VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ZPRACOVÁNÍ A VYHLEDÁVÁNÍ DOKUMENTŮ S VYUŽITÍM VEKTOROVÝCH DATABÁZÍ A JAZYKOVÉHO MODELU

PROCESSING AND RETRIEVAL OF TEXT DOCUMENTS WITH USE OF VECTOR DATABASES AND A LANGUAGE MODEL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ADAM VALÍK**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

163475

Ústav:	Ústav informačních systémů (UIFS)
Student:	**Valík Adam**
Program:	Informační technologie
Název:	**Zpracování a vyhledávání dokumentů s využitím vektorových databází a jazykového modelu**
Kategorie:	Data mining
Akademický rok:	2024/25

Zadání:

1. Seznamte se s problematikou zpracování přirozeného jazyka a vyhledávání textových dokumentů. Seznamte se s vektorovými databázemi.
2. Analyzujte požadavky na systém pro vyhledávání dokumentů, který rozdělí vstupní text na menší části vhodné velikosti, doplní k těmto částem dodatečné informace a kontext a upraví je do vektorové podoby. Dále umožní tvorbu dotazu uživatelem, provede vyhledání v databázi a zpracování odpovědi jazykovým modelem.
3. Navrhněte systém dle požadavků. Návrh konzultujte s vedoucím.
4. Implementujte navržený systém.
5. Otestujte vytvořený systém na vhodném vzorku dat a experimentálně ověřte úspěšnost vyhledávání.
6. Zhodnoťte dosažené výsledky a další možnosti pokračování tohoto projektu.

Literatura:

- Vaidyamath, R.C.: Vector Databases Unleashed: Navigating the Future of Data Analytics. Independent, 2024. ISBN 979-8321023488.
- Buttcher, S.: Information Retrieval: Implementing and Evaluating Search Engines. MIT Press Ltd., 2016. ISBN 978-0262528870.

Při obhajobě semestrální části projektu je požadováno:
Body 1-3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce:	**Bartík Vladimír, Ing., Ph.D.**
Vedoucí ústavu:	Kolář Dušan, doc. Dr. Ing.
Datum zadání:	1.11.2024
Termín pro odevzdání:	14.5.2025
Datum schválení:	22.10.2024

Abstrakt

Abstract

Klíčová slova

Keywords

Citace

VALÍK, Adam. *Zpracování a vyhledávání dokumentů s využitím vektorových databází a jazykového modelu*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Zpracování a vyhledávání dokumentů s využitím vektorových databází a jazykového modelu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Další informace mi poskytli... Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Adam Valík
26. ledna 2025

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant apod.).

Obsah

Kapitola 1

Úvod

Především za poslední 2 roky lze zaznamenat napříč světem vzestup zájmu o umělou inteligenci [6]. Vzniká například mnoho velkých jazykových modelů podporující generativní AI ve formě chatbotů, kteří jsou schopni odpovídat na dotazy na základě dohledatelných dat. Tradiční databázové systémy, jako relační a objektově-relační databáze, jsou efektivní při zpracování strukturovaných dat. Vyhledávání v těchto databázích probíhá nejčastěji konkrétními SQL dotazy. Je to efektivní způsob uchovávání a vyhledávání dat například pro informační systémy. Revoluce AI však přináší práci s daty, která není efektivně nebo vůbec proveditelná těmito databázovými systémy, a proto s ní přichází rozvoj vektorových databází. Ty totiž dokáží pracovat i s nestrukturovanými daty, které jsou reprezentovány pomocí vektorů (matematickou reprezentací n-tice čísel). Svou formou dokáží zachytit význam dat, jako jsou slova nebo celé texty, obrázky a zvuková data. Právě proto patří vektorové databáze k moderním přístupům NLP [12] [11].

Spojením metod získávání informací na základě významu pomocí vektorových databází a předtrénovaného jazykového modelu můžeme vytvořit systém, kterému se říká Retrieval-Augmented Generation (RAG), neboli generování rozšířené o získávání informací. Vývoj vlastního LLM je velmi náročný, a proto tento přístup využívá dostupných jazykových modelů. Obohacení dotazování jazykového modelu staví na vlastních zdrojích informací získaných z vektorové databáze, kde je uložen velký objem dat dokumentů. Z navrácených dat se pak formuje dotaz pro jazykový model společně s dotazem uživatele, který produkuje uživateli odpověď. Tento přístup umožňuje například vyhledávání nad velkým množstvím interních dokumentů firmy. Možnosti využití můžeme nalézt i v akademickém prostředí při vyhledávání v odborné literatuře. Důležité je zde správné předzpracování dat, jelikož typicky zdroje tvoří velké množství dokumentů různých typů souborů. Vektorové databáze mohou vyhledávat texty, které mají podobný význam jako dotaz, místo aby se spoléhaly na přesnou shodu klíčových slov, které v tomto případě nemusí být efektivní. Data je však třeba vhodně rozkouskovat, přidat k nim metadata a pomocí embedding modelu vytvořit vektorovou reprezentaci, která je uložena do vektorové databáze. Cílem kapitoly 2 je tak shrnout fungování vektorových databází pro pochopení následné integrace s jazykovým modelem (systém RAG), které je popsáno v kapitole 3.

Výsledkem této práce je pak vytvoření takového komplexního systému, který nabídne uživatelům získávat informace z velkého množství dokumentů. Je toho dosaženo kombinací moderních přístupů NLP, jakou jsou embedding modely, práce s nestrukturovanými daty, vektorové databáze a kombinovaný přístup k získávání informací pomocí full-textového a

vektorového vyhledávání pro co nejlepší výsledky. Data si uchovávají metadata k filtrování a přímým odkazům na původní zdroje. Dále je využit jazykový model pro navrácení lidsky formulované odpovědi.

Kapitola 2

Vektorové databáze

"Vektorová databáze je typ databáze, která ukládá data jako vysokorozměrné vektory, což jsou matematické reprezentace rysů nebo atributů." [7]. Vektorové databáze jsou relativně moderní technologií, která vznikla především jako řešení na zpracování a vyhledávání nestrukturovaných dat. Již v roce 1998 bylo odhadováno, že více než 85% dat neexistuje ve strukturované formě, a tedy je nelze uložit do relačních databází [3]. Před zhruba deseti lety začaly s vývojem zpracování přirozeného jazyka a hlubokého učení vznikat trénované modely. Tyto modely dokáží převést nestrukturovaná data do vektorové reprezentace, která zachovává jejich význam a vlastnosti. Aby však byla existovala možnost efektivně s těmito vektory pracovat, musela vzniknout nová specializovaná databáze. V tom důsledku začal během posledních let vývoj tvorby vektorových databází, kde uložení velkého množství vektorů je jen prvním krokem. Síla těchto systémů plyne z efektivního vyhledávání na základě sémantické podobnosti, zatímco relační databáze provádějí vyhledávání skrze sloupce a porovnávání hodnot. To je klíčové pro moderní aplikace, jako jsou doporučovací systémy, chatboti nebo vyhledávače obrázků. V následujících podkapitolách je detailněji popsáno, jak vektorové databáze zpracovávají nestrukturovaná data, jak funguje vektorová reprezentace dat a jak se provádí indexace pro efektivní vyhledávání.

2.1 Nestrukturovaná data

Lidé i stroje produkují velké množství dat, které nesplňují předem daný formát nebo schéma. Taková data jsou pak náročná na vyhledávání a vyžadují předzpracování a analýzu. Nestrukturovaná data nezapadají do relačních databázových systémů, jelikož je nelze reprezentovat hodnotami uloženými do sloupců tabulky jako data strukturovaná. V jejich kontextu spadají do datového typu BLOB (Binary Large Object), přes které nelze snadno vyhledávat.

Příklady nestrukturovaných dat:

- **Data vytvořená člověkem**: E-maily, textové dokumenty, poznámky, obrázky, audio nebo video nahrávky.
- **Data vytvořená strojem**: Údaje ze senzorů, data počítačového vidění, webová a aplikační data, logy, data z IoT zařízení [21].

Mezi strukturovanými a nestrukturovanými daty existuje mezikrok, který se nazývá polostrukturovaná data. Vznikají přidáním metadat nebo významových tagů, čímž je dodáno nestrukturovaným datům jistá forma organizace nebo hierarchie. Například k obrázkům,

které jsou binárním popisem vlastností jednotlivým pixelů, lze přidat údaje o autorovi a datu a místa vzniku, což umožňuje snadnější filtrování a organizaci dat.

Nestrukturovaná data jsou nejen ukládána, ale procesem vektorizace je jim přidělena sémantika [18]. Proto pro efektivní uložení a vyhledávání nestrukturovaných dat hrají klíčovou roli systémy pro správu vektorových databází.

2.2 Vektorová reprezentace dat

Vektor je v matematice formálně definován jako prvek vektorového prostoru. Pakliže má tento prostor konečné rozměry, lze hovořit o uspořádané n-tici čísel, kde n udává dimenzi vektoru [**?**]. S vektory lze provádět operace vektorové aritmetiky, kde nejdůležitější roli hrají Euklidovská vzdálenost a kosinová podobnost.

V kontextu vektorových databází jsou vektory tvořeny transformací dat na matematickou reprezentaci pomocí modelů strojového učení, které se nazývají embedding modely [7]. Vektor si tedy lze představit jako soubor čísel, kde každé z nich reprezentuje číselné vyjádření určité vlastnosti dat. Mějme 2D prostor, pakliže jednotlivým složkám dvourozměrného vektoru přidělíme nějaký rys, můžeme ohodnotit např. slova označující ovoce, kde první hodnota bude sladkost a druhá hodnota kyselost (na škále 1-10). Vzdálenost mezi vektory ovoce podobné chuti pak bude malá.

Komplexní data však vyžadují mnohem více popisů atributů, a tak podobně jako nestrukturovaná data, vysocedimenziální vektory nejsou pro lidi čitelné a uchopitelné. Pro práci s vektory není nutné užití specializovaných vektorových databází, avšak jak počet vektorů a jejich dimenze roste, vektorový databázový systém poskytuje efektivní řešení pro uložení a vyhledávání nad těmito komplexními daty [18]. Počet dimenzí se snadno pohybuje od stovek po tisíce [**?**], s tím že s vývojem poroste na desítky tisíc. Toto číslo se odvíjí od architektury použitého embedding modelu.

2.3 Embedding modely

Proces vektorizace lze označit jako "*transformaci dat na vysocedimenzionální vektorovou reprezentaci, která zachycuje smysluplné vztahy a vzorce*"[18]. Cílem embedding modelů je zachytit význam dat v numerické podobě, aby byly použitelné pro výpočetní modely. Výsledné vektory, též nazývané *embeddingy*, umožňují strojům práci s nestrukturovanými daty, jelikož jejich poloha v prostoru odráží vzájemné vztahy s ostatními vektory. Významově podobná vstupní data tak budou blízko sebe. Demonstraci vektorové aritmetiky a vzájemného vztahu mezi daty uloženými ve vektorovém prostoru lze ilustrovat na klasickém příkladu [1]: Mějme vektorový prostor obsahující vektorové reprezentace slov ze slovníku. Mějme vektorovou reprezentaci slova *král*. Odečteme-li od slova *král* slovo *muž* a následně přičteme slovo *žena*, nejbližší vektor tomuto vypočtenému vektoru bude ten, který reprezentuje slovo *královna*.

Embedding modely jsou jedním z produktů strojového učení. Obecně vznikají trénováním na rozsáhlých sadách nestrukturovaných dat, např. textových korpusech, obrázcích

nebo zvukových souborech. Tato práce se specificky zaměřuje na zpracování textových dokumentů, kde embedding modely patří k významnému nástroji zpracování přirozeného jazyka.

Word Embeddings

Prvním průlomem ve vývoji embedding modelů byly *word embeddings*, které vektorizují jednotlivá slova. Pro proces konverze se používají nástroje jako Word2vec, který využívá neuronové sítě k naučení asociace slova z velkého textového korpusu. Byly navrženy 2 modely Word2vec:

- CBOW (Continuous Bag-Of-Words): Zaměřuje se na předpovídání aktuálního slova na základě okolního kontextu.
- Skip-gram: Na rozdíl od CBOW se zaměřuje na předpovídání kontextu pro dané slovo [4].

Dalším významnou architekturou je GloVe [15].
Cíl reprezentovat každé slovo bodem ve vektorovém prostoru má hlavní nedostatek, jelikož ignoruje možnosti slova, které má v různých kontextech různý význam. Problémem word embeddings tak je, že vícevýznamová slova jsou přeložena v každém svém významu ve větě na identickou vektorovou reprezentaci. Zachycení správné sémantiky vícevýznamových slov však hraje klíčovou roli v porozumění jazyku NLP systémů. Může se tak stát, že slova *krysa* a *počítač* budou blízko sebe, ačkoliv spolu sémanticky nesouvisí, jelikož se budou obě blížit slovu *myš* [4].

Kontextové modely

Moderní kontextové embedding modely, jako jsou BERT nebo GPT, patří mezi velké jazykové modely (LLM). Mimo prediktivní generování textu dokáží tyto LLM tvořit vektorové reprezentace textu. Umí zachytit dynamiku vícevýznamových slov tak, že pro různé významy slov generují různé vektorové reprezentace. Zohledňují tak kontext konkrétní věty na základě okolních slov.

GPT (Generative Pre-Trained Transformer)

Příkladem kontextového modelu je GPT, který byl primárně navržen pro generování textu. Predikuje další slovo ve větě na základě předchozích slov a kontext slova tak dokáže zachytit tím, že čte větu jednosměrně zleva doprava. Mimo generování textu však dokáže díky porozumění kontextu vytvářet vektorové repezentace textu [19].

BERT (Bidirectional Encoder Representations from Transformers)

BERT je předtrénovaný model (převážně na celé anglické Wikipedii), který používá transformery k zachycení kontextu slova ve větě, kterou navíc čte oběma směry, zleva doprava i zprava doleva. Tím dokáže lépe zachytit kontext daného slova a vytvořit pro něj vektorovou reprezentaci [5].

S-BERT (Sentence-BERT)

S-BERT je rozšíření modelu BERT. Oproti předchozím modelům je S-BERT efektivnějším nástrojem pro zpracování textových dokumentů, jelikož optimalizuje tvorbu vektorové

reprezentace vět (*sentence embeddings*). Přidává speciální tréninkovou fázi, aby mohl efektivně vytvářet vektory z celých kusů textu [16]. Toho lze využít ve vektorových databázích při uložení textových dokumentů a následném vyhledávání pomocí dotazu vektorizovaném tímto modelem.

2.4 Systém správy vektorových databází

Systém správy vektorových databází je technologie, která práci s vektorovými databázemi seskupuje dohromady. Je to specializovaný typ systému správy databází zaměřující se především na efektivní správu vysokorozměrných vektorových dat. Toto nezahrnuje nízkorozměrná vektorová data (jako například 2D souřadnice) s nimiž práce je podstatně jednodušší a nevyžaduje dále uvedené optimalizační techniky. VDBMS (z anglického Vector Database Management System) většinou podporuje vyhledávání na základě podobnosti skrze indexování, které je nevyhnutelné pro rychlé vyhledávání ve vektorovém prostoru [18].

Termín vektorová databáze se často používá jako synonymum pro VDBMS. Sama vektorová databáze je však jen soubor dat, zatímco systém správy vektorových databází je celý software. Ten se stará nejen o uložení dat, ale i efektivní navracení relevantních informací a často si ukládá k samotnému vektoru i nějaký identifikátor nebo metadata, která mohou sloužit k filtrování nebo k poskytování personalizovaných doporučení [18].

2.5 Indexace a vyhledávání

Po vektorizaci a uložení dat je třeba nad nimi vytvořit index. Indexace slouží k organizaci dat pro rychlejší vyhledávání. Vektorové indexy zrychlují proces vyhledávání minimalizací porovnávání vektorů rozdělením vektorového prostoru na datové struktury, které lze snadno procházet. Porovnává se tak pouze malá podmnožina prostoru k nalezení nejbližších sousedů [13].

Výběr algoritmu k indexaci záleží na požadavcích. Obecně bývá výpočetně náročný, v ideálním případě ho však není třeba při rozšíření datasetu přepočítávat a umožňuje velmi rychlé vyhledávání. Právě proto se bez něho VDBMS neobejdou [18].

Dotazování pak předchází vyhledávání. Dotazy jsou podávány systému často v přirozeném jazyce, a proto potřebují být taktéž vektorizovány a to stejným modelem jako vektory uložené v databázi. Dotazy si obvykle uchovávají metadata k vyhledávání jako například počet nejbližších vektorů k nalezení. [18]. Vektor dotazu je tak "zanesen"mezi vektory databáze a probíhá vyhledávání. Jak již bylo zmíněno, vyhledávání ve vektorových databázích probíhá na základě významové podobnosti, jinak řečeno vzájemné vzdálenosti mezi vektory, jelikož sémanticky blízké vektory jsou i výpočetně blízko sebe. Tomuto principu se říká vyhledání nejbližších sousedů (anglicky *Nearest Neighbor Seach*, NNS). Je to *"optimalizační problém nalezení bodu v dané množině, který je nejblíže (= nejpodobnější) danému bodu"* [7]. Motivace k zavedení optimalizace indexy a aproximace výsledků je znázorněna na následující metodě.

Naivní metoda

Nejjednodušším přístupem k nalezení nejbližšího vektoru je naivní metoda (v anglické literatuře se též používá pojem *brute-force approach* [7], neboli přistup hrubou silou). Naivní algoritmus počítá vzdálenost mezi každým bodem vektorového prostoru a vektorem dotazu. Tento přístup má absolutní úspěšnost pro navrácení požadovaného počtu opravdových nejbližších bodů, avšak je velice neefektivní. Je-li použita pro výpočet vzdálenosti např. Euklidovská vzdálenost, která se mezi dvěma body p a q v n-rozměrném prostoru vypočítá jako odmocnina součtu čtverců rozdílů odpovídajících souřadnic,

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} [10] \tag{2.1}$$

bude časová náročnost naivního algoritmu *O(NM)*, kde N je počet vektorů v datasetu a M je dimenze vektoru (velikost embeddingu). Při menším počtu nízkodimenzionálních vektorů je tato intuitivní metoda efektivní díky své přesnosti. U vektorových databází však počet dimenzí vektorů roste exponenciálně a dochází k poklesu efektivity mnoha algoritmů strojového učení, což popisuje pojem prokletí dimnezionality (z anglického *curse of dimensionality*) [14]. Navíc počet vektorů uložených v databází velkých projektů může dosahovat několika miliard, a proto VDBMS zavádějí aproximaci a indexační metody, které budou popsány v následujících podkapitolách.

Přibližné vyhledávání nejbližšího souseda

Zatímco metody NNS prohledávají vektorový prostor vyčerpávajícím způsobem porovnáváním vektoru se všemi ostatními, metody ANNS (z anglického *Approximate Nearest Neighbor Search*) řeší problém jejich neefektivity na velkém množství vysokodimenziálních vektorů. Na úkor přesnosti vyhledávají nejbližšího souseda pouze přibližně s určitou přesností, poskytují však velmi výrazný nárůst výkonu umožňující provádění vyhledávání v řádech tisíců na sekundu. Je třeba u nich optimalizovat volbu parametrů dle specifikací databáze a vyvážit tak poměr mezi výkonem a přesností. Při hledání k nejbližších sousedů se metody označují k-ANNS, pro jednoduchost bude toto označení vynecháno (jde o nastavení parametru, princip však zůstává stejný). Výběrem nejpoužívanějších implementací ANNS jsou následující metody a jejich algoritmy.

Hashovací metody

Při zpracování vysocedimenzionálních dat staví tyto techniky na transformaci vektorů pomocí hashovacích funkcí na kompaktnější reprezentaci ve formě hashovacích klíčů. Redukují složitost porovnávání dat a urychlují tak vyhledávání i díky specifické sadě hashovacích funkcí, které zajišťují, že není třeba prohledávat celou databází. Příkladem je následující metoda, která při hashování zachovává podobnost mezi daty.

Locality Sensitive Hashing (LSH, v překladu "hashování citlivé na lokalitu") je algoritmus urychlující přibližné vyhledávání sousedů v rozsáhlých databázích. Jeho principem je použití sady hashovacích funkcí, které jsou citlivé na lokalitu a zachovávají tak informace o podobnosti mezi vektory. To znamená, že blízké vektory (podle zvolené metriky) hashuje do stejného "hashovacího koše"s vyšší pravděpodobností než vektory vzdálené. LSH pro každý vektor aplikuje sadu hashovacích funkcí, které přídělí vektorům hashovací klíče odpovídající

jejich lokalitě. Dle těchto klíčů lze určit, do kterého hashovacího koše daný vektor spadá. Při vyhledávání je vektor dotazu zpracován stejným principem, takže lze snadno nalézt kandidáty na nejbližší sousedy ve stejném hashovacím koši. Jelikož není potřeba prohledat celá databáze, dochází k výraznému urychlení vyhledávání [17] [20].

Stromové metody

Smyslem stromových metod je zmenšit prohledávaný prostor následováním větví stromu, které s největší pravděpodobností obsahují nejbližší sousedy vektorizovaného dotazu [7].

Approximate Nearest Neighbors Oh Yeah (ANNOY) je algoritmus vyvinutý společností Spotify, původně pro doporučování hudby. Pracuje nejefektivněji ve střednědimenzionálních vektorového prostorech (nižší stovky domenzí), zatímco při velmi vysoké dimenzionalitě může efektivita klesat. ANNOY funguje na principu náhodných projekcí a využívá stromové struktury k rozdělení datového prostoru. V každém uzlu stromu je prostor rozdělen na podprostory pomocí náhodně zvolené hyperroviny[1]. Hyperrovina je zvolena na základě dvou náhodně vybraných bodů, mezi kterými prochází středová rovina, která je od obou stejně vzdálená. Tento proces pokračuje rekurzivně, dokud podprostor neobsahuje dostatečně málo bodů (v závislosti na parametru). Výsledkem je les binárních stromů, kde každý vektor je přiřazen k listovému uzlu na základě své polohy vůči hyperrovinám.

Při vyhledávání nejbližších sousedů prochází ANNOY každý strom od kořene k listovému uzlu, kam připadá vektor dotazu a shromažďuje všechny vektory ve stejných listových uzlech. Poté vypočítá přesnou vzdálenost mezi dotazovaným vektorem a těmito kandidáty a vrátí nejbližší sousedy. Počet stromů a hloubka stromu lze optimalizovat ke kompromisu rychlosti a přesnosti [7] [2].

Grafové metody

Grafové metody poskytují řešení ukládání a vyhledávání vektorů pomocí grafových struktur. Při vytváření indexu přidávají bod po bodu, přičemž je podle algoritmu spojují do grafu pro následný jednoduchý průchod grafem k přibližnému nalezení nejbližších sousedů.

Navigable small world (NSW, v překladu "Prohledávatelný malý svět") je algoritmus tvořící graf spojováním vektorů s jeho nejbližšími sousedy. NSW graf je zkonstruován přidáváním vektorů do prostoru v náhodném pořadí. Každý přidaný vektor spojí hranou s určitým počtem nejbližších vektorů (heuristikou nejlepší možnosti je vypočtená vzdálenost mezi vektory), přičemž prohledávání začíná na několika náhodných vstupních bodech a funguje hladově, tedy může skončit v lokálním minimu. Výsledný graf se pak blíží konceptu "malého světa"[2], jelikož body přidány v prvotní fázi vytvoří vzdálená spojení mezi výslednými shluky. Výsledkem je až logaritmické prohledávání grafu k získání přibližných nejbližších sousedů [9].

Hierarchical navigable small world (HNSW, v překladu "Hierarchický prohledávatelný malý svět") je vylepšením algoritmu NSW o hierarchickou strukturu a představuje

[1] rovina, která dělí více dimenzionální prostor na dvě části

[2] fenomén malého světa a šest stupňů odloučení ve společnosti je myšlenka, že jsou všichni lidé v průměru 6 sociálních konktaktů od sebe

jednu z nejvýkonnějších metod ANNS. Vytváří vícevrstvý graf. Body jsou přidávány od nejvyšší vrstvy, kde je jich nejméně. Jakmile nalezne lokální minimum, pokračuje na nižší úrovni s daným bodem jako výchozím. Proces se opakuje, s tím že každá vyšší úroveň obsahuje řádově menší počet uzlů, než vrstva nižší. To umožňuje přesvědčivě dosáhnout logaritmické náročnosti, jelikož vyhledávání zprvu udělá "velké kroky"a jakmile se dostane na nižší úrovně, blíží se relevantnějším uzlům představující nejbližší sousedy. V realitě pak nejvyšší vrstvu může představovat stovky uzlů z celkového počtu milionů [9].

Princip vyhledávání by se pak dal ilustrovat na analogii hledání bydlení. Nezačali bychom prohledáváním všech nabídek jedné po druhém. V kontextu HNSW bychom se podívali na jedno bydlení z různých států. To by byla nejvyšší vrstva hierarchie. Po zvolení nejvhodnějšího je pak třeba vybrat jedno bydlení z různých měst v daném státě, kde jsou typicky koncentrovány. Po výběru vyhovující globálnější lokace bychom až přešli o úroveň níže, k výběru bydlení dle části města, než bychom se propracovali k výběru konkrétní budovy. Na nejvyšší úrovni si tedy vektory spolu nejsou vůbec podobné a chovají se jako dálnice spojující lokality, která jsou daleko od sebe. (Tento příklad byl převzat z podcastu Semantic Search, Developer Voices [**?**])

Kvantizační metody

Kvantizační metody jsou techniky používané ke snížení výpočetní složitosti při práci s vysoce dimenzionálními daty. Kvantizace signálů převádí spojité hodnoty na diskrétní, čímž umožňuje efektivnější ukládání. Kvantizace vektorů pak aproximuje jejich hodnoty na reprezentativní vzorky, čímž dochází k výrazné kompresi dat. Zároveň z této aproximace vzniká kvantizační chyba, která je však přijatelným kompromisem přinášejícím nárůst výkonu a snížení paměťových nároků. Následující metoda je jedna z nejvýznamějších kvantizačních metod v oblasti ANNS [8].

Product Quantization (PQ, v překladu "Kvantizace produktu") je další z technik efektivního vyhledávání nejbližších sousedů vysokodimenzionálních prostorů. Hlavní myšlenkou je rozdělit vektorový prostor na menší podporostory a každý z nich kvantizovat samostatně. Vektory se tedy rovnoměrně rozdělí na daný počet částí (podprostory). Pro každý podprostor jsou pomocí shlukovacího algoritmu (např. *k-means*) určeny centroidy a zapsány do tzv. kódovací knihy. Vektor, který je následně přidáván do indexu, je rozdělen na podprostory a každý tento podprostor je přiřazen nejbližšímu centroidu. Výsledný záznam v databázi se skládá z identifikátorů centroidů v kódovací knize každého z podprostorů [8].

Při vyhledávání je vektor dotazu rozdělen na podprostory a kvantizován stejným způsobem jako vektory uložené v databázi. Na základě identifikátorů centroidů z kódovacích knih jsou vybráni kandidáti, jejichž indexy odpovídají nebo se nejvíce podobají dotazu. Pro tyto kandidáty lze původní vektory rekonstruovat a následně vypočítat přesné vzdálenosti k dotazu, což umožňuje nalezení nejbližších sousedů [8].

2.6 Využití vektorových databází

Kapitola 3

RAG: Retrieval-Augmented Generation

Kapitola 4

Architektura systému

4.1 Zpracování dokumentů

Převod dokumentů na plain text

Chunking a uchování metadat

Embedding chunků

4.2 Vyhledávání dokumentů

Embedding dotazu

Kombinované vyhledávání

Reranking

Prompt

LLM call a navracení odpovědi

Kapitola 5

Implementace

Kapitola 6

Testování

Kapitola 7

Závěr

Literatura

[1] Alammar, J. *The Illustrated Word2vec*. 2018. Dostupné z: https://jalammar.github.io/illustrated-word2vec/.

[2] Bernhardsson, E. *Annoy: Approximate Nearest Neighbors Oh Yeah* https://github.com/spotify/annoy. GitHub, 2015.

[3] Blumberg, R. a Atre, S. The Problem with Unstructured Data. *DM Review*, 2003, sv. 13, č. 2, s. 42–49. Dostupné z: https://www.soquelgroup.com/wp-content/uploads/2010/01/dmreview_0203_problem.pdf.

[4] Camacho-Collados, J. a Pilehvar, M. T. From Word to Sense Embeddings: A Survey on Vector Representations of Meaning. *CoRR*, 2018, abs/1805.04032. Dostupné z: http://arxiv.org/abs/1805.04032.

[5] Devlin, J.; Chang, M.; Lee, K. a Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, 2018, abs/1810.04805. Dostupné z: http://arxiv.org/abs/1810.04805.

[6] Google. *Google Trends: ai vyhledávání*. Dostupné z: https://trends.google.com/trends/explore?date=all&q=ai.

[7] Han, Y.; Liu, C. a Wang, P. *A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge*. 2023. Dostupné z: https://arxiv.org/abs/2310.11703.

[8] Jégou, H.; Douze, M. a Schmid, C. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Institute of Electrical and Electronics Engineers, 2011, sv. 33, č. 1, s. 117–128. Dostupné z: https://hal.inria.fr/inria-00514462v2.

[9] Malkov, Y. A. a Yashunin, D. A. Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs. *CoRR*, 2016, abs/1603.09320. Dostupné z: http://arxiv.org/abs/1603.09320.

[10] MathWorld Contributors. *Distance*. N.d. Dostupné z: https://mathworld.wolfram.com/Distance.html. Accessed: 2025-01-26.

[11] Michal Rost. *Co jsou vektorové databáze a proč jsou vhodné pro AI?* Dostupné z: https://www.michalrost.cz/co-jsou-vektorove-databaze-a-proc-jsou-vhodne-pro-ai.

[12] Musadiq Peerzada. *Vectors Unleashed: Navigating the Future with Vector Databases*. Dostupné z: https://blogs.musadiqpeerzada.com/vectors-unleashed-navigating-the-future-with-vector-databases.

[13] Pan, J. J.; Wang, J. a Li, G. *Survey of Vector Database Management Systems*. 2023. Dostupné z: https://arxiv.org/abs/2310.14021.

[14] Peng, D.; Gui, Z. a Wu, H. *Interpreting the Curse of Dimensionality from Distance Concentration and Manifold Effect*. 2024. Dostupné z: https://arxiv.org/abs/2401.00422.

[15] Pennington, J.; Socher, R. a Manning, C. Glove: Global Vectors for Word Representation. In:. Leden 2014, sv. 14, s. 1532–1543.

[16] Reimers, N. a Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *CoRR*, 2019, abs/1908.10084. Dostupné z: http://arxiv.org/abs/1908.10084.

[17] Shinde, R.; Goel, A.; Gupta, P. a Dutta, D. Similarity Search and Locality Sensitive Hashing using TCAMs. *CoRR*, 2010, abs/1006.3514. Dostupné z: http://arxiv.org/abs/1006.3514.

[18] Taipalus, T. Vector database management systems: Fundamental concepts, use-cases, and current challenges. *Cognitive Systems Research*. Elsevier BV, červen 2024, sv. 85, s. 101216. ISSN 1389-0417. Dostupné z: http://dx.doi.org/10.1016/j.cogsys.2024.101216.

[19] Yenduri, G.; M, R.; G, C. S.; Y, S.; Srivastava, G. et al. *Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions*. 2023. Dostupné z: https://arxiv.org/abs/2305.10435.

[20] Zhang, B.; Liu, X. a Lang, B. Fast Graph Similarity Search via Locality Sensitive Hashing. In: Ho, Y.-M.; Sang, J.; Ro, Y. M.; Kim, C.-S. a Wu, F., ed. *Advances in Multimedia Information Processing – PCM 2015*. Springer, Cham, 2015, sv. 9314, s. 352–361. Lecture Notes in Computer Science. Dostupné z: https://doi.org/10.1007/978-3-319-24075-6_60.

[21] Zilliz. *Introduction to Unstructured Data*. 2022. Dostupné z: https://zilliz.com/learn/introduction-to-unstructured-data. Accessed: 2025-01-26.