RRI PoVejMo

Rezultat **D3.3: Demonstracijska aplikacija za OCR**
Vodilni partner: **Semantika, d.o.o.**
Sodelujoči partnerji:
Fakulteta za računalništvo in informatiko UL
Inštitut za novejšo zgodovino

Avgust 2025

Uvod

Programska oprema GalisOnline in njen predhodnik (Galis) že od samega začetka podpirata napredno in kompleksno iskanje, vendar je moral uporabnik poznati prave izraze ali uporabljati napredne filtrirne obrazce, da je našel relevantne informacije. V okviru projekta PoVejMo smo zato razvili storitev "AI podprto Semantično iskanje", ki omogoča uporabo naravnega jezika pri iskalnih poizvedbah, hkrati pa omogoča tudi naprednejše indeksiranje vsebine. To v praksi pomeni, da lahko uporabnik zastavi vprašanje ali zahtevo z uporabo naravnega jezika, sistem pa razume pomen in najde ustrezne zadetke tudi, če ne vsebujejo dobesednih ključnih besed. S tem postane iskanje prijaznejše širši javnosti in obenem učinkovitejše za strokovnjake, ki iščejo po obsežnih zbirkah muzejskih in arhivskih podatkov.

V tem poročilu predstavljamo razvoj in trenutno stanje rešitve Semantičnega iskanja (dosežena raven tehnološke zrelosti TRL 6) v sklopu projekta PoVejMo. Opisali bomo, kako storitev povezuje cilje projekta (zlasti delovni sklop C3.3), kakšne tehnologije in jezikovne modele uporablja (vključno z velikimi modeli GaMS-27B in Micka) ter kako deluje celoten postopek iskanja - od vektorizacije in razširitve poizvedbe do hibridnega iskanja, deduplikacije, ponovnega rangiranja z umetno inteligenco in prikaza rezultatov. Predstavili bomo integracijo rešitve v obstoječi muzejski dokumentacijski sistem GalisOnline, primer uporabe na muzejskih podatkih ter možnost prenosa v druge domene.

Projekt PoVejMo in cilj C3.3 - Semantični iskalnik

Projekt PoVejMo ("Prilagodljiva obdelava naravnega jezika z velikimi jezikovnimi modeli") je usmerjen v raziskave in razvoj novih rešitev na področju umetne inteligence in jezikovnih tehnologij, s poudarkom na slovenščini. Projekt vključuje več delovnih sklopov, ki naslavljajo različne izzive, od ustvarjanja dialogov in ukazov v slovenskem jeziku do strojnega prevajanja in govornega vmesnika.

Semantično iskanje je osrednja tema sklopa C3.3, ki je opredeljen kot "demonstracija semantičnega iskalnika s podporo sporočanju v naravnem jeziku"; cilj C3.3 je torej pokazati, kako lahko napredni jezikovni modeli omogočijo iskanje po zbirkah dokumentov zgolj z

uporabo običajnega jezika, brez uporabe kompleksnih poizvedb ali poznavanja skriptnih jezikov. Implementacija je bila izvedena na primeru iskalnika po internih dokumentih - v našem primeru po podatkih iz muzejskih zbirk podjetja Semantika - z izvedbo PoC na dveh področjih: digitalna humanistika in visoko regulirana industrija (energetika), v tem dokumentu pa se zaradi občutljivosti vsebine iz drugega sklopa osredotočamo na digitalno humanistiko.

Razvoj rešitve in uporabljene tehnologije

Razvoj semantičnega iskanja se je opiral na najsodobnejše dosežke na področju velikih jezikovnih modelov (LLM) in vektorskega iskanja (t.i. RAG), kjer smo združili rezultate projekta PoVejMo z obstoječimi pristopi. V projektu PoVejMo smo tako imeli dostop do novo razvitih slovenskih jezikovnih modelov, ki so plod sodelovanja raziskovalcev na UL FRI, ki smo jih dodatno trenirali s pomočjo razvite zbirke 10000 vprašanj in odgovorov s področja humanistike (ki jih je pripravil partner INZ).

Osnova je družina modelov GaMS (Generativni model za slovenščino), zlasti model GaMS-27B, ki vsebuje 27 milijard parametrov - izdelan na osnovi arhitekture Gemma 2, in dodatno usposobljen na velikih količinah slovenskih, angleških ter nekaj hrvaških, srbskih in bosanskih besedil. GaMS-27B predstavlja enega največjih in najzmogljivejših odprtokodnih modelov za slovenski jezik, kar pomeni, da zmore razumevati kompleksna besedila in odgovarjati na zahtevna vprašanja v slovenščini.

Poleg splošnega modela GaMS je Semantika razvila tudi lastne specializirane modele znotraj projekta, s kodnim imenom Micka; ti modeli so usmerjeni v specifične naloge, kot so strukturiranje besedil, označevanje entitet in pridobivanje informacij iz besedil. Gre za modele z encoder/decoder arhitekturo, ki so specializirani za RAG opravila (iskanje z vključevanjem zunanjega znanja, t.i. *Retrieval-Augmented Generation*) in strukturiranje/tagging besedil. Med njimi so različice, kot na primer *Micka-gen3* in *Micka-mass-v3*, ki dokazujejo možnosti prilagoditve jezikovnih modelov za potrebe specifičnih domen.

S tehnološkega vidika je semantično iskanje zgrajeno kot nadgradnja obstoječega sistema GalisOnline. GalisOnline je celovit dokumentacijski sistem za muzeje, skladen z mednarodnimi standardi (SPECTRUM, MARC, ISAD(G), idr.) in ga v več jezikovnih različicah uporablja prek 70 organizacij v 3 državah; rešitev že podpira klasično iskanje (zgrajeno na tehnologiji Lucene), v okviru projekta pa smo dodali komponento za vektorsko iskanje in modul z umetno inteligenco. Pomembno je poudariti, da semantično iskanje ne nadomešča obstoječih funkcionalnosti, temveč jih dopolnjuje - uporabnikom ponuja nov, bolj intuitiven način dostopa do podatkov, ob tem pa še vedno spoštuje obstoječe varnostne mehanizme, pravice dostopa in strukturo podatkovnega sistema.

Kako deluje semantični iskalnik

Delovanje semantičnega iskanja lahko opišemo kot več zaporednih korakov, ki združujejo klasične metode iskanja po podatkovnih zbirkah z umetno inteligenco. V nadaljevanju je predstavljen celoten potek iskanja - od priprave podatkov, preko obdelave uporabnikove poizvedbe, do končnega prikaza rezultatov. Ključni koncept je, da tako podatke kot poizvedbe pretvorimo v vektorski prostor (prostorsko predstavitev pomena), kjer lahko iščemo po pomenski podobnosti med opisi in poizvedbo, ne zgolj po ključnih besedah. Vzporedno pa se upošteva tudi polno besedilno ujemanje za boljše rezultate.

Pregled postopka:

1. Vektorizacija vsebine (priprava podatkovnih vektorjev): za vsako enoto, po kateri želimo iskati (npr. muzejski predmet, dokument), najprej samodejno ustvarimo opis na podlagi vseh razpoložljivih polj v podatkovni bazi. Cilj je dobiti strnjen tekst, ki najbolje opisuje dano entiteto (predmet) - ta opis lahko vključuje naziv, avtorja, leto nastanka, material, zbirko ipd. Nato s pomočjo jezikovnega modela ta opis pretvorimo v vektorsko predstavitev (*embedding*). V našem primeru smo uporabili model Micka, ki vsakemu besedilu priredi visokodimenzionalni vektor števil. Ti vektorji predstavljajo pomensko vsebino opisa - sorodni predmeti bodo imeli podobne vektorje. Za vsako entiteto shranimo vektor v bazo (Postgres/pgvector) skupaj z identifikatorjem entitete in tipom entitete, da vemo, h kateri tabeli pripada (npr. predmet, avtor, dogodek). Ta korak izvedemo vnaprej (offline) za vse obstoječe podatke in periodično ponovimo za nove vnose.

2. Vnos poizvedbe v naravnem jeziku: uporabnik storitve semantičnega iskanja v iskalno vrstico zapiše poljubno poizvedbo v slovenščini, podobno kot bi zastavil vprašanje osebi. Primer takšne poizvedbe je: *“Najdi mi predmete iz železarske zbirke ali predmete, ki so iz železa.”*. Uporabnik ni omejen s tem, kako oblikuje stavek - sistem poskuša razumeti namen in ključne pojme v poizvedbi (npr. v navedenem primeru “železarska zbirka” in “predmeti iz železa”). To pomeni veliko bolj naravno uporabniško izkušnjo v primerjavi s klasičnim iskanjem, kjer bi moral uporabnik posebej nastaviti filter na *Zbirka = Železarska* ALI *Material = železo*.

3. Samodejna razširitev poizvedbe: da bi sistem kar najbolje pokril namen uporabnika, poizvedbo najprej razširi s sopomenkami in povezanimi izrazi. Uporabili smo pristop, kjer je v ozadju jezikovni model seznanjen z bazami terminov (*šifranti*) s področja muzejskih zbirk, materialov in tehnik, ki jih poznamo v našem podatkovnem sistemu. Na osnovi teh domenskih znanj model generira alternativne izraze, ki imajo podoben pomen kot izvorni pojmi v poizvedbi. V našem primeru bi sistem iz izvirne poizvedbe razpoznal pojem *“železarska zbirka”* in ga razširil še na povezane pojme, npr. *“kovaška zbirka”* (če obstaja) ali *“predmeti iz železa”*, *“železni predmeti”*. Prav tako bi za besedno zvezo *“predmeti, ki so iz železa”* dodal sinonim *“kovinski predmeti”* ipd. Končni rezultat tega koraka je torej razširjen seznam poizvedb, ki vsebuje izvorno uporabnikovo besedilo *in* vse relevantne razširitve. S tem povečamo možnost, da najdemo ustrezne zadetke, tudi če so v podatkih uporabljeni nekoliko drugačni izrazi.

4. Pretvorba poizvedb v vektorje: vse poizvedbe (izvorna in razširjene) nato podobno kot podatke pretvorimo v vektorske predstavitve. Uporabimo isti postopek in isti embedding model kot pri vektorizaciji opisov, saj želimo, da so vektorji primerljivi v istem vektorskem prostoru. Pred primerjavo vektorje še normaliziramo (priredimo na enotno dolžino), da bo računanje podobnosti konsistentno.

5. Hibridno iskanje (vektorsko + polno besedilno): za vsako poizvedbo iz razširjenega seznama izvedemo iskanje po bazi. Tukaj uporabimo hibridni pristop, ki združuje pomensko ujemanje in klasično ujemanje ključnih besed. Na PostgreSQL strežniku imamo tabelo (npr. *vectordata*), kjer so shranjeni vsi vektorji opisov, po njej pa poizvedujemo z uporabo razširitve pgvector, ki omogoča iskanje najbolj podobnih vektorjev glede na poizvedbenega. Kot mero podobnosti uporabljamo klasično kosinusno razdaljo med vektorji. Hkrati v isti poizvedbi uporabimo še PostgreSQL full-text search (tsvector indeks) za ujemanje besedila - s tem poskrbimo, da če se kakšen izraz natančno pojavi (npr. točno “železo” ali točno ime zbirke) v opisu, dobi tak zadetek dodatno težo. Rezultat tega hibridnega iskanja je za vsako poizvedbo seznam kandidatov: nabor (entiteta, njeno besedilo oz. “chunk”, vektorska metrika podobnosti, ujemajoča poizvedba). Vsak tak kandidat ima torej informacijo, katera entiteta (predmet) je bila najdena, kateri del besedila je bil najbolj relevanten, kakšen je vektorski rang (nižje je bolje, pomeni večjo podobnost) in iz katerega razširjenega iskalnega niza izvira ujemanje.

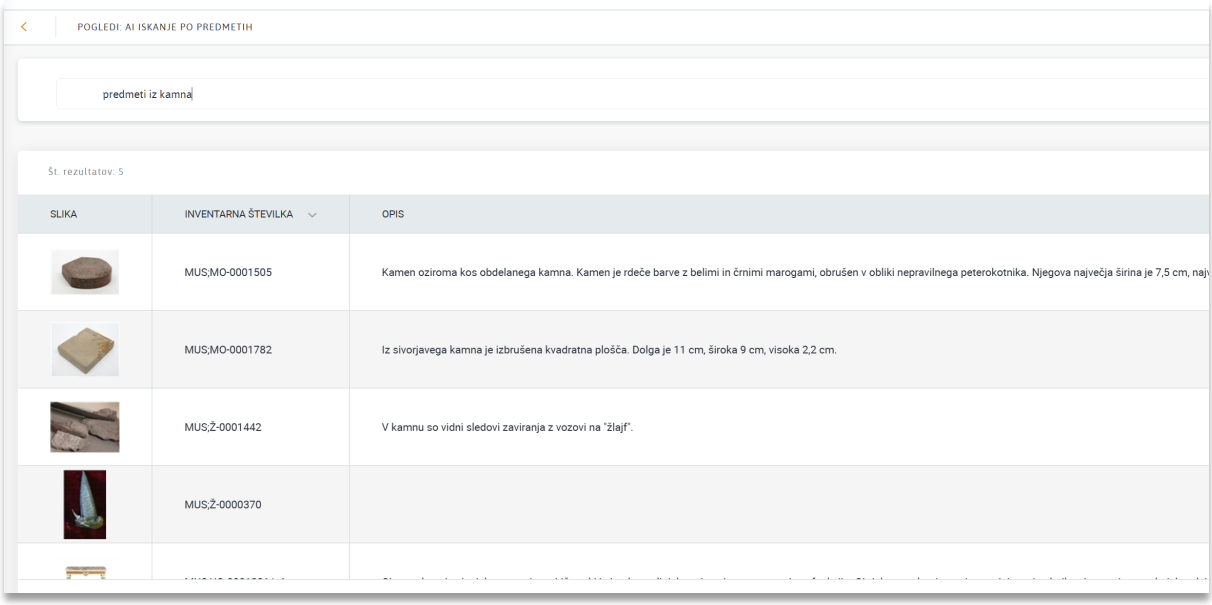
6. Združevanje in odprava dvojnikov: ko zberemo rezultate za vse razširjene poizvedbe, jih združimo v en skupen seznam. Tu lahko pride do podvajanja, saj se ista entiteta lahko najde prek več različnih poizvedb (npr. neki predmet ustreza različnim iskalnim pogojem: npr. “železarska zbirka” in “železni predmeti”). Da uporabniku ne bi prikazovali podvojenih rezultatov, opravimo deduplikacijo: če več “chunkov” ali zadetkov pripada istemu predmetu, obdržimo samo tistega z najboljšo oceno (najboljše ujemanje). S tem zagotovimo, da se vsak relevantni predmet pojavi le enkrat in to z najbolj relevantnim delčkom opisa. Nato rezultate sortiramo po kombiniranem hibridnem ocenjevalnem kriteriju (upošteva vektorsko podobnost in besedilno ujemanje) ter omejimo seznam na najboljših N (npr. 10 ali 20) najrelevantnejših entitet. S tem imamo pripravljene najboljše kandidate za prikaz, a naš proces semantičnega iskanja tu še ni končan - sledi še dodatna inteligentna razvrstitev.

7. Pridobitev celotnih zapisov: identifikatorje izbranih N entitet uporabimo za poizvedbo po celotni zbirki podatkov (v našem primeru po Lucene indeksu oz. iskalniku znotraj GalisOnline); na ta način pridobimo celoten dokument oziroma zapis za vsak zadetek - to vključuje vse podrobnosti predmeta (vsa polja, slike itd.), ne le kratek opis, ki smo ga uporabili za vektorsko iskanje. Ta korak je pomemben, ker želimo uporabniku prikazati celoten kontekst zadetka (npr. vse podatke o najdenem predmetu). Ob tem vsakemu dokumentu pripnemo tudi meta-informacije iz prejšnjih korakov: kakšna je bila podobnost ter kateremu iskalnemu nizu je ustrezal.

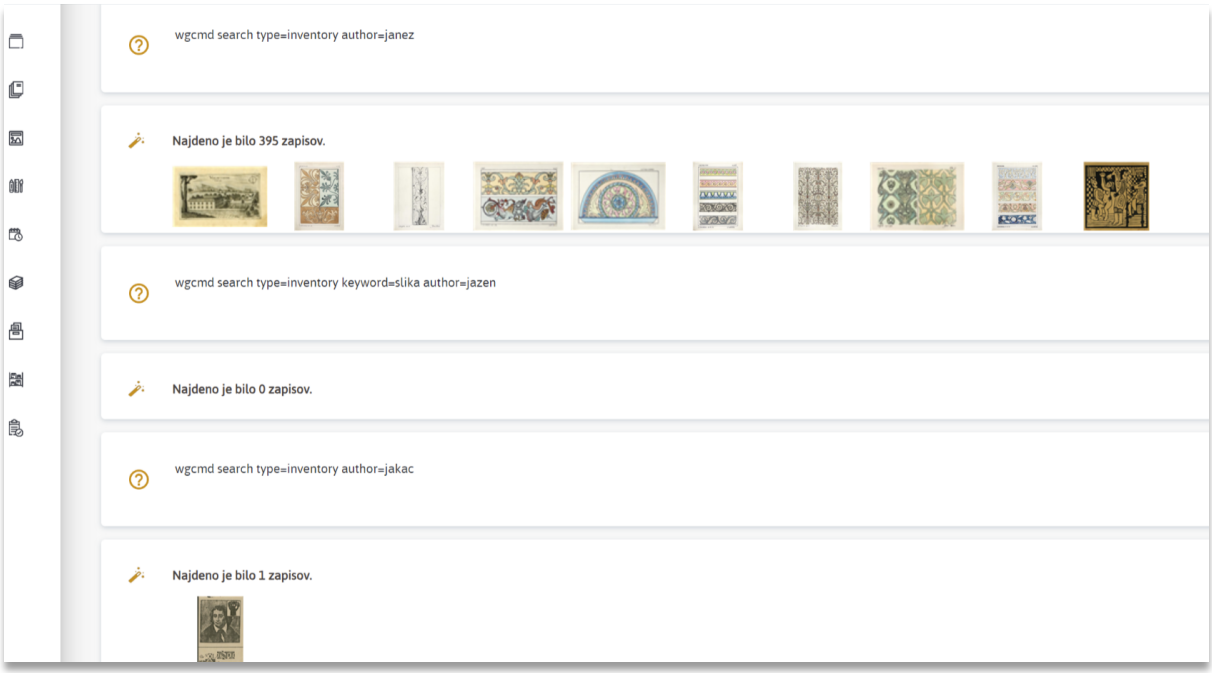
8. Pametno razvrščanje rezultatov (re-ranking z generativnim modelom): čeprav že imamo seznam relevantnih zadetkov, želimo izkoristiti moč velikega jezikovnega modela, da te zadetke še kvalitativno oceni in uredi po dejanski pomenski relevantnosti. Zato sestavimo poseben poziv (prompt) za jezikovni model, v katerega za vsakega kandidata vključimo njegov opis (npr. ime predmeta in opis), prvotni rang in informacijo, kateri iskalni niz ga je našel. Modelu torej predstavimo nabor potencialnih zadetkov z nekaj konteksta in ga povprašamo, *kako relevantni* so posamezni zadetki glede na poizvedbo. Jezikovni model za vsak zadetek vrne naslednje podatke: identifikator entitete, izvirni rang (zaradi sledljivosti), oceno relevantnosti med 0.0 in 1.0 (kjer 1.0 pomeni izjemno relevantno, 0.0 pa nerelevantno) ter razlago (Reason) - kratek stavek, zakaj naj bi bil zadetek relevanten ali ne. Ko te ocene zberemo, dobimo končni seznam strukturiran kot (EntityId, Score, Relevance, Reason). Končni rang rezultatov lahko nato določimo glede na Relevance (npr. od najvišje do najnižje), saj ta številka najbolje odraža skupno presojo AI o tem, kaj je zares najbolj ustrezno glede na uporabnikov poizvedbo.

9. Prikaz rezultatov uporabniku: izvajalni del iskanja se zaključi s prikazom rezultatov na uporabniškem vmesniku. Uporabnik dobi seznam najdenih predmetov oziroma dokumentov, urejen po relevantnosti. Vsak zadetek je predstavljen s ključnimi podatki (npr. naziv predmeta, slika, osnovni opis itd.), lahko pa so vključene tudi dodatne informacije, ki jih je prispevala AI. Tako lahko prikažemo npr. “Razlog” - kratko obrazložitev, zakaj je ta zadetek v kontekstu poizvedbe smiseln.

Integracija v GalisOnline in uporabniški scenariji



Slika 1: Primer rezultata semantičnega iskanja v vmesniku GalisOnline.



Slika 2: Primer poizvedbe, ki jo umetna inteligenca tvori v ozadju.

Ena od prednosti razvite rešitve je, da je bila neposredno preizkušena v obstoječi aplikaciji GalisOnline, ki je bila tehnično izvedena tako, da je GalisOnline dobil nov iskalni modul. V

uporabniškem vmesniku to v praksi pomeni novo iskalno polje, kjer je označeno, da lahko uporabnik postavi vprašanje v naravnem jeziku.

Ob vnosu poizvedbe (in sprožitvi iskanja) GalisOnline v ozadju namesto običajnega iskanja po Lucene indeksu pokliče postopke, opisane v prejšnjem poglavju - generiranje vektorjev, poizvedba po pgvector in generativni re-ranking. Rezultati se nato pretvorijo nazaj v format, ki ga GalisOnline razume, in se prikažejo na standardni strani (maski) z zadetki.

Da bi ponazorili uporabniško vrednost semantičnega iskanja, opišimo primer scenarija v muzejski dokumentaciji. Recimo, da muzejski kustos želi najti predmete, povezane z določenim materialom in zbirko, ne da bi vedel točno, v katerih poljih baze se to nahaja. V klasičnem iskanju bi moral posebej nastaviti kriterije (Material = "železo" OR Zbirka = "Železarska zbirka"). Z novim iskalnikom pa preprosto napiše stavek: *"Najdi mi predmete iz Železarske zbirke ali predmete, ki so iz železa."* Sistem razume, da sta ključni dve ideji: *Železarska zbirka* (ime zbirke) in predmeti iz *železa* (material). Uporabi vnaprej naučene povezave in sopomenke (ve, da "železo" lahko pomeni tudi "kovinski") ter najde predmete, ki spadajo v železarsko zbirko ali imajo material železo. Kustos dobi združen seznam relevantnih predmetov, med katerimi so npr. kovaška orodja iz železa, železni izdelki ter predmeti iz železarske muzejske zbirke, četudi morda opis teh predmetov ne vsebuje dobesedno "železarska" (morda je zapisano "kovaška zbirka").

Pomemben del integracije so tudi pravice - GalisOnline upravlja z različnimi ravnmi dostopa (nekateri podatki so javni, drugi interni, nekateri morda občutljivi); semantično iskanje pa v celoti spoštuje te omejitve, saj zgolj nadgrajuje obstoječo infrastrukturo. Za visoko regulirane industrije (kot npr. farmacija, energetika ipd.) je to ena ključnih lastnosti: imeti inteligentno iskanje znotraj zasebnega okolja z uporabo lokalnih modelov (GaMS, Micka), tako da lahko rešitev namestimo znotraj podjetja, celo brez povezave v internet.