Vektorske baze podataka za obradu nestrukturiranih podataka

Blašković, Luka

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

Luka Blašković

VEKTORSKE BAZE PODATAKA ZA OBRADU NESTRUKTURIRANIH PODATAKA

Završni rad

Pula, 2022.

Sveučilište Jurja Dobrile u Puli
Fakultet informatike u Puli

VEKTORSKE BAZE PODATAKA ZA OBRADU NESTRUKTURIRANIH PODATAKA

Završni rad

JMBAG: 0303088177, redovni student
Studijski smjer: Informatika
Kolegij: Baze podataka II
Znanstveno područje: Društvene znanosti
Znanstveno polje: Informacijske i komunikacijske znanosti
Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Goran Oreški

Pula, srpanj 2022.

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisani Luka Blašković, kandidat za prvostupnika informatike ovime izjavljujem da je ovaj Završni rad isključivo rezultat mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Završnog rada nije napisan na nedozvoljeni način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

Luka Blašković

U Puli, 17. srpnja 2022. godine

IZJAVA O KORIŠTENJU AUTORSKOG DJELA

Ja, Luka Blašković, dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj Završni rad pod nazivom „Vektorske baze podataka za obradu nestrukturiranih podataka“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

Student

Luka Blašković

U Puli, 17. srpnja 2022. godine

Sadržaj

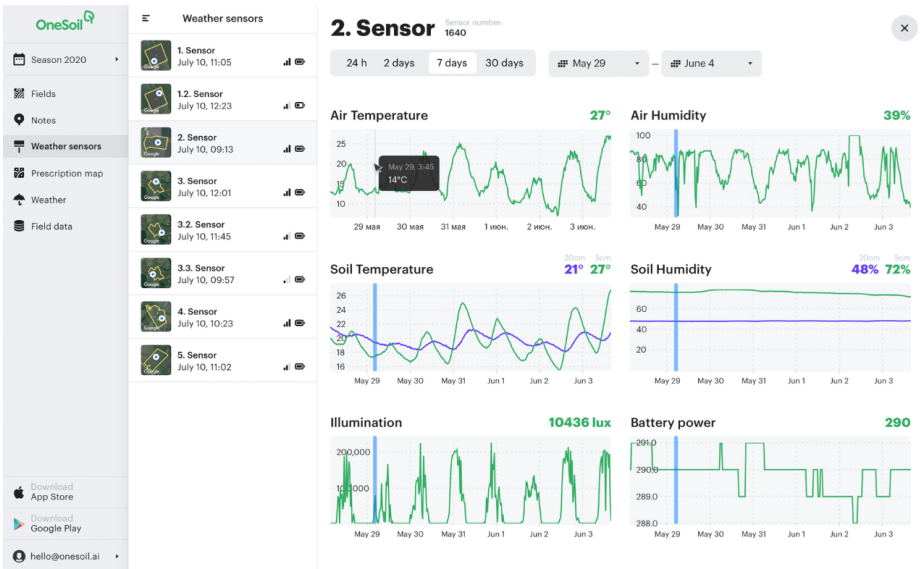
1. UVOD

Podaci koji se pohranjuju u današnjim bazama podataka, prema svojemu obliku, mogu se podijeliti u tri skupine: **strukturirani**, **nestrukturirani i polu-strukturirani**. Premda ne postoji jasno definirana granica između strukturiranih i nestrukturiranih podataka, može se zaključiti da su strukturirani podaci oni podaci koji odgovaraju formalno definiranom podatkovnom modelu te koje obilježavaju jasno definirani tipovi podataka. Za usporedbu, nestrukturirani podaci nemaju prepoznatljivu strukturu (MongoDB, n.d.). Drugim riječima, nisu raspoređeni prema unaprijed postavljenom podatkovnom modelu ili shemi. Najpoznatiji primjeri takvih podataka su fotografije, videozapisi, zvuk i sl. Danas se bilježi značajan porast uporabe i skladištenja nestrukturiranih podataka. Između 80 i 90 posto podataka koje velike firme i organizacije stvaraju je nestrukturirano (Harbert, 2021). Međutim, prema anketi koju je provela *Deloitte* kompanija, samo 18% organizacija je spremno za obradu i skladištenje takvih podataka (Deloitte Insights, 2019). Kroz povijest, nestrukturirani podaci bili su vrlo teški za analizirati te ih je vrlo nezgodno, gotovo nemoguće pohraniti i obrađivati kroz standardne sustave za upravljanje relacijskim bazama podataka (RDBMS). Nestrukturirani podaci mogu se pohranjivati u raznim oblicima noSQL baza podataka, *data lake-ovima* ili skladištima podataka (MongoDB, n.d.). Strojno učenje (engl. Machine learning), kroz razne modele omogućuje pretvorbu kompleksnih nestrukturiranih podataka u numeričke reprezentacije zvane vektorske ugradnje (engl. vector embeddings). Drugim riječima, moguće je stvarne objekte poput slika, zvuka, tekstualnih datoteka ili videozapisa reprezentirati pomoću matematičkih vektora (Bergstein, 2022). Vektorske baze podataka su dizajnirane da rade s karakteristikama vektorskih ugradnji. One indeksiraju podatke na način koji olakšava pretraživanje i dohvaćanje objekata prema njihovim numeričkim vrijednostima. Glavna funkcionalnost koju pružaju vektorske baze podataka su jednostavno pretraživanje i dohvaćanje sličnih vektora. Semantička pretraga teksta, pretraživanje po sličnosti, rangiranje web stranica ili detekcija anomalija, najčešći su slučajevi upotrebe vektorskih baza podataka (Turriff, n.d.). U ovom radu objašnjeni su sljedeći koncepti: strukturirani i nestrukturirani podaci, neuronske mreže i „vektorizacija“ podataka pomoću modela dubokog učenja te vektorske baze podataka i njihova primjena. Kako bi se preciznije prikazao princip rada vektorskih baza podataka, za potrebe ovog završnog rada izrađena je aplikacija za pretraživanje slika po sličnosti (engl. Reverse image search). Aplikacija je implementirana pomoću Milvus.io vektorske baze podataka i Towhee.io – Python modula za kodiranje nestrukturiranih podataka.

2. NESTRUKTURIRANI PODACI

Podaci su sastavni dio digitalnih usluga i digitalne transformacije. Veliki podaci odnose se na skupove podataka koji su preveliki i previše kompleksni da bi se mogli obraditi standardnim alatima i metodama. Danas se iz više izvora, bili to strojevi ili ljudi, vrlo brzo generiraju velike količine podataka koji se pritom prikupljaju i analiziraju kako bi se stekla nova saznanja. Obzirom da se radi o velikim skupovima podataka koji su vrlo često nečitljivi i pre kompleksni, koriste se nove tehnologije poput umjetne inteligencije i strojnog učenja za njihovu obradu (Oracle, n.d.). Primjerice, meteorolozi prikupljaju podatke raznom opremom poput radara, senzora ili sondi kako bi predvidjeli vremensku prognozu. Takvi uređaji generiraju veliku količinu kompleksnih podataka za koju je potrebna računalna obrada da bi oni bili od značaja (Essearth, 2020). Radi se o nestrukturiranim podacima budući da ih ne možemo svrstati u formalno definirani podatkovni model sa jasno definiranim tipovima podataka. Na slici 1 možemo vidjeti primjer grafičkog sučelja aplikacije koja prikazuje informacije o temperaturi i vlažnosti zraka i tla u realnom vremenu, koje prikupljaju senzori.

Slika 1. Primjer grafičkog sučelja obrađenih podataka koje generiraju senzori

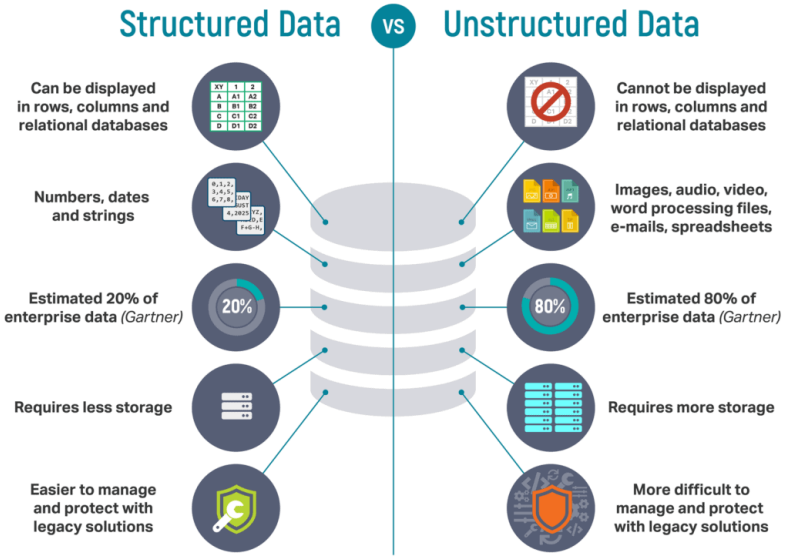


Izvor: https://sensors.onesoil.ai/

Vjerojatno svima najpoznatiji primjeri nestrukturiranih podataka, s kojima se svakog dana gotovo svi susrećemo su multimedijalni zapisi - slike, videozapisi i zvučni zapisi. Količina nestrukturiranih podataka danas munjevito raste budući da ih gotovo svaki pojedinac ili

poduzeće stvara svakog dana. Kako je već navedeno u uvodu, eksperti navode da je danas 80-90% podataka nestrukturirano. Glavni nedostatak je što nestrukturirane podatke nije moguće prikazati kroz relacijsku bazu podataka, već zahtijevaju kompleksnija rješenja za pohranu i obradu. Dodatno, zahtijevaju i veći prostor za pohranu te ih je teže zaštititi. Poslovna analitika i poslovna inteligencija (engl. Business analytics and Business intelligence) skupina je tehnologija, praksi i strategija koje velika poduzeća koriste kako bi upravljali i analizirali velike količine podataka i izvukli korisne informacije i znanje iz njih (Elliott, 2021). Slika 2 prikazuje glavne karakteristike strukturiranih i nestrukturiranih podataka i njihove međusobne razlike.

Slika 2. Usporedba strukturiranih i nestrukturiranih podataka



Izvor: https://k21academy.com/microsoft-azure/dp-900/structured-data-vs-unstructured-data-vs-semi-structured-data/

Međutim, najveći izazov leži upravo u obradi i analizi nestrukturiranih podataka zbog velikog broja različitih formata u kojima mogu biti. **Veliki podaci** (engl. Big Data) je termin koji se koristi za velike količine teško obradivih strukturiranih i nestrukturiranih podataka, koji su često pomiješani (polu-strukturirani) te su samim time pre kompleksni za obradu tradicionalnim softverom za obradu podataka (Oracle, n.d.).

Štoviše, obrada takvih podataka zahtjeva moderne tehnologije poput:

- umjetne inteligencije,
- strojnog učenja,
- neuronskih mreža,
- genetičkih algoritama,
- rudarenja podataka.

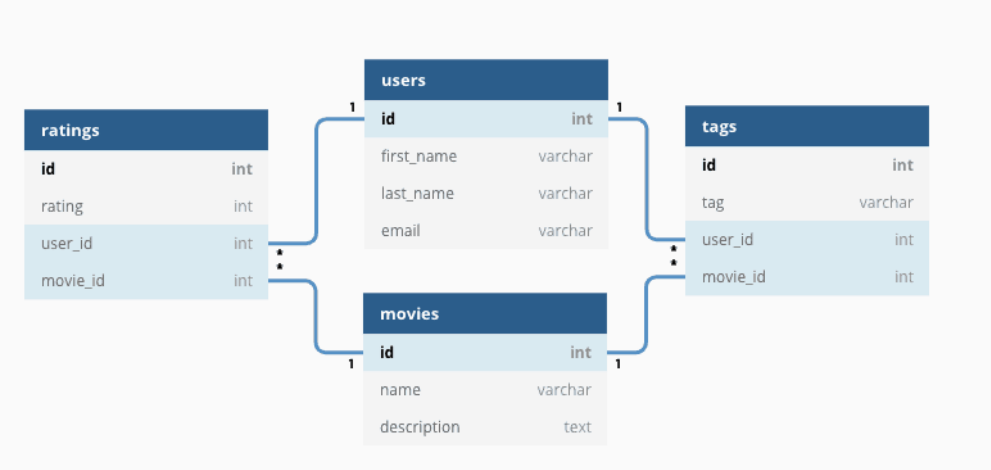
3. KONCEPTI BAZA PODATAKA

U sljedećim poglavljima ukratko su opisana dva osnovna koncepta baza podataka – relacijski i ne-relacijski.

3.1 Relacijske baze podataka

Relacijske baze podataka organiziraju podatke u tablice (relacije) koje se sastoje od redaka i stupaca te se međusobno povezuju. Radi se o najčešće korištenoj bazi podataka koju koriste današnje firme (Google Cloud, n.d.). Kako bi se osigurala valjanost podataka pri izvođenju transakcija sa bazom podataka, ACID (engl. atomicity, consistency, isolation, durability) svojstva moraju biti ostvarena (Keboola, 2021). Tablice se sastoje od: primarnih ključeva (koji su jedinstveni identifikatori svakog zapisa u tablici), atributa i njihovih tipova podataka te veza među njima. Slika 3 prikazuje logički model relacijske baze podataka.

Slika 3. Logički model relacijske baze podataka



Izvor: https://www.heavy.ai/technical-glossary/relational-database

Relacijske baze podataka u pravilu omogućuju pohranu isključivo strukturiranih podataka. Postoje iznimke poput *FileTables* funkcionalnosti koju pruža SQL Server DBMS.

Filetables omogućava pohranu nestrukturiranih podataka u datotečni sustav i njihov dohvat pomoću pokazivača (Microsoft Docs, 2020). Naravno, takav pristup ne omogućuje obradu nestrukturiranih podataka.

3.2 Ne-relacijske (NoSQL) baze podataka

Zadnjih godina sve veću popularnost stječu takozvane ne-relacijske ili NoSQL baze podataka. Radi se o bazama podataka koje ne zahtijevaju unaprijed definiranu shemu. Glavna

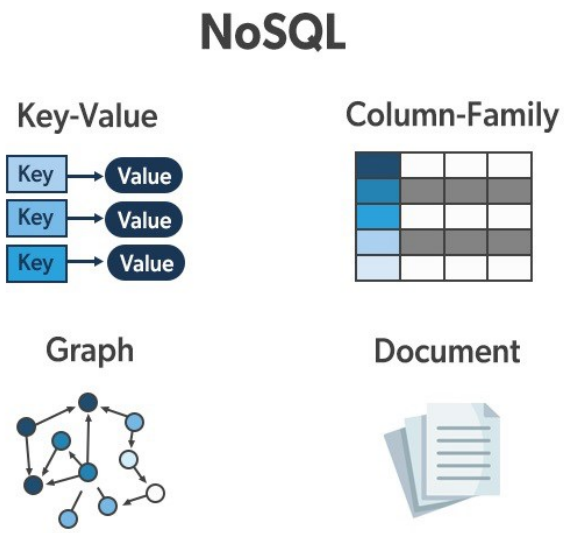
karakteristika je što umjesto spajanja tablica normaliziranih podataka, NoSQL baze pohranjuju nestrukturirane i polu-strukturirane podatke, često u obliku ključ-vrijednost parova ili *JSON* objekata (Integrate.io, n.d.).

Najpoznatiji oblici NoSQL baza podataka su:

- ključ-vrijednost par (engl. key-value pair),
- stupčaste baze podataka (engl. Column-oriented),
- grafovske baze podataka (engl. Graph-based) i
- dokumentne baze podataka (engl. Document-oriented).

Slika 4 prikazuje grafičke reprezentacije navedenih NoSQL baza podataka.

Slika 4. Grafičke reprezentacije NoSQL baza podataka



Izvor: https://www.geeksforgeeks.org/types-of-nosql-databases/

Fleksibilan podatkovni model koji pružaju ne-relacijske baze podataka omogućava lakšu pohranu i analitiku nestrukturiranih podataka, naravno uz pomoć umjetne inteligencije i tehnika strojnog učenja. Ovisno o formatu, veličini i operacijama koje se izvode nad pohranjenim podacima, koriste se različiti oblici NoSQL baza podataka.

4. NEURONSKE MREŽE I DUBOKO UČENJE

Duboko učenje (engl. Deep learning) je skup tehnika strojnog učenja koji se bavi algoritmima inspiriranim strukturom i principom rada ljudskog mozga - preciznije neuronskim mrežama. Neuron u ovom kontekstu predstavlja najjednostavniju matematičku operaciju koja prima ulazni podatak, množi ga težinskim faktorom (engl. weight) i potom šalje sumu kroz aktivacijsku funkciju ostalim neuronima. Težinski faktor je parametar koji predstavlja jačinu veze među čvorovima neuronske mreže (Hackernoon, 2017). Bez obzira što tradicionalne neuronske mreže mogu dati vrlo dobre rezultate, dodatni skriveni slojevi doprinose preciznosti i točnosti rezultata. Iako sama ideja dubokog učenja potječe još od 1940-ih godina, njegov utjecaj na IT industriju započinje tek u ranim 2000-im, dok je veća industrijska primjena započela 2010-ih godina značajnijim razvojem hardvera, posebice grafičkih kartica (Foote, 2022).

U kontekstu vektorskih baza podataka, duboko učenje omogućava rješavanje problema u *computer vision* i *natural language processing* područjima budući da omogućava treniranje modela pomoću kojih je moguće dobiti vektorske reprezentacije raznih nestrukturiranih podataka, koje je potom moguće pohranjivati u vektorsku bazu podataka (Heller, 2019).

U sljedećim poglavljima detaljnije će se govoriti o:

- vektorskim reprezentacijama nestrukturiranih podataka,
- modelima dubokih neuronskih mreža (DNN, RNN i CNN),
- pretraživanju vektorskog prostora (KNN i ANN)

4.1 Vector embeddings

Vektorske ugradnje (engl. Vector embeddings), u strojnom učenju, su reprezentacije nestrukturiranih podataka kao točke u n-dimenzionalnom prostoru na način da se slične točke grupiraju zajedno (engl. clustering). Drugim riječima, to je preslikavanje diskretne kategoričke varijable u vektor koji se sastoji od kontinuiranih brojeva. Matematički gledano, to je skup decimalnih (engl. floating-point) brojeva koji reprezentira neki objekt.

U sljedećem paragrafu, obradit će se *Word2Vec* model koji se ne smatra dubokim učenjem već dvoslojnom neuronskom mrežom. Bez obzira na to, predstavlja dobar uvod u razumijevanje vektorskih reprezentacija podataka.

Natural language processing (NLP) je tehnologija koja se bavi interakcijom između računala i ljudskog jezika. Cilj je obrada i analiza velike količine podataka prirodnog jezika i govora kako bi se, primjerice, precizno mogle izvući korisne informacije i saznanja iz dokumenata, automatsko prevođenje prepoznavanjem govora ili teksta ili pak davanje instrukcija računalu kroz glasovne naredbe. Jedna od najpoznatijih skupina modela i tehnika koja se i danas koristi za *NLP* je „Word2Vec“. Izradio ju je i patentirao češki računalni znanstvenik Tomáš Mikolov u Google-u 2013. godine. Word2Vec algoritam koristi model dvoslojne neuronske mreže koji kada se istrenira velikim tekstom, može „predvidjeti“ sljedeću riječ ili riječi u nepotpunoj rečenici. Algoritam funkcionira na način da kao ulazni podatak uzima veliki tekst te kao izlazni podatak generira vektorski prostor, tipično sastavljen od nekoliko stotina pa i više dimenzija. Svaka jedinstvena riječ u tekstu dobiva zaseban vektor u prostoru. Vektorske ugradnje koje reprezentiraju slične riječi (sinonime) nalaze se bliže jedna drugoj u vektorskom prostoru, odnosno u zajedničkom su klasteru (Markowitz, 2022).

Kako bismo bolje razumjeli Word2vec algoritam, prikazat ćemo njegov princip rada u dvodimenzionalnom vektorskom prostoru na jednostavnom primjeru. Tablica na slici 5 sadrži stupac s 14 životinja i dva stupca s vrijednostima atributa $cuteness$ i $size$ u rasponu od 0-100.

Slika 5. prikaz životinja i njihovih karakteristika

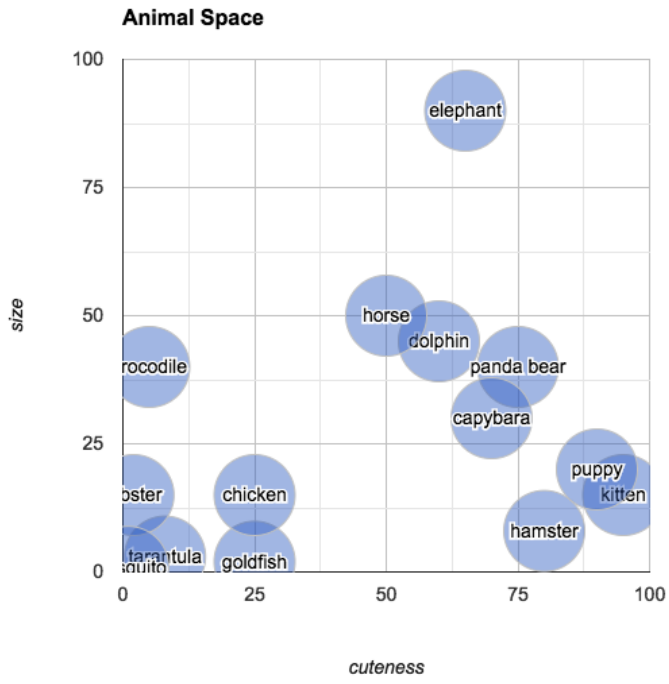
	cuteness (0–100)	size (0–100)
kitten	95	15
hamster	80	8
tarantula	8	3
puppy	90	20
crocodile	5	40
dolphin	60	45
panda bear	75	40
lobster	2	15
capybara	70	30
elephant	65	90
mosquito	1	1
goldfish	25	2
horse	50	50
chicken	25	15

Izvor: https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469

Dalje, kada bi si htjeli postaviti pitanje koja je životinja najsličnija mačiću (gledajući vrijednosti 2 atributa koje imamo), najjednostavnije bi bilo vizualizirati podatke kao točke u dvodimenzionalnom prostoru.

Na slici 6 se prikazuju podaci vizualizirani u dvodimenzionalnom prostoru, može se vidjeti da je psić najsličniji mačiću prema danim vrijednostima atributa.

Slika 6. prikaz podataka pomoću točaka u dvodimenzionalnom prostoru



Izvor: https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469

Jedan od najjednostavnijih načina pretraživanja vektorskog prostora je euklidska udaljenost. Njome možemo izračunati koliko su dvije točke „udaljene“ u prostoru, u našem slučaju dvije životinje. Slika 7 prikazuje implementaciju euklidske udaljenosti u Pythonu.

Slika 7. Implementacija euklidske udaljenosti u Pythonu

```
import math
def distance2d(x1, y1, x2, y2):
  return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

Sada možemo vrijednosti uvrstiti u funkciju te dobiti numeričke udaljenosti između dvije različite točke (životinje). Primjerice, možemo dobiti udaljenost (sličnost) između dabra (70, 30) i pande (74, 40):

Slika 8. Euklidska udaljenost između pande i dabra

distance2d(70, 30, 75, 40) #panda i dabar

Rješenje: 11.180339887498949

Izračunajmo i udaljenost između tarantule (8, 3) i slona (65, 90):

Slika 9. Euklidska udaljenost između tarantule i slona

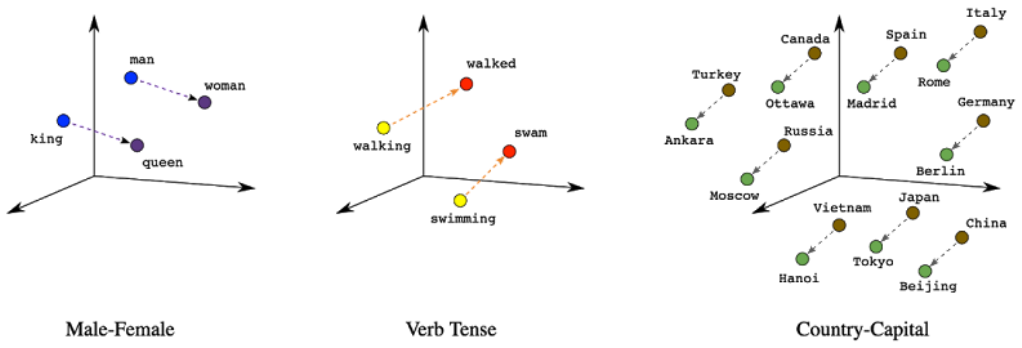
distance2d(8, 3, 65, 90) #tarantula i slon

Rješenje: 104.0096149401583

Možemo zaključiti da su dabar i panda (u ovom kontekstu) puno sličnije životinje od tarantule i slona. Primjer s programskim kodom preuzet je s Github repozitorija - (Parrish, 2017).

Na isti princip možemo prikazati trodimenzionalni vektorski prostor, kao što vidimo na slici 10., slične riječi se grupiraju zajedno tako da se vektori (točke) - kralj, kraljica i princ „klasteriraju“ u neposrednoj blizini. Ista stvar se događa kod sinonima (hodao, šetao, trčao).

Slika 10. Riječi prikazane u 3-dimenzionalnom prostoru..



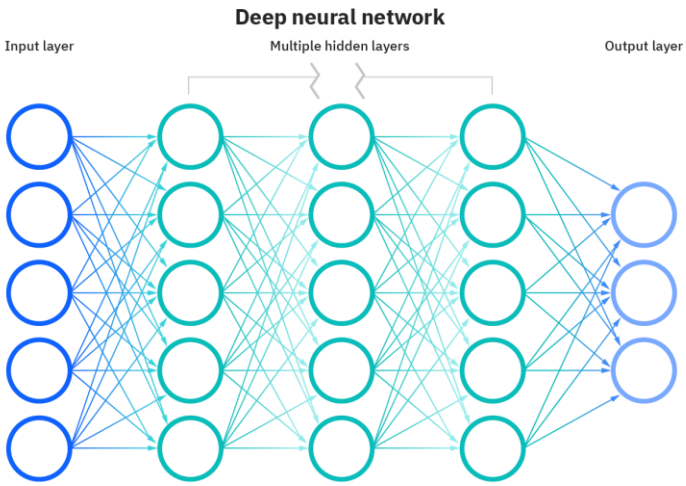
Izvor: https://cloud.google.com/blog/topics/developers-practitioners/meet-ais-multitool-vector-embeddings

Nakon što su objašnjene vektorske ugradnje podataka kroz mali broj dimenzija i njihova vizualizacija u vektorskom prostoru, u sljedećim poglavljima obradit će se relevantni modeli dubokih neuronskih mreža te na koji način oni mogu „generirati“ vektorske ugradnje od stotinu pa i tisuću dimenzija.

4.2 Duboke neuronske mreže

Kako bi razumjeli kako funkcioniraju poznate arhitekture koje generiraju vektorske ugradnje, poput BERT-a, ResNet50-a ili Inception-a, potrebno je detaljnije objasniti same pojmove dubokog učenja i dubokih neuronskih mreža. Duboko učenje bazirano je na dubokim neuronskim mrežama (engl. Deep Neural Networks - DNNs) koje se sastoje od niza slojeva između input i output sloja - tkz. skrivenih slojeva. Riječ „duboko“ u dubokom učenju se referira na dodatne skrivene slojeve između *input* i *output* sloja. *Input* slojevi predstavljaju ulazne podatke, dok *output* slojevi predviđaju krajnji rezultat. Većina računanja se odvija upravo u srednjim, skrivenim slojevima. U pravilu, svaka neuronska mreža s više od 3 sloja, uključujući *input* i *output* sloj, može se smatrati DNN modelom. Slika 11 prikazuje strukturu duboke neuronske mreže.

Slika 11. Prikaz strukture duboke neuronske mreže



Izvor: https://www.ibm.com/cloud/learn/neural-networks

Općenito, rad svih neuronskih mreža možemo podijeliti u dvije faze:

1. Faza učenja (treniranja)
2. Faza obrade podataka (inferencije, iskorištavanja)

Faza učenja je iterativan postupak gdje algoritam u svakoj iteraciji podešava vrijednosti težinskih faktora veza između neurona. Kod DNN modela, podaci se tipično kreću od *input* prema *output* sloju bez petlji unazad.

Algoritam rada DNN modela objasnit ćemo na primjeru neuronske mreže koja predviđa životni vijek u nekoj državi temeljem pokazatelja razvoja poput: BDP-a, stope nezaposlenosti, minimalne plaće i rasta BDP-a. Svaki ulazni podatak možemo definirati kao vektor gdje svaki element predstavlja određeni pokazatelj. Kao primjer ćemo uzeti 4 navedena pokazatelja za Republiku Hrvatsku.

Republika Hrvatska - pokazatelji razvijenosti

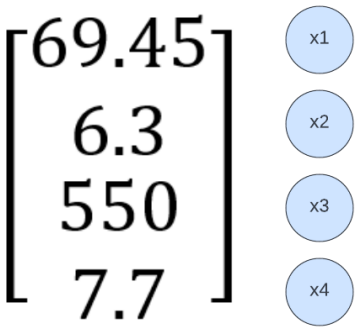
1. **Nominalni BDP** – 69.45 milijardi dolara (x_1)
2. **Stopa nezaposlenosti** – 6.3% (x_2)
3. **Minimalna neto plaća** – 550 EUR (x_3)
4. **Rast BDP-a** – 7.7% (x_4)

Izvor: Državni zavod za statistiku

$$x_{cro} = \begin{bmatrix} 69.45 \\ 6.3 \\ 550 \\ 7.7 \end{bmatrix}$$

U sljedećem koraku ćemo prikazati vrijednosti parametara vektora x_{cro} u obliku neurona koji pohranjuju te vrijednosti. Slika 12 prikazuje primjer *input* vektora i 4 neurona.

Slika 12. Input vektor

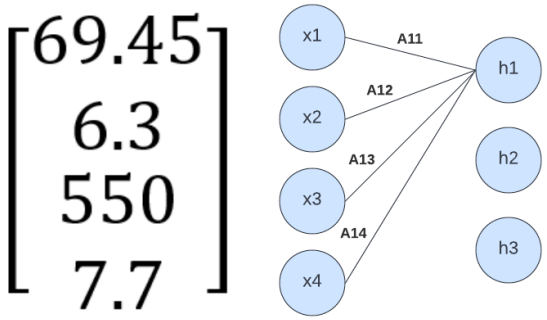


Izvor: autor

Dalje, dodat ćemo novi čvor koji ćemo označiti sa h_1. To je čvor skrivenog sloja neuronske mreže te je jednak vektoru koji nastaje linearnom kombinacijom *input* vektora i težinskih faktora ($A_{11}, A_{12}, A_{13}\ i\ A_{14}$). Matematički možemo izraziti kao sljedeću jednadžbu:

$$h_1 = A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + A_{14}x_4$$

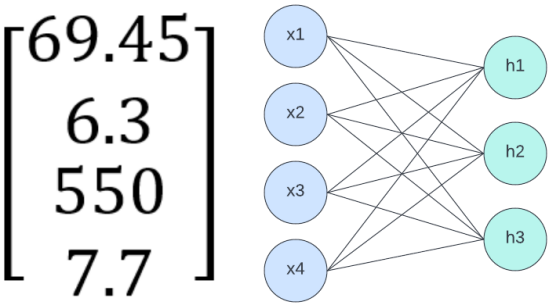
Slika 13. Veze između h1 skrivenog neurona i input vektora



Izvor: autor

Istu stvar ćemo ponoviti za čvorove h_2 i h_3.

Slika 14. Sve veze između input vektora i skrivenog h vektora



Izvor: autor

Time dobivamo sustav 3 linearne jednadžbe:

$$h_1 = A_{11}x_1 + A_{12}x_2 + A_{13}x_3 + A_{14}x_4$$

$$h_2 = A_{21}x_1 + A_{22}x_2 + A_{23}x_3 + A_{24}x_4$$

$$h_3 = A_{31}x_1 + A_{32}x_2 + A_{33}x_3 + A_{34}x_4$$

Vektor h također možemo izraziti kao umnožak vektora x i matrice težinskih faktora A.

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

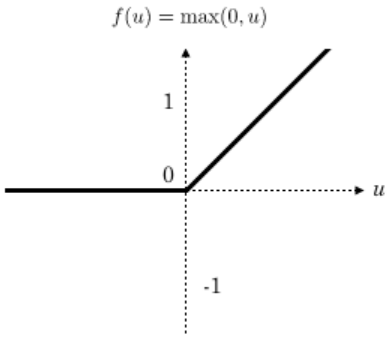
$$\boldsymbol{h = Ax}$$

Međutim, u realnom okruženju je potrebno umnožak vektora i težinskih faktora Ax omotati takozvanom aktivacijskom funkcijom koja je zadužena za donošenje odluke hoće li se sljedeći neuron „upaliti“.

„*Algoritam za računanje vrijednosti aktivacije neurona kao funkcije njenog ukupnog ulaza. Ukupni ulaz je tipično težinska suma ulaza u neuron.*“ (Eberhart, 1990).

Nelinearna aktivacijska funkcija je funkcija koja ograničava izlaz neurona na interval [0,1] - kako bi se spriječila linearnost rezultata. Tipično ju označavamo oznakom φ (fi). Ovisno o arhitekturi i tipu neuronske mreže, koriste se različite aktivacijske funkcije. Jedna od najpoznatijih aktivacijskih funkcija je funkcija **rectifier** ili **ReLU** (engl. Rectified Linear Unit) (Brownlee, 2019)**.** Graf ReLU funkcije možemo vidjeti na slici 15.

Slika 15. ReLU aktivacijska funkcija



Izvor: https://www.researchgate.net/figure/ReLU-activation-function_fig3_319235847

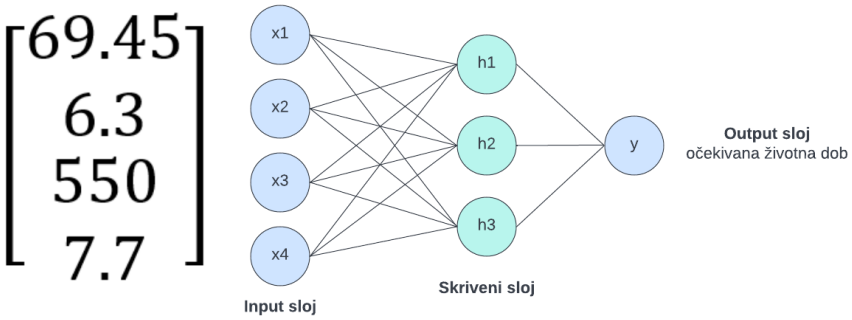
Stoga ćemo vektor h ćemo izraziti kao:

$$\boldsymbol{h = f_1(Ax)}$$

Posljednji *output* čvor y (očekivana životna dob) jednak je umnošku vektora h i matrice B težinskih koeficijenata, omotanih aktivacijskom funkcijom. Slika 16 prikazuje osmišljeni DNN model.

$$\boldsymbol{y = f_2(Bh)}$$

Slika 16. Prikaz osmišljene neuronske mreže za predviđanje očekivanje životne dobi



Izvor: autor

Naposlijetku, cjelokupnu funkciju možemo zapisati kao:

$$\boldsymbol{y = f_2(Bf_1(Ax))}$$

Odabir A i B parametara se odnosi na treniranje neuronske mreže. Unosom testnih podataka (koji se sastoje od ulaznog vektora temeljnih pokazatelja razvoja države i korespondirajućeg izlaznog vektora prosječne životne dobi), neuronsku mrežu treniramo te ju naposlijetku možemo i koristiti za obradu podataka. Preciznost neuronske mreže ovisi o kvaliteti i količini testnih podataka. Ukoliko mreža ne prepozna precizno određeni uzorak, algoritam će prilagođavati vrijednosti težinskih faktora dok to ne bude slučaj. Na taj način algoritam „uči" tj. pronalazi odgovarajuće parametre sve dok mreža ne počne davati precizne i točne rezultate.

U ovom primjeru obrađen je temeljni princip rada jednostavne neuronske mreže. Međutim, modeli dubokih neuronskih mreža (DNN) sadrže više od jednog skrivenog sloja te se samim time drastično povećava vrijeme treniranja, koje može trajati tjednima, mjesecima pa i godinama. Osim velikog vremenskog okvira, potrebna je i velika količina podataka za treniranje (samim time i hardverskih resursa) kako bi DNN modeli bili učinkovitiji od ostalih tehnika strojnog učenja. Glavna prednost je upravo njihovo automatsko podešavanje i prilagođavanje kako bi servirali optimalne rezultate. Još jedna od velikih prednosti je fleksibilnost modela, odnosno sposobnost modela da se primjeni na više različitih aplikacija i tipova podataka (Developers Google, n.d.).

Kako je već spomenuto, kod DNN modela podaci kolaju od ulaznog prema izlaznom sloju bez vraćanja te su veze između neurona jednosmjerne (prema naprijed). *Recurrent Neural Network* (RNN) je oblik duboke neuronske mreže koji rješava taj nedostatak na način da veze između neurona mogu imati cikluse, dopuštajući pritom da izlaz iz nekih neurona utječe na kasniji unos (u sljedećim iteracijama) u iste čvorove. RNN modeli idealni su za sekvencijalne podatke budući da se njima može oponašati način na koji čovjek „povezuje nizove". Ta karakteristika čini model idealnim za analizu teksta i govora. Primjerice, neki od scenarija upotrebe su prepoznavanje govora i prepoznavanje rukopisa (IBM, n.d.).

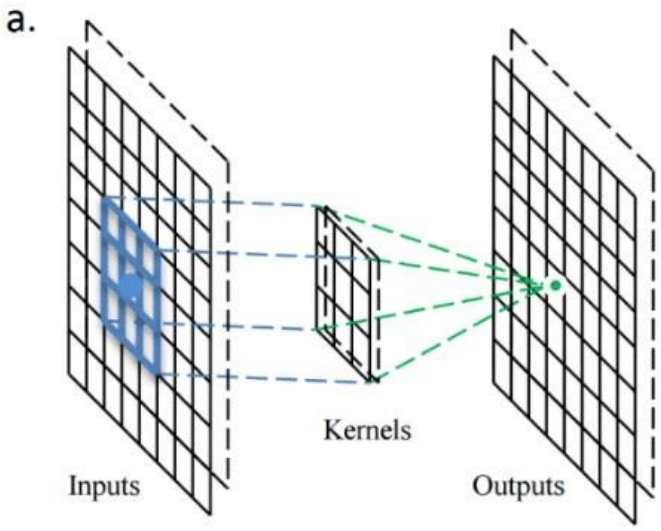
Međutim, kako se ovaj rad bavi prvenstveno vektorskim bazama podataka i vektorskim reprezentacijama nestrukturiranih podataka, u sljedećem poglavlju će se obraditi posebna klasa neuronskih mreža koja se pretežito koristi u *computer vision* području te za sustave preporuke (Tutorialspoint, n.d.).

4.3 Konvolucijske neuronske mreže

Konvolucijska neuronska mreža (engl. Convolutional neural network – CNN ili ConvNet) je vrsta modela duboke neuronske mreže koja služi za obradu podataka koji imaju tkz. *grid pattern*, poput slika. Princip rada se temelji na postepenoj obradi manjih dijelova podatka, umjesto cijelog odjednom. Arhitektura se sastoji od višeslojnih perceptora u više dimenzija te se pokazala najboljom za obradu nestrukturiranih podataka. Svaki sloj u CNN modelu sadrži niz filtera koji se nazivaju konvolucijske jezgre. Filtar je matrica cijelih brojeva koja se koristi na podskupu ulaznih vrijednosti piksela, iste veličine kao jezgra. Filtri mogu započeti s vrlo jednostavnim značajkama slike (poput svjetline i rubova) te postepeno povećavati složenost do značajki koje jedinstveno definiraju neki objekt (mathworks, n.d.). Svaki piksel se množi s korespondirajućom vrijednosti u jezgri, a rezultat se potom zbraja radi jednostavnosti prezentiranja ćelije mreže (Hons, 2019).

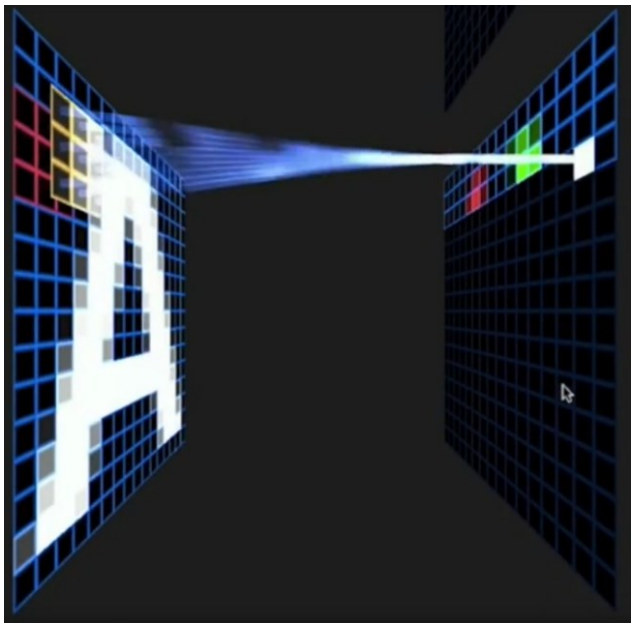
U *computer vision* području, ulazni podatak je često tro-kanalna RGB slika. Međutim, radi pojednostavljenja kao primjer uzmimo crno bijelu sliku koja ima 1 kanal (dvodimenzionalna matrica) i 3x3 konvolucijsku jezgru. Jezgra „korača“ preko ulazne matrice prolazeći horizontalno redak po redak, skenirajući preko prvih redaka u matrici koji sadrže vrijednosti piksela slike. Zatim se jezgra okomito spušta prema sljedećim redovima. Slike 17 i 18 prikazuju opisani postupak rada konvolucijske jezgre.

Slika 17. Ilustracija principa rada konvolucijske jezgre



Izvor: https://www.researchgate.net/figure/a-Illustration-of-the-operation-principle-of-the-convolution-kernel-convolutional-layer_fig2_309487032

Slika 18. Vizualizacija - Konvolucijska jezgra skenira vrijednosti u ulaznoj matrici (koja predstavlja slovo A)



Izvor: https://www.youtube.com/watch?v=f0t-OCG79-U

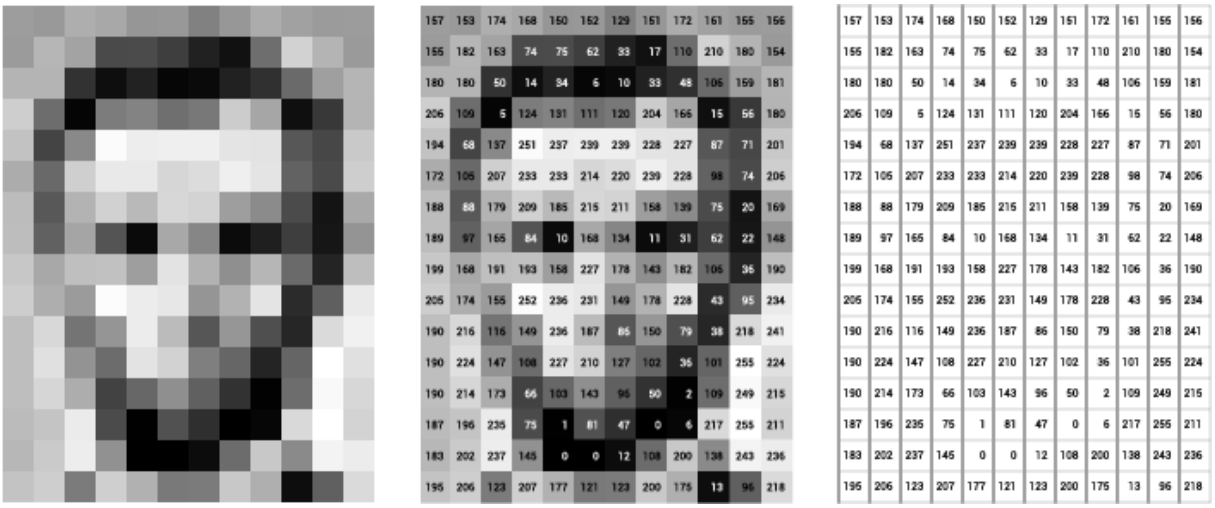
Kako bi se obradili rubni pikseli, postoji nekoliko tehnika. Jedna od njih je *reflection padding* gdje se broj piksela potreban da konvolucijska mreža obradi rubne piksele dodaje na vanjsku stranu kopirajući piksele s ruba slike. Kod 3x3 konvolucijske jezgre, jedan piksel treba dodati izvana, dok bi se kod primjerice 7x7 jezgre tri piksela reflektirala izvana. Kada se više jezgara konvolucijskih mreža primjeni unutar konvolucijskog sloja, stvaraju se mape kanal/značajki – jedna za svaki konvolucijski sloj. Slika 19 prikazuje vizualizaciju tih mapa (Hons, 2019).

Slika 19. Vizualizacija mapa kanala/značajki

Izvor: https://www.youtube.com/watch?v=f0t-OCG79-U

Razmotrimo li sljedeći primjer na Slici 20, krajnje lijevo možemo vidjeti portret osobe u crno-bijelim pikselima. Središnja slika sadrži istu sliku s vrijednostima piksela u rasponu od 0 do 255 gdje 0 predstavlja crnu, a 255 bijelu boju. Krajnje desno možemo vidjeti dvodimenzionalnu matricu. Može se primijetiti da vrijednosti u matrici definiraju dvodimenzionalnu vektorsku reprezentaciju slike (Tripathi, n.d.).

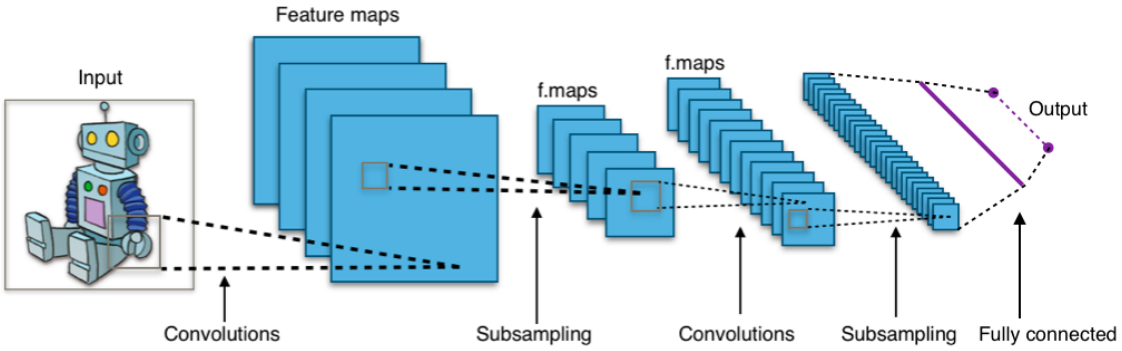
Slika 20. Primjer vektorske reprezentacije slike pomoću CNN modela



Izvor: https://www.pinecone.io/learn/vector-embeddings/

Isti princip može se primijeniti na ostale oblike nestrukturiranih podataka. Naravno, ovisno o složenosti podataka, broj dimenzija se drastično povećava. Tako da je normalno da se vektorska reprezentacija sastoji od nekoliko stotina pa i tisuća dimenzija. Ovakve vektorske ugradnje su odlične jer zadržavaju semantičke informacije o susjednim pikselima, međutim osjetljive su na transformacije slike poput rotacija, rezanja i sl. Slika 21 prikazuje tipičnu CNN arhitekturu. Rezultirajuća vektorska reprezentacija se dobiva iz potpuno povezanog sloja.

Slika 21. CNN arhitektura



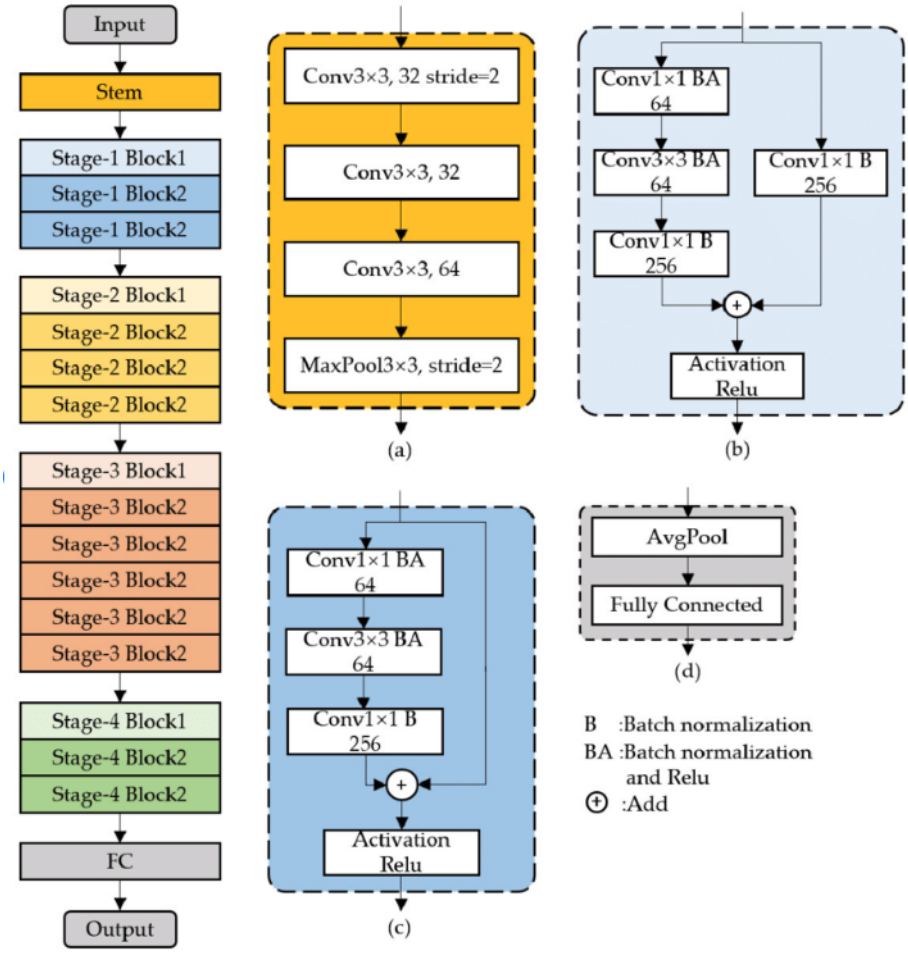
Izvor: https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Typical_cnn.png

Jedna od najčešće korištenih funkcija u konvolulcijskim neuronskim mrežama je prethodno spomenuta *ReLU* aktivacijska funkcija budući da omogućava brže i lakše učenje dubokih neuronskih mreža nad velikim i kompleksnim skupom podataka.

Što se tiče učenja, najčešće korišteni algoritam za učenje konvolucijskih neuronskih mreža je *backpropagation* algoritam koji je dizajniran da pronalazi pogreške povratkom na prethodne čvorove.

U praktičnom dijelu rada koji je dokumentiran u 6. poglavlju, za kodiranje nestrukturiranih podataka (slika u ovom slučaju) koristi se **resnet50** konvolucijska neuronska mreža koja je 50 slojeva duboka. Mreža je istrenirana sa preko milijun slika te omogućava klasifikaciju slika u 1000 različitih kategorija (mathworks, n.d.). Mreža se danas smatra jednim od standarda u *computer vision* području. Slika 22 prikazuje složenu arhitekturu resnet50 mreže.

Slika 22. Arhitektura resnet50 mreže



Izvor: https://www.researchgate.net/figure/The-architecture-of-ResNet-50-vd-a-Stem-block-b-Stage1-Block1-c-Stage1-Block2_fig4_349646156

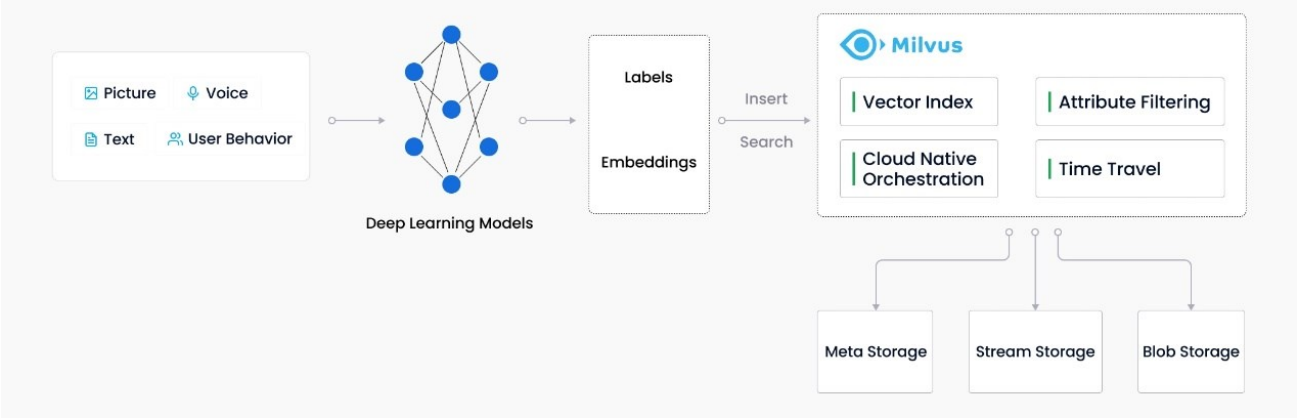
5. VEKTORSKE BAZE PODATAKA

Kako im i sam naziv nalaže, vektorske baze podataka pohranjuju vektore odnosno vektorske reprezentacije nestrukturiranih podataka. Pogodne su za brz pristup podacima i pretraživanje po sličnosti koristeći algoritme pretraživanja vektorskog prostora. Vektorske baze podataka omogućavaju standardne *CRUD* operacije (*create*, *read*, *update* i *delete*) nad nestrukturiranim podacima.

5.1 Milvus.io

Milvus.io vektorska je baza podataka otvorenog koda dostupna online. Izrađena je 2019. godine s jedinstvenim ciljem pohrane, indeksiranja i upravljanja velikim količinama vektorskih ugradnji koje generiraju neuronske mreže i ostali modeli strojnog učenja. Koriste ju mnoge poznate kompanije poput Ebay-a, Walmart-a i Ikee. Na slici 23 može se vidjeti tijek rada Milvus baze podataka. Od nestrukturiranih podataka koji se obrađuju modelima dubokog učenja, do pretraživanja i ubacivanja generiranih vektorskih ugradnji u samu bazu (Milvus, n.d.).

Slika 23. Milvus workflow



Izvor: https://milvus.io/docs/v2.0.x/overview.md

5.1.1 Indeksiranje

Indeksiranje je proces učinkovitog organiziranja podataka i igra važnu ulogu u implementaciji pretraživanja po sličnosti budući da dramatično ubrzava dugotrajne upite nad velikim skupovima podataka. Milvus omogućava specifikaciju različitih indeksa za svako vektorsko polje. Većina indeksa koje Milvus podržava koriste ANN algoritam pretraživanja vektorskog prostora. U usporedbi s preciznim dohvaćanjem koje je u pravilo vrlo sporo, ANN algoritam će tražiti samo susjede vektora kojeg pretražujemo. Tablica 1 prikazuje nekoliko najčešćih indeksa koje Milvus podržava i njihove scenarije upotrebe.

Tablica 1. Tipovi indeksa koje podržava Milvus.io

Podržani indeks	Klasifikacija	Scenarij upotrebe
FLAT	N/A	scenariji koji traže savršeno točne i egzaktne rezultate pretraživanja na malom skupu podataka
IVF_FLAT	quantization-based index	najprikladniji za scenarije koji traže idealnu ravnotežu između točnosti i učinkovitosti
IVF_SQ8	quantization-based index	scenariji koji teže značajnom smanjenju potrošnje memorije diska, CPU-a i GPU-a
IVF_PQ	quantization-based index	scenariji koji zahtijevaju veliku brzinu upita žrtvujući preciznost
HNSW	graph-based index	scenariji koji zahtijevaju visoku učinkovitost
ANNOY	tree-based index	scenariji koji zahtjevu visoku preciznost

Izvor: https://milvus.io/docs/v2.0.x/overview.md

5.1.2 Metrike sličnosti

Metrike sličnosti koriste se za mjerenje sličnosti između vektora. Kao i kod indeksa, odabir primjerene metrike poboljšava točnost i performanse klasifikacije i klasteriranja. Metrika se odabire ovisno o tipu ulaznih podataka. Tablica 2 prikazuje metrike sličnosti koje Milvus podržava.

Tablica 2. Metrike sličnosti koje podržava Milvus.io

Metrika	Scenarij upotrebe	Tip vektora
Euclidian distance (L2)	generalno se koristi u području *computer vision* (CV) području UI	Floating-point
Inner product (IP)	generalno se koristi u *natural language processing* (NLP) području UI	Floating-point
Hamming	generalno se koristi u *natural language processing* (NLP) području UI	Binarni
Jaccard	generalno se koristi za pretraživanje sličnosti molekularnih struktura	Binarni
Tanimoto	generalno se koristi za pretraživanje sličnosti molekularnih struktura	Binarni
Superstructure	generalno se koristi za pretraživanje sličnosti nadgrađa molekula	Binarni
Substructure	generalno se koristi za pretraživanje sličnosti nadgrađa molekula	Binarni

Izvor: https://milvus.io/docs/v2.0.x/overview.md

5.2 Pretraživanje vektorskog prostora

Pretraga vektora po sličnosti (engl. Vector similarity search) je proces usporedbe vektora s vektorskom bazom podataka s ciljem pronalaska sličnih vektora. Algoritmi pretraživanja poput KNN (k-nearest neighbors) i ANN (approximate nearest neighbor) se koriste za izračunavanje udaljenosti između vektora te ubrzavanje procesa pretraživanja (Tibshirani, 2022). Ako su dvije vektorske ugradnje slične, znači da su i originalni podaci također slični. Neki od primjera uporabe takvih algoritama:

- de-duplikacija podataka (engl. data deduplication),
- sustav za preporuke (engl. recommender system),
- detekcija anomalija (engl. anomaly detection),
- pretraživanje slika po sličnosti (engl. reverse image search).

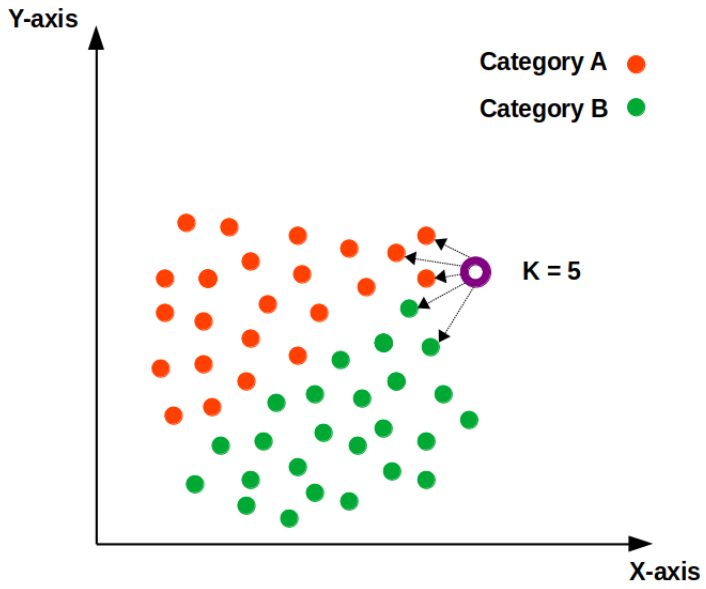
5.2.1 K-nearest neigbours (KNN)

Algoritam **k-najbližih susjeda** (engl. k-nearest neigbours - KNN) poznati je klasifikacijski algoritam koji se koristi u strojnom učenju za klasifikaciju i regresiju. To je neparametarska metoda nadziranog učenja koja računa blizinu objekata za izradu klasifikacija. Najčešće koristi algoritam euklidske udaljenosti za računanje udaljenosti (IBM, n.d.). Kako se radi o nadziranom učenju, algoritam unaprijed zahtjeva skup podataka kako bi odradio klasifikaciju. Međutim, KNN metoda ne zahtjeva prethodno treniranje, već pri pokretanju pretražuje cijeli skup podataka i pamti ga (engl. *lazy learning algorithm*) (Raj, 2021).

Prvi korak KNN algoritma je inicijalizacija. Slika 24. prikazuje dvodimenzionalni prostor s već raspoređenim točkama iz *Category A* i *Category B* klasa. Kako bi ispravno klasificirali novu točku potrebno je izračunati udaljenost k najbližih susjeda od te točke (u primjeru $k = 5$), primjerice kroz funkciju euklidske udaljenosti. Vrijednost k je proizvoljna, no optimalna k vrijednost je u pravilu drugi korijen broja uzoraka N.

$$k = \sqrt{N}$$

Slika 24. Princip klasifikacije KNN algoritma



Izvor: https://ai.plainenglish.io/why-is-the-k-nearest-neighbors-knn-called-a-lazy-algorithm-ba165f723005

Algoritam klasificira novu točku tako što uspoređuje klase k najbližih susjeda točke te bira onu klasu koja ima najveći broj susjeda. U primjeru iznad vidi se da većina susjeda spada u klasu A pa će i nova točka biti svrstana u tu klasu.

5.2.2 Approximate nearest neighbours (ANN)

Većina indeksa koje Milvus baza podataka podržava koriste **approximate nearest neighbours** algoritam pretraživanja vektorskog prostora. Algoritam je nešto složeniji od KNN algoritma i razlikuje se u fazi predikcije. U KNN fazi predikcije, svi elementi (susjedi) su uključeni prilikom pretraživanja k najbližih susjeda, dok kod ANN algoritma pretraživanje započinje nad manjim podskupom podataka (Ignite Apache, n.d.). U kontekstu pretraživanja vektora, temeljna ideja ANN pretraživanja nije vraćanje najtočnijeg rezultata, već samo traženje susjeda ciljanog vektora. ANN algoritam poboljšava učinkovitost žrtvujući točnost unutar prihvatljivog raspona (Milvus, n.d.).

5.3 Primjene vektorskih baza podataka

Vektorske baze podataka koriste se za mnoga programska rješenja gdje je potrebna obrada nestrukturiranih podataka pretraživanjem vektorskih ugradnji po sličnosti. Taj postupak, uz odabir odgovarajućih indeksa i metrika sličnosti, omogućava implementaciju mnogih korisnih aplikacija.

Semantičko pretraživanje jedna je od tehnika pretraživanja podataka koja za cilj ima, osim standardne pretrage ključnih riječi, određivanje konteksta i semantike u rečenicama ili riječima koje korisnik koristi pri pretraživanju. Vektorske baze podataka mogu u tome pomoći na način da pohranjuju vektorske ugradnje NLP modela. „Pinecone.io“ primjer je vektorske baze podataka upravo za takve namjene. Uz pomoć NLP modela, omogućuje semantičku pretragu riječi, rečenica ili pak cijelih dokumenata (Briggs, n.d.).

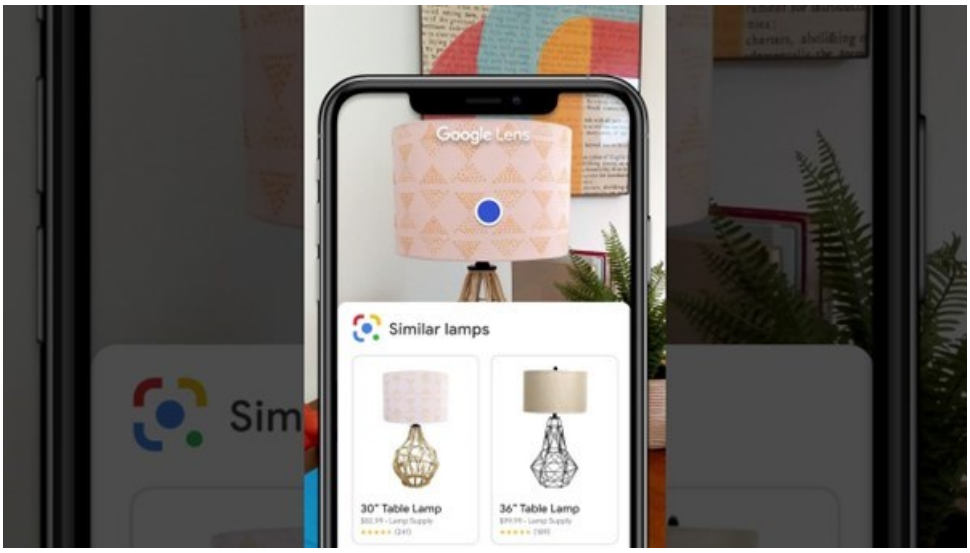
Pretraživanje po sličnosti (engl. Similarity search) je termin koji se koristi za skup principa i tehnika za klasifikaciju i klasteriranje podataka. Pretraživanje nestrukturiranih podataka po sličnosti u tradicionalnim relacijskim bazama podataka bilo bi vrlo nezgrapno jer zahtjeva ručno dodavanje ključnih riječi, opisa i drugih metapodataka za svaki pojedini objekt. Pretraživanje i usporedba vektorskih ugradnji rješava upravo taj nedostatak budući da omogućava automatsku klasifikaciju kompleksnih podataka.

Neki od najčešćih scenarija upotrebe pretraživanja po sličnosti su:

- pretraživanje slika po sličnosti – omogućava brzu i preciznu pretragu slika te dohvaćanje sličnih iz istreniranog skupa podataka. Primjer jedne takve aplikacije je „Google Lens“ koja uz to što pretražuje slične fotografije, pretražuje i samu fotografiju u Google-ovoj tražilici te izbacuje poveznice na prodajna mjesta i druge web stranice gdje je pronađen

sličan, ali vrlo često i identičan proizvod. Slika 25 prikazuje sučelje Google Lens aplikacije.

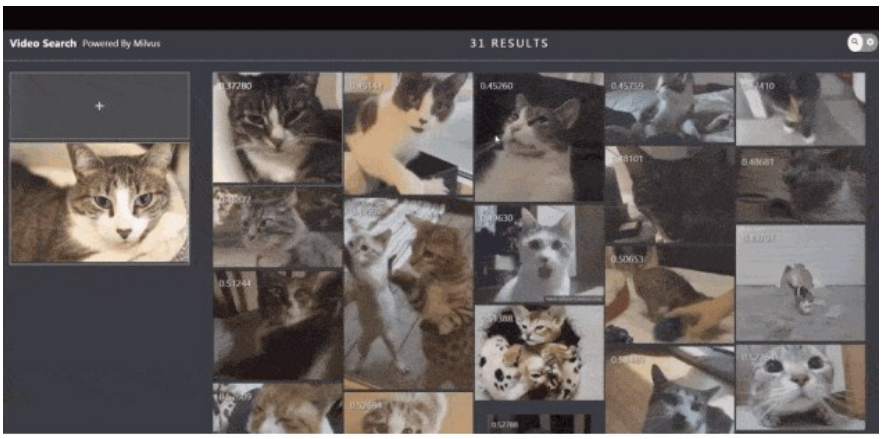
Slika 25. Sučelje Google Lens aplikacije.



Izvor: https://www.youtube.com/watch?v=26mCdeuvnzI

- pretraživanje videozapisa po sličnosti – pretvorbom ključnih kadrova u vektore moguće je pretraživati ili preporučiti korisniku slične videozapise u realnom vremenu. Takva aplikacija omogućava prepoznavanje različitih videozapisa, primjerice filmova i serija. Korisnici mogu učitati foto ili video isječak iz videozapisa ili filma te zatim dobiti natrag videozapise ili filmove koji sadrže slične isječke. Slika 26 prikazuje primjer aplikacije za pretragu videozapisa po sličnosti.

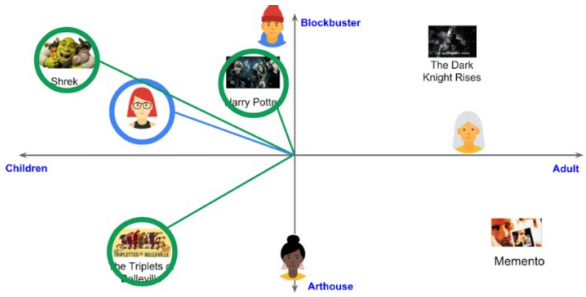
Slika 26. Prikaz aplikacije za pretragu videozapisa po sličnosti izrađene pomoću Milvus.io



Izvor: https://milvus.io/docs/video_similarity_search.md

- Sustavi za preporuke (engl. Recommender systems) – radi se o sustavima za filtriranje informacija koji za cilj imaju previdjeti preferencije korisnika i preporučiti korisniku: stavke koje ga zanimaju, koje bi mogao dobro ocijeniti ili koje bi kupio. Danas ih koriste mnoge velike kompanije kako bi unaprijedile svoje poslovanje. Neki od poznatijih sustava preporuke su: Amazon-ov sustav koji predlaže proizvode za kupnju, Spotify-ev sustav preporuke glazbe te Netflix-ov sustav preporuke filmova ili serija koji bi nam se mogli svidjeti. Slika 27 ilustrira princip rada sustava za preporuke filmova u 2 dimenzije:
    1. x-os je ciljana dobna skupina,
    2. y-os je tip filma namijenjen masi ili eksperimentalni, intimni film autora namijenjen manjem broju ljudi.

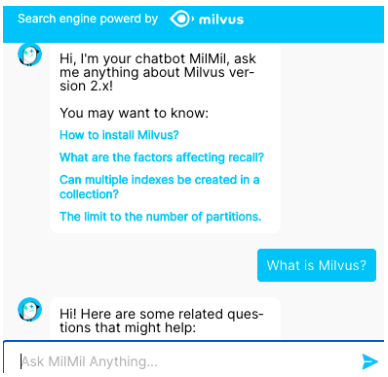
Slika 27. Prikaz dvodimenzionalnog prostora vektora koji za filmske preporuke



Izvor: https://developers.google.com/machine-learning/recommendation/collaborative/basics

- Chatbot (Question answering system) – aplikacija koja pruža automatizirani pristup odgovaranja na pitanja koja čovjek postavlja prirodnom jezikom. Ovakav tip aplikacije može se sve češće pronaći na web stranicama. Slika 28 prikazuje primjer *QA chatbot-a* izrađenog pomoću Milvus.io baze i BERT modela strojnog učenja za NLP.

Slika 28. Primjer Chatbot-a izrađenog u Milvus.io

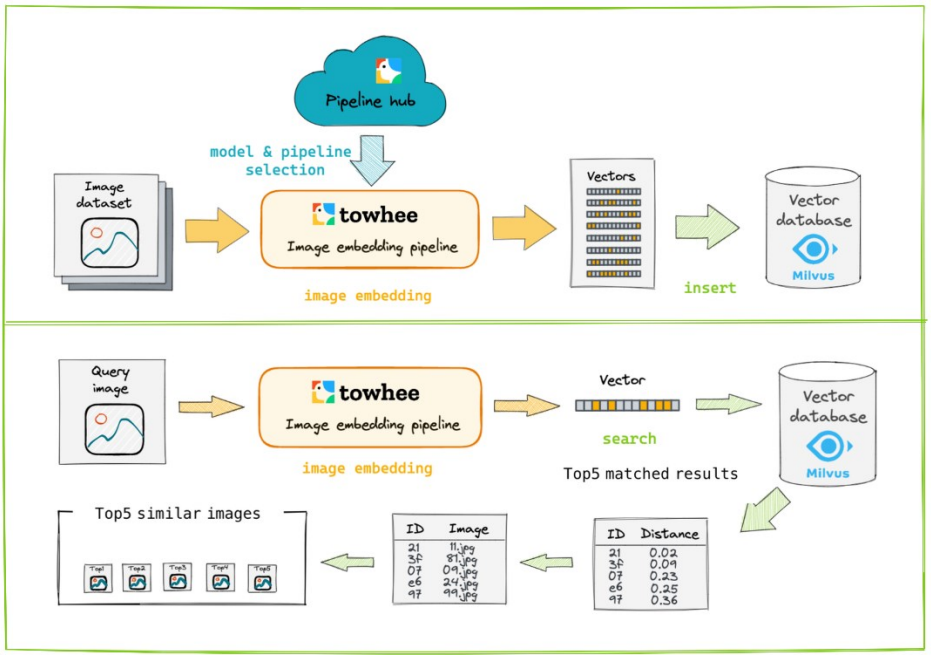


Izvor: https://milvus.io/docs/v2.0.x/question_answering_system.md

6. REVERSE-IMAGE-SEARCH IMPLEMENTACIJA

Za potrebe ovog završnog rada izrađena je jednostavna aplikacija za pretraživanje slika po sličnosti u Pythonu koristeći *Milvus.io* vektorsku bazu podataka i *Towhee.io* modul za strojno učenje koji omogućava kodiranje nestrukturiranih podataka u vektorske ugradnje (Milvus, n.d.). Slika 29 prikazuje skicu implementacije aplikacije. Kroz nekoliko sljedećih poglavlja proći će se kroz postupak instalacije i pokretanja Milvus baze, pregled osnovnih funkcionalnosti te korake implementacije same aplikacije.

Slika 29. Skica implementacije aplikacije prepoznavanja slika po sličnosti



Izvor: https://codelabs.towhee.io/build-a-reverse-video-search-engine-in-minutes/index#0

Izvorni kod aplikacije može se pronaći u Github repozitoriju na sljedećoj poveznici:
https://github.com/lukablaskovic/reverse-image-search

Važno je prije instalacije same baze provjeriti hardverske zahtjeve koji su definirani u dokumentaciji.

6.1 Instalacija Milvus.io baze podataka

Postoji više načina instalacije Milvus baze podataka. Ovdje će se prikazati instalacija Milvus *Standalone* paketa koristeći **Docker Compose**. Radi se o inačici Dockera koja podržava čitanje konfiguracijskih podataka iz YAML datoteke te dopušta konfiguraciju i pokretanje više od

jednog kontejnera od jednom (u usporedbi sa klasičnim Dockerom koji može pokretati samo 1 kontejner istovremeno). Bazu podataka je moguće pokrenuti i putem Kubernetees sustava.

Prvi korak je preuzimanje *milvus-standalone-docker-compose.yml* datoteke te njeno preimenovanje u *docker-compose.yml* kako bi ju Docker Compose prepoznao. Datoteka sadrži sve potrebne informacije kako bi Docker Compose ispravno izvršio orkestraciju. To je moguće učiniti ručno preuzimanjem datoteke sa Milvus.io dokumentacije ili kroz sljedeću naredbu u terminalu:

```
$ wget https://github.com/milvus-io/milvus/releases/download/v2.1.0/milvus-standalone-docker-compose.yml -O docker-compose.yml
```

Nalazeći se u istom direktoriju kao i *docker-compose.yml* datoteka, pokrećemo Milvus bazu podataka sa sljedećom naredbom:

```
sudo docker-compose up -d
```

Nakon toga, ako je sve prošlo u redu, Docker Compose bi trebao pokrenuti 3 Milvus kontejnera. To se može provjeriti sljedećom naredbom:

```
sudo docker-compose ps
```

6.2 Instalacija potrebnih Python paketa

Milvus nudi mogućnosti implementacije koristeći Python i Node.js. Ovdje će se prikazati instalacija koristeći Python i njegov $pymilvus$ paket za rad s Milvus bazom. Potrebno je koristiti Python 3.71 verziju ili stariju, međutim korištena je 3.8.10 verzija budući da se pokazala najstabilnijom.

Dalje, potrebno je instalirati sljedeće dodatne pakete za implementaciju same aplikacije:

- $towhee$ – modul za strojno učenje koji omogućava kodiranje nestrukturiranih podataka u vektore
- $gradio$ – paket koji nudi sučelje za izgradnju i javnu objavu aplikacije na web
- $opencv-python$ – paket koji služi za rad u CV (computer vision) području
- $pillow$ – paket koji dodaje osnovne funkcije obrade slike u Python prevoditelj
- $pyarrow$ – razvojna platforma koja sadrži set tehnologija za rad sa velikim podacima

Sve ove pakete moguće je instalirati istovremeno sljedećom naredbom:

```
pip install -q pymilvus towhee gradio opencv-python pillow pyarrow
```

Također, koristit će se Jupyter Notebook budući da omogućava pregledniji i lakši rad s većim skupom podataka i pregled slika. Jupyter Notebook možemo instalirati sljedećom naredbom:

pip install notebook

6.3 Priprema podataka

Preuzet ćemo dataset koji se sastoji od 100 klasa i 10 slika za svaku klasu, ukupno 1000 slika. Dataset je organiziran u 3 dijela:

- $train$ – podaci za treniranje
- $test$ – podaci za testiranje
- $reverse\_image\_search.csv$ – datoteka koja sadrži ID svake slike, $path$ gdje je slika pohranjena te $label$, odnosno oznaku slike

Dataset se može preuzeti i raspakirati pomoću sljedećih naredbi:

curl -L https://github.com/towhee-io/examples/releases/download/data/reverse_image_search.zip -O

unzip -q -o reverse_image_search.zip

U našem projektnom direktoriju napravit ćemo novu $\boldsymbol{data.py}$ datoteku iz koje ćemo dohvaćati lokalno spremljeni dataset, prikazat ćemo nekoliko prvih redaka kako bi se uvjerili jesu li podaci ispravno pročitani ($pandas$ paket) te ćemo implementirati funkciju $read\_images$ koju ćemo kasnije koristiti za vraćanje sličnih slika. Slika 30 prikazuje *data.py* izvorni kod.

Slika 30. Prikaz koda - data.py

```
import pandas as pd
import cv2
from towhee._types.image import Image

source = 'D:/data/reverse_image_search.csv'
#Read data
df = pd.read_csv(source)
print(df.head())
id_img = df.set_index('id')['path'].to_dict()

#Helper function - returns found images that are similar to the test image
def read_images(results):
    imgs = []
    for re in results:
        path = id_img[re.id]
        imgs.append(Image(cv2.imread(path), 'BGR'))
    return imgs
```

6.4 Spajanje na Milvus

U novoj ***db.py*** datoteci, koristeći *pymilvus* paket, spojit ćemo se na bazu koja se lokalno pokreće na portu 19530. Deklarirat ćemo funkciju koja izrađuje novu kolekciju u Milvusu te shemu koja se sastoji od *ID* i *embedding* polja za identifikatore slika i njihove korespondirajuće vektorske ugradnje.

Nakon toga ćemo izraditi indeks za novokreiranu kolekciju. Kao metriku sličnosti odabrat ćemo *L2* – euklidsku udaljenost i tip indeksa *IVF_FLAT* koji nam daje idealnu ravnotežu između preciznosti i efikasnosti. Slika 31 prikazuje *db.py* datoteku s programskim kodom.

Slika 31. Prikaz koda - db.py

```
from pymilvus import connections, FieldSchema, CollectionSchema, DataType, Collection, utility

#Milvus connection
connections.connect(host='127.0.0.1', port='19530')

def create_milvus_collection(collection_name, dim):
    if utility.has_collection(collection_name):
        utility.drop_collection(collection_name)

    fields = [
    FieldSchema(name='id', dtype=DataType.INT64, descrition='ids', is_primary=True, auto_id=False),
    FieldSchema(name='embedding', dtype=DataType.FLOAT_VECTOR, descrition='vectors', dim=dim)
    ]
    schema = CollectionSchema(fields=fields, description='reverse image search')
    collection = Collection(name=collection_name, schema=schema)

    # create IVF_FLAT index for collection
    index_params = {
        'metric_type':'L2',
        'index_type':"IVF_FLAT",
        'params':{"nlist":2048}
    }
    collection.create_index(field_name="embedding", index_params=index_params)
    return collection
```

6.5 Učitavanje vektora u Milvus

Sada kada imamo poveznicu na bazu podataka i funkciju za izradu novu kolekcije, možemo napraviti ***loader.py*** datoteku koja će služiti za pohranu vektorskih ugradnji u bazu. Prvi korak je izrada nove kolekcije koju ćemo nazvati „reverse_image_search“ i podesiti dimenziju vektora na 2048. Za samo učitavanje i dekodiranje podataka koristit ćemo *towhee framework*.

U nastavku se nalazi objašnjenje towhee funkcija za kodiranje podataka:

- **runas_op[**'id', 'id'](func=lambda x: int(x)) → za svaki redak pretvori tip podataka u stupcu ID iz str u int
- **image_decode**['path', 'img']() → za svaki redak, pročitaj i dekodiraj sliku koja se nalazi na $path$ lokaciji i u novi redak img spremi podatke o pikselima
- **image_embedding.timm**['img', 'vec'](model_name='resnet50') → Generiraj vektorske ugradnje koristeći $resnet50$ CNN
- **to_milvus**['id', 'vec'](collection=collection, batch=100) → spremi vektorske ugradnje slike u Milvus bazu podataka

Slika 32 prikazuje datoteku *loader.py* s izvornim kodom.

Slika 32. Prikaz koda - loader.py

```
import towhee
from data import source
from db import create_milvus_collection

collection = create_milvus_collection('reverse_image_search', 2048)

dc = (
   towhee.read_csv(source)
    .runas_op['id', 'id'](func=lambda x: int(x))
    .image_decode['path', 'img']()
    .image_embedding.timm['img', 'vec'](model_name='resnet50')
    .to_milvus['id', 'vec'](collection=collection, batch=100)
)

print('Total number of inserted data is {}.'.format(collection.num_entities))
```

Ako je sve prošlo u redu, ubačeno bi trebalo biti 1000 entiteta.

6.6 Pretraga sličnih slika

Implementaciju same pretrage slika pisat ćemo u Jupyter notebooku kako bi imali pregledniji, grafički prikaz slika. Napravit ćemo $main.pynb$ notebook u kojem ćemo prikazati rad *towhee* funkcija te naposlijetku implementirati sam kod za pretraživanje slika po sličnosti. Prvo možemo pročitati podatke iz csv datoteke te ih prikazati kako bi se uvjerili da su u redu.

```
import towhee
from data import source
towhee.read_csv(source).show()
```

Slika 33. Tablični prikaz prvih 5 podataka

id	path	label
0	D:/data./train/brain_coral/n0191...	brain_coral
1	D:/data./train/brain_coral/n0191...	brain_coral
2	D:/data./train/brain_coral/n0191...	brain_coral
3	D:/data./train/brain_coral/n0191...	brain_coral
4	D:/data./train/brain_coral/n0191...	brain_coral

Izvor: autor

Dalje, prikazat ćemo prva 3 zapisa u podacima te ćemo $image\_decode$ funkcijom kodirati sliku koja se nalazi na $path$ lokaciji i prikazati ju u novom img stupcu (Slika 34).

```
towhee.read_csv(source) \
   .head(3) \
   .image_decode.cv2['path', 'img']() \
   .show()
```

Slika 34. Dodavanje slika u tablični prikaz

id	path	label	img
0	D:/data./train/brain_coral/n0191...	brain_coral	
1	D:/data./train/brain_coral/n0191...	brain_coral	
2	D:/data./train/brain_coral/n0191...	brain_coral	

Sada ćemo kodirati i izvući vektorski zapis $image\_embedding.timm$ funkcijom koristeći $resnet50$ model i prikazati ga u novom vec stupcu (Slike 35, 36).

Slika 35. Primjer koda - kodiranje slika pomoću resnet50

```
towhee.read_csv(source) \
    .head(3) \
    .image_decode['path', 'img']() \
    .image_embedding.timm['img', 'vec'](model_name='resnet50') \
    .show()
```

Slika 36. Dodavanje vektorskog zapisa slika u tablični prikaz

id	path	label	img	vec
0	D:/data./train/brain_coral/n0191...	brain_coral		[0.0, 0.0, 0.0, ...] shape=(2048,)
1	D:/data./train/brain_coral/n0191...	brain_coral		[0.0, 0.007748181, 0.0, ...] shape=(2048,)
2	D:/data./train/brain_coral/n0191...	brain_coral		[0.0, 0.024515001, 0.0, ...] shape=(2048,)

Izvor: autor

Vektorski zapis je puno veći budući da je slika kodirana u vektor s 2048 dimenzija. Naravno, ovdje se prikazuje samo početak zbog ograničenja prozora.

Napokon, samo pretraživanje slike odradit ćemo $milvus\_search$ funkcijom te ćemo prikazati pronađene slike uz pomoć $read\_images$ funkcije uvezene iz $data.py$ datoteke. Koristit ćemo $glob$ funkciju koja kao parametar prima $path$ testnih podataka koje želimo pretraživati. Postavili smo $path$ tako da se izvrši pretraga nad svim JPEG slikama koje se nalaze u klasi koja počinje sa slovom k (Slika 37).

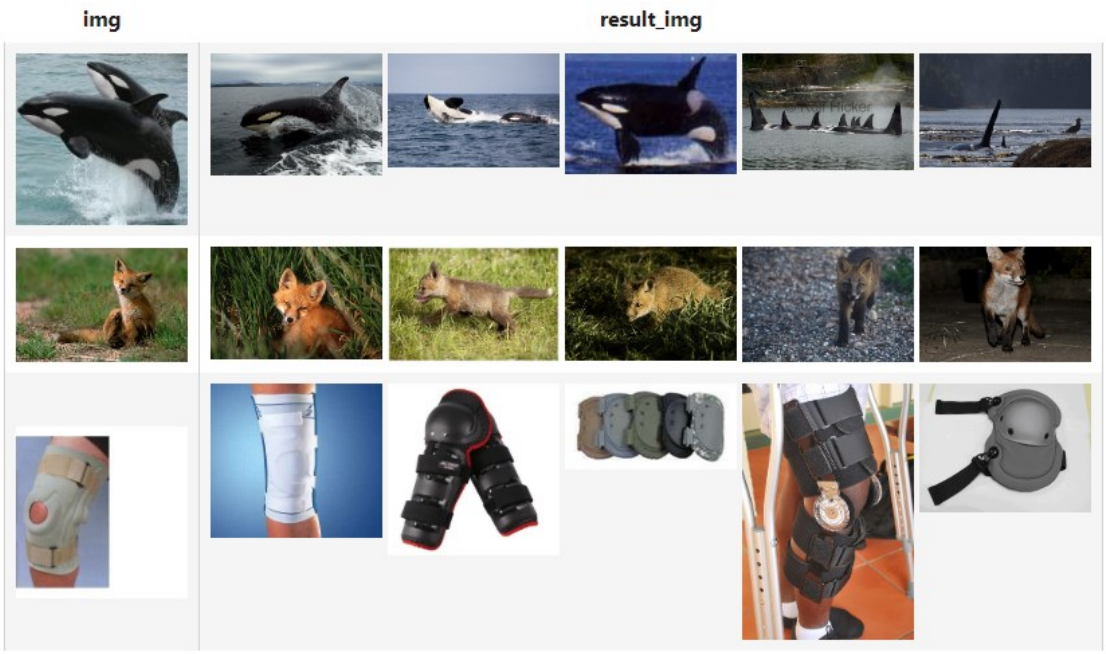
Slika 37. primjer koda - pretraživanje po sličnosti

```
from data import read_images
from loader import collection
(
   towhee.glob['path']('D:/data/test/k*/*.JPEG')
     .image_decode['path', 'img']()
     .image_embedding.timm['img', 'vec'](model_name='resnet50')
     .milvus_search['vec', 'result'](collection=collection, limit=5)
     .runas_op['result', 'result_img'](func=read_images)
     .select['img', 'result_img']()
     .show()
)
```

Nakon pokretanja skripte možemo vidjeti 2 stupca (Slika 38):

1. *img* - predstavlja testni podatak odnosno sliku koju pretražujemo te
2. *result_img* stupac koji prikazuje pronađene slične slike.

Slika 38. Rezultati pretraživanja slika po sličnosti



Izvor: autor

7. ZAKLJUČAK

Nestrukturirani podaci mogu predstavljati problem za mnoga poduzeća budući da ih je teže pohranjivati i obrađivati, odnosno izvući korisno znanje iz velikog skupa podataka. Poslovna inteligencija i poslovna analitika su tehnologije i prakse koje firme upotrebljavaju kako bi dobile detaljnije uvide i saznanja o svojim klijentima i unutarnjim procesima. Takva praksa omogućava kvalitetno i pravovremeno donošenje poslovnih odluka.

U usporedbi sa strukturiranim podacima, nestrukturirani podaci nisu podobni za transakcijsku obradu koju pružaju relacijske baze podataka. Nestrukturirane podatke pohranjujemo u tkz. ne-relacijske ili *NoSQL* baze podataka koje nisu strogo uvjetovane unaprijed definiranom shemom. Vektorske baze podataka jedan su od oblika *NoSQL* baza podataka. One pohranjuju vektore, odnosno vektorske reprezentacije nestrukturiranih podataka, kao entitete.

U četverom poglavlju objašnjen je osnovni koncept neuronskih mreža i dubokog učenja na primjeru jednostavnog DNN modela. Objašnjena je i konvolucijska neuronska mreža koja se većinom koristi za implementaciju softvera u području *computer vision-a*. Štoviše, objašnjen je sam koncept vektorskog zapisa nestrukturiranih podataka kroz *Word2Vec* model.

Vektorske ugradnje mogu reprezentirati gotovo sve nestrukturirane podatke kroz nekoliko stotina ili tisuća dimenzija. Zbog velikog broja dimenzija i podataka, koriste se modeli strojnog učenja i dubokih neuronskih mreža (poput *resnet50*) za obavljanje samog kodiranja. Matematička definicija vektora omogućava izračunavanje međusobne udaljenosti između dvaju vektora. To omogućava pretraživanje vektorskog prostora koristeći KNN ili ANN algoritme te samim tim mnoge korisne implementacije poput: pretraživanja slika, videozapisa ili zvučnih zapisa po sličnosti, sustava za preporuke, rangiranja web stranica, detekcije anomalija te semantičko pretraživanje teksta. Pritom treba izabrati odgovarajuće metrike po kojoj će se računati sličnost te indeks koji će učinkovito organizirati podatke prije samog pretraživanja.

Šesto poglavlje ovog rada dokumentira proces izgradnje aplikacije za prepoznavanje slika po sličnosti koristeći *Milvus.io* vektorsku bazu podataka i *towhee.io framework* za strojno učenje i kodiranje podataka u vektore. Prikazana je instalacija i podizanje baze koristeći *Docker* Compose te priprema i instalacija svih paketa potrebnih za izradu aplikacije u Pythonu. Sama osnovna funkcionalnost implementirana je kroz 20-tak linija koda, međutim moguća je i poželjna dodatna optimizacija radi veće preciznosti i bolje učinkovitosti.

8. POPIS LITERATURE

1. Bergstein, D. (8. ožujak 2022.). *InfoWorld*. Preuzeto 4. travanj 2022. iz Solving complex problems with vector databases: https://www.infoworld.com/article/3651360/solving-complex-problems-with-vector-databases.html

2. Briggs, J. (n.d.). *Pinecone*. Preuzeto 25. srpanj 2022. iz Semantic Search: Measuring Meaning From Jaccard to Bert: https://www.pinecone.io/learn/semantic-search/

3. Brownlee, J. (9. siječanj 2019). *Machine Learning Mastery*. Preuzeto 28. kolovoz 2022 iz A Gentle Introduction to the Rectified Linear Unit (ReLU): https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/

4. *Deloitte Insights*. (25. srpanj 2019.). Preuzeto 25. srpanj 2022. iz Analytics and AI-driven enterprises thrive in the Age of With: https://www2.deloitte.com/us/en/insights/topics/analytics/insight-driven-organization.html

5. *Developers Google*. (n.d.). Preuzeto 28. kolovoz 2022 iz Deep Neural Network Models: https://developers.google.com/machine-learning/recommendation/dnn/softmax#:~:text=Deep%20neural%20network%20(DNN)%20models,improve%20the%20relevance%20of%20recommendations.

6. Eberhart, R. C. (lipanj 1990). Standardization of neural network terminology. *IEEE, 1*(2), str. 244-245. doi:10.1109/72.80238

7. Elliott, A. (19. srpanj 2021). *Mixpanel*. Preuzeto 28. srpanj 2022. iz Business Intelligence vs Business Analytics: https://mixpanel.com/blog/business-intelligence-vs-business-analytics/#:~:text=The%20primary%20distinction%20between%20business,to%20happen%20in%20the%20future.

8. *Essearth*. (7. rujan 2020). Preuzeto 24. srpanj 2022. iz Supercomputing for Weather Forecasting.: https://www.essearth.com/supercomputing-how-it-is-used-in-weather-forecasting/

9. Foote, K. D. (4. veljača 2022). *Dataversity*. Preuzeto 26. kolovoz 2022 iz A Brief History of Deep Learning: https://www.dataversity.net/brief-history-deep-learning/

10. *Google Cloud*. (n.d.). Preuzeto 25. srpanj 2022. iz What is a relational database?: https://cloud.google.com/learn/what-is-a-relational-database#:~:text=A%20relational%20database%20is%20a,structures%20relate%20to%20each%20other.

11. *Hackernoon*. (1. studeni 2017). Preuzeto 25. kolovoz 2022 iz Everything you need to know about Neural Networks: https://hackernoon.com/everything-you-need-to-know-about-neural-networks-8988c3ee4491

12. Harbert, T. (1. veljača 2021.). *MIT Sloan*. Preuzeto 24. srpanj 2022. iz Tapping the power of unstructured data: https://mitsloan.mit.edu/ideas-made-to-matter/tapping-power-unstructured-data

13. Heller, M. (24. svibanj 2019). *InfoWorld*. Preuzeto 26. kolovoz 2022 iz What is deep learning? Algorithms that mimic the human brain: https://www.infoworld.com/article/3397142/what-is-deep-learning-algorithms-that-mimic-the-human-brain.html

14. Hons, C. T. (27. svibanj 2019). *Towards data science*. Preuzeto 29. kolovoz 2022 iz An introduction to Convolutional Neural Networks: https://towardsdatascience.com/an-introduction-to-convolutional-neural-networks-eb0b60b58fd7

15. *IBM*. (n.d.). Preuzeto 27. kolovoz 2022 iz Recurrent Neural Networks: https://www.ibm.com/cloud/learn/recurrent-neural-networks

16. *IBM*. (n.d.). Preuzeto 29. kolovoz 2022 iz K-Nearest Neighbors Algorithm: https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20algorithm%2C%20also%20known%20as%20KNN%20or,of%20an%20individual%20data%20point.

17. *Ignite Apache*. (n.d.). Preuzeto 29. kolovoz 2022 iz ANN (Approximate Nearest Neighbor): https://ignite.apache.org/docs/latest/machine-learning/binary-classification/ann#:~:text=The%20difference%20between%20KNN%20and,small%20subset%20of%20candidates%20points.

18. *Integrate.io*. (n.d.). Preuzeto 24. srpanj 2022. iz What is NoSQL?: https://www.integrate.io/glossary/what-is-nosql/#:~:text=NoSQL%20is%20a%20database%20management,NoSQL%20is%20not%20standardized.

19. *Keboola*. (12. kolovoz 2021). Preuzeto 21. srpanj 2022. iz What are the ACID properties of transactions and why do they matter in data engineering?: https://www.keboola.com/blog/acid-transactions#:~:text=The%20acronym%20ACID%20stands%20for,power%20failures%2C%20or%20other%20issues.

20. Markowitz, D. (23. ožujak 2022). *Google Cloud Blog*. Dohvaćeno iz Meet AI's multitool: Vector embeddings: https://cloud.google.com/blog/topics/developers-practitioners/meet-ais-multitool-vector-embeddings

21. *mathworks*. (n.d.). Preuzeto 29. kolovoz 2022 iz resnet50: https://www.mathworks.com/help/deeplearning/ref/resnet50.html#:~:text=ResNet%2D50%20is%20a%20convolutional,%2C%20pencil%2C%20and%20many%20animals.

22. *mathworks*. (n.d.). Preuzeto 30. kolovoz 2022 iz How CNNs Work: https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html

23. *Microsoft Docs*. (7. siječanj 2020). Preuzeto 26. srpanj 2022. iz FileTables (SQL Server): https://docs.microsoft.com/en-us/sql/relational-databases/blob/filetables-sql-server?view=sql-server-ver16

24. *Milvus*. (n.d.). Preuzeto 21. srpanj 2022. iz What is Milvus vector database?: https://milvus.io/docs/v2.1.x/overview.md

25. *Milvus*. (n.d.). Preuzeto 29. kolovoz 2022 iz ANNS vector indexes: https://milvus.io/docs/index.md

26. *MongoDB*. (n.d.). Preuzeto 26. srpanj 2022. iz What is Unstructured Data?: https://www.mongodb.com/unstructured-data#:~:text=Unstructured%20data%20is%20information%20that,common%20types%20of%20unstructured%20content.

27. *Oracle*. (n.d.). Preuzeto 25. srpanj 2022. iz What Is Big Data?: https://www.oracle.com/big-data/what-is-big-data/

28. Parrish, A. (2017). *Github*. Preuzeto 27. kolovoz 2022 iz Understanding word vectors: https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469

29. Raj, A. (8. travanj 2021). *Towards data science*. Preuzeto 29. kolovoz 2022 iz Introduction to Classification Using K Nearest Neighbours: https://towardsdatascience.com/getting-acquainted-with-k-nearest-neighbors-ba0a9ecf354f

30. Tibshirani, J. (10. veljača 2022). *Elastic Blog*. Preuzeto 27. srpanj 2022. iz Introducing approximate nearest neighbor search in Elasticsearch 8.0: https://www.elastic.co/blog/introducing-approximate-nearest-neighbor-search-in-elasticsearch-8-0

31. Tripathi, R. (n.d.). *Pinecone.io*. Preuzeto 28. srpanj 2022. iz What are Vector Embeddings?: https://www.pinecone.io/learn/vector-embeddings/

32. Turriff, B. (n.d.). *Pinecone*. Preuzeto 25. srpanj 2022. iz What is a Vector Database?: https://www.pinecone.io/learn/vector-database/

33. *Tutorialspoint*. (n.d.). Preuzeto 28. kolovoz 2022 iz TensorFlow - CNN And RNN Difference: https://www.tutorialspoint.com/tensorflow/tensorflow_cnn_and_rnn_difference.htm

9. POPIS SLIKA

10. POPIS TABLICA