Univerza v Ljubljani
Fakulteta za računalništvo in informatiko
Fakulteta za matematiko in fiziko

Rok Mušič

Napredne s poizvedovanjem obogatene tehnike generiranja (RAG)

diplomsko delo

interdisciplinarni univerzitetni
študijski program prve stopnje
računalništvo in matematika

Mentor: izr. prof. Slavko Žitnik

Ljubljana, 2024

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Kandidat: Rok Mušič
Naslov: Napredne s poizvedovanjem obogatene tehnike generiranja (RAG)
Vrsta naloge: Diplomska naloga na univerzitetnem programu prve stopnje Računalništva in matematike
Mentor: izr. prof. Slavko Žitnik

Opis:
Cilj naloge je implementirati pravnega svetovalca, ki je sposoben uporabniku odgovoriti na poljubno vprašanje s pravnega področja. Za generacijo odgovora naj se uporabi velik jezikovni model. Zaradi pomanjkanja internega znanja v velikem jezikovnem modelu, naj se uporabijo različne s kontekstom obogatene tehnike generiranja; odlično predstavljene v članku pregleda teh tehnik [8]. Generirani odgovori se preverijo na kvalitetni ročno izdelani testni množici in so ocenjeni s strani strokovnjaka, da se preveri do kakšne mere bi se implementirane metode lahko začele uporabljati v kritičnih sistemih.

Title: Advanced RAG techniques

Description:
The goal of this work is to implement a law assistant able to answer complex law related questions. A large language model should be used for the generation of the answer. Due to the absence of specific domain knowledge in the large language model, different retrieval augmented techniques are to be utilized to alleviate incorrectness and hallucinations. There is an excellent article that surveys retrieval augmented generation techniques [8]. Generated answers are graded by an expert and a high-quality, hand-crafted test dataset. The objective of testing is to find out which of the methods are suitable to be used in critical systems, and to what degree of certainty.

Rad bi se zahvalil svojim staršem, še posebej mami in očetu.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
VJM	large language model	velik jezikovni model
RAG	retrieval augmented generation	s kontekstom obogoteno generiranje
RNN	recurrent neural network	rekurentne nevronske mreže
GZ	knowledge graph	graf znanja
GZ-RAG	RAG with knowledge graph	RAG z grafom znanja
PISRS	law-informational system of the Republic of Slovenia	pravno informacijski sistem Republike Slovenije
BLEU	bilingual evaluation understudy	dvojezična ocenjevalna študija
ROUGE	recall-oriented understudy for gisting evaluation	spominsko usmerjena študija za ocenjevanje povzemanja
ZGD	Law on commercial companies	Zakon o gospodarskih družbah

Povzetek

Naslov: Napredne s poizvedovanjem obogatene tehnike generiranja (RAG)

Avtor: Rok Mušič

Slovenska zakonodaja je obsežna in pravni delavci porabijo veliko časa vsak dan za iskanje ustrezne literature. V ta namen smo raziskali uspešnost velikih jezikovnih modelov (VJM) kot pravnih asistentov. VJM-ji so uspešni v številnih nalogah, a zahtevna domenska vprašanja so ena izmed njihovih večjih pomankljivosti; pogosto pride do halucinacij. S kontekstom obogateno generiranje (RAG) je tehnika, ki zaobide pomanjkanje domenskega znanja VJM-jev s tem, da se glede na vprašanje v zakonodaji najde vsebina s katero je moč pravilno odgovoriti na vprašanje. Z najdenim znanjem VJM pravilno odgovori in ne halucinira. Raziskali in implementirali smo več različnih RAG tehnik. Vse metode smo preizkusili na ročno izdelani testni množici, ki vsebuje 4 testne scenarije, s katerimi preverimo kako uspešne so metode v različnih situacijah. Naprednejše različice RAG-a, napredni in modularni RAG, kažejo dobro uspešnost pri direktnih vprašanjih, a nižjo uspešnost za bolj splošna vprašanja kot so npr. dejanski primeri.

Ključne besede: velik jezikovni model, s poizvedovanjem obogateno generiranje, obdelava naravnega jezika.

Abstract

Title: Advanced RAG techniques

Author: Rok Mušič

Abstract: Slovenian legislation is extensive, causing legal professionals to spend a significant amount of time each day searching for relevant literature. To address this, we explored the effectiveness of large language models (LLMs) as legal assistants. LLMs have been successful in various tasks, but handling complex domain-specific questions remains one of their major weaknesses; often producing hallucinations. Retrieval-Augmented Generation (RAG) is a technique that bypasses the lack of domain knowledge in LLMs by retrieving content from legislation based on the question, allowing for accurate responses. With the retrieved knowledge, the LLM can correctly answer the question without hallucinating. We explored and implemented several different RAG techniques. All methods were tested on a manually crafted test set containing four test scenarios to evaluate how successful the methods are in various situations. More advanced versions of RAG, such as advanced and modular RAG, show good performance in direct questions but lower success in more general questions, such as real-world examples.

Keywords: large language model, retrieval augmented generation, natural languange processing.

Poglavje 1

Uvod

Živimo v svetu informacij. Uporaba besedne zveze "eksponentna rast" je zadnja leta na porastu v širši družbi. Čeprav se povprečen državljan niti približno ne zaveda kako divje raste ta funkcija, smo morda okusili vzorec ob spremljanju števila okužb z virusom Covid-19 (vsaj v prvem navalu porasta okužb). Verjetno ne bo nikogar presenetilo, ko rečemo, da število informacij, ki jih proizvede človeštvo, strmi k eksponentni rasti in ne kaže nobenih znakov upada [5].

Kako iz tega morja informacij izluščit relevantne informacije, je primerna prispodoba iskanja šivanke v bali sena. Tehnološki velikan Google je z odgovorom na to vprašanje zaslužil milijarde evrov.

Magnituda vseh teh podatkov je za navadnega smrtnika prevelika; ter če smo iskreni, nepraktična. V praksi se mnogo bolj pogosto pojavi primer, da bi potrebovali natančne informacije iz nekega specifičnega področja na katerem delamo. Če razvijamo programsko opremo za urejanje besedila, nam znanje (podatki) trkov atomov iz pospeševalnika atomov v Cernu ne koristijo kaj dosti.

Konkreten primer, ki ga bomo obravnavali v tej diplomski nalogi, je odgovarjanje na pravna vprašanja, kar vključuje iskanje po slovenski zakonodaji za potrebne informacije.

V nadaljevanju tega poglavja obravnavamo problem tega raziskovalnega

dela, rešitev na ta problem, ter vpliv te rešitve. V naslednjem poglavju predstavimo tehnologije uporabljene v tem delu in sorodna dela. V tretjem poglavju opišemo s katerimi podatki bomo delali. V četrtem poglavju opišemo vse metode, ki smo jih uporabili in vse njihove implementacijske podrobnosti. V petem poglavju poročamo o rezultatih in uspešnosti metod. V zadnjem poglavju zaključimo to delo.

1.1 Problem

Slovenska zakonodaja je obsežna. Čeprav kot država obstajamo šele 33 let (od leta 1991), se je v tem času nabral že zajeten korpus pravnih besedil. Vredno je omeniti, da kljub temu da pred tem nismo bili pravnomočna država, smo vseeno bili del druge države, ki je imela svoje pravne predpise. Tako veliko naše zakonodaje izvira iz stare jugoslovanske zakonodaje, ter nemške iz avstro-ogrskih časov. Nadalje je v slovenski zakonodaji več kot 30 različnih vrst aktov [16], kot so zakon, odločba US, sklep, itd.

Vsak zakon vsebuje množico členov, kateri so lahko združeni v strukturne enote, kot so poglavja in odseki. Člen je osnovna strukturna enota v zakonu, ki celovito zajema neko tematiko. Členi so nadalje lahko deljeni na odstavke, točke in alineje. Poseben tip členov so končne in prehodne določbe, ki posebaj določajo kdaj kakšen zakon ali del zakona stopi v veljavo, ter pod kakšnimi pogoji. Recimo pri spremembi zakona mora biti določeno, kaj velja v prehodnem obdobju. Določilo je načeloma bolj splošen pojem kot člen, vendar se v tem delu izraza uporabljata bržkone sinonimno.

Sedaj pa se postavimo v čevlje osebe, ki bi potrebuje pravno asistenco. Že tu lahko naredimo delitev na dva primera:

- ko je oseba učena na pravnem področju, pozna terminologijo ipd.,
- ter vse preostale državljane, ki o pravu ne vemo kaj dosti.

Začnimo z osebo veščo na pravnem področju (od tu dalje naj bo poimenovana oseba A). Oseba A je ob opravljanju svojega dela pogosto v stiku z

zakonodajo. Ko dela na rešitvi za nek primer, mora pogosto najti primerne informacije v zakonodaji. Pa naj bo to interpretacija člena zakona, primer sodne prakse, ali kaj drugega. Ko oseba A ve točno kaj išče in kje lahko to najde, je super. Problem nastopi, ko temu ni tako.

Mnogokrat je to iskanje ustrezne zakonodaje kot lov na zaklad. Če človek nima zemeljevida do zaklada, ne ve niti kje začeti iskat; kaj šele kako nadaljevati iskanje. S konkretnimi besedami: oseba A ne ve kje sploh začeti iskati literaturo (t.j. pravni akti). Morda niti ne ve, da tak pravni akt obstaja.

Pa recimo, da smo določili izhodiščno točko. Naslednje vprašanje je, kako zdaj nadaljevati, oz. kako iz izhodiščne točke najdemo nadaljno pravno literaturo. Bilo bi sila priročno, če bi za vsako enoto znanja imeli množico puščic, ki bi kazale na podobne in relevantne vsebine. Najpreprostejši primer bi bila povezava med členom zakona in primerom uporabe tega člena v sodni praksi.

Nazadnje je še problem veljavnosti (verodostojnosti) podatkov. Tu imamo predvsem v mislih morebitne spremembe zakonov ali posameznih členov. Prehodne in končne odločbe spremenijo delovanje nekaterih členov. Če nismo pozorni, lahko nehote uporabimo neveljavno zakonodajo, zoper česa je razveljavljena pravomočnost. Posebaj nelagodno je, ko so te posebne določbe, ki spreminjajo druge določbe, razpršena med več pravnih aktov, kar povzroči preglavice ob iskanju vseh teh omemb. Pride do kršitve iz nevednosti.

Če povzamemo: zaradi obsežnosti zakonodaje se porabi veliko dragocenega časa z iskanjem potrebnih informacij, kateri bi lahko bil bolje porabljen na bolj domensko specifičnih in pomembnih opravilih.

Glavni, a ne ekskluzivni, razlogi so:

1. nevednost obstoja določenega pravnega akta,
2. nevednost kje in kako začeti iskanje,
3. nevednost kako nadaljevati iskanje in najti še več relevantnih informacij,
4. ter kako zagotoviti celovitost podatkov.

Trenuten pristop reševanja takšnih primerov je iskanje po pravih aktih s ključnimi besedami. Očitno je uporabnost tega pristopa omejena na primere, ko oseba A ve v katerem pravnem aktu se informacije nahajajo in kaj so ključne besede iskanja. Sicer osebi A ne preostane drugega, kot da povpraša sodelavce, če se kdo spozna na dotičnem področju, ali pa slepo išče po velikem številu pravnih aktov.

Drugi primer je končni uporabnik (oseba B), ki se ne spozna na pravnem področju. To je tipično oseba, ki ko potrebuje pravne rešitve, najame ustrezenga pravnega delavca, kateremu v poljudnem jeziku razloži kakšno težavo ima, nakar je dolžnost pravne osebe, da stranki priskrbi rešitev. Take storitve so bolj kot ne drage in včasih bi oseba B le želela kakšno začetno misel, preden s primerom pristopi k pravnemu strokovnjaku.

Ker se oseba B bržkone ne spozna na prvo stvar v pravu, je edina rešitev, kako odgovoriti na njeno vprašanje, strokovnjak na področju (kot je npr. oseba A), ki lahko v celovitosti razume problem osebe B, kako razrešiti ta problem, ter kako odgovor podati osebi B, da ga bo razumela. Zatorej tu ne moremo problema razdeliti na več manjših podproblemov in vsakega obravnavati posebej, kot smo to naredili pri osebi A, vendar smo prisiljeni problem osebe B smatrati kot enovito celoto.

1.2 Rešitev

Z uporabo dobrega iskalnika, dobro postavljene podatkovne baze v ozadju in občasno pomočjo strokovnjaka (če primer seže na področje, na katerem oseba A ni strokovnjak), bi oseba A lahko razrešila večino svojih težav.

Ključna faktorja sta kako dobro je postavljena podatkovna baza in kako dobro deluje iskalnik. Če je podatkovna baza pomankljivo ali slabo postavljena, bodo takšni tudi podatki. Da lahko razrešimo prej omenjene težave osebe A, tj. kako zagotoviti celovitost in veljavnost podatkov in kako razširiti prostor iskanja po tem ko najdemo neko informacijo, je smotrno med posameznimi entitetami zakonodaje (enota informacije) vzpostaviti povezave.

Najlažje si je to predstavljati v okviru, ko se neka določba nanaša (referencira) neko drugo določbo. Če bi imeli indeksirane vse te povezave—katere definicije uporablja določba, katera druga določila referencira, katera druga določila referencirajo to določbo (ki mu lahko spreminjajo vsebino in/ali veljavnost), kje v strukturi pravnega besedila se nahaja—se pred nami izriše bogata mreža informacij.

Če označimo entite (enote informacij) za vozlišča in zgoraj omenjena razmerja za povezave, dobimo usmerjen graf. Usmerjen graf znanja, če smo bolj natančni.

Če določimo neko vozlišče in grobo združimo povezave v dve množici, vhodne in izhodne, opazimo naslednje:

- vhodne povezave nam v celoti podajo informacijo katera druga določila se nanašajo na to določbo, kar vključuje kakršnekoli morebitne spremembe. Nadalje vključujejo povezave do "nadaljnega branja".

- izhodne povezave pa vsebujejo vse informacije za razumevanje določila. Lahko si predstavljamo, da povsod v vsebini določila, kjer je referenca, "razširimo" referenco v celotno referencirano besedilo.

Izhodne povezave osebi A niso tako ključnega pomena, saj ob branju določila vedno vidi vse reference in lahko kadarkoli pogleda ciljno vozlišče. Vendar še vedno je želja po nemotenem branju zakonodaje (da so vsa referencirana besedila dostopna na licu mesta) dovolj velika, da je izključno iz tega razloga nastalo spletišče *zakonodaja.com*.

Bistveno bolj ključne so vhodne povezave. Kratek primer: Oseba A v svojem primeru uporablja določilo X, vendar ni videla določila Y, ki spreminja vsebino (referencira) določilo X. Izhodne povezave se vedno lahko razberejo iz besedila samega, vendar da lahko zagotovimo celovitost vhodnih povezav, bi morali pregledati celotno zakonodajo.

Nadalje nosijo vhodne povezave bolj pomembne informacije. To so kakršnekoli morebitne spremembe določila, katere bi drugače spregledali, ter povezave do drugih aktov / določil, ki uporabljajo vsebino tega določila. Slednje je glavni

način kako najti nadaljne informacije. Predvsem uporabno je to v konkretnih primerih kot jih vidimo v sodni praksi, kjer lahko tako zasledimo interpretacijo določila.

S tem smo pod streho spravili težavo celovitosti podatkov in težavo kako najti sorodno vsebino oz. nadaljno literaturo. Ostane še težava kako začeti iskanje, tj. kako najti relevantne vsebine, kar je zadolžitev iskalnika.

V primeru ko točno vemo ključne besede iskanja, so klasične metode iskanja, kot je npr. BM-25 ali TF-IDF, dokaj uspešne. Vendar kot sem prej omenil, večji problem nastane v primeru, ko ne poznamo teh ključnih besed za iskanje. Nadalje so omejene le na iskanje s ključnimi besedami, brez podpore za bolj zahtevne poizvedbe [19]. Za nadaljne težave lahko navedemo problem sklanjatve slovenskega jezika (če bi v predpocesiranju lemizirali vse besede bi se tej težavi lahko ognili) in še kakšne druge. Vendar največja pomankljivost je pomanjkanje semantičnosti.

Z uvedbo iskanja po semantičnosti vsebine, namesto po ključnih besedah, lahko veliko bolj natančno in zanesljivo najdemo iskano vsebino [19]. Če tudi iskano besedilo ne vsebuje nobene ključne besede, ki se nahaja v poizvedbi, ga bo semantični iskalnik našel, če se poizvedba semantično ujema z iskano vsebino.

To močno pripomore k težavi, kako najti relevantno vsebino, ko ne vemo povsem najbolje kaj iščemo. Oseba A lahko opiše primer in v poizvedbo poda ključne informacije, nakar dobi željene rezultate, do katerih drugače ne bi morala priti. Osebi B, ki pa ne pozna prav nič pravne terminologije, se pa odpre možnost da v poljudnem jeziku povpraša kaj jo zanima, česar drugače ne bi morala.

Vse kar še preostane je povezati vse dele v delujočo celoto. Namesto, da mora končen uporabnik prvo napisati poizvedbo, nato izbrati med najdenimi rezultati, za vsakega od rezultatov pogledati dodatne informacije preko ustreznih referenc, ter vse skupaj združiti v odgovor na prvotno vprašanje, si zamislimo asistenta. Avtonomnega asistenta, ki opravi vse zgoraj našteto. Končena rešitev (asistent) sprejme uporabnikovo poizvedbo, naredi vse po-

trebne poizvedbe, ter uporabniku vrne odgovor na poizvedbo, ter seznam virov od koder je dobil informacije.

Namen te naloge je ugotoviti, kako učinkovito lahko ta avtonomni sistem deluje na področju prava. Kako uspešne so različne implementacije v različnih situacijah, da uporabniku pomagajo na pravnem področju. Končena rešitev je torej avtonomni pravni asistent.

1.3 Vpliv

Količina časa, ki jo pravniki in drugi delavci z znanjem (angl. knowledge worker) porabijo za iskanje informacij, je vse prej kot zanemarljiva. Ocena na področju bančništva je, da vsak zaposleni porabi v povprečju uro na dan za iskanje po dokumentih. Gledano na ravni organizacije, je to več tisoče ur zapravljenih vsak dan. Ur, ki bi se lahko koristile za pomembnejše odločitve.

Inteligentni sistemi in orodja, ki so na voljo razvijalcem programske opreme, se včasih zdijo kot čarovnija. Veliko dela je ponovljivega, kjer ta orodja zasvetijo in nam prihranijo veliko časa, ki ga lahko namenimo za globje razmišljanje.

Tehnologija nam omogoči, da stvari, ki smo jih že delali, naredimo na enostavnejši in lažji način. Pravni asistent je le ena od novih inovacij tehnologije. S tem ko pravnikom avtomatizira iskanje literature in ponuja natančne odgovore na vprašanja, jim čas, ki bi ga sicer porabili za te dejavnosti, ostane za bolj kompleksne naloge. Kot so razmišljanje o tem kakšen pristop vzeti, kako povezati ideje, ipd.

Vendar povečana efektivnost pravnih delavcev je le ena stran kovanca. Druga izboljšava je možnost, da ljudje, ki si ne morejo privoščiti pravnih storitev, jih preko teh rešitev lahko dobijo. Kitajci imajo to možnost z uporabo sistema ChatLaw, medtem ko imamo pri nas Pravka (oba podrobnejše opisana v naslednjem poglavju).

Pomembno je omeniti, da namen teh sistemov ni, da bi nadomestili strokovnjake na področju. Tako kot od ChatGPT-ja ne pričakujemo, da nam

bo samostojno napisal produkcijsko aplikacijo, ne pričakujeno od pravnega asistenta, da nas bo zagovarjal na sodišču ali nam napisal pogodbo. Vendar redek je programer danes, ki si ne pomaga s temi sistemi za prototipiranje in še marsikatero drugo stvar. Pravni asistent je želja razširiti ta orodja še na pravno področje.

Poglavje 2

Tehnologije in sorodno delo

Avtonomni sistem, ki od uporabnika prejme vprašanje oz. opis primera in mu nazaj vrne generiran odgovor, je velik jezikovni model (VJM). Vsi VJM-ji v uporabi danes so zgrajeni na arhitekturi transformerja [21].

2.1 Transformer

Transformer je inovativen in trenutno najboljša arhitektura za "razumevanje" jezika (modeli zgrajeni na arhitekturi transformerja so najuspešnejši na področju procesiranja naravnega jezika). Za razliko od prejšnjih poskusov z rekurentnimi nevronskimi mrežami (RNN), je transformer veliko bolj uspešen pri razumevanju povezav med besedami (in s tem pomena besed) z uporabo pozornosti (angl. attention). Pozornost v grobem deluje tako, da se za vsak žeton v vhodnem nizu za vsak drug žeton v nizu določi utež, ki predstavlja kako povezana oz. pomembna za razumevanje drug drugega sta si žetona.

Nadalje je transformer sestavljen iz več zaporednih kodirnikov in dekodirnikov. Kodirnik dobi celotno besedilo naenkrat in ga pretvori v vektor, ki predstavlja semantičnost besedila. Kodirnike omenimo že tu, saj bodo skupaj s sematičnimi vektorji pomembni kasneje. Vektor iz kodirnika se skupaj z do sedaj že zgeneriranim izhodnim nizom da v dekodirnik, kateri vrne verjetnost za vsakega od žetonov v besedišču modela. V transformer modelu

si sledi 6 zaporednih parov kodirnik-dekodirnik.

2.1.1 Veliki jezikovni modeli

Primeri VJM-jev v uporabi danes so GPT-4 [15] in Claude 3 [2] na strani zaprtih modelov, ter Llama 3 [1] na strani odprtih modelov. Vsi našteti modeli so zgrajeni na arhitekturi transformerja, vendar uporabljajo le dekodirnik. To jih naredi izključno generativne modele. Za vhoden niz zgenerirajo izhoden niz.

Na drugi strani imamo vložilne (angl. embedding) modele, ki uporabljajo le kodirnik. Ti modeli vhoden niz pretvorijo vhoden niz v semantični vektor.

VJM je podlaga na kateri lahko gradimo pravnega asistenta. Razumevanje jezika, zmožnost generiranja odgovora in ogromna količina znanja (podatki na katerih je nevronska mreža bila natrenirana) so dobra odskočna deska.

To je že prvi poskus implementacije, ki ga bomo uporabili: pred-treniran VJM. Služil bo kot osnovna točka za primerjavo drugih metod.

2.1.2 Pomankljivosti velikih jezikovnih modelov

Ta metoda je povsem odvisna od internega znanja modela. VJM lahko odgovori le na podlagi informacij na katerih je bil učen. To pomeni, da če slovenska zakonodaja ni bila v korpusu besedil, ki so se uporabile za treniranje modela, VJM ne bo moral pravilno odgovoriti na uporabnikovo vprašanje, saj informacij nima.

Pomanjkanje domenskega znanja se je že zgodaj uveljavil kot velik problem pri generacije natančnih in verodostojnih odgovorov. V tem primeru se zgodijo halucinacije - VJM si izmisli informacije, ki niso resnične. Halucinacije nastanejo kot posledica arhitekture in učenja modela: ker je VJM v osnovi nevronska mreža in verjetnostni model, za vhoden niz zgenerira ustrezen izhod. Ter ker je za določeno obliko vhoda naučen, da zgenerira izhod, ki ima določeno obliko, lahko izhod zgleda povsem pravilen, a so dejstva v

njem neresnična in izmišljena (halucinirana). Primer: če vprašamo za članke na temo "posledice uporabe samorogove krvi v kliničnih primerih", nam bo VJM lahko nazaj podal naslove člankov in povezave do njih, ki zgledajo povsem stvarni, a v resnici ne obstajajo. Vredno je omeniti, da modeli tega ne delajo namerno in s slabimi nameni, vendar so naučeni, da če ga vprašamo za članke, nam jih poda; ter verjetnostno oceni kako bi najverjetneje zgledali naslovi teh člankov, četudi ne obstajajo.

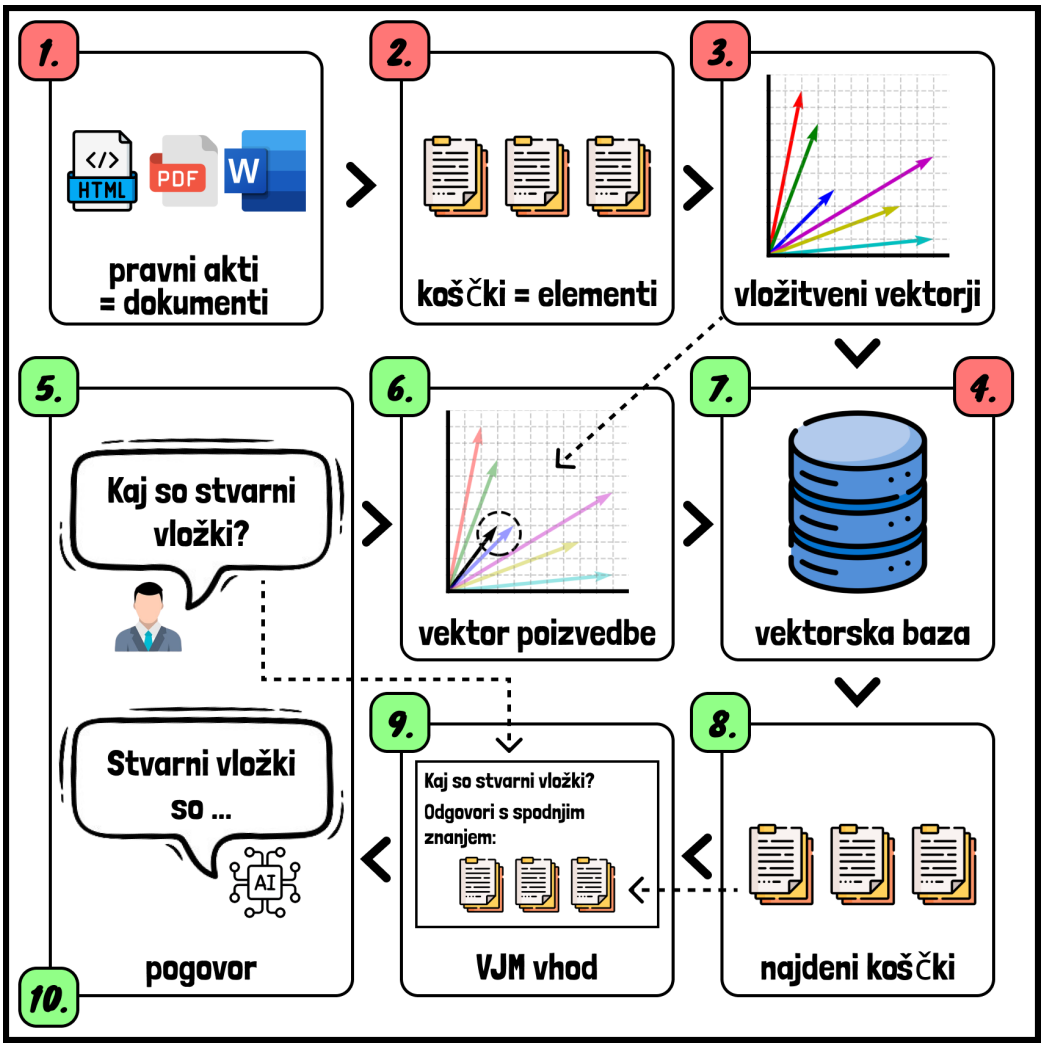
Halucinacije so včasih težko preverljive in ena izmed večjih ovir, da bi se VJM-ji lahko brez zadržkov uporabljali v kritičnih sistemih [26]. Pravo, kjer je skorajda vrstni red besed pomemben za pravilno interpretacijo zakonodaje, je dober primer, kjer napačne informacije lahko povzročijo velike in neprijetne posledice. Raziskave so pokazale, da celo VJM-ji specializirani na pravnem področju, halucinirajo do 33% časa [14], kar je zaskrbljivo.

2.2 S kontekstom obogateno generiranje (RAG)

Da bi rešili zgoraj omenjene težave, se je pojavila tehnika RAG [11]. Ideja je preprosta. Pred samo uporabo pripravimo vsa besedila z domenskim znanjem. Ko uporabnik postavi vprašanje, namesto da VJM odgovori s svojim notranjim znanjem, na podlagi uporabnikovega vprašanja najdemo najbolj relevantne vsebine v domenskih besedilih, jih podamo VJM-ju in mu naročimo, da naj na vprašanje odgovori le iz teh vsebin.

Domenska besedila vlagamo (angl. embed) s kodirnikom in s tem za vsakega dobimo semantični vektor. Ko uporabnik postavi vprašanje, se tudi za vprašanje izračuna semantični vektor, kateri se primerja z vektorji domenskih besedil. Najbolj pogosto uporabljena metrika je kosinusna razdalja. Uporabimo neko različico algoritma najbližjih sosedov, iz česar dobimo k najbližjih koščkov domenskega znanja. Na sliki 2.1 se lahko shematsko vidi potek te tehnike.

Izbira kodirnega modela, VJM-ja, vektorske baze, itd. so parametri, ki



Slika 2.1: RAG shema.

vplivajo na uspešnost metod. V vseh metodah konsistentno uporabljamo enake parametre (jih fiksiramo), s čimer zagotovimo enako podlago za vse metode. Vsi uporabljeni modeli so podrobneje opisani v poglavju implementacije.

2.2.1 Prednosti uporabe RAG-a

RAG tehnika je preprosta v ideji, a močna v izvedbi. Čeprav še zmeraj obstajajo številni izivi, RAG močno izboljša natančnost odgovorov in zmanjša halucinacije, predvsem na domenskem področju [4].

Dve dodatni veliki prednosti uporabe RAG-a sta nadzor nad znanjem, ter transparentnost virov. Ker so podatki ločeni od VJM-ja, shranjeni v zunanji bazi, imamo popoln nadzor nad tem, kateri podatki so notri. Tako lahko na primer zastarele podatke umaknemo in smo prepričani, da bodo informacije v odgovoru vedno aktualne. Če bi le te bile interni del VJM-ja, preko učenja ali dodatnega treniranja (angl. fine-tuning), nimamo nobenega nadzora kako jih bo VJM uporabil, niti zagotovila da so aktualne. Nadalje za vsako informacijo, ki podamo VJM-ju, vemo od kod prihaja, je enostavno našteti vire preko katerih lahko uporabnik preveri verodostojnost odgovora, ali pa si več prebere na to temo.

Zgoraj je opisan preprost ali naiven RAG (angl. naive RAG): prvo razdeli domenska besedila na manjše enote, pridobi vložitve, ko pride vprašanje pridobi vložitev vprašanja ga primerja z ostalimi, da dobi vsebine najbolj semantično podobne vprašanju in vprašanje skupaj z domenskim znanjem poda VJM-ju.

2.2.2 Pomankljivosti naivnega RAG-a

Drži, da že s tem ko uporabimo naiven RAG močno izboljšamo geniriran odgovor, vendar ima naiven RAG številne pomankljivosti, med njimi [8]:

- slabo zastavljeno vprašanje in nerelevante vsebine: če naredimo poizvedbo tako, da dobesedno primerjamo uporabnikovo zastavljeno vprašanje

z vsebinami, lahko dobimo nerelevante vsebine, če je vsebina poizvedbe zgrešena.

- ponavljanje vsebin: ne natančna priprava podatkov lahko povzroči, da se v eni poizvedbi vrne več različnih tekstov z isto vsebino.
- vračanje vsebin: namesto, da besedila VJM poveže v smiselno celoto in s tem uporabniku odgovori na vprašanje, preprosto le parafrazira vsebine (kot papiga).
- halucinacije: vedno je nevarnost, da se VJM "odloči" ne upoštevati navodil, ter se ne zmeni za podane informacije, ter poskusi odgovoriti iz internega znanja.

2.2.3 Napredni RAG

V ta namen se je razvila naprednejša tehnika: *napredni* RAG (angl. advanced RAG). Napredni RAG vzame temelje naivnega RAG-a, ter doda številne izboljšave.

Namesto da za poizvedbo uporabi uporabnikovo vprašanje takšno kot je, jo prvo pretvori v bolj primerno obliko. Obstaja več načinov, med katerimi sta razširitev poizvedbe in sprememba poizvedbe najbolj razširjeni. Preden naredimo poizvedbo, razširimo uporabnikovo vprašanje in ga pretvorimo v bolj primerno obliko. S tem dosežemo, da je poizvedba bližje končni iskani vsebini.

Naslednja velika sprememba se zgodi po tem ko najdemo dokumente. Preden so dokumetni podani v VJM, jih še enkrat predelamo. Dokumente z prerazaporjevalcem (angl. reranker) prerazporedimo, da so vsebine, ki bolje odgovarjajo na vprašanje, višje v rangirnem listu. Prerazporjevalec deluje namreč tako, da za vhodno poizvedbo in najden dokument vrne kako relevanten je dokument za poizvedbo [13]. Medtem ko v prvem koraku s kodirnikom vložimo poizvedbo in primerjamo semantično podobnost z ostalimi dokumenti, tukaj primerjamo kako relevantna je najdena vsebina za poizvedbo. Drobna, a pomembna razlika.

Še dva pogosta pristopa v temu delu sta povzetek in združevanje besedil (poleg prerazvrščevanja) [10]. Na področju prava ti tehniki izpadeta, saj bi parafrazacija ali izpuščanje zakonodaje hitro vodilo v napačno podane informacije. Vendar širša ideja za druga področja ostane ista: po tem ko najdemo dokumente, jih predelamo na način, ki bo VJM-ju omogočil kar se da dobro odgovoriti na uporabnikovo vprašanje.

Ta dva ključna dodatna dela, sprememba pred poizvedbo in sprememba po poizvedbi, močno izboljšata naivni RAG.

Vredno je omeniti, da je več poudarka postavljeno tudi na povsem prvi korak, ko pridobivamo podatke. "Garbage in, garbage out", je ena izmed prvih fraz, ki jo vsak nadobudni podatkovni znanstvenik sliši. Izboljšava tega koraka seveda ni omejena na napredni RAG, saj enako močno velja za naivni RAG.

2.2.4 Modularni RAG

Najnovejša različica RAG-a je modularni RAG. Namesto fiksne strukture kot jo imata naivni in napredni RAG, je tu vse ... modularno. Razvijalec za vsak primer sam presodi, katere komponente so najbolj ustrezne. Shematsko lahko module združimo v iskalne, spominske, preusmerjevalne, napovedovalne, itd. [8]. Osnovna ideja ostane enaka: glede na uporabnikovo vprašanje najdemo najbolj ustrezne dokumente, s katerimi VJM odgovori na uporabnikovo vprašanje. Vse ostalo je variabilno; razvijalec/raziskovalec presodi kateri moduli bodo za problem najustreznejši, ter jih vstavi v RAG cevovod.

Po temeljitem pregledu literature nisem zasledil modularne implementacije RAG-a na področje prava. Zatorej predlagam novo arhitekturo opisano v podpoglavju implementacije.

Do zdaj je bilo implicitno dorečeno, da je vse zunanje znanje (t.j. zakonodaja) shranjena v vektorski bazi. Kot pa sem že v uvodu namigoval, je grafna baza veliko bolj primerna za shranjevanje povezav med entitetami, česar v zakonodaji ne primankuje.

Hu in sod. delijo metodi za generiraje jezika s pred-treniranim VJM-jem z zunanjimi viri na (a) poizvedbene metode (RAG) in (b) grafno bazo. Znanje podgrafa vsebuje dodatne in poglobljene informacije določenega koncepta [9].

Konkretno to pomeni, da za določeno entiteto (npr. člen zakona) zraven povlečemo še dodatne relavatne vsebine, kot so referencirani členi, uporabljene definicije, itd.

2.3 Sorodni projekti

2.3.1 PISRS

Pravno informacijski sistem Republike Slovenije je "de facto" spletišče za pravnike. Vsebuje celotno slovensko zakonodajo, sodno prakso, evropsko zakonodajo, itd. Poleg dostopa do dejanskih datotek pravnih aktov, ponuja številne druge uporabne storitve, kot so:

- neuradna prečiščena besedila: za vsak zakon je na spletišču dostopno neuradno prečiščeno besedilo. Kakršnikoli popravki ali morebitne spremembe v drugih aktih, ki spreminjajo določen zakon, so vidni v predogled.
- povezava do sodne prakse: vsak sodni primer, ki se nanaša ka kakšno določbo iz zakona, je naštet.
- podatek o veljavnosti akta: ali je pravni akt veljaven ali ne.
- povezave med akti: povezave do ostalih aktov, ki vplivajo ali posegajo v besedilo. To so razni popravki zakonov ipd.
- pravna podlaga in podrejeni akti: iz česa izvira ta pravni akt in kateri drugi pravni akti so napisani na podlagi tega akta.
- napredno iskanje: možnost iskanja po naslovih ali vsebini, vrsti pravnega akta, času objave v uradnem listu, identifikacijske številke, itd.

Kot vidimo, ponuja pravno informacijski sistem pester nabor funkcionalnosti. Ni pravnega delavca, ki ne bi uporabljal PISRS. Vednar treba je poudariti, da je PISRS le iskalnik in ne vsebuje nobenih naprednih rešitev kot je umetna inteligenca.

Za vse kar ponuja, pa ne ponuja pogovora z velikim jezikovnim modelom, ki bi se vedel kot pravni strokovnjak. Prav tako je iskanje po vsebini omejeno na ključne besede in ne omogoča semantičnega iskanja.

2.3.2 Pravko

Pravko izgleda kot podoben projekt temu projektu. Gre za velik jezikovni model, ki je strokovnjak na področju prava. Glede na informacije na spletišču:

- razume vprašanja v pogovornem jeziku,
- poišče uradne vire za odgovor,
- preišče vire in poda najverjetnejšo rešitev,
- vrne uradne vire za nadaljno raziskovanje.

Žal implementacijskih podrobnosti ne vemo, vendar ključne komponente izgledajo analogne temu projektu. Za razumevanje pogovornega jezika se najverjetneje uporablja sprememba poizvedbe (angl. query transformation), za iskanje vsebin pa semantično iskanje. Vračanje uporabljenih virov na koncu je trivialno. Izgleda kot napredni ali modularni RAG.

Zdi se nam, da je Pravko uspešnejša implementacija glede na naše. Primerjavo naših metod s Pravkom lahko najdete na koncu poglavja o vrednotenju.

2.3.3 IUS-INFO

IUS-INFO je še eno spletišče za pravnike, ki se promovira kot "umetna inteligenca za pravnike". Storitev je plačljiva in avtorji tega članka nimamo izkušenj z uporabo, kot tudi ne drugih informacij glede implementacije, saj

gre za zaprto rešitev, tako da razen omembe ne moremo narediti nadaljne primerjave.

2.3.4 ChatLaw

ChatLaw [7] je podoben projekt temu projektu. Na področju kitajskega prava so želeli razviti sistem, ki je sposoben odgovarjati in se pogovarjati o poljubnih pravnih temah. Arhitektura je podobna, a bolj napredna in sofisticirana, kot kar smo do zdaj opisali.

Chatlaw je ekipa agentov, kjer ima vsak svojo vlogo. Agent je specializiran VJM, ustvarjen z namenom, da reši specifično težavo. Pogosto imajo agenti na voljo tudi orodja, ki jim omogočajo klicanje arbitrarnih funkcij [24]. Primer bi bil, da je orodje iskanje po vektorski bazi, kjer VJM poda iskalni niz. VJM se tako lahko odloči / presodi kdaj bi bilo smotrno iskanje po bazi.

Chatlaw je sestavljen iz naslednjih agentov: svetovalec, raziskovalec, odvetnik in urejevalec. Vsi agenti so tudi posebna vrst VJM-ja - mešanica strokovnjakov (angl. mixture of experts). Gre za posebno arhitekturo VJM-ja, ki je sestavljen iz več specializiranih enot in usmerjevalnika (router), ki glede na vhodni niz določi kateri od strokovnjakov je najbolj ustrezno sposoben odgovoriti na vprašanje.

Svetovalec uporablja pogoste vzorce poizvedb v pravnih svetovanjih, da uporabnika iterativno vodi skozi fazo pridobivanja informacij. Iz teh informacij sproti sestavlja graf znanja, ki predstavlja uporabnikovo unikatno situacijo. Ta korak je analogen s preoblikovanjem poizvedbe.

Raziskovalec je zadolžen za iskanje literature. Med iskanimi objekti so pravne entitete, povezave med njimi in pravni primeri. Vsak najden dokument je prerazporejen z uporabo VJM-ja. Ta korak je bržkone identičen koraku poizvedbe v naprednem RAG-u.

Odvetnik in urejevalec najdene vsebine predelata in vrneta uporabniku v ustrezni obliki (formi). Odvetnik pogleda najdene dokumente, se za vsakega odloči ali je ustrezen za primer, ter poveže znanje iz različnih virov v končen odgovor. Urejevalec vzame odgovor in ga pretvori v ustrezno obliko.

Vse komponente so do-trenirane na kitajskih pravnih dokumentih, kar izboljša pravno terminologijo agentov, ter izboljša učinkovitost iskalnika.

Poglavje 3

Podatki

Podatke delimo na dva sklopa: učno in testno množico. Učna množica je podmnožica slovenskih pravnih aktov, kateri določajo obseg znanja velikega jezikovnega modela. V praksi to pomeni: katere akte bomo razkosali, vložili, indeksirali in med njimi vzpostavili povezave; kateri akti bodo na voljo za iskanje. Testna množica so vprašanja na katera se lahko odgovori z znanjem iz aktov iz učne množice.

3.1 Zakonodaja

Ker je vprašanje te diplomske naloge, kako efektiven je lahko VJM kot pravni svetovalec na podoročju slovenskega prava, se omejimo na slovensko zakonodajo.

Kot je bilo rečeno že v uvodu, obstaja vrsta različnih pravnih aktov. Vendar zaradi obsežnosti ni izvedljivo, da bi za namen te diplomske naloge lahko indeksirali vso.

Iz tega namena in želje, da lahko dobro definiramo metriko in testno množico, se bomo omejili le na en pravni akt - Zakon o gospodarskih družbah.

Vse zakone je možno dobiti v PDF obliki na pravno informacijskem sistemu Republike Slovenije (*pisrs.si*).

Zakon je sestavljen iz več poglavij in podpoglavij, vsak od katerih ima

enega ali več členov. Člen je osnovna strukturna enota, ki celovito zajema en pojem.

Za nadaljno raziskovanje bi bilo treba dodati še več zakonov, kot drugih pravnih aktov, predvsem sodno prakso in popravke zakonov. Popravki zakonov bi zagotovili, da so informacije vedno aktualne in pravilne, sodne prakse pa ponudijo vselej nujno interpretacijo o tem, kako sodišče odloča v konkretnih primerih.

3.2 Testna množica

Da bi lahko primerjali uspešnost različnih implementacij smo določili 4 testne scenarije:

- **specifično vprašanje**: natančno zastavljeno vprašanje v pravni terminologiji, kjer so vse informacije dostopne na licu mesta. V praksi to pomeni, da iz zakona vzamemo neko trditev, primer, definicijo, ipd. in iz nje postavimo vprašanje. Vse informacije potrebne za pravilen odgovor so v celoti omejene na del besedila v katerem se nahajajo (npr. točka ali alineja člena).

- **specifično vprašanje z referencami**: enako kot zgoraj, a z eno razliko: v besedilu so omenjene vsebine drugih členov. Ko zakon bere dejanska oseba, ne računalniški sistem, in v besedilu piše "v skladu s prvim odstavkom 42. člena tega zakona", bo pogledala to dodatno vsebino, da lahko bolje in v celoti razume določilo. Tako naiven kot napredni RAG imata to pomankljivost, da tem referencam ne sledita, zoper česa ima VJM pomankljiv kontekts. Uporaba grafne baze bi morala rešiti ta problem.

- **splošno vprašanje**: medtem ko sta zgornji dve metriki omejeni na primere, ko so vse informacije lokalno dostopne, pri splošnem vprašanju ni tako. Tu preverjamo zmogljivost programa, da najde pomembne infor-

macije iz več različnih mest zakona, ter jih smiselno poveže v odgovor. Vprašanje je še vedno postavljeno v pravni terminologiji.

- **konkreten primer**: nazadnje se še preverja učinkovitost sistema za vsakodnevnega uporabnika. Vprašanje je sedaj opis primera v vsakodnevnem jeziku, kateremu sledi dejansko vprašanje. Tu se primerja še kako učinkovito lahko sistem nadomesti pravno osebo (v nekem smislu tolmača) v postopku.

Vsi testni primeri so omejeni le na obdelano literaturo, t.j. Zakon o gospodarskih družbah.

Ker nismo našli javno dostopne testne množice, ki bi preverjala vse željene kriterije, smo testno množico sestavili sami v sodelovanju s strokovnjaki na tem področju.

Imeli smo izbiro med manjšim številom kakovostnih testnih primerov in večjim številom manj kakovostnih testnih primerov. Po posvetovanju smo se odločili za manjše število visoko kakovostnih testnih primerov. Ker želimo preveriti ali bi se veliki jezikovni modeli kot pravni svetovalci lahko uporabljali v kritičnih sistemih, je bolj pomembno, da smo prepričani, da dobro delujejo na primerih, ki se dejansko pojavijo v praksi in niso le umetno postavljeni.

Z manjšim številom primerov smo tudi uspeli uresničiti ročno ocenjevanje generiranih odgovorov, česar v primeru velikega števila testnih primerov ne bi bilo mogoče. Človeška ocena nam zaradi potrebnega domenskega znanja veliko bolj osvetli luč na uspešnost metod kot računalniške metrike. To na omogoči, da veliko bolj realno lahko ocenimo uspešnost metod.

Pomankljivost manjšega števila testnih primerov je seveda ta, da obstaja nevarnost, da ne ocenimo metod na zadostnem številu podatkov, ter da testna množica ni reprezentativen vzorec. To težavo smo zmanjšali tako, da smo s testnimi primeri pokrili čim večji spekter vsebine.

Perspektiva študenta pri opravljanju izpita je morda smotrna na tem mestu: profesor ima omejeno število vprašanj, ki jih lahko postavi na izpitu,

s katerimi želi pokriti čim večji delež snovi. Študentovo znanje se oceni le na tem vzorcu snovi, pa je lahko ostalo snov znal bolje ali slabše. Podoben pristop smo ubrali tudi mi: za vsak scenarij smo sestavili kar se da reprezentativen vzorec. Ker so generirane odgovore ocenili strokovnjaki, smo lahko mnogo bolj prepričani v uspešnost metod, kot če bi uporabili le avtomatske metrike.

Za vsak scenarij smo tako pripravili 10 parov vprašanje-odgovor, kar skupaj nanese 40 parov.

V tabeli spodaj smo zbrali nekaj statistik za vsakega od scenarijev:

Tabela 3.1: Statistike podatkov glede na scenarij.

statistika	scenarij 1	scenarij 2	scenarij 3	scenarij 4
št. vprašanj	10	10	10	10
povpr. št. besed / vpr.	11	8	9	19
povpr. št. žetonov / vpr.	23	20	20	40
povpr. št. besed / odg.	38	102	139	125
povpr. št. žetonov / odg.	89	244	333	302
št. koščkov	0	329	3271	3271
povpr. št. besed / košček	0	273	32	32
povpr. št. žetonov / košček	0	644	74	74

Za vsak scenarij si spodaj lahko ogledate primer vprašanja (nekateri odgovori so dokaj dolgi, zato jih zaradi preglednosti raje izpustimo):

1. Na podlagi česa se določi sistem vodenja poslovnih knjig podjetnika?
2. Za katere družbe slovenski računovodski standardi ne določajo vsebine konsolidiranega letnega poročila?
3. Kako vodijo poslovne knjige družbe, ki se po velikostnih kriterijih razvrščajo med velike družbe?
4. Sem samostojni podjetnik posameznik in sem dejavnost prenesel na ženo. Je to mogoče? Kaj moram storiti?

Poglavje 4

Implementacija

Kot je bilo že omenjeno v pregledu literature, bomo primerjali 4 različne metode, vsaka naprednejša različica od prejšnje.

Velik jezikovni model, ki ga uporabljamo v vseh metodah, je kvantiziran Gemma 2 (gemma2:9b-instruct-q6_K) [20]. Po krajšem testiranju smo ugotovili, da za velikost modela, ki ga lahko spravimo na grafično kartico, se ta model najbolje odziva na ukaze in razume slovenski jezik. Veliko modelov je bolj zmogljivih, a še več jih ima pomankljivost nerazumevanja slovenskega jezika, zoper česa nam ne koristijo. Odločili smo se za odprte modele, zaprtih kot so GPT-4 in Claude 3 nismo preverjali.

4.1 VJM

Ta pristop služi za izhodiščno točko. VJM-ju se iz testne množice postavi vprašanje brez dodantih navodil ali konteksta. VJM mora tako iz internega znanja odgovoriti na vprašanje.

Največji pomankljivosti sta seveda pomanjkanje znanja (če slovenska zakonodaja ni bila v množici učnih podatkov), zaradi česar model lahko začne halucinirati, in aktualnost informacij.

Navodilo, ki ga VJM dobi, se glasi:

```
Si pravni pomočnik.
```

```
Uporabnik ti bo postavil vprašanje.
Nanj odgovori po najboljših zmogljivostih.
```

4.2 VJM + naivni RAG

Zakon o gospodarskih družbah razčlenimo na manjše enote, t.j. koščke (angl. chunks), fiksne dolžine in za vsako posebej izračunamo vložitveni vektor s kodirnikom. Zakon razčlenimo naključno, za kriterij delitve vzamemo znak za prelom vrstice. Razlog, zakaj zakon razdelimo na manjše dele, je več stopenjski. Kot prvo imata oba, kodirnik in velik jezikovni model, omejeno število žetonov, ki jih prejmeta na vhodu. Celoten zakon se tako ne more procesirati v enem koraku. Kot drugo je problem preveč konteksta - če iščemo informacijo, ki se nahaja v enem odstavku, a imamo za enoto celotno poglavje, bo vložitveni vektor poglavja mnogo bolj splošen in drugačen od vektorja odstavka. Nastopita težavi, da ali relevantnega besedila ne najdemo, ali da ima VJM preveč informacij in ne uspe izluščit potrebnih.

Vektorje shranimo v vektorski bazi. Za vektorsko bazo uporabljamo odprto-kodno bazo imenovano Chroma [6]. Vsi vektorji so shranjeni v isti kolekciji.

Vsi vložitveni vektorji so pridobljeni s kodirnikom multiligual-e5-large [22]. Gre za kodirnik zgrajen na arhitekturi SBERT-a oz. transformerja stavkov (angl. sentence transformer) [18].

Za razliko od navadnega kodirnika, kot ga poznamo iz transformerja, se tu vzporedno uporabljata dva identična kodirnika, ki ju naučimo prepoznavati katera vhodna besedila so si podobna in katera ne. Ta premik perspektive omogoča modelom, da veliko bolje razlikujejo katere vsebine so si sorodne; kar je ključnega pomena, saj na ta način lahko za vhodno poizvedbo zanesljivo najdemo del besedila, ki je podoben tej poizvedbi (pri predpostavki da nam ta podobnost nudi informacije za odgovor na poizvedbo).

Ker uporabljamo slovenski jezik, uporabljamo tudi večjezično različico transformerja stavkov [17]. Gre za razširitev enojezičnih transforjev stavkov

na večjezične.

Ta pristop je enostaven, a pomankljiv. Ker ne delimo besedila glede na strukturo, niti ne na semantiko, ampak na določene znake, obstaja nevarnost, da se pomembna vsebina ne nahaja na istem mestu. Primer: člen prelomimo na polovici, s čimer potem nimamo na voljo celotne vsebine člena, ko najdemo eno od polovic.

Uporabnikovo vprašanje se uporabi takšno kot je, brez kakšnih transformacij. Vprašanje se kot vhod poda vložitvenemu modelu, od koder dobimo vložitveni vektor. Vektor vprašanja primerjamo z ostalimi vektorji v vektorski bazi, da dobimo k vsebinsko najbližjih koščkov. Za iskanje sosedov se uporabi različica metode najbližjih sosedov.

Vprašanje skupaj z najdenimi vsebinami podamo VJM-ju, ter mu ukažemo naj na vprašanje odgovori le z znanjem iz najdenih vsebin. Navodilo se glasi:

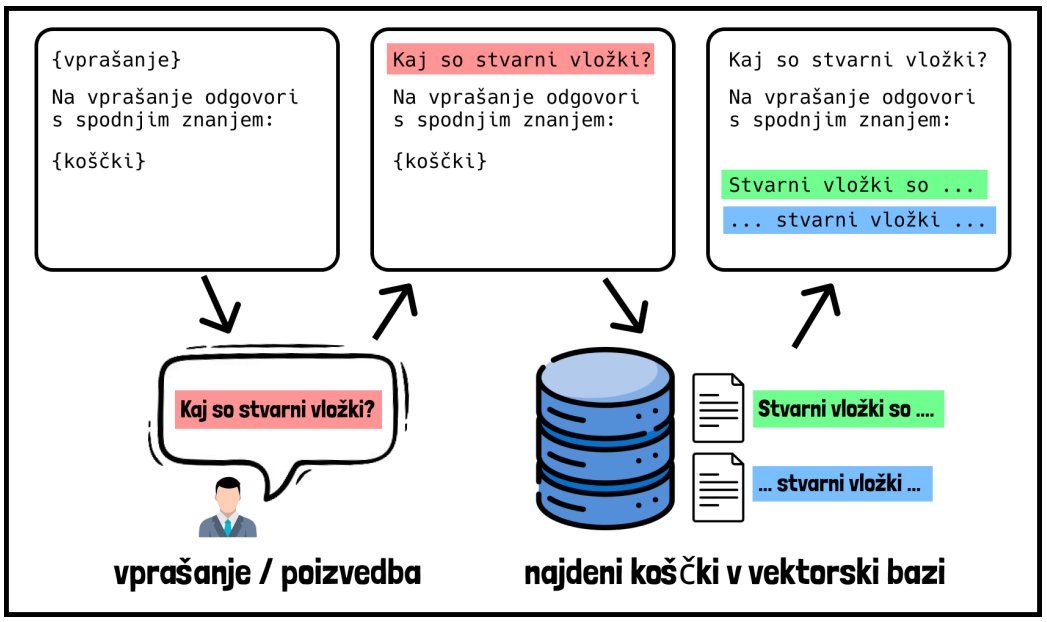
```
Si pravni pomočnik.
Uporabnik ti bo postavil vprašanje.
Nanj odgovori z znanjem iz spodnjih podatkov:

{znanje najdeno v vektorski bazi}
```

Kako poteka postopek sestave navodila za VJM pri naivnem RAG-u, tako kot pri naprednem in modularnem, je prikazano na sliki 4.1.

4.3 VJM + napredni RAG

Zakon sedaj za razliko od prejšnega pristopa razčlenimo glede na strukturne enote - člene. Vsak člen nadalje rekurzivno razdelimo na manjše enote, kot so odstavki, točke in alineje. Vsako enoto posebaj vložimo. Tako imamo večje in manjše enote, nekatere zelo specifične, nekatere bolj splošne - od vsebine celotnega člena do posamezne alineje. Ko iskalnik najde košček, pogleda v katerem je vsebovan (če je) in razširi vsebino, da vključuje vse informacije. Ta tehnika se imenuje small2big [25]. Primer: če je kot košček najdena alineja,



Slika 4.1: Postopek sestave vhoda za VJM pri RAG tehnikah.

se za besedilo ne vzame le ta alineja, temveč besedilo celotnega odstavka v katerem se ta alineja nahaja.

Preden se vsebine razširijo, se jih še prerazporedi. To zagotovi, da je možna informacija dejansko relevantna za odgovor na vprašanje, ne le vsebinsko podobna.

Za prerazporejevalnik (angl. reranker) uporabljamo model bge-reranker-v2-m3 [12] [3].

Še preden pa se izvede kakršnakoli poizvedba, se uporabnikovo vprašanje spremeni v bolj ustrezno obliko. To naredi VJM z navodili, naj vprašanje zastavi na bolj jasen način in uporablja pravno terminologijo. Ta komponenta je v ChatLaw-ju bila dodatno natrenirana na več milijon ročno ustvarjenih primerih, kar omogoča veliko večjo učinkovitost. VJM uporabljen v tej nalogi ni dodatno natreniran. Tako kot v vseh ostalih primerih je za VJM tu uporabljena Gemma 2 z naslednjimi navodili:

```
# Navodilo
```

```
Pretvori  besedilo spodaj v pravni jezik primerem
za iskanje po pravnih dokumentih (zakonih).

Osredotoči se, da iz besedila vzameš vse pomembne informacije
in jih pretvoriš v pravno terminologijo.
Pravni dokumenti so napisani v pravni terminologiji.
Dokumente iščemo s primerjanjem sematičnosti besedil
(tj. rag vectorstore retrieval).
V ta namen želimo spodnje besedilo pretvoriti v ustrezno pravno obliko,
da najdemo vsebine, ki so najbolj primerne za uporabnikovo poizvedbo.

Pretvorba naj bo v preprosti tekstovni obliki,
brez kakršnegakoli formatiranja.

# Besedilo
{prevorjena poizvedba}
```

Zadnji del ostane enak kot pri naivnem RAG-u: najdeni koščki se v vrstnem redu podajo VJM-ju skupaj s prvotnim vprašanjem in navodilom, naj na vprašanje odgovori samo s podatki iz konteksta.

4.4 VJM + modularni RAG

V predprocesiranju zakona zdaj izluščimo še reference in jih shranimo v grafno bazo. Torej za vsako omembo drugega dela zakona, ki se pojavi v besedilu, kot je npr. "... če je to v skladu s prvim odstavkom 42. člena tega zakona", v grafni bazi shranimo povezavo med tema dvema entitetama. Za grafno bazo uporabljamo Neo4j [23].

Vsega skupaj je 4278 vozlišč pridobljenih iz zakona o gospodarskih družbah. Vsako vozlišče je označeno kot dokument (ta je en sam), poglavje (strukturna enota, vključuje poglavja, odseke, podoseke, itd.) ali element (vsebina zakonov). Vozlišča tvorijo drevesno strukturo z dokumentom v korenini,

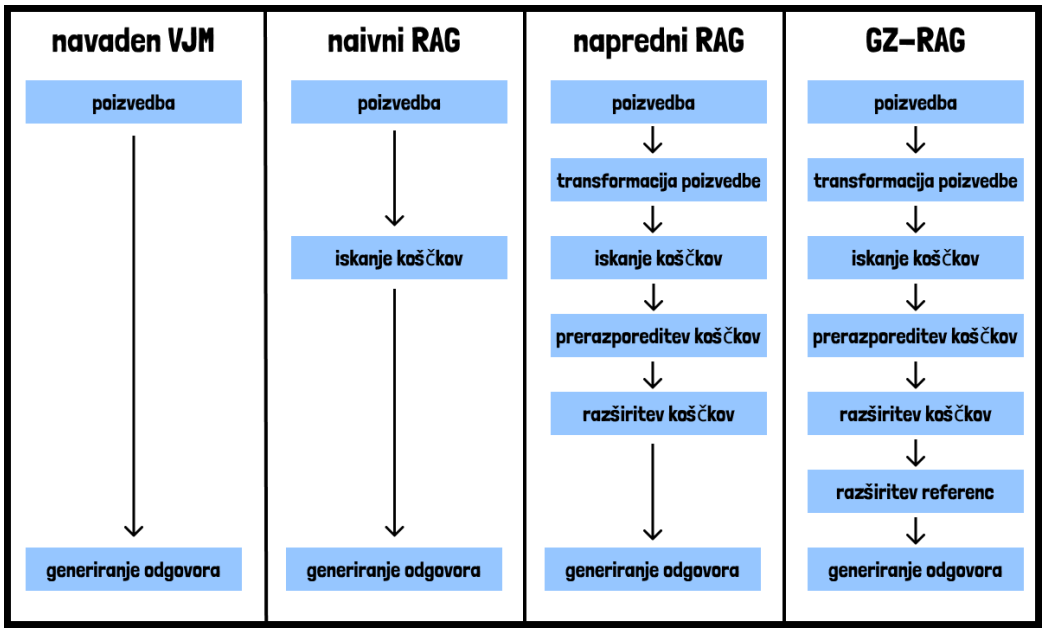
vse do najmanših elemntov kot so alineje preko poglavij, če upoštevamo le strukturne povezave. Stukturna povezava je kaže od vozlišča do starša tega vozlišča.

Drug tip povezave v grafu so referenčne povezave. Za vsako omembo drugega dela zakona, ki se pojavi v kakšnem besedilu elementa, vzpostavimo povezavo med tema dvema vozliščema.

Postopek pridobivanja konteksta je enak kot pri naprednem RAG-u, le da imamo na koncu še en dodaten korak: Za vsak košček, ki ga damo VJM-ju, najprej združimo vse reference, ki so omenjene v besedilu koščka (s tem razširimo besedilo na vse informacije), ter vse reference, ki referencirajo ta košček in ga na kakršenkoli način spreminjajo.

S tem podamo VJM-ju vse potrebne informacije. Pri naprednem RAG-u za zgornji primer ne moremo vedeti kaj so pogoji prvega odstavka 42. člena (ker le ta ni bil najden ob poizvedbi), zaradi česar je odgovor pomankljiv, če ne celo napačen.

Za vizualno primerjavo teh štirih metod, si oglejte sliko 4.2.



Slika 4.2: Primerjava sestavnih delov implementiranih metod v tem delu.

Poglavje 5

Vrednotenje in diskusija

Vse 4 različne metode opisane v poglavju implementacije smo testirali na 4 različnih scenarijih opisanih v poglavju podatkov. Za testiranje smo uporabili testne primere, ki smo jih pripravili specifično v namen tega raziskovalnega dela.

5.1 Metrike

Uspešnost metod smo preverili na več načinov. Uporabljene metrike so naštete spodaj, kot tudi njihov opis in zakaj so bile uporabljene:

- **BLEU**: preverja število ujemajočih se N-gramov med željenim in geniriranim besedilom. Tipično se ta metrika uporablja v prevajanju, t.j. kako dobro računalniški sistem prevede besedilo v primerjavi s človeškim prevodom. Za N se uporabijo različne vrednosti, nakar se te vrednosti še spremeni s faktorji, ki penalizirajo pogosto uporabljene N-grame in prekratke prevode, da se doseže večja natančnost. Vrednosti se gibljejo med 0 in 1, kjer 0 pomeni, da med besediloma ni nobenega ujemanja, 1 pa predstavlja popolno ujemanje. Za primer te diplomske naloge ta metrika preverja, ali se v generiranem odgovoru ključne besede pojavijo v določenem vrstnem redu (v lokalnem merilu). Velika pomankljivost je seveda ta, da se ne preverja sematničnost odgovora.

Zato četudi VJM pravilno odgovori na vprašanje, vendar z drugačnimi besedami, bo dobil slabo oceno.

- **ROUGE-1**: se večinoma uporablja pri preverjanju kakovosti prevedenih besedil. Tako kot BLEU za gradnik primerjave vzame N-gram, pri ROUGE-1 je to 1-gram (besede). Ta metrika preverja, če se pojavijo ključne besede v generiranem odgovoru in v kolišnem številu. Tako kot pri BLEU je razpon od 0 do 1, kjer je 1 popolno ujemanje in 0 popolno ne ujemanje. Pomankljivosti so enake kot pri BLEU metriki.

- **ROUGE-L**: meri najdaljši (nepovezan) niz besed med generiranim in testnim besedilom. Tu pride veliko bolj do pomena v kakšnem vrstnem redu se besede pojavijo v primerjavi z zgornjima dvema primeroma. Prav tako ni omejen na dejstvo, da se morajo besede vrstiti neposredno ena za drugo, vendar se lahko pojavijo v različnih (kasnejših) delih besedila. To nam omogoča, da bolje preverimo tekočnost odgovora, vendar še vedno ne zagotovi sematičnosti. Če se VJM namreč izrazi z drugačnimi besedami kot so v testnem primeru, bo rezultat slab. Vendar v pravu je dosti domenske terminologije, kar bi moralo to težavo nekoliko uravnovesiti.

- **človeška ocena (splošno)**: je daleč najpomembnejša metrika, saj nam da vedeti, kako bi strokovnjak ocenil odgovore. Veliko bolj smo lahko prepričani kako natančni in uporabni so odgovori dejansko v praksi. Pa prva različica, splošna, različica človeške ocene preverja pravilnost odgovora v bolj splošnem pomenu, medtem ko spodnja različica preverja le v okviru predelane zakonodaje, t.j. ZGD. Na primer vprašanje "Kaj so obveznosti revizijske komisije?" se da odgovoriti zunaj okvirja ZGD. Z znanjem, ki ga je JVM morda videl v drugih virih ali drugih oblikah med treniranjem.

- **človeška ocena (ZGD)**: za razliko od splošne ocene v prejšnjem primeru se tu odgovor oceni izključno v okvirih ZGD-ja. Revizijska komisija ima po ZGD-ju neko definicijo in nanjo vezane določbe, ki se lahko

razlikujejo od splošne prispodobe revizijske komisije in so bolj specifične. Z ločevanjem teh dveh primerov lahko razločimo kako uspešno metode odgovarjajo v splošnem in v ozkem pogledu določenih besedil. Obe človeški oceni imata celodecimalsko oceno med 0 in 1 (odgovori so bili ocenjeni od 0 do 10, ter kasneje deljeni z 10, da imajo ocene enak razpon kot prejšnje metrike).

5.2 Rezultati

Zaradi boljše preglednosti v tabelah rezultatov uporabljamo naslednje oznake za metode:

- VJM: je prva metoda, t.j. navaden VJM.
- RAG: je naivni RAG.
- n. RAG: je napredni RAG.
- GZ-RAG: pomeni graf znanje RAG. Gre za zadnjo metodo, modularni RAG oz. napredni RAG z grafno bazo.

5.2.1 Specifična vprašanja

Tabela 5.1: Ocene za specifična vprašanja.

metrika	VJM	RAG	n. RAG	GZ-RAG
BLEU	0,00	0,00	**0,16**	0,13
ROUGE-1	0,13	0,10	0,35	**0,41**
ROUGE-L	0,08	0,07	0,32	**0,35**
človeška ocena splošno	0,46	0,25	0,62	**0,76**
človeška ocena ZGD	0,16	0,04	0,62	**0,76**

Prva opazka je, da sta prvi dve metodi, navaden VJM in naivni RAG, občutno inferiorni v primerjavi z naprednim in modularnim RAG-om. Z

razliko splošne človeške ocene, vse ostale metrike nakazujejo na nezmogljivost teh dveh pristopov, da uspešno odgovorita na poizvedbe. Da splošna človeška ocena ni tako slaba namiguje na dejstvo, da so modeli sorodne koncepte videli v drugih situacijah (izven okvira ZGD-ja) in so uspeli nekoliko bolje odgovoriti.

Da je navaden VJM uspešnejši od naivnega RAG-a si razlagamo tako, da s tem ko modelu v naivnem RAG-u ukažemo, da naj odgovori na vprašanje z najdenim znanjem, vendar se s podanimi podatki ne da odgovoriti na vprašanje (najdene vsebine se ne navezujejo na vprašanje), poskuša z napačnimi podatki odgovoriti na vprašanje, zaradi česar je odgovor napačen. Da je že napredni RAG toliko bolj uspešen od naivnega RAG-a potrdi hipotezo, da je izbira velikosti koščkov pri predprocesiranju podatkov zelo pomembna, saj nam omogoči, da bolj natančno najdemo pravilno vsebino.

Nekoliko zanimivo je videti, da GZ-RAG uspešnejši od naprednega RAG-a. V tem scenariju to ni bilo za pričakovati, saj glede na to, da so vse potrebne informacije za pravilen odgovor dostopne na enem mestu (in je treba to košček le pravilno najti), nam besedilo iz referenc ne bi kaj dosti pomagalo.

Da sta obe človeški oceni enaki pri naprednem RAG-u in GZ-RAG-u pomeni, da sta modela odgovarjala ekskluzivno z najdeno literaturo, t.j. najdenim znanjem iz ZGD-ja, in da se splošen in ZGD kontekst nista razlikovala.

Druga opazka je ta, da avtomatske metrike, BLEU in ROUGE, ne uspeta efektivno oceniti uspešnost odgovorov. Recimo pri navandem VJM-ju sta BLEU in ROUGE metriki blizu 0, medtem ko je splošna človeška ocena nekoliko pod 0.5. N-grami in prekrivanja besedila kot kaže nista zadostni merili za preverjanje semantične pravilnosti odgovora.

Odločitev manjšega števila testnih primerov in posledično možnost ročnega ocenjevanja se je izkazala kot dobra odločitev, saj bi v nasprotnem primeru narobe interpretirali rezultate glede na ostale metrike.

5.2.2 Specifična vprašanja z referencami

Tabela 5.2: Ocene za specifična vprašanja z referencami.

metrika	VJM	RAG	n. RAG	GZ-RAG
BLEU	0,02	0,01	0,12	**0,17**
ROUGE-1	0,22	0,22	0,45	**0,46**
ROUGE-L	0,14	0,13	0,38	**0,38**
človeška ocena splošno	0,36	0,38	0,58	**0,81**
človeška ocena ZGD	0,07	0,07	0,59	**0,81**

Tu vidimo podobno stanje kot v prejšnjem testnem scenariju. Prvi dve metodi sta dokaj manj uspešni od zadnjih dveh. ROUGE-1 je višja pri vseh metodah, kar pomeni, da so modeli v tem scenariju uporabljali več ključnih besed oz. besed, ki se pojavijo v pravilnem odgovoru.

Človeška ocena za prvi dve metodi je podobna kot prej: splošno sta nekoliko uspešni, kar nakazuje na osnovno splošno razumevanje koncepta, vendar ne v okviru ZGD-ja.

GZ-RAG je odločno najuspešnejši. Dodatne informacije, ki jih VJM dobi preko referenc, naredijo veliko razliko s tem da modelu podajo več potrebnih informacij za pravilen odgovor. To je pomankljivost, ki se vidi v naprednem RAG-u. Medtem ko je v prejšnjem scenariju dohajal GZ-RAG-u, referencirani koščki niso med najdenimi, zaradi česar odgovor ni nujno napačen, le pomankljiv.

5.2.3 Splošna vprašanja

Tabela 5.3: Ocene za splošna vprašanja.

metrika	VJM	RAG	n. RAG	GZ-RAG
BLEU	0,00	0,00	0,03	**0,04**
ROUGE-1	0,25	0,25	0,30	**0,30**
ROUGE-L	0,13	0,12	**0,21**	0,18
človeška ocena splošno	0,22	0,18	**0,58**	0,54
človeška ocena ZGD	0,07	0,02	**0,58**	0,53

Napredni RAG je v tem scenariju uspešnejši kot GZ-RAG! Razlike se majhne, tako da ne moremo z gotovostjo trditi kaj je odločilni faktor, vendar si lahko razlagamo sledeče: v tem scenariju se preverja, če so najdeni vse relevantne vsebine, ki se nahajajo v več različnih členih. Zatorej ni tako pomembno, da za najdene člene vključimo njihove reference, marveč dejstvo da jih sploh najdemo. Torej se je v naprednem RAG-u našlo več relevantnih členov s katerimi je bilo moč odgovoriti na vprašanja. Dodatno razširjeno besedilo pridobljeno preko referenc pa ni niti približno prineslo take teže kot v prejšnjem scenariju.

Dodatna razlaga je možnost predolgega konteksta v primeru GZ-RAG-a. Saj namreč za vsakega od najdenih koščkov povlečemo še vso referencirano besedilo, kar v nekaterih primerih lahko pomeni vsebino celotnih členov, je vhod v VJM mnogo daljši kot pri naprednem RAG-u. Po predstavlja dve težavi: prvič, informacij je veliko več in presoditi katere so pomembne ima več možnosti za napako; drugič, če je skupek dolžine vseh besedil predolg in preseže maksimalno dolžino vhoda VJM-ja, se besedilo grobo odseka pri maksimalnem številu žetonov. Tako se del znanja (najdenih koščkov) sploh ne pojavi v vhodu VJM-ja.

Opazi se tudi manjša uspešnost naprednega RAG-a in GZ-RAG-a, vsaj glede na človeške ocene. Obe ROUGE metriki sta višji kot v prejšnjih dveh scenarijih, kar nakazuje na to, da so generirani odgovori bolj skladni s pravilnimi iz testne množice.

Navaden VJM in naivni RAG tudi tu ne izkazujeta dobrih rezultatov.

5.2.4 Konkretni primeri

Tabela 5.4: Ocene za konkretne primere.

metrika	VJM	RAG	n. RAG	GZ-RAG
BLEU	0,00	0,00	0,05	**0,05**
ROUGE-1	0,22	0,21	0,31	**0,32**
ROUGE-L	0,11	0,11	**0,21**	0,18
človeška ocena splošno	0,11	0,15	**0,71**	0,69
človeška ocena ZGD	0,00	0,00	**0,73**	0,66

Prva opazka so višje človeške ocene za zadnjih dve metodah v primerjavi s prejšnjim scenarijem. Pričakovali smo nižje. Vprašanja v prejšnjem scenariju so bila postavljena natančno in v ustrezni terminologiji, medtem ko so vprašanja v tem scenariju poslošitev tistih vprašanj na njihovo bolj poljudno obliko. Transformerji poizvedbe so naredili odlično delo, da so modeli uspeli najti ustrezne koščke napisane v pravni terminologiji. Vendar ne smemo zanemariti dejstva, da tako kot vprašanja so tudi odgovori podani v bolj poljudni obliki. Znano dejstvo je, da so VJM-ji bolj uspešni na splošnih področjih in vse manj uspešni in natančni na vse bolj domenskem znanju. Torej so generirani odgovori v tem scenariju bili primernejši od prejšnjega, saj so VJM-ji uspešnejši v poljudnem okolju.

5.2.5 Primerjava naših metod s Pravkom

Sedaj naredimo še primerjavo s Pravkom, da ocenimo, kako uspešne so naše metode v primerjavi z že obstoječimi rešitvami. Ker je za neplačniške uporabnike število vprašanj, ki jih lahko postavimo Pravku, omejeno na 5, smo tako za vsak scenarij Pravku postavili le eno vprašanje iz naše testne množice. S tem dobimo bolj informativno primerjavo, za konkretnejšo bi se bilo potrebno naročiti na Pravkove storitve in narediti bolj temeljito primerjavo. Za

vprašanja smo izbrali tista, pri katerih so bile naše metode (dokaj) uspešne, vendar je še prostor za izboljšavo. Na ta način lahko bolje razločimo kako uspešen je Pravko v primerjavi z našimi metodami.

Tabela 5.5: Primerjava naših metod s Pravkom.

scenarij	VJM	RAG	n. RAG	GZ-RAG	Pravko
scenarij 1	0,70	0,40	0,60	**0,90**	0,80
scenarij 2	0,40	0,40	0,80	0,80	**0,90**
scenarij 3	0,40	0,20	**0,80**	**0,80**	0,20
scenarij 4	0,30	0,30	0,70	0,80	**0,90**

Pravko ima uspešne rezultate, primerljive z naprednim in modularnim RAG-om. Razen tretjega scenarija, kjer ima Pravko slabši rezultat kot celo navaden VJM, so rezultati na isti ravni kot modularni RAG. Naše metode, predvsem napredni in modularni RAG, so podobno uspešne kot Pravko.

Pomembno je omeniti, da smo za vsak scenarij preverili le eno vprašanje in to vprašanje je bilo izbrano tako, da so naše metode uspešno odgovorile nanj. Da bi z gotovostjo lahko kaj trdili, bi bila potrebna bolj obsežna primerjava.

5.3 Diskusija

Navaden VJM in naivni RAG nikakor nista primerna za kakršnokoli uporabo. Od navadnega VJM-ja je to bilo za pričakovati, saj (zelo verjetno) slovenska zakonodaja in s tem ZGD ni bil v korpusu učnih podatkov, torej potrebnega znanja za pravilen odgovor nima. Po splošnih človeških ocenah vidimo, da do neke mere še ponudi uporabno informacijo, vendar premalokrat in premalo zanesljivo.

Od naivnega RAG-a smo pričakovali več, če ne drugega vsaj boljšo uspešnost kot pri navadnem VJM-ju, ki je služil kot primerjalna točka. Da je uspešnost slabša lahko pomeni le, da najdena besedila niso prav nič pripomogla k pravilnosti odgovora, če niso na mestih celo škodila. Očiten razlog so napačno

najdena besedila, t.j. besedila členov, ki smo jih dali VJM-ju, niso relevantna za vprašanje. Vprašanja so bila strokovno postavljena, zato pomanjkanje transformacije poizvedbe ne bi smela biti težava. Ostaneta še velikost koščkov in prerazporejanje. Avtorji se nagibamo k temu, da je večji vpliv na slabo uspešnost imela velikost koščkov, kar je povzročilo, da najdeni koščki niso bili relevantni za odgovor.

Največji preskok je med naivnim RAG-om in naprednim RAG-om. S tem, da se podatki bolje pripravijo in bolje najdejo, se je pokazala velika razlika v natančnosti odgovorov. Med naprednim in modularnim RAG-om so razlike dosti manjše, kar je bilo do določnene mere tudi za pričakovati. Medtem ko je med naivnim in naprednim RAG-om kopica izboljšav, se GZ-RAG od naprednega RAG-a razlikuje le v zadnjem koraku, ko za vsak koščke zbere še vsa referencirana besedila. Ko je poudarek na tej lastnosti, kot je bil v scenariju 2, se vidi, da je GZ-RAG uspešnejši od naprednega RAG-a. V primerih, ko so vse (ali večina) informacij zbrane na licu mesta (člena), ni kaj za nagraditi. Ali če referencirani informacije ne prinesejo koristnih informacij, so lahko le motilec, zaradi česar napredni RAG ne rabi razločevati med pomembnimi in nepomembnimi informacijami in je v teh primerih uspešnejši, kot je bil v scenarijih 3 in 4.

Kot rečeno, prvi dve metodi za pravnega asistenta nista primerni. Izgleda da je pravo še zmeraj posebno področje, ki potrebuje specializirane rešitve namesto generičnih. Zadnji dve metodi kažeta upanje, da bi se z nadaljnimi izboljšavami te sistemi lahko začeli uporabljati v praksi. Ne še na nivoju splošnega svetovalca, vendar mogoče kot pomočnik pravnemu delavcu, kjer je oseba v stalnem kontaktu z zakonodajo.

Poglavje 6

Zaključek

Cilj tega raziskovalnega dela je bilo odgovoriti na vprašanje "Ali se pravni asistent (VJM) lahko uporablja v vsakodnevnih situacijah?". Ker je pravo kompleksno področje, smo predstavili več različnih metod. Za največji problem se je izkazalo pomanjkanje pravnega znanja VJM-ja, predvsem kar se tiče zakona o gospodarskih družbah, zaradi česar smo predlagali tehniko RAG, da bi VJM-ju umetno podali potrebno znanje.

Rezultati nam pustijo še marsikaj želenega. Navaden VJM in naivni RAG nista uspešna praktično v nobenem primeru, njuni rezultati so zelo slabi v vseh scenarijih. Napredni RAG in GZ-RAG dosežeta primerljivo uspešnost, ki je približno na enakem nivoju. Največja razlika med njima se je pokazala v drugem scenariju, kjer je referencirano besedilo bilo ključno za odgovor.

V prvih dveh scenarijih sta se tako napredni RAG kot GZ-RAG izkazala za močna kandidata. Uspešnost je bila zadostna, da bi se z nekaj nadaljnimi popravki te metode že lahko začele uporabljati v praksi, z GZ-RAG-om v ospredju. Za bolj splošna vprašanja je prihodnost nekoliko bolj meglena. Rezultati še niso na zadostni ravni, da bi se te metode lahko zanesljivo uporabljale v kritičnih sistemih.

Za nadaljno raziskovanje bi bilo smotrno do-trenirati VJM na slovenski zakonodaji. S tem bi omogočili veliko boljše razumevanje vprašanj z domenskega vidika, poznavanje terminologije in interno znanje zakonov. Tedaj bi

tudi pri uporabi RAG-a VJM lahko bolj natančno ubesedil in izrazil odgovor (strokovnjak se bo izrazil bolj strokovno dosledno kot povprečen človek). Pretvorba poizvedbe iz vsakodnevnega jezika v pravno terminologijo bi bila močno izboljšana na ta način.

Naslednja očitna izboljšava je razširitev korpusa. Vključiti bi bilo treba več pravnih aktov, vzpostaviti povezave med njimi in preveriti kako se metode obnašajo v primeru te večje množice. Nazadnje bi se dalo raziskati, kako se izboljša učinkovitost, če vpeljemo več specializiranih in avtonomnih enot v proces, podobno kot je to bilo videno v projektu ChatLaw.

Kot nakazuje dejstvo, da sta Pravko in ChatLaw javno dostopna in na voljo za uporabo, potreba po pravnih asistenih obstaja. Tako kot vsaka nova tehnologija bo tudi to področje z nadaljnim raziskovanjem postalo naše vsakodnevno orodje, s katerim lahko izboljšamo življenja ljudi.

Literatura

[1] Meta AI. *Meta LLaMA 3*. Blog post. 2024. URL: https://ai.meta.com/blog/meta-llama-3/.

[2] Anthropic. *Claude 3 Model Card (Technical Report)*. Model card PDF. 2024. URL: https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

[3] Jianlv Chen in sod. *BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation*. 2024. arXiv: 2402.03216 [cs.CL].

[4] Jiawei Chen in sod. "Benchmarking Large Language Models in Retrieval-Augmented Generation". V: *ArXiv* (dec. 2023). arXiv:2309.01431 [cs]. URL: http://arxiv.org/abs/2309.01431 (pridobljeno 22. 7. 2024).

[5] Luca Clissa, Mario Lassnig in Lorenzo Rinaldi. "How big is Big Data? A comprehensive survey of data production, storage, and streaming in science and industry". V: *Frontiers in Big Data* 6 (okt. 2023), str. 1271639. ISSN: 2624-909X. DOI: 10.3389/fdata.2023.1271639. URL: https://www.frontiersin.org/articles/10.3389/fdata.2023.1271639/full (pridobljeno 19. 5. 2024).

[6] Chroma Core Contributors. *Chroma Core*. GitHub repository. 2024. URL: https://github.com/chroma-core/chroma.

[7] Jiaxi Cui in sod. "Chatlaw: A Multi-Agent Collaborative Legal Assistant with Knowledge Graph Enhanced Mixture-of-Experts Large Language Model". V: *arXiv* (maj 2024). arXiv:2306.16092 [cs]. URL: http://arxiv.org/abs/2306.16092 (pridobljeno 21. 7. 2024).

[8] Yunfan Gao in sod. "Retrieval-Augmented Generation for Large Language Models: A Survey". V: *ArXiv* abs/2312.10997 (mar. 2023). arXiv:2312.10997 [cs]. URL: http://arxiv.org/abs/2312.10997 (pridobljeno 18. 7. 2024).

[9] Linmei Hu in sod. "A Survey of Knowledge Enhanced Pre-Trained Language Models". V: *IEEE Transactions on Knowledge and Data Engineering* 36.4 (apr. 2024), str. 1413–1430. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2023.3310002. URL: https://ieeexplore.ieee.org/document/10234662/ (pridobljeno 21. 7. 2024).

[10] Ivan Ilin. *Advanced RAG Techniques: an Illustrated Overview*. URL: https://pub.towardsai.net/advanced-rag-techniques-an-illustrated-overview-04d193d8fec6 (pridobljeno 25. 7. 2024).

[11] Patrick Lewis in sod. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". V: *Advances in Neural Information Processing Systems*. Ur. H. Larochelle in sod. Zv. 33. Curran Associates, Inc., 2020, str. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

[12] Chaofan Li in sod. *Making Large Language Models A Better Foundation For Dense Retrieval*. 2023. arXiv: 2312.15503 [cs.CL].

[13] Xinbei Ma in sod. "Query Rewriting for Retrieval-Augmented Large Language Models". V: *arXiv* (okt. 2023). arXiv:2305.14283 [cs]. URL: http://arxiv.org/abs/2305.14283 (pridobljeno 24. 7. 2024).

[14] Varun Magesh in sod. *Hallucination-Free? Assessing the Reliability of Leading AI Legal Research Tools*. arXiv:2405.20362 [cs]. Maj 2024. URL: http://arxiv.org/abs/2405.20362 (pridobljeno 24. 7. 2024).

[15] OpenAI in sod. *GPT-4 Technical Report*. arXiv:2303.08774 [cs]. Mar. 2024. URL: http://arxiv.org/abs/2303.08774 (pridobljeno 21. 7. 2024).

[16] PIRSR. *PISRS API vrsta akta*. URL: https://pisrs.si/swagger/sifrant?naziv=%C5%A0ifrant%20Vrsta%20akta%20RS. (accessed: 10.06.2024).

[17] Nils Reimers in Iryna Gurevych. "Making Monolingual Sentence Embeddings Multilingual using Knowledge Distillation". V: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, nov. 2020. URL: https://arxiv.org/abs/2004.09813.

[18] Nils Reimers in Iryna Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". V: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, nov. 2019. URL: https://arxiv.org/abs/1908.10084.

[19] Priya Shelke in sod. "A Systematic and Comparative Analysis of Semantic Search Algorithms". V: *International Journal on Recent and Innovation Trends in Computing and Communication* 11.11s (okt. 2023), str. 222–229. ISSN: 2321-8169. DOI: 10.17762/ijritcc.v11i11s.8094. URL: https://ijritcc.org/index.php/ijritcc/article/view/8094 (pridobljeno 24. 7. 2024).

[20] Gemma Team. "Gemma". V: (2024). DOI: 10.34740/KAGGLE/M/3301. URL: https://www.kaggle.com/m/3301.

[21] Ashish Vaswani in sod. "Attention is all you need". V: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, str. 6000–6010. ISBN: 9781510860964. (Pridobljeno 24. 7. 2024).

[22] Liang Wang in sod. "Multilingual E5 Text Embeddings: A Technical Report". V: *arXiv preprint arXiv:2402.05672* (2024).

[23] Jim Webber. "A programmatic introduction to Neo4j". V: *Proceedings of the 3rd Annual Conference on Systems, Programming, and Applications: Software for Humanity*. SPLASH '12. Tucson, Arizona, USA: Association for Computing Machinery, 2012, str. 217–218. ISBN: 9781450315630. DOI: 10.1145/2384716.2384777. URL: https://doi.org/10.1145/2384716.2384777.

[24] Zhiheng Xi in sod. "The Rise and Potential of Large Language Model Based Agents: A Survey". V: *ArXiv* (sep. 2023). arXiv:2309.07864 [cs]. URL: http://arxiv.org/abs/2309.07864 (pridobljeno 26. 7. 2024).

[25] Sophia Yang. *Advanced RAG 01: Small-to-Big Retrieval*. URL: https://towardsdatascience.com/advanced-rag-01-small-to-big-retrieval-172181b396d4 (pridobljeno 29. 7. 2024).

[26] Yue Zhang in sod. "Siren's Song in the AI Ocean: A Survey on Hallucination in Large Language Models". V: *ArXiv* abs/2309.01219 (sep. 2023). arXiv:2309.01219 [cs]. URL: http://arxiv.org/abs/2309.01219 (pridobljeno 21. 7. 2024).