehovin za strokovni pregled. Na koncu se še zahvaljujem gospodu Boštjanu Ketišu za uvodno svetovanje.

Vsebina

Seznam slik in tabel

POVZETEK

Umetna inteligenca je dandanes naš vsakdan. Vsak dan se izboljšuje, že veliko ljudi izumlja nove aplikacije na podlagi AI, s katerimi si lajšamo vsakdanje življenje. V svoji raziskovalni nalogi sem zasnovala aplikacijo za pomoč slepim in slabovidnim pri lažji orientaciji v trgovini. Glavna naloga raziskovalnega dela je bilo ustvariti funkcijo OCR ter funkcijo glasovnega branja (s programiranjem v programskem jeziku python). To sta najpomembnejša procesa v aplikaciji, saj je v le-teh uporabljena umetna inteligenca. Uporabila sem razno programsko opremo – sistemske programe, kot so programske knjižnice, in razvojno programsko opremo, ki omogoča razvoj drugih programov. Skozi eksperimentalno delo sem odkrivala, kako delujejo izbrane programske knjižnice, njihove prednosti in slabosti ter njihovo zanesljivost. Da pa bi sploh lahko izvedla funkciji, sem potrebovala testno fotografijo, katere besedilo bi bilo pretvorjeno v tekst oziroma govor. Pretvorjeno besedilo je rezultat, ki sem ga lahko primerjala z rezultati dela drugih programskih knjižnic in tako določila zanesljivost vsake od izbranih knjižnic glede na kazalnike. Poleg uspešne pretvorbe me je pri knjižnici zanimalo, ali ima slovensko jezikovno področje, torej ali je bil rezultat v slovenskem jeziku. Glede na to, ali sem izvedla funkciji in če sta bili zanesljivi, sem lahko ugotovila, ali lahko pomagam slepim in slabovidnim.

1 UVOD

Ko v trgovini iščemo artikle, prebiramo etikete, na katerih so podatki o artiklu. Pisava na etiketah pa je zelo majhna, podatki so lahko tudi nenatančni in ker se nam s staranjem vid slabša ali pa imamo katero od motenj vida, je tem težje brati in še razumeti, o čem besedilo govori. Tako se nam lahko zgodi, da kupimo artikel, ki ga ne potrebujemo, kupimo napačen proizvod oziroma ga sploh ne najdemo, skratka imamo težave, ki nam lahko povzročajo skrbi.

V današnjih časih si z uporabo umetne inteligence (AI) lahko zelo olajšamo vsakdan. Zdi se mi, da bi bilo z uporabo umetne inteligence lažje tudi prebrati etiketo nekega proizvoda in izvedeti več o njem. Zamislila sem si aplikacijo, ki ima funkcije, s katerimi lahko pretvoriš besedilo iz slike (ga povečaš, deklamiraš ali pa prilagodiš barvo za dislektike oziroma nasploh spremeniš barvo besedila in/ali ozadja), lahko pa tudi vprašaš za več informacij o izdelku, če na primer želiš izvedeti, katero zdravilo potrebuješ za neko bolezen, kakšen odmerek ipd. Večinoma sem se posvetila optični prepoznavi znakov (OCR) oziroma besedila in govorni različici besedila. Cilj moje naloge je, da dokažem, da je možno zasnovati takšno aplikacijo s programiranjem in izvedbo teh ključnih funkcij. Pri delu sem se seznanila tudi s programskim jezikom python, ki sem ga uporabila pri raziskovalnem delu.

1.1 NAMEN

Z izvedbo raziskovalne naloge in potrditvijo ali zavrnitvijo hipoteze bom prikazala, kakšen bi lahko bil lažji način branja in razumevanja besedila na etiketah med iskanjem artiklov za ljudi z motnjami vida, starejše ljudi, navsezadnje pa tudi za otroke in ostale, ki morda prvič sami nakupujejo in imajo težave pri iskanju izdelkov ter branju njihovih etiket. Zastavila sem si raziskovalno vprašanje:

AIi lahko s pomočjo umetne inteligence pomagam ljudem z motnjami vida ter drugim, da se bodo lažje znašli v trgovini?

1.2 HIPOTEZE

Postavila sem si eno delovno hipotezo, ki odgovarja na zgornje raziskovalno vprašanje.

Domnevam, da lahko s pomočjo računalniških algoritmov in umetne inteligence zagotovim slepim in slabovidnim, dislektikom in drugim, da se bodo preko take aplikacije lažje znašli v trgovini.

1.3 METODE DELA

Kot se začne vsako raziskovalno delo, sem tudi sama najprej poiskala informacije in literaturo, ki pa je opisana v teoretičnem delu. Pri svojem raziskovalnem delu sem naredila diagram postopkov, ki jih izvede aplikacija, da pride do končnega produkta, ki

pa je besedilo pretvorjeno v določeno obliko. Vse postopke in ugotovitve sem si skrbno zapisovala.

Seveda pa sem se seznanila tudi z nekaterimi programskimi knjižnicami, ki so mi pomagale pri izvajanju teh dveh funkcij. V tem delu sem se seznanila s programskim jezikom python, ki sem ga uporabila v programu PyCharm (Community Edition). Knjižnice, ki sem jih uporabila, so:

- Pythonova knjižnica pytesseract, ki je vmesnik za program Tesseract OCR, namenjen optični prepoznavi znakov.
- Pythonova knjižnica pillow (PIL), ki Pythonu doda zmožnosti obdelave slik.
- Google Text-to-Speech (gTTS) je Pythonova knjižnica, ki omogoča glasovno branje besedila v mp3 formatu.
- Pythonova knjižnica (pyttsx3), ki omogoča glasovno branje besedila.

Pri eksperimentu sem uporabila tudi telefon, s katerim sem zajela testno fotografijo. V programu sem zapisala kodo, pridobljeno s spletnih strani pytesseract-PyPI in GeeksforGeeks za sklic in izvedbo funkcij. Po nekaj manjših popravkih sem dobila rezultat, vse o poskusu pa bom podrobneje opisala v praktičnem delu.

2 TEORETIČNI DEL

2.1 UMETNA INTELIGENCA (ang. Artificial Intelligence – AI)

Za začetek bom opredelila pojem umetne inteligence, njena področja uporabe in nekaj posledic njene uporabe.

»Umetna inteligenca je ena od tehnologij, ki naj bi bistveno zaznamovale prihodnost. Kaj pravzaprav je umetna inteligenca in kako že vpliva na naša življenja?«

»Umetna inteligenca je zmožnost stroja, da izkazuje človeške lastnosti, kot so mišljenje, učenje, načrtovanje in kreativnost. AI omogoča tehničnim sistemom, da zaznavajo okolje, obdelajo, kar zaznajo, in rešijo problem, pri čemer ravnajo v skladu z določenim ciljem. Računalnik sprejema podatke, ki so predhodno pripravljeni, ali pa jih zbere sam s senzorji, denimo kamero, jih obdela in se odzove. Sistemi, ki delujejo na podlagi umetne inteligence, lahko na podlagi analize učinkov svojih predhodnih dejanj do določene mere samostojno prilagajajo svoje vedenje«. (»Kaj je umetna inteligenca in kako se uporablja v praksi?«, European Parliament, 2020)

»Zakaj je umetna inteligenca pomembna?«

»Nekatere tehnologije umetne inteligence so prisotne že več kot 50 let, napredek v zmogljivosti računalnikov, dostopnost ogromnih količin podatkov in razvoj novih algoritmov pa so v zadnjih letih privedli do velikih prebojev. Umetna inteligenca je prednostna naloga Evropske unije, saj bo imela glede na napovedi ključno vlogo v digitalni preobrazbi gospodarstva in družbe. Novi, še neuporabljeni ali neodkriti načini

uporabe umetne inteligence naj bi povzročili ogromne spremembe, v praksi pa je v naših življenjih močno prisotna že danes.« (European Parliament, 2020)

2.2 OPTIČNA PREPOZNAVA ZNAKOV (ang. Optical character recognition – OCR)

OCR je pri mojem raziskovalnem delu zelo pomemben. Spodaj je opisan njegov namen in delovanje. Besedilo sem skrajšala.

Optično prepoznavanje znakov (OCR) je tehnologija, ki je modernizirala način, kako obdelujemo in shranjujemo dokumente. OCR omogoča strojem, da pretvorijo slike natisnjenega ali ročno napisanega besedila v digitalno besedilo, ki ga računalniki lahko berejo in obdelujejo. Ta tehnologija je omogočila digitalizacijo ogromnih količin informacij, kar olajša iskanje, shranjevanje in analizo podatkov. Ta članek bo raziskal, kako OCR deluje, vključno z algoritmi in tehnikami, ki se uporabljajo za prepoznavanje znakov in njihovo pretvorbo v digitalno besedilo. (»How OCR Works«, b. d.)

Razumevanje OCR

OCR je kompleksen postopek, ki vključuje več stopenj. Na visoki ravni OCR deluje tako, da najprej digitalizira sliko besedila, nato prepozna posamezne znake in nazadnje pretvori te znake v digitalno besedilo. Za dosego tega OCR uporablja kombinacijo prepoznavanja vzorcev, strojnega učenja in algoritmov umetne inteligence. (»How OCR Works«, b. d.)

Skeniranje z OCR

Prvi korak pri OCR je digitalizacija besedila. Skener zajame sliko in ustvari digitalno kopijo dokumenta. Digitalno kopijo nato obdela OCR programska oprema, ki analizira sliko in prepozna posamezne znake. (»How OCR Works«, b. d.)

Prepoznavanje znakov

Ko je slika digitalizirana, OCR programska oprema analizira sliko in prepozna posamezne znake. Algoritmi OCR uporabljajo prepoznavanje vzorcev za prepoznavo oblik in vzorcev znakov. Ta postopek vključuje primerjavo oblik in vzorcev znakov na skenirani sliki z zbirko znanih znakov. Algoritmi OCR lahko prepoznajo tako natisnjeno kot ročno napisano besedilo, čeprav je prepoznavanje rokopisnega besedila zahtevnejše kot prepoznavanje natisnjenega besedila. (»How OCR Works«, b. d.)

Pretvorba znakov

Ko je OCR programska oprema prepoznala posamezne znake na digitalizirani sliki, je naslednji korak pretvoriti te znake v digitalno besedilo. OCR programska oprema uporablja tehniko, imenovano optično pretvarjanje znakov (OCT), da pretvori znake v digitalno besedilo. OCT vključuje ujemanje prepoznanih znakov z njihovimi ustrezajočimi Unicode vrednostmi. Unicode je standard, ki vsakemu znaku v jeziku dodeli edinstveno

kodo. OCR programska oprema uporablja te Unicode vrednosti za pretvorbo prepoznanih znakov v digitalno besedilo. (»How OCR Works«, b. d.)

Natančnost OCR

Natančnost OCR je ključen dejavnik za uspeh tehnologije OCR. Natančnost OCR je odvisna od več dejavnikov, vključno s kakovostjo skenirane slike, zapletenostjo besedila in natančnostjo algoritmov OCR. Natančnost OCR je mogoče izboljšati z uporabo visokokakovostnih skenerjev, izboljšanjem kakovosti slike in uporabo naprednih algoritmov OCR. Za izboljšanje natančnosti OCR lahko OCR programska oprema uporablja algoritme strojnega učenja in umetne inteligence. Ti algoritmi se lahko učijo iz preteklih napak in se prilagajajo novim vzorcem besedila. Algoritmi strojnega učenja lahko tudi pomagajo OCR programski opremi prepoznati rokopisno besedilo bolj natančno, kar je zahtevna naloga zaradi spremenljivosti rokopisa. (»How OCR Works«, b. d.)

2.3 VISION AI

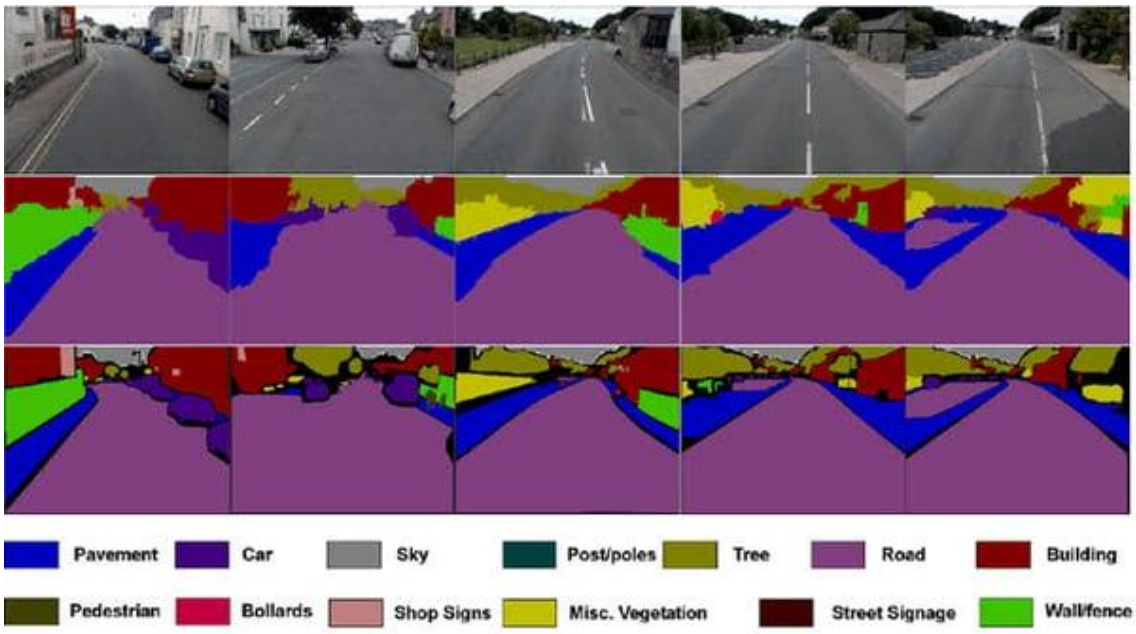
Besedilo sem deloma povzela iz članka Vision AI: What is it and Why does it Matter?

Vision AI (poznan tudi kot računalniški vid) je področje računalniške znanosti, ki izobražuje računalnike, da posnemajo človeški vidni sistem. To omogoča digitalnim napravam (kot so detektorji obrazov, bralniki QR kod) prepoznati in obdelovati predmete na slikah in videih, podobno kot to počnejo ljudje. Prilagojeno iskanje slik v spletnih trgovinah, izdelava 3D-modelov (fotogrametrija), zračne slike na zemljevidu, optično prepoznavanje znakov (OCR) v trgovinah, prepoznavanje obrazov, detekcija slik, rekonstrukcija MRI so nekateri inovativni primeri uporabe računalniškega vida, ki jih imamo danes. (N. Sachdeva, »Vision AI«, b. d.)

Računalniški vid deluje na principu sestavljanke. Sestavljena je iz več kosov, ki skupaj tvorijo celoto – sliko. Nevronske mreže pri računalniškem vidu delujejo na enak način. Le-te razlikujejo med različnimi deli slike in prepoznajo zelo majhne podrobnosti. Namesto namigov za prepoznavo predmeta, računalniku podamo slike, ki mu lahko pomagajo pri natančni prepoznavi predmeta. Če bi računalnik želeli na primer naučiti prepoznati mačko, bi namesto namiga, kot so različni repi, obrazi, oči, v sistem shranili na tisoče ali več slik mačk. Tako ustvarjen model se uči o različnih značilnostih, ki sestavljajo mačko ali jo ločuje od drugih podobnih živali.

Uporaba računalniškega vida AI zahteva nekaj vmesnih postopkov, med katerimi je prvi segmentacija. Je postopek razdeljevanja slike na več regij in delov, glede na lastnosti slikovnih pik v sliki. Navadno se uporablja za pregledovanje, kar vključuje ločevanje ozadja od ospredja ali združevanje delov slike po slikovnih pikah glede na podobnost barve ali oblike.

Spodnja slika prikazuje primer segmentacije slik, kjer so deli slike različno označeni z barvami.



Slika 1: Segmentacija slike

Vir slike: https://insights.daffodilsw.com/blog/vision-ai-what-is-it-and-why-does-it-matter/, dostop 5. 10. 2023

Sledeč postopek vsebuje zaznavanje objektov. To področje računalniškega vida AI se ukvarja z zaznavanjem enega ali več objektov na sliki ali v videu. Na primer, nadzorne kamere pametno zaznavajo ljudi in njihove dejavnosti (brez gibanja, prisotnost predmetov, kot so orožje ali nož itd.), tako da opozorijo na sumljive dejavnosti.

Nato sledi tehnika prepoznavanja obrazov, ki si prizadeva zaznati objekt ali človeški obraz na sliki. Gre za eno od kompleksnih uporab računalniškega vida zaradi različnosti človeških obrazov – izraz, položaj, barva kože, razlika v kakovosti kamere, položaj ali usmerjenost, ločljivost slike itd. Čeprav je precej kompleksna, se ta tehnika uporablja pogosto, na primer v pametnih telefonih, ki jo uporabljajo za identifikacijo uporabnika.

Za tem sledi zaznavanje robov, ki se ukvarja z iskanjem mej objektov na sliki, kar se doseže z zaznavanjem prekinitev v svetlosti. Zaznavanje robov lahko pripomore k pridobivanju podatkov in pri segmentaciji slik.

Sledi še prepoznavanje vzorcev. To je sposobnost sistema za zaznavanje razporeditev značilnosti ali podatkov. Tu je vzorec lahko ponavljajoče se zaporedje podatkov ali nabor podatkov.

Zadnji postopek je klasifikacija slik. Le-ta vključuje razvrščanje slike glede na vizualno vsebino v njej. Postopek vključuje osredotočanje na razmerje med bližnjimi slikovnimi pikami. Klasifikacijski sistem vsebuje bazo podatkov, ki vsebuje predhodno določene vzorce, ti vzorci pa se primerjajo z zaznanim objektom, da se ta določi. Klasifikacija slik

ima pomembne aplikacije na področjih, kot so navigacija vozil, biometrija, video nadzor, biomedicinska slikovna diagnostika, itd. (Povzeto po: N Sachdeva, »Vision AI«, b. d.)

2.4 RAČUNALNIŠKI ALGORITMI

Literaturo sem povzela iz dveh člankov. To sta članka Algoritmi Janeza Demšarja in Understanding algorithms in Computer Science, čigar avtor je International Institute in Geneva.

Algoritem je postopek, kako rešiti nek problem. V računalništvu algoritem pomeni seznam navodil za izvedbo neke naloge oziroma rešitev nekega problema na podlagi razumevanja razpoložljivih variant. Algoritmi niso le programiranje, uporabljajo se za iskanje najboljšega možnega načina za rešitev problema, ti pa temeljijo na specifikacijah za izvajanje računov, obdelavi podatkov, avtomatiziranem razmišljanju ali na primer sprejemanju odločitev. Računalniki brez določenih programov ne bi mogli narediti ničesar. V programiranju obstaja veliko metod reševanja problemov, vendar pa so nekatere bolj učinkovite, saj dajejo natančnejše rezultate. Algoritem lahko uporablja naključne operacije (»izberi naključne elemente zaporedja«), a ne more vsebovati operacij, za katere ni jasno, kako naj bi jih izvedli (»izberi element, ki te bo najhitreje pripeljal do rešitve«, kjer ni jasno navedeno, kateri element je to). Pri opazovanju algoritmov nas najprej zanima, ali so pravilni, torej ali vedno podajo pravi rezultat. Poleg točnih algoritmov so še hevristični. Takšni algoritmi morda ne dajejo pravega odgovora temveč le približek. Če bi radi izvedeli nek rezultat, ki ustreza določenim pogojem, vendar pa bi njegovo iskanje vzelo na primer preveč časa ali pa preveč pomnilnika, bomo navadno zadovoljni že s približkom tega rezultata. Zato je potrebno razumeti, kaj pomeni pravilen algoritem ter obravnavani problem rešiti natančno. Drugo, kar nas zanima pri algoritmih, je njihova hitrost, hitrosti pa ne merimo absolutno, temveč relativno, odvisno od velikosti problema. Če na primer želimo poiskati v tuji hiši, v katerem predalu je nek določen predmet, bomo za to potrebovali več časa, kot pa bi ga za isto nalogo potrebovali doma, saj svoj dom poznamo bolje kot tujega.

V računalništvu poznamo nešteto algoritmov, ti pa se lahko delijo na pomembnejše in ne tako pomembne. Pomembno je vedeti, da so lahko algoritmi izraženi v različnih jezikih, vključno z naravnimi, programskimi, diagrami poteka in podobno. Obdelava informacij in algoritem sta v računalništvu povezana. Podatke lahko preberemo iz vhodnega vira, te zapišemo v izhodno napravo in hranimo za nadaljnjo obdelavo. Podatkovna struktura je oblika za organiziranje, upravljanje in shranjevanje podatkov, ki omogoča dostop in spreminjanje. Je zbirka podatkovnih vrednosti, odnosov med njimi, funkcij oziroma operacij, ki jih je mogoče uporabiti kot podatke. (Povzeto po: J. Demšar, »Algoritmi«, str. 1, b. d.; »Understanding algorithms in Computer Science«, International Institute in Geneva, b.d.)

2.5 TESSERACT OCR, PYTESSERACT IN PILLOW

Tesseract je odprtokodni mehanizem OCR, ki iz slik izvleče natisnjeno ali napisano besedilo. Prvotno ga je razvil Hewlett-Packard, kasneje pa je razvoj prevzel Google, zato

je zdaj znan kot »Google Tesseract OCR«. Toda kaj je odprtokodni OCR? To preprosto pomeni, da je na voljo vsem za prosto uporabo, bodisi neposredno bodisi z uporabo vmesnika za programiranje aplikacij (API). S Tesseract OCR lahko uporabniki izvlečejo besedilo iz slik z učinkovitim vgrajenim in znakovnim prepoznavanjem vzorcev motorja OCR. Do zdaj Tesseract že podpira prepoznavanje jezikov za več kot 100 jezikov »izven škatle«. Najnovejša različica Tesseract (4.0) ima integracijo AI prek nevronske mreže LSTM za boljše zaznavanje in prepoznavanje vnosov različnih velikosti. Ena od velikih prednosti Tesseracta je, da je združljiv s številnimi programskimi jeziki in ogrodji, ki uporabljajo ovoje, kot je pytesseract, znan tudi kot Python-Tesseract. Oglejmo si natančno to povezavo med Tesseract OCR in Pythonom. (»Tesseract OCR«, 2022).

Odprtokodna programska oprema Python OCR

Pytesseract ni samo OCR v Pythonu, odprtokodni programski opremi ali knjižnici Python, ampak služi tudi kot ovoj za Googlov Tesseract OCR Engine. Kar naredi je, da kodo Python ovije okoli Tesseract OCR, kar zagotavlja združljivost in sposobnost delovanja z različnimi strukturami programske opreme. Upoštevati moramo, da obstajajo druge knjižnice in ovoji Python OCR, ki jih je mogoče povezati s Tesseractom, vključno z:

- PYOCR – omogoča več možnosti za zaznavanje stavkov, števk in besed,
- Textract – omogoča ekstrakcijo podatkov PDF za velike datoteke in pakete,
- OpenCV – odprtokodna knjižnica programskih funkcij, ki se osredotočajo na računalniški vid (CV) v realnem času,
- Leptonica – omogoča funkcije obdelave slik in aplikacije za analizo slik s svojo knjižnico slik.
- *Pillow – še ena slikovna knjižnica Python, ki podpira odpiranje, upravljanje in shranjevanje obsežnega seznama formatov slikovnih datotek.* (»Tesseract OCR«, 2022)

Kako deluje (Py)Tesseract?

Zaenkrat vemo, da je pytesseract ovoj za Googlov Tesseract OCR v Pythonu z dodatnimi funkcionalnostmi, ki jih sam Tesseract nima. Katere so torej te funkcije in kako delujejo? Pytesseract se lahko uporablja kot samostojen skript za Tesseract, ki mu omogoča tiskanje prepoznanega besedila, namesto da ga pretvori v datoteko. Pytesseract lahko bere vse slikovne datoteke, ki jih podpirajo knjižnice slik, kot sta leptonica in pillow, vključno z JPEG, PNG, GIF, BMP, TIFF in številnimi drugimi. Zato se pogosto uporablja v primerih uporabe Python OCR za pretvorbo slike v besedilo. Pytesseract deluje tako, da besedilo in grafične elemente skenirane slike pretvori v bitno sliko. Ta bitna slika je preprosto konstrukcija belih in črnih pik. Tako kot pri katerem koli OCR-ju gre slika skozi

fazo predhodne obdelave za prilagoditev svetlosti in kontrasta pred ekstrakcijo podatkov in pretvorbo. Ogrodje pytesseract je optimizirano za boljše zaznavanje jezika, kar koristi tudi Googlovemu Tesseract OCR. Poleg tega je ta okvir odličen pri zaznavanju uporabljenih pisav in orientaciji besedila na vhodni sliki. Ugotovi lahko smer, v katero so postavljene besede in črke v določenem besedilu. Vendar pa je ena njegovih najpomembnejših lastnosti ta, da vam lahko zagotovi informacije o omejevalnem polju OCR. (»Tesseract OCR«, 2022)

Omejitve Tesseracta

Tesseract OCR je lahko zelo uporaben v mnogih primerih uporabe. Vendar pa je, tako kot pri vsaki drugi odprtokodni rešitvi, vedno treba upoštevati nekatere pomanjkljivosti. V tem razdelku bomo eno za drugo osvetlili te omejitve:

- ni tako natančen kot naprednejše rešitve, vgrajene z AI,
- nagnjen je k napakam, če ločitev ospredja in ozadja slike ni pomembna,
- za razvoj lastne rešitve z uporabo Tesseract OCR potrebujete veliko virov in časa,
- sam po sebi ne podpira vseh formatov datotek,
- ne prepozna rokopisa,
- kakovost slike mora doseči določen prag točk na palec (DPI), da deluje,
- treba ga je še naprej razvijati in potrebuje integracijo umetne inteligence, da bi lahko avtomatiziral določene procese dokumentov (npr. preverjanje, navzkrižno preverjanje validacije itd.),
- nima grafičnega uporabniškega vmesnika (GUI), kar pomeni, da ga morate povezati s svojim obstoječim GUI ali ga imeti razvitega.

Na splošno, če je naš primer uporabe OCR preprost in imamo interno znanje o tem, kako razviti rešitve OCR z uporabo Pythona, potem je lahko Googlov Tesseract zadostna rešitev za nas. (»Tesseract OCR«, 2022)

2.6 GOOGLE TEXT-TO-SPEECH IN PYTTSX3

Tukaj sta na kratko predstavljeni Pythonovi knjižnici, ki besedilo pretvorita v govor.

Google Text-to-Speech

gTTS (Google Text-to-Speech) je Pythonova knjižnica in orodje CLI za povezavo z API-jem Google Translate, namenjena pretvorbi besedila v govor. Zapiše izgovorjene podatke v datoteko mp3 formata, datoteki podoben objekt (bytestring) za nadaljnjo obdelavo zvoka ali stdout. Vsebuje prilagodljivo predhodno obdelavo in tokenizacijo. (»gTTS«, P. N. Durette, b. d.)

Pyttsx3

Pyttsx3 je knjižnica za pretvorbo besedila v govor v Pythonu. Za razliko od alternativnih knjižnic deluje brez povezave in je združljiva s Pythonom 2 in 3. Aplikacija prikliče

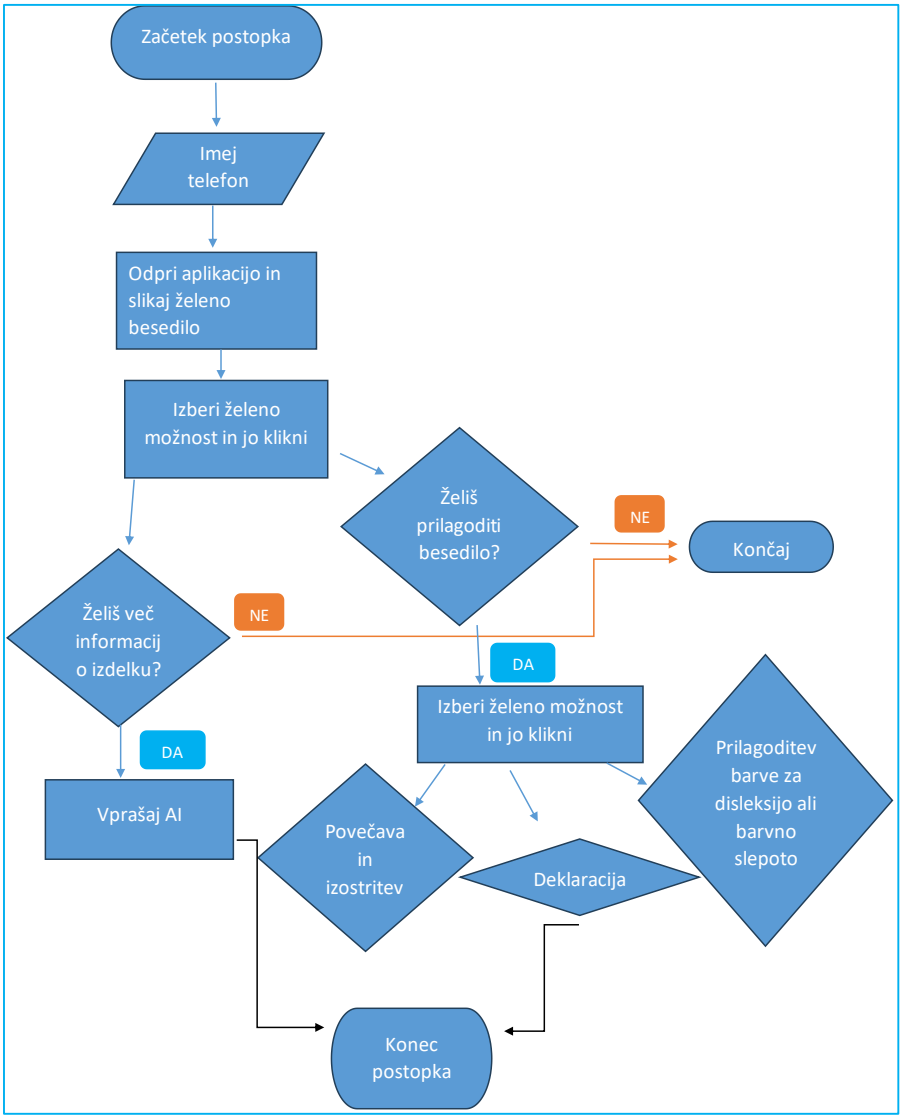
tovarniško funkcijo pyttsx3.init, da dobi sklic na pyttsx3. Je zelo enostavno orodje, ki pretvori vneseno besedilo v govor. (»Python Text to Speech by using pyttsx3«, b. d.)

3 PRAKTIČNI DEL

V tem poglavju bom predstavila svoje eksperimentalno delo, ki zajema postopek le-tega ter opis eksperimenta. Eksperiment je razdeljen na dva dela, in sicer na optično prepoznavo znakov iz besedila ter glasovno branje. Prikazala sem tudi diagram vseh procesov, ki jih izvede aplikacija.

3.1 POSTOPEK DELA

Sprva sem si za lažjo orientacijo naredila splošen diagram postopkov, ki jih izvede aplikacija, nato sem se posvetila najpomembnejšima procesoma, kar sta funkciji pretvorbe besedila in govorna različica besedila.



Slika 2: Diagram postopkov aplikacije

Vir: osebni arhiv

Nato sem zajela testno fotografijo, iz katere sem v nadaljnjem procesu s pomočjo umetne inteligence prepoznala besedilo ter ga pretvorila v govor.

Slika 3: Testna fotografija

Vir: osebni arhiv

Za poskus sem uporabljala že omenjen program PyCharm (Community edition). Naredila sem nov projekt kot bazo, v katero bom pisala.

Pred začetkom dela sem si morala naložiti omenjene knjižnice in program Tesseract OCR. Brez teh ne bi mogla sklicati funkcij, ki so potrebne za nastanek kratkega programa, s katerim bi bila izvedena funkcija aplikacije.

3.2 EKSPERIMENT

Eksperiment sem razdelila na dva manjša projekta oziroma programa, da sem ju lahko kasneje primerjala, ter preizkusila obe glasovni knjižnici. V prvem sta uporabljeni knjižnici pytesseract in gTTS, v drugem pa pytesseract in pyttsx3.

3.2.1 OPTIČNA PREPOZNAVA ZNAKOV IZ BESEDILA

Najprej sem si s spletne strani pytesseract-PyPI s poglavja Tesseract-ocr (pytessehttps://pypi.org/project/pytesseract/ract · PyPI) pridobila kodo ter jo vnesla v projekt. Najprej sem klicala knjižnici PIL ter pytesseract, nato sem določila pot do slike ter klicala funkcijo, iz katere je program prebral besedilo iz slike (simple image to string).

Izpisalo se mi je besedilo, to pa je vsebovalo številne napake, ker v kodi ni bil naveden slovenski jezik, posledično pa OCR ni mogel prebrati šumnikov. To sem v drugem projektu odpravila, saj sem dodala slovensko jezikovno področje.

3.2.2 GLASOVNO BRANJE

Poleg preproste pretvorbe besedila iz slike sem naredila korak še dlje. Nadaljevala sem s kodo, ki omogoča glasovno branje. To sem prikazala v prej omenjenih dveh projektih.

V prvem projektu sem uporabila knjižnico gTTS. Koda je bila pridobljena s strani GeeksforGeeks. Na začetku sem klicala knjižnico gTTS. Nato sem klicala funkcijo, ki je določila besedilo, ki sem ga prej dobila kot rezultat OCR-ja. Za govor je bilo potrebno še klicati funkcijo za zvok ter določiti jezik. Ker gTTS deluje na principu mp3, se je govorna različica najprej shranila, nato pa sem prejela rezultat.

```
# Iz knjižnice PIL prenesi sliko
from PIL import Image

# naloži Google Text to Speech
from gtts import gTTS
import os
# naloži pytesseract
import pytesseract

# Celotna pot do Tesseract-OCR
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Da se izognemo pretvorbi formata, ki je rezultat uporabe pytesseracta, izberemo relativno ali absolutno pot.
# Relativna pot je pot glede na trenutno delovno mapo (S1.jpg), absolutna pa celotna pot ne glede na trenutno delovno mapo.
# Uporabljena je absolutna pot do slike.
# V tej verziji ni naveden jezik, zato je rezultat manj natančen in brez šumnikov.
print(pytesseract.image_to_string(Image.open('C:\SLIKE\S1.jpg')))

# V tej verziji sem uporabila gTTs (Google Text to Speech), ki je manj zanesljiv, saj ne podpira slovenskega jezika.
mytext = (pytesseract.image_to_string(Image.open('C:\SLIKE\S1.jpg')))
audio = gTTS(text=mytext, lang="sr", slow=False)
audio.save("example.mp3")
os.system("start example.mp3")
```

Slika 4: Prvi projekt

Vir: osebni arhiv

Ta knjižnica je bila manj zanesljiva, saj v kodi ni bila podprta večjezičnost in ni bilo možnosti izbora slovenskega jezikovnega področja. Google-Text-to-Speech se je izkazal kot manj učinkovit. Do boljšega rezultata sem lahko prišla le s slovenščini jezikovno najbolj podobnim jezikom, a podprtim s strani gTTS – srbščino.

V drugem projektu sem uporabila knjižnico pyttsx3, ki se je izkazala za bolj učinkovito, a tudi ta sama po sebi ni imela slovenskega jezikovnega področja. Zato sem želela izvedeti, če lahko nekako dodam slovenski glas – izkazalo se je, da lahko. Ta proces lahko vzame kar nekaj časa in ni enostaven.

Privzeto sta v Windows-u dva glasova, in sicer angleški (Microsoft Zira Desktop – English (United States) in slovenski (Microsoft Lado – Slovenian (Slovenia). Najdemo ju v nastavitvah za glas pripovedovalca. Namenjena sta za govor v določenem jeziku.

Da bi lahko besedilo spremenila v slovenski govor, sem morala v Windows registru dodati oziroma posebej registrirati slovenski glas. Osnovna namestitev ima zgolj angleški glas. Čeprav lahko v registru najdemo tudi slovenskega, ta ni aktiven.

Zakaj registrirati? Registracija glasu za Windows omogoča dostop do vseh njegovih naprednih funkcij. (»Registration of the vOICe«, odst. 1, b. d.)

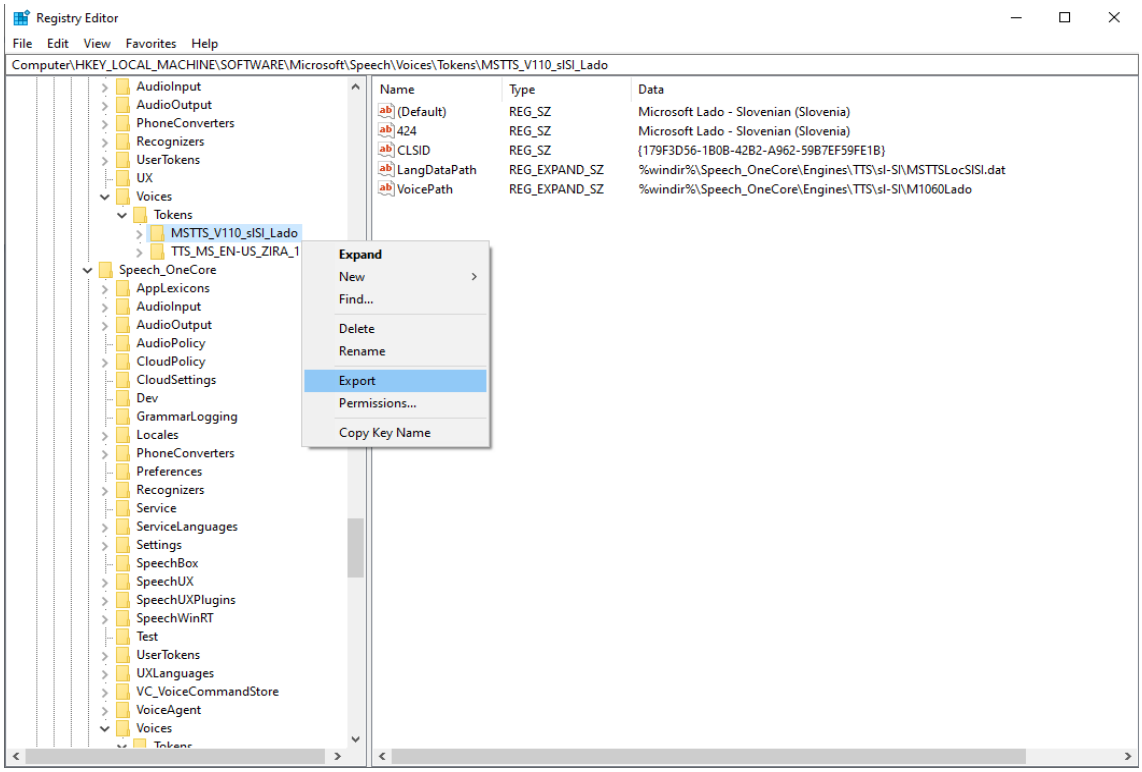
Čez nekaj časa sem spletni strani https://www.thewindowsclub.com/unlock-extra-text-to-speech-voices-in-windows našla navodila za dodajanje novega glasu v Windows

register. Tu je treba biti pazljiv, saj spreminjanje podatkov v Windows registru lahko povzroči, da računalnik ne deluje več pravilno.

Postopek je bil sledeč:

Najprej sem s kombinacijo dveh tipk *Windows key + R* odprla okno za zagon. Nato sem vanj vtipkala *regedit* in pritisnila enter. Odprl se je register, v katerem sem poiskala pot »HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech_OneCore\Voices\Tokens«

Poiskala sem glas *Lado*, kliknila nanj z desno tipko in iz priročnega menija izbrala *izvozi*. Nato sem datoteko shranila pod naziv *SLO.reg*.



Slika 5: Windows register – pot do privzetih glasov

Vir: https://www.thewindowsclub.com/unlock-extra-text-to-speech-voices-in-windows/, dostop 25. 11. 2023.

Datoteko sem nato odprla s programom za urejanje besedila in naredila naslednje spremembe:

Prej:

- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech_OneCore\Voices\Tokens\MSTTS_V110_slSI_Lado]

Potem:

- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSTTS_V110_slSI_Lado]

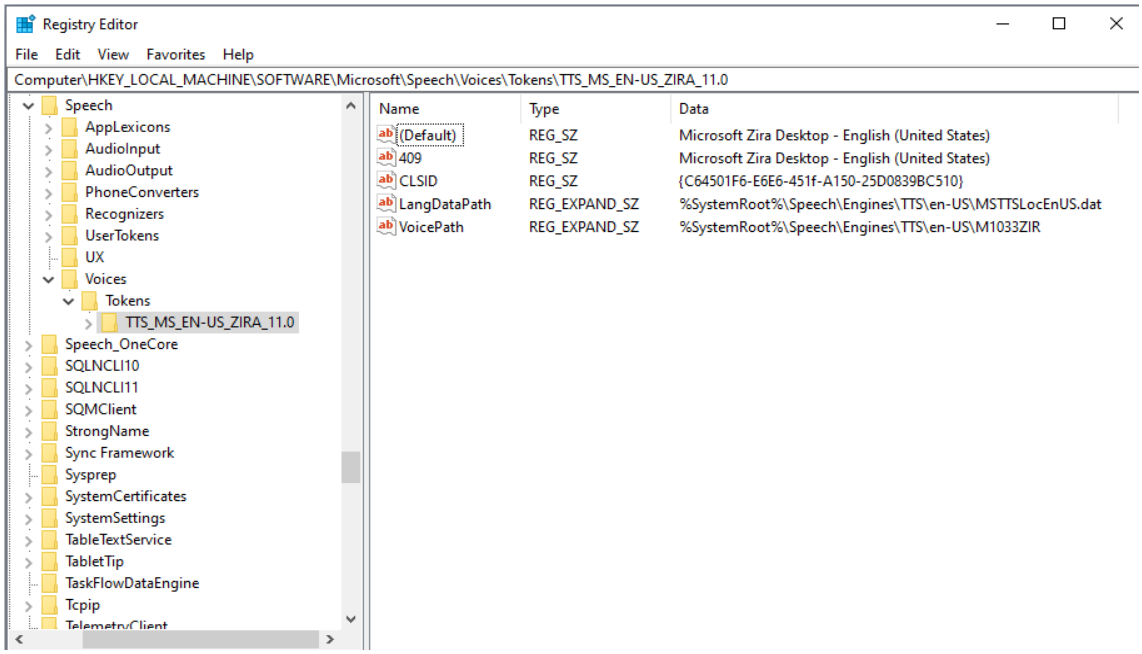
Prej:

- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech_OneCore\Voices\Tokens\MSTTS_V110_slSI_Lado\Attributes]

Potem:

- [HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSTTS_V110_slSI_Lado]

Kot sem že omenila, register v osnovi ponuja zgolj angleški glas (MS_EN-US_ZIRA_11.00).



Slika 6:Windows register – datoteka za angleški glas

Vir: https://www.thewindowsclub.com/unlock-extra-text-to-speech-voices-in-windows/, dostop 25. 11. 2023.

Sedaj je datoteka izgledala takole:

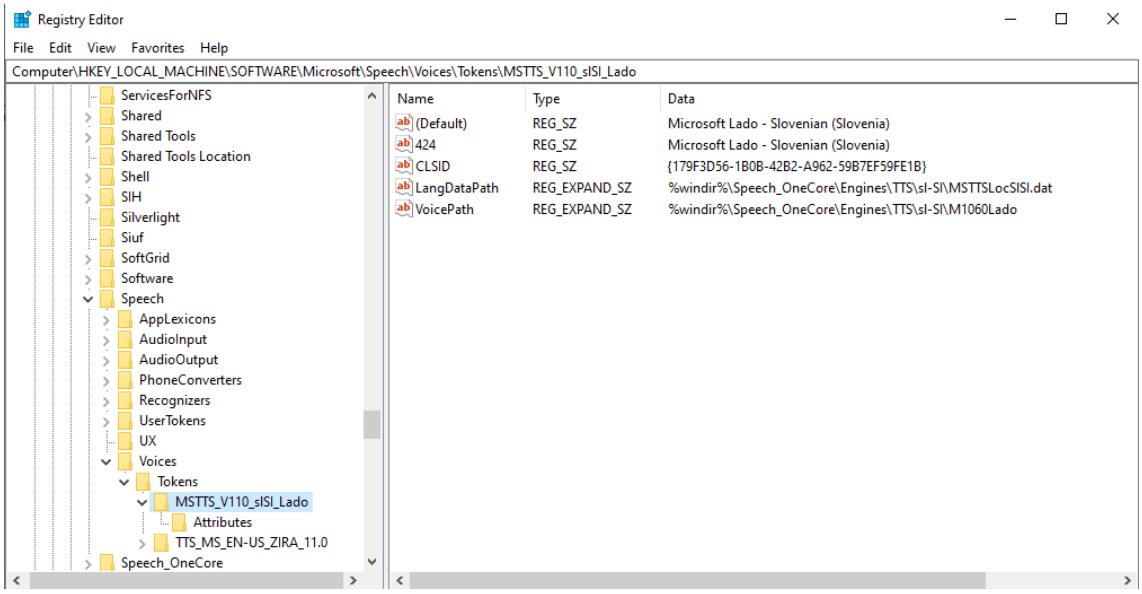
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSTTS_V110_slSI_Lado]
@="Microsoft Lado - Slovenian (Slovenia)"
"424"="Microsoft Lado - Slovenian (Slovenia)"
"CLSID"="{179F3D56-1B0B-42B2-A962-59B7EF59FE1B}"
"LangDataPath"=hex(2):25,00,77,00,69,00,6e,00,64,00,69,00,72,00,25,00,5c,00,53,\
  00,70,00,65,00,65,00,63,00,68,00,5f,00,4f,00,6e,00,65,00,43,00,6f,00,72,00,\
  65,00,5c,00,45,00,6e,00,67,00,69,00,6e,00,65,00,73,00,5c,00,54,00,54,00,53,\
  00,5c,00,73,00,6c,00,2d,00,53,00,49,00,5c,00,4d,00,53,00,54,00,54,00,53,00,\
  4c,00,6f,00,63,00,53,00,6c,00,53,00,49,00,2e,00,64,00,61,00,74,00,00,00
"VoicePath"=hex(2):25,00,77,00,69,00,6e,00,64,00,69,00,72,00,25,00,5c,00,53,00,\
  70,00,65,00,65,00,63,00,68,00,5f,00,4f,00,6e,00,65,00,43,00,6f,00,72,00,65,\
  00,5c,00,45,00,6e,00,67,00,69,00,6e,00,65,00,73,00,5c,00,54,00,54,00,53,00,\

5c,00,73,00,6c,00,2d,00,53,00,49,00,5c,00,4d,00,31,00,30,00,36,00,30,00,4c,\
00,61,00,64,00,6f,00,00,00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Speech\Voices\Tokens\MSTTS_V110_slSI_Lado\Attributes]
"Age"="Adult"
"DataVersion"="11.0.2017.0525"
"Gender"="Male"
"Language"="424"
"Name"="Microsoft Lado"
"SayAsSupport"="spell=NativeSupported; alphanumeric=NativeSupported"
"SharedPronunciation"=""
"Vendor"="Microsoft"
"Version"="11.0"

Spremembe sem shranila v novo datoteko *SLO_DODAJ.reg* in nato z dvoklikom nanjo v register vnesla podatke. V registru je sedaj registriran tudi slovenski govor, ki ga lahko v nadaljevanju uporabim v svoji aplikaciji.



Slika 7:Windows register – datoteka za slovenski glas

Vir: https://www.thewindowsclub.com/unlock-extra-text-to-speech-voices-in-windows, dostop 25. 11. 2023.

Sedaj sem lahko nadaljevala s kodo za glasovno branje. Dodala sem govor v slovenščini. To sem izvedla z naslednjo kodo:
engine.setProperty('voice', voices[1].id)
#določiš, v katerem jeziku naj program bere.

Sledeč postopek je bil podoben kot pri prejšnji različici, le z drugo glasovno knjižnico. Po klicanju funkcije, ki je določila besedilo, ki bo pretvorjeno v govor, sem klicala funkcijo za glasovno branje, nato pa sem le-tej dodala lastnosti, med katerimi je najpomembnejša govor v slovenščini.

Rezultat je bil zelo zanesljiv, registriranje glasu pa je bilo ključno, saj lahko zdaj program prebere besedilo v slovenskem jeziku.

```python
# Naloži knjižnico pyttsx3
import pyttsx3

# Iz knjižnice PIL prenesi sliko
from PIL import Image

# naloži pytesseract
import pytesseract

# Celotna pot do Tesseract-OCR
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

# Da se izognemo pretvorbi formata, ki je rezultat uporabe pytesseracta, izberemo relativno ali absolutno pot.
# Relativna pot je pot glede na trenutno delovno mapo (S1.jpg), absolutna pa celotna pot ne glede na trenutno delovno mapo.
# Uporabljena je absolutna pot do slike.
# Ta koda je bolj učinkovita, saj prepozna šumnike, ker je naveden jezik pretvorbe.
besedilo = (pytesseract.image_to_string(Image.open('C:\FOTO\Lekadol.jpg'), lang='slv'))

# Določanje parametrov oz. njihovih vrednosti novemu objektu.
engine = pyttsx3.init()


# Spodnja lastnost je hitrost branja.
engine.setProperty( name: 'rate',  value: 150)    # Speed percent
```

Slika 8: Drugi projekt – 1. stran

Vir: osebni arhiv

```python
# Spodnja lastnost je glasnost branja.
engine.setProperty( name: 'volume',  value: 0.9)  # Volume 0-1
voices = engine.getProperty('voices')
# Izberi glas, v katerem želiš prebrano besedilo.
engine.setProperty( name: 'voice', voices[1].id)

# Natisni lastnosti glasov, ki so v Windows.
voices = engine.getProperty('voices')
for voice in voices:
    print("Voice:")
    print(" - ID: %s" % voice.id)
    print(" - Name: %s" % voice.name)
    print(" - Languages: %s" % voice.languages)
    print(" - Gender: %s" % voice.gender)
    print(" - Age: %s" % voice.age)

# Natisni besedilo.
print(besedilo)

# Govori besedilo.
engine.say(besedilo)
engine.runAndWait()
```

Slika 9: Drugi projekt – 2. stran

Vir: osebni arhiv

4 REZULTATI

Rezultate sem uvrstila glede na določene kazalnike, ki so odraz zanesljivosti. Zanimalo me je, kako dobro se je odrezal Tesseract OCR oziroma pytesseract pri prepoznavi besedila iz testne fotografije in kako dobro sta se odrezala gTTS in pyttsx3 pri glasovnem branju. Ugotovitve sem skrbno podkrepila z razlago. OCR sem primerjala še glede na zanesljivost prvega in drugega poskusa, pri glasovnem branju pa sem med seboj primerjala opisani programski knjižnici. Svoja opažanja ter primerjave sem prikazala s pomočjo preglednic.

Tabela 1: Primerjava glasovnih knjižnic glede na rezultat

KAZALNIK	GTTS	PYTTSX3
PREPOZNAVA TER GOVOR BESEDILA	DA	DA
BRANJE ŠUMNIKOV	DA	DA
SLOVENSKI GOVORNI NAGLAS	NE	DA
BRANJE ŠTEVIL V SLOVENŠČINI	NE	DA
SKUPAJ	2 x DA 2 x NE	4 x DA

V prvi tabeli je prikazana primerjava glasovnih knjižnic glede na navedene kazalnike. Obe knjižnici sta prepoznali besedilo ter ga brez večjih težav pretvorili v govor.

GTTS ne podpira slovenskega jezikovnega področja, zato sem mu dodala srbskega, ki je sicer znal prebrati šumnike, vendar je v govoru zelo razločno zaznaven srbski naglas, pa tudi števila je bral v srbščini.

Pyttsx3 je lahko s pomočjo postopka izven te knjižnice – registracije slovenskega glasu v Windows registru – prebral šumnike in števila v slovenščini, njegov govor pa je imel slovenski naglas.

Pyttsx3 je zahvaljujoč registraciji slovenskega glasu in možnosti dodajanja le-tega v kodo uspešno opravil svojo nalogo ter se v primerjavi z gTTS-jem izkazal za bolj zanesljivega in s tem boljšega za uporabo v prihodnji aplikaciji.

Tabela 2: Primerjava rezultata pri obeh poskusih pytesseracta za OCR

KAZALNIK	PYTESSERACT 1. POSKUS	PYTESSERACT 2. POSKUS
PREPOZNAVA BESEDILA	DA	DA
PREPOZNAVA ŠUMNIKOV	NE	DA
NAPAKE V KONČNEM BESEDILU	VELIKO	MALO
ZANESLJIVOST	MAJHNA	VELIKA

V drugi tabeli so prikazana opažanja prvega in drugega poskusa pytesseracta pri optični prepoznavi znakov iz besedila. Oba sta uspešno prepoznala besedilo, saj sem v drugem poskusu le dodala slovensko jezikovno področje, posledično je v drugem poskusu OCR prepoznal šumnike in s tem zmanjšal količino napak v končnem besedilu. Seveda je s tem drugi poskus tudi bolj zanesljiv.

Tabela 3: Primerjava vseh treh knjižnic glede na zanesljivost funkcij za aplikacijo

KAZALNIK	PYTESSERACT	GTTS	PYTTSX3
DELOVANJE APLIKACIJE S POMOČJO **PROGRAMSKIH KNJIŽNIC**	DOBRO	SLABO	DOBRO

Tretja tabela kaže na zanesljivost knjižnic glede na zgornje ugotovitve meritev in primerjav prejšnjih dveh tabel. S tem je pokazatelj zanesljivosti delovanja le-teh v prihodnji aplikaciji.

5 RAZPRAVA

Z raziskovanjem sem želela ugotoviti, ali lahko s pomočjo umetne inteligence ter računalniških algoritmov pomagam slepim in slabovidnim tako, da se bodo med branjem etiket artiklov v trgovini lažje znašli, saj je besedilo na etiketah res majhno in morda tudi nerazločno. Zasnovala sem aplikacijo, ki deluje na osnovi umetne inteligence; posvetila sem se načrtovanju in izvedbi glavnih funkcij te aplikacije. Želela sem oblikovati funkciji, ki omogočata pretvorbo besedila iz slike v tekst, kar posledično da bolj razločno in povečano besedilo, in pretvoriti besedilo v govor, saj to lahko bistveno pomaga slepim in slabovidnim, ki bi s poslušanjem besedilo lažje pomnili oziroma ga razumeli. Računalniški algoritmi so na kratko vmesni procesi, ki potekajo, da privedejo do neke rešitve oziroma rezultata. Ti so potekali tudi med sklicevanjem teh dveh funkcij. Sestavna enota teh algoritmov je umetna inteligenca, ki je bistveno potrebna za omogočanje izvedbe takih vrst funkcij. Tako optična prepoznava znakov kot pretvorba besedila v govor sta omogočena le tako, da je v program prek specifičnih programskih knjižnic vnesena umetna inteligenca. Poleg programskih knjižnic, ki so sistemski programi, je uporabljena programska oprema vključevala še razvojno programsko opremo, katere glavni namen, kot nakazuje ime, je omogočiti razvijanje novih programov.

Da bi lahko izvedla funkciji, je zelo pomembno poudariti namembnost vsake programske opreme, ki sem jo uporabila. Glede na njihovo namembnost lahko hitro ugotovimo, ali je bila programska oprema primerna za določeno nalogo. Za primer si lahko zastavimo, da želimo besedilo pretvoriti v govor. Če bomo za to nalogo uporabili programsko knjižnico pytesseract (ki je vmesnik programa Tesseract OCR), bomo hitro ugotovili, da take funkcije ne moremo izvesti, saj je bila uporabljena programska knjižnica neprimerna, njen namen pa je povsem drugačen. Programske knjižnice je bilo po potrebi treba še dodatno namestiti oziroma jih poiskati v programu, ki sem ga uporabila za programiranje – PyCharm.

S telefonom sem zajela še testno fotografijo, in sicer fotografijo navodil za uporabo lekadola.

Po pridobitvi potrebne programske opreme in testne fotografije se je izvedba praktičnega dela šele začela. Nato je bilo treba pridobiti primerno kodo, ki bo omogočala potek procesov in izvedbo funkcij, ker sama nimam dovolj znanja programiranja, da bi lahko ustvarila svojo kodo. Koda, ki sem jo pridobila iz spletne strani prav za ta namen, je morala biti dokaj razločna in preverljiva, v vsaki vrstici pa sem dodala svoj komentar, ki je opis, kaj določena vrstica kode pomeni. V kodi je bilo potrebno podati pot do

fotografije. Nato sem program zagnala ter dobila rezultat, po potrebi je bilo ob napakah potrebno še kaj prilagoditi.

Ker sem vso programsko opremo, testno fotografijo in kodo skrbno raziskala in preverila namembnost ter zanesljivost glede na mnenja drugih na spletu, lahko potrdim, da so bili primerni, saj sem dobila zanesljiv rezultat, pa tudi funkcije so bile izvedene v skladu z mojimi pričakovanji. Metode dela, kot je testiranje, izvedba eksperimenta, načrtovanje procesov aplikacije in načrtovanje funkcij ter postavitev hipoteze, so bile prav tako primerne.

V poglavju o rezultatih sem analizirala uspešnost programskih knjižnic. Glede na podatke analize se je od glasovnih knjižnic najbolje odrezala pyttsx3. Menim pa, da bi bil rezultat brez registracije glasu zelo podoben rezultatu knjižnice gTTS. A mislim, da ima tu odločilno vrednost tudi koda, ki ni enaka za obe knjižnici. Zato sem se za dodajanje registriranega glasa odločila za pyttsx3 in ne za gTTS, saj menim, da bi bil v nasprotnem primeru proces zahtevnejši in daljši in ni bistveno pomembno, kateri knjižnici sem dodala glas s pomočjo registracije, saj je namen mojega raziskovalnega dela drugačen. Če bi registriran glas dodala v kodo, ki pripada knjižnici gTTS, bi najverjetneje izgubila preveč časa, ki bi ga sicer lahko namenila primerjavi rezultatov teh dveh knjižnic. Uspešnost knjižnic sem merila na podlagi kazalnikov. Vsem knjižnicam je bilo skupno, da so bile zanesljive vsaj s tem, ko so opravile svoje delo. Drugo, kar me je zanimalo, pa je bila vsebnost slovenskega jezikovnega področja in temu primeren rezultat – v slovenščini. Najprej bom izpostavila glasovni knjižnici, saj je za slovensko jezikovno področje bilo zadanih več kazalnikov. Glede na podatke iz rešitev se je gTTS odrezal slabše, pyttsx3 pa bolje, zato je pri izdelavi aplikacije bolj pametno uporabiti pyttsx3. Za knjižnico optične prepoznave znakov sem uporabila le pytesseract, in vredno je povedati, da bi z nekaj dodatki ta knjižnica v aplikaciji delovala zanesljivo.

Pomembno je dodati, da je razvojni program PyCharm, ki deluje v programskem jeziku python, zelo dobra podlaga za razvijanje drugih programov, pa naj bodo ti zelo kratki in nezapleteni ali obratno. Menim, da je ta program primeren tako za začetnike in ljubitelje programiranja, kot za poklicne programerje, ter da je zelo primeren za razvijanje programov in funkcij, ki delujejo na principu umetne inteligence.

Dodala bi še, da lahko dandanes kaj hitro razviješ kak program oziroma aplikacijo, saj je namestitev potrebne programske opreme dokaj enostavna, kodo pa lahko najdeš povsod na spletu, v nadaljevanju pa jo lahko prilagajaš in preoblikuješ za potrebe svoje aplikacije.

5.1 POTRDITEV HIPOTEZE

S tem ko sem dokazala, da je mogoče ustvariti delujoči, a hkrati zanesljivi funkciji, sem tudi dokazala, da lahko s pomočjo umetne inteligence ter računalniških algoritmov zasnujem aplikacijo ter s tem pomagam slepim in slabovidnim ljudem. Hipoteza je potrjena.

Umetna inteligenca je še zmeraj precej neraziskano področje, zato je uporaba optične prepoznave znakov in glasovnega branja s pomočjo le-te zelo zanimiva in sveža tema, pri kateri se da še veliko raziskati.

6 ZAKLJUČEK

Pri raziskovanju in izvedbi eksperimenta sem ugotovila, da je aplikacijo mogoče razviti in s tem pomagati slepim ter slabovidnim, saj sem uspešno ustvarila in izvedla funkciji. Uporaba programskega jezika python je pri umetni inteligenci zelo popularna in tudi zares učinkovita metoda, saj z malo truda in nekaj znanja dobimo kar se da zanesljive rezultate.

Pri prvi funkciji – optični prepoznavi znakov iz besedila – sta bila program Tesseract OCR in njegova Pythonova knjižnica zanesljiva ter zelo uporabna za začetnike, kot sem jaz. Pri drugi funkciji – glasovnem branju – je bila bolj zanesljiva knjižnica pyttsx3, registracija slovenskega glasu pa je bila ključna.

Z izvedbo eksperimenta sem prikazala, da lahko na ne preveč zahteven način ustvarim funkciji, ki imata nek potencial, s tem pa sem posledično dokazala, da je aplikacijo mogoče ustvariti. Hkrati sem po zanesljivosti knjižnic dokazala, kateri programski knjižnici uporabiti za ti dve nalogi. Dokazala sem, da lahko pomagam slepim in slabovidnim ljudem, ki potrebujejo nekaj takega, kar jim lahko bistveno olajša vsakdan, s pomočjo umetne inteligence.

S tem sem potrdila svojo hipotezo ter dosegla cilj raziskovanja.

V prihodnje bi se dalo tako aplikacijo razviti. Ko sem na začetku raziskovanja dobila idejo o tej aplikaciji, sem imela še nekaj zamisli, npr.: kako bi naredila funkcijo, ki spremeni barvo besedila za dislektike; kako bi lahko dodala AI asistenta, ki bi odgovarjal ter dajal podatke o artiklu, ki ga iščemo.

Aplikacija bi imela še nekaj procesov, kot so vprašalnik za uporabnika na začetku ob namestitvi o njegovih zmožnostih in nezmožnostih, s pomočjo *ibeacon* bi lahko vodila uporabnika do artikla, skratka AI bi najprej uporabnika vodila do izdelka, nato prilagodila besedilo s slike etikete, mu podala še več informacij o artiklu ter mu svetovala o primernosti in uporabnosti izdelka zanj.

7 VIRI IN LITERATURA

European Parliament. (Zadnja posodobitev: 26. 3. 2021). Kaj je umetna inteligenca in kako se uporablja v praksi? Najdeno dne 5. 10. 2023 na spletnem naslovu: https://www.europarl.europa.eu/news/sl/headlines/society/20200827STO85804/kaj-je-umetna-inteligenca-in-kako-se-uporablja-v-praksi/

How OCR Works: An In-Depth Explanation of Optical Character Recognition. (b. d.). Na welabeldata.com. Najdeno dne 5. 10. 2023 na spletnem naslovu: https://www.welabeldata.com/post/how-ocr-works-an-in-depth-explanation-of-optical-character-recognition/

Sachdeva N. (Zadnja posodobitev: 23. 10. 2023). Vision AI: What is it and Why does it Matter? Najdeno dne 5. 10. 2023 na spletnem naslovu: https://insights.daffodilsw.com/blog/vision-ai-what-is-it-and-why-does-it-matter/

International Institute in Geneva. (b. d.). Algorithms: their meaning in computer science. Najdeno dne 10. 10. 2023 na spletnem naslovu: *Algorithm & computer science: definition and understanding (iig.ch)*

Demšar J. (b. d.). Algoritmi. Najdeno dne 10. 10. 2023 na spletnem naslovu: https://ucilnica.fri.uni-lj.si/pluginfile.php/10499/mod_resource/content/5/algoritmi.pdf?forcedownload=1/

Tesseract OCR: What is it, and Why Would You Choose it in 2023? (Zadnja posodobitev: 20. 10. 2022). Na klippa.com. Najdeno dne 16. 10. 2023 na spletnem naslovu: Tesseract OCR: What is it and Why Would You Choose It? (klippa.com)

Durette P. N. (b. d.). gTTS. Najdeno dne 22. 10. 2023 na spletnem naslovu: https://gtts.readthedocs.io/en/latest/

Python | Text to Speech by using pyttsx3. (b. d.). Na geeksforgeeks.org. Najdeno dne 22. 10. 2023 na spletnem naslovu: https://www.geeksforgeeks.org/python-text-to-speech-by-using-pyttsx3/

Registration of The vOICe. (b. d.). Na seeingwithsound.com. Najdeno dne 19. 11. 2023 na spletnem naslovu: The vOICe Registration (seeingwithsound.com)

IZJAVA

Mentorica Marjeta Gradišnik Mirt v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom OPTIČNO BRANJE IN PRETVORBA BESEDILA IZ SLIK ZA POMOČ SLEPIM, SLABOVIDNIM IN DISLEKTIKOM,

katere avtorica je Noemi Luiza Flis:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

*

Celje, 6.3.2024

Žig šole

Podpis mentorja
Marjeta Gradišnik Mirt

Podpis odgovorne osebe

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.