

Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky



No-code webscraping pomocí platformy Make.com a jeho využití s RAG databází.

BAKALÁŘSKÁ PRÁCE

Studijní program: Aplikovaná informatika

Autor: Dinh Hoang Nguyen

Vedoucí bakalářské práce: Ing. Jiří Korčák

Praha, Květen 2025

Poděkování

Chtěl bych vyjádřit svoji vděčnost vedoucímu této bakalářské práce, Ing. Jiří Korčák, za zpětné vazby, cenné rady a odborné vedení, které mi pomohlo dokončit tuto práci.

Dále děkuji své rodině a přátelům za jejich podporu a trpělivost během studia. Bez kterých by studium bylo nepřekonatelnou překážkou.

Abstrakt

Tato bakalářská práce se zabývá využitím automatizační platformy Make.com pro webscraping, uložením získaných dat do RAG databáze a následnou integrací s chatbotem využívající velké jazykové modely (LLM). Cílem práce je ověřit efektivitu tohoto no-code přístupu a změřit kvalitu generovaných odpovědí s využitím RAG databáze i bez ní.

V teoretické části jsou popsány principy webscrapingu, automatizačních platforem a modelů RAG, včetně možností uložení dat pomocí vektorových databází jako Pinecone. Dále je rozebráno fungování velkých jazykových modelů a metrik pro měření relevance odpovědí, jako jsou BLEU, METEOR, RAGAS a BERTScore.

Praktická část se zaměřuje na návrh a implementaci systému, který automatizuje získávání herních dat z vybraných zdrojů (např. Steam API, IGDB) pomocí Make.com. Tato data jsou následně uložena do RAG databáze a využita chatbotem k odpovědím na dotazy uživatelů. Výsledky experimentu jsou analyzovány na základě testovacích dotazů, přičemž je porovnávána kvalita odpovědí generovaných s RAG a bez něj.

Výsledky práce poskytují ucelený pohled na možnosti využití no-code přístupu k webscrapingu a RAG databázím. Ukazují, do jaké míry tento přístup zlepšuje relevanci generovaných odpovědí a jaké jsou jeho limity při integraci s LLM.

Klíčová slova

webscraping, Make.com, RAG databáze, chatbot, velké jazykové modely, vektorové databáze, herní data

Abstract

This bachelor's thesis focuses on utilizing the Make.com automation platform for web scraping, storing extracted data in a RAG database, and integrating it with a chatbot powered by large language models (LLM). The goal is to evaluate the effectiveness of this no-code approach and measure the quality of generated responses using a RAG database compared to responses without it.

The theoretical part explores the principles of web scraping, automation platforms, and RAG models, including data storage options using vector databases like Pinecone. It also examines the workings of large language models and metrics for evaluating response relevance, such as BLEU, METEOR, RAGAS, and BERTScore.

The practical part focuses on designing and implementing a system that automates the retrieval of gaming data from selected sources (e.g., Steam API, IGDB) using Make.com. The collected data is stored in a RAG database and used by a chatbot to respond to user queries. The results of the experiment are analyzed based on test queries, comparing the quality of responses generated with and without RAG.

The findings of this work provide a comprehensive insight into the potential of a no-code approach to web scraping and RAG databases. They demonstrate the extent to which this approach improves response relevance and its limitations when integrated with LLM.

Keywords

web scraping, Make.com, RAG database, chatbot, large language models, vector databases, gaming data

Obsah

Úvod.....	7
1 Webscraping a automatizační platformy	9
1.1 Obecné principy a metody webscrapingu.....	9
1.1.1 Metody webscrapingu	9
1.2 Přehled automatizačních platforem (Make.com a alternativy).....	11
1.2.1 Make.com (dříve Integromat)	11
1.2.2 Zapier.....	12
1.2.3 n8n.....	12
1.2.4 Další nástroje a platformy.....	12
1.2.5 Srovnání a volba platformy	13
1.3 Výhody a omezení no-code přístupu	13
1.3.1 Výhody no-code přístupu	13
1.3.2 Omezení no-code přístupu.....	14
2 RAG databáze a velké jazykové modely (LLM)	16
2.1 Co je Retrieval-Augmented Generation (RAG)	16
2.2 Možnosti uložení dat: tradiční vs. vektorové databáze.....	18
2.2.1 Tradiční databáze vs. vektorové databáze	18
2.3 Principy fungování velkých jazykových modelů (LLM)	21
2.3.1 Architektura transformerů jako základ LLM.....	21
2.3.2 Zero-shot a few-shot prompting	21
2.3.3 GPT-4 vs. open-source LLM (LLaMA, Mistral) a práce s externím kontextem .	22
2.4 Návrh struktury vektorové databáze pro projekt.....	23
3 Návrh a implementace systému	26
3.1 Architektura systému	26
3.2 Možnosti scrapování dat.....	27
3.3 Zvolené postupy pro získání dat v práci	27
3.4 Výzvy spojené s extrakcí a strukturováním dat.....	28
3.4.1 Zpracování a uložení dat pro potřeby chatbota	30
3.5 Možnosti propojení Make.com s RAG databází a chatbotem	32
3.5.1 Konkrétní řešení systému	33
4 Měření relevance odpovědí	35
4.1.1 Testovací sada dotazů	35
4.1.2 Struktura dotazů a kategorizace (obecné vs. klíčová slova).....	35

4.1.3 Metriky pro hodnocení fáze vyhledávání (retrieval)	37
4.1.4 Vyhodnocení retrieval fáze pomocí metrik.....	39
4.2 Metriky pro hodnocení fáze generování odpovědi	40
4.3 Metodologie evaluace v projektu	42
4.3.1 Využití LLM-as-a-Judge pro hodnocení relevance a správnosti.....	44
4.4 Srovnání kvality odpovědí s RAG a bez něj.....	45
5 Diskuse a limity	48
Závěr	50
Použitá literatura.....	51
Přílohy.....	I
Příloha A: První sada dotazů	I
Příloha B: Druhá sada dotazů.....	I
Příloha C: Evaluace retrievalu	I
Příloha D: JSON soubor s plnou evaluací.....	I
Příloha E: Připravené textové soubory pro každý záznam hry	I

Úvod

V současné době došlo k rychlému rozvoji umělé inteligence, zejména velkých jazykových modelů (LLM), které zásadně mění způsoby, jakými pracujeme s informacemi. Tyto modely nacházejí uplatnění ve vyhledávačích, asistentech i chatbotech, které generují lidsky srozumitelné odpovědi na dotazy uživatelů. Ale aby jejich výstupy byly skutečně relevantní a aktuální, je potřeba jim poskytovat aktuální kontext, často ve formě externích dat, která nejsou součástí jejich tréninkové paměti. Právě zde se uplatňuje koncept Retrieval-Augmented Generation (RAG), který kombinuje výhody klasického vyhledávání s generativními schopnostmi LLM.

Současně roste význam přístupů, které zpřístupňují práci s daty i netechnickým uživatelům. No-code platformy, jako je Make.com, umožňují automatizaci úloh bez potřeby programování. Tyto nástroje představují svět datové integrace, sběru a zpracování širšímu okruhu lidí.

Tato práce se zaměřuje na propojení těchto dvou trendů. Zkoumá, jak lze pomocí no-code nástrojů realizovat webscraping, a následně využít takto získaná data pro zlepšení odpovědí jazykového modelu v rámci systému RAG. Zaměření na herní doménu přitom poskytuje dobře vymezený, ale zároveň datově bohatý prostor pro experimentální ověření přínosu tohoto přístupu.

Metody

V této práci byly využity následující metody:

- **Literární rešerše:** Pro vytvoření teoretického základu práce byla provedena literární rešerše zaměřená na klíčová témata, jako jsou webscraping, no-code automatizační platformy (zejména Make.com), vektorové databáze pro RAG, velké jazykové modely (LLM) a metriky hodnocení relevance
- **Komparace:** Metoda komparace sloužila k posouzení přínosu integrace RAG databáze do systému chatbota. Porovnány byly výstupy generované chatbotem ve dvou režimech. S využitím externí znalostní databáze (RAG) a bez ní (tj. čistě generativní přístup modelu bez dodatečného kontextu).
- **Pozorování:** Pro zjištění funkčnosti systému byla v práci pozorována relevantnost výstupů.
- **Analýza a interpretace výsledků:** Výsledky byly kvantitativně shrnuty do tabulek a interpretovány z hlediska rozdílů mezi přístupy, typů dotazů a limitací řešení.

Motivace a cíle

Motivací pro zpracování tohoto tématu je snaha ukázat, že i bez hlubokých znalostí programování lze vytvořit pokročilý systém kombinující webscraping, ukládání znalostí a jazykové modely. Využití no-code platformy jako Make.com výrazně snižuje vstupní bariéru pro vývoj funkčních datových pipeline a umožňuje tak rychlé prototypování řešení, které lze dále integrovat s moderními AI technologiemi.

Zároveň je cílem ověřit, do jaké míry může být takto navržený systém efektivní při odpovídání na dotazy uživatelů. Vzhledem k tomu, že jazykové modely samy o sobě často halucinují nebo nejsou schopné pracovat s aktuálními a specifickými daty, je přínos RAG přístupu klíčovou výzkumnou otázkou.

Hlavním cílem této bakalářské práce je navrhnout, implementovat a experimentálně ověřit efektivnost no-code řešení pro automatizovaný webscraping herních dat a integraci těchto dat do chatbota využívající velký jazykový model. K dosažení tohoto cíle byly stanoveny následující dílčí cíle:

- **Analyzovat** možnosti webscrapingu pomocí no-code nástrojů, se zaměřením na platformu Make.com.
- **Navrhnout a implementovat** datovou pipeline, která zahrnuje scraping, převod textu na embeddingy a uložení do RAG databáze (např. Pinecone).
- **Propojit** tuto databázi s jazykovým modelem a vytvořit chatbota schopný využít získaný kontext.
- **Změřit** kvalitu generovaných odpovědí pomocí zvolených metrik jako je BERTScore, Precision@K a porovnat výstupy s RAG a bez RAG.
- **Vyhodnotit přínosy a limity** zvoleného přístupu z hlediska přístupnosti pro netechnické uživatele.

Přínos práce spočívá v poskytnutí pohledu na možnosti využití no-code nástrojů v kontextu moderních AI systémů. Práce má ambici být praktickým průvodcem pro implementaci podobných řešení i pro uživatele bez pokročilých programátorských znalostí, což usnadňuje přístup k AI technologiím.

1 Webscraping a automatizační platformy

1.1 Obecné principy a metody webscrapingu

Web scraping (též web harvesting nebo web data extraction) označuje proces získávání dat z webových stránek a jejich převod do vhodného formátu pro další analýzu (Yashaswini, 2024). Tato technika umožňuje systematicky sbírat informace z různých online zdrojů většinou bez nutnosti ručního kopírování. V praxi existuje několik přístupů a metod web scrapingu, od základního parsování statického HTML kódu po pokročilé využití API rozhraní či simulace celého prohlížeče. V následující části jsou rozebrány hlavní principy a metody.

1.1.1 Metody webscrapingu

Existuje několik technických přístupů k webscrapingu, přičemž volba konkrétní metody závisí na dostupnosti dat a složitosti webové stránky:

- **Parsování HTML** je nejběžnější formou scrapingu, která spočívá v načtení HTML kódu stránky a extrakci požadovaných dat z tohoto kódu. Každá webová stránka je strukturována pomocí značek (tagů) jazyka HTML, které definují jednotlivé prvky obsahu (texty, odkazy, obrázky, tabulky atd.). Pomocí HTML parseru lze procházet tuto strukturu a vyhledat v ní konkrétní informace na základě tagů či atributů. Typicky se využívají knihovny jako BeautifulSoup nebo lxml (v jazyce Python) umožňující prohledávání HTML dokumentu a získání cílových dat (Zanini, 2025). Parsování HTML je vhodné pro statické webové stránky, které nevyžadují interakci uživatele. Např.: extrakce produktových informací z e-shopu, kde je obsah již přímo součástí HTML kódu stránky (Yashaswini, 2024).
- **DOM scraping** a dynamický obsah: Mnoho moderních webových stránek načítá část obsahu dynamicky pomocí JavaScriptu (např. nekonečné scrollování, asynchronní načítání dat) a samotný HTML kód neobsahuje všechny údaje, které se uživateli zobrazí v prohlížeči. V takových případech samotné parsování HTML nestačí. DOM scraping označuje techniku, kdy se simuluje skutečný webový prohlížeč, aby bylo možné pracovat s kompletním Document Object Model (DOM) stránky po vykonání JavaScriptu (Yashaswini, 2024). Prakticky toho lze dosáhnout pomocí nástrojů pro automatizaci prohlížeče, jako jsou Selenium, Puppeteer či jiné headless prohlížeče (Zanini, 2025), které načtou stránku obdobně jako běžný prohlížeč a umožní skriptem interagovat s DOM (vyhledávat elementy, kliknout na tlačítka, scrollovat apod.) (Yashaswini, 2024). Tento přístup umožňuje získat i dynamicky generovaný obsah, například aktualizace produktových nabídek na e-shopu, které se načítají po posunu stránkou nebo po filtrování uživatelem. Výhodou je možnost napodobit chování uživatele a tím získat kompletní data,

nevýhodou jsou vyšší nároky na výkon a složitost, kdy je nutné spustit skriptovaný prohlížeč a čekat na načtení stránky.

- **Headless prohlížeče:** Specifickou kategorií nástrojů pro dynamický scraping jsou tzv. headless prohlížeče, tedy webové prohlížeče bez grafického uživatelského rozhraní. Headless Chrome, PhantomJS aj. umožňují načíst stránku na pozadí, vykonat potřebný JavaScript a poskytnout skriptu přístup k DOM (Zanini, 2025), aniž by bylo cokoli vidět na obrazovce. Web scraper tak může programově ovládat prohlížeč, procházet stránky, vyplňovat formuláře či stránku posouvat, a následně extrahovat data obdobně jako při klasickém parsování HTML. Headless prohlížeče jsou užitečné pro scraping moderních webových aplikací a jednotnost s reálným prostředím. Scraper získá stránku přesně v takové podobě, jak ji vidí uživatel v prohlížeči. Nevýhodou jsou opět větší nároky na prostředky a rychlost (spouštění celého prohlížeče pro každou stránku je pomalejší než stažení čistého HTML). Pro mnoho scénářů jsou však headless nástroje nezbytné, zejména pokud web vyžaduje přihlášení, kliknutí nebo překonání jiných interakcí pro zobrazení dat (Yashaswini, 2024).
- **Scraping přes API:** Některé webové služby samy nabízejí oficiální API (Application Programming Interface) pro přístup k datům. V těchto případech není nutné „scrapovat“ HTML a místo toho lze poslat požadavek na API rozhraní a získat data v strukturované podobě, typicky JSON nebo XML (Tal, 2025). API scraping tedy spočívá ve volání dokumentovaných webových API dané služby (Yashaswini, 2024). Výhodou je, že data získaná přes API jsou často přímo ve využitelném formátu a API poskytuje standardizovaný a stabilní přístup k informacím. Například řada platforem elektronického obchodu, sociálních sítí či databází nabízí vývojářské API, přes které lze získat potřebné údaje (za podmínek daných poskytovatelem). Příkladem relevantním pro tuto práci je databáze videoher IGDB (Internet Games Database) (*An Evaluation of the Impact of Twitch's Content Classification Labelling*, 2024), která nabízí veřejné API pro vyhledávání informací o hrách. Scrapování webových stránek IGDB lze tedy využít přímo toto API, které vrátí detailní informace o hře ve formátu JSON. Scraping přes API je efektivnější a spolehlivější, ovšem má i omezení: ne všechny platformy poskytují veřejné API a často platí různé limity či nutnost autentizace (API klíče, omezení počtu dotazů za den apod.) (Yashaswini, 2024). V případě, že API není dostupné nebo neobsahuje požadovaná data, je pak web scraping (HTML/DOM) jedinou možností, jak data získat.

U web scrapingu je nutné zmínit, že kromě technických výzev (např. překonání dynamického načítání obsahu či anti-scraping opatření webu) existují i etické a právní aspekty. Web scraping se musí provádět v souladu s podmínkami užití daného webu a platnými zákony týkajícími se ochrany dat a autorských práv. Mnohé webové servery nasazují ochranné mechanismy proti automatizovanému sběru dat např. soubory robots.txt mohou indikovat, které části webu není žádoucí prohledávat, některé stránky používají CAPTCHA testy, blokování IP adres při podezření na robotickou aktivitu či omezování rychlosti dotazů (rate limiting) (Yashaswini, 2024). Scrapeři pak musí

případně implementovat protiopatření, jako je rotace IP adres, zpomalování rychlosti požadavků nebo řešení CAPTCHA, aby se chovali co nejvíce jako běžný uživatel (Yashaswini, 2024). Z právního hlediska je klíčové dodržovat terms of service daného webu a dbát na ochranu soukromí, například v EU může nešetrné získávání a ukládání osobních údajů narazit na GDPR. Obecně platí, že scraper dat by měl respektovat pravidla webu a využívat získaná data v souladu s právním rámcem a etickými normami (Yashaswini, 2024). Legitimita web scrapingu tedy závisí na kontextu. Scraping veřejně přístupných dat pro nekomerční účely může být považován za přijatelný, zatímco scraping citlivých či chráněných dat nebo obcházení silných ochranných opatření může být nelegální. Důležité je, aby výzkumníci i firmy při scrapingu dat z webu dbali na principy etiky a compliance, tedy získávat pouze data, která smí využít, minimalizovat zátěž serverů a vždy zvažovat možný dopad své činnosti (Yashaswini, 2024).

1.2 Přehled automatizačních platforem (Make.com a alternativy)

Realizace web scrapingu může probíhat programátorským způsobem (psáním kódu s využitím výše zmíněných knihoven a nástrojů) nebo stále častěji také formou no-code řešení. No-code platformy umožňují tvorbu aplikací a automatizaci procesů bez nutnosti programování. Uživatelé sestavují logiku pomocí grafického rozhraní, předpřipravených bloků a integrací (Caballar, 2020). Tyto nástroje si získaly popularitu tím, že usnadňují vývoj softwaru a zpřístupňují jej i lidem bez znalosti programování (Caballar, 2020). V oblasti automatizace workflow a integrace webových aplikací patří mezi nejznámější no-code platformy Zapier, Make.com (dříve známý jako Integromat) a n8n. Umožňující propojovat různé služby, spouštět akce na základě událostí a zpracovávat data mezi systémy. To lze využít i pro realizaci web scrapingu v rámci širších automatizačních scénářů.

1.2.1 Make.com (dříve Integromat)

Make.com je pokročilá no-code integrační platforma, původně vyvinutá v ČR pod názvem Integromat, která se zaměřuje na tvorbu automatizačních scénářů. Make se vyznačuje velkou flexibilitou, která nabízí bohatou sadu modulů pro práci s HTTP požadavky, formátování dat, iterace, podmínky, ale i integrace stovek aplikací podobně jako Zapier. Uživatel tak může vytvořit scénář, který periodicky spustí HTTP GET požadavek na určitou URL (např. produktová stránka e-shopu), získá HTML, následně pomocí modulů stránku rozparsuje (Make má vestavěné funkce pro extrakci textu mezi značkami, případně podporuje regulární výrazy či JSON extrakci), a pak třeba uloží výstup do Google Sheets nebo databáze (Kumaresh, 2024). Velkou výhodou Make.com je právě modul http, který umožňuje volat libovolné URL s metodami GET a POST, tím pádem se dá použít jak pro přístup k oficiálním API, tak pro přímé stahování webových stránek. V rámci této práce byla využita platforma Make k automatizaci sběru dat o videoreclích.

1.2.2 Zapier

Zapier je jednou z průkopnických platform v oblasti no-code automatizace (založena 2011). Jejím cílem je uživatelům co nejvíce usnadnit propojování webových aplikací a automatizaci opakujících se úloh (Kumparak, 2020). Zapier poskytuje prostředí, ve kterém lze vytvářet „Zaps“, tedy pracovní postupy sestávající ze spouštěče (trigger, např. příchozí e-mail nebo nový řádek v tabulce) a navazujících akcí v různých aplikacích. Platforma integruje přes 5000 různých aplikací a služeb, takže uživatel může například nastavit: „Když přijde nový e-mail s přílohou, nahraj přílohu na Dropbox a pošli upozornění na Slack.“ Vše konfiguruje ve webovém rozhraní bez psaní kódu. Zapier tak poskytuje integraci webových aplikací pro využití v automatizovaných workflow (*Automate without limits* | Zapier, 2025). Pro samotný web scraping Zapier nenabízí specializované moduly, ale lze jej využít v kombinaci s dostupnými akcemi: Zapier umožňuje volat webhooks a spouštět vlastní skripty (např. krok „Code“ pro běh JavaScriptu/Pythonu) v rámci workflow, což se dá využít k načtení webové stránky a parsování dat. Obecně je však Zapier orientován spíše na integraci existujících API (tj. spojování služeb, které mají konektory) a méně na libovolné stahování obsahu webu.

1.2.3 n8n

n8n je relativně novější nástroj (uveden kolem roku 2019) v kategorii no-code/low-code integrace, který je open-source (*n8n.io - a powerful workflow automation tool*, 2025). Jde o alternativu k Zapieru a Make, kterou si uživatel může sám provozovat na vlastním serveru. N8n nabízí workflow editor podobný Make, kde uživatel propojuje uzly představující jednotlivé kroky automatizace. K dispozici jsou integrace na mnoho služeb (přes 300 konektorů) a také generické funkce jako HTTP request, práci s daty, čekání, rozhodovací logiku apod. Velkým lákadlem n8n je, že je zdarma dostupný (licencován jako „fair-code“, zdrojový kód je k dispozici) a umožňuje libovolné přizpůsobení. Pro implementaci web scrapingu to znamená, že n8n může být nasazen například na vlastním serveru blízko zdroje dat a může spouštět scrapingové úlohy bez omezení třetí stranou. N8n umožňuje vkládat i vlastní kód (JavaScript) v uzlech typu „Function“, takže mezery v integracích lze doplnit napsáním malého skriptu. Příklad: Pomocí n8n lze vytvořit workflow, které si každé ráno stáhne HTML ze stránky s aktuálními cenami zboží, spustí vlastní JS funkci pro parsování HTML (pokud není k dispozici vestavěný parser), a výsledky odešle do e-mailu nebo uloží do souboru. Tím se n8n blíží plnohodnotnému nástroji pro web scraping, přičemž stále poskytuje pohodlí vizuálního návrhu toku a integraci následných kroků (ukládání, notifikace atd.). Oproti Make a Zapieru však n8n může vyžadovat o něco více technických znalostí na zprovoznění (self-hosting, nastavení prostředí) (*n8n.io - a powerful workflow automation tool*, 2025). Každopádně z pohledu funkcionality představuje n8n velmi silnou alternativu a uživatel není vázán cenovou politikou SaaS platformy a má plnou kontrolu nad daty i chováním workflow.

1.2.4 Další nástroje a platformy

Kromě výše jmenovaných existuje řada dalších nástrojů pro automatizaci a scraping. Například Apify je platforma specializovaná přímo na web scraping ve velkém měřítku.

Poskytuje předdefinované scrapers (tzv. *actors*) a možnost je spouštět bez starostí o infrastrukturu (Apify, 2025). Apify však vyžaduje určitou znalost programování (scrapers se typicky píšou v jazyce JavaScript/Python). Z oblasti RPA (*Robotic Process Automation*) stojí za zmínku UiPath či Selenium IDE (Wewerka & Reichert, 2020), které umožňují nahrávat a automaticky přehrávat činnosti uživatele v prohlížeči. Tyto nástroje se někdy využívají pro scraping webu s důrazem na vizuální postup (např. kdy uživatel nakliká, co se má odkud zkopírovat). Pro úplnost, v ekosystému no-code integrací existuje také IFTTT (If This Then That) (Yu et al., 2021), nicméně to je velmi jednoduchý nástroj zaměřený spíše na spotřebitelské automatizace a na web scraping se příliš nehodí (má omezené množství podporovaných služeb a žádnou možnost vlastního parsování obsahu).

1.2.5 Srovnání a volba platformy

Make.com se mezi těmito platformami vyznačuje svou širokou škálou integrací a pokročilými funkcemi pro webscraping, čímž poskytuje adekvátní řešení pro automatizaci procesů spojených se sběrem a zpracováním dat. Díky těmto vlastnostem je Make.com ideální volbou pro implementaci v rámci této práce.

1.3 Výhody a omezení no-code přístupu

Nasazení no-code platform pro úlohy, jako je web scraping, přináší řadu výhod, ale také určitá omezení ve srovnání s tradičním programováním. V této části shrneme obojí a uvedeme příklad z praktické implementace. Konkrétně využití platformy Make.com pro scraper napojený na vektorovou databázi v rámci systému RAG.

1.3.1 Výhody no-code přístupu

- **Nižší vstupní bariéra a rychlejší vývoj:** No-code nástroje umožňují i neprogramátorům realizovat poměrně komplexní automatizace. Grafické rozhraní a předpřipravené integrace zkracují křivku učení a je snazší začít skládat bloky, než psát kód od nuly (*Programming Without Code*, 2025). To vede také k rychlejšímu prototypování a nápady lze ověřit během hodin či dní místo týdnů vývoje.
- **Demokratizace a dostupnost pro doménové experty:** No-code řešení „rozdává moc tvořit software všem“ (*Programming Without Code*, 2025). To znamená, že ti, kdo nejlépe rozumí danému problému nebo datům (např. analytici, vědci), si mohou sami nastavit scraper a datový workflow podle svých potřeb, místo aby zadávali požadavky vývojářům. Odpadá riziko nepochopení mezi zadavatelem a programátorem. Tvůrce automatizace je přímo ten, kdo data potřebuje (*Programming Without Code*, 2025). V praxi tak například marketingový specialista může pomocí no-code platformy sám získávat data o konkurenčních produktech z webu, aniž by musel umět programovat nebo zaměstnat programátora.
- **Bohatý ekosystém integrací:** Platformy jako Zapier či Make mají tisíce propojení na různé služby (Parimi, 2025). To znamená, že výstup scrapingu lze

hned použít v dalším kroku. Uložení do Google Sheets, poslat e-mailem, vložit do databáze, spustit další proces apod. Nemusíme ručně psát kód pro autentizaci k API Google nebo posílání e-mailů, stačí přidat příslušný modul. No-code nástroje tak fungují jako mostem mezi systémy.

- **Menší starost o infrastrukturu:** SaaS platformy (Zapier, Make cloud) zajišťují provoz na svých serverech, škálování, fronty úloh atd. Uživatel se nemusí zabývat nastavením serveru, na kterém by běžel jeho scraper, řešit paralelní běh či ukládání mezivýsledků, to vše obstarává platforma. Například Make.com umožňuje nastavit interval spouštění scénáře a sama hlídá, aby se scénář spouštěl včas a v případě chyby modulů ji zalogue. Tato serverless povaha vývoje zrychluje nasazení řešení do praxe (Parimi, 2025).

1.3.2 Omezení no-code přístupu

- **Omezená flexibilita a výkon:** No-code nástroje obvykle neposkytují takovou míru kontroly jako psaní vlastního kódu. Jsou navrženy, tak aby pokryly většinu běžných případů, ale pokud nastane nestandardní požadavek, může být obtížné jej realizovat (Programming Without Code, 2025). V kontextu scrapingu to může být například situace, kdy je potřeba speciální zpracování obsahu (proprietární formát nebo nestandardní způsob stránkování). Pokud to platforma nepodporuje, nelze ani vložit vlastní kód, narazili bychom na strop možností no-code řešení. Dále, pro velmi rozsáhlý scraping (desítky tisíc stránek denně) mohou být no-code platformy pomalejší a nákladnější než vlastní řešení přímo nad HTTP knihovnou a paralelním zpracováním. Platformy mohou mít limity na počet operací nebo velikost dat za měsíc což se dá finančně prodražit.
- **Závislost na poskytovateli (vendor lock-in):** Použití konkrétní no-code služby znamená svěřit jí nejen běh, ale i často samotnou logiku procesu. Pokud bychom se rozhodli přejít jinam, neexistuje snadný způsob, jak exportovat scénáře, museli bychom je ručně přebudovat na jiné platformě nebo v tradičním kódu. Tato závislost může být problém z hlediska dlouhodobé udržitelnosti (služba může změnit podmínky, zdrazit, nebo dokonce ukončit provoz) a také bezpečnosti (data procházejí skrz servery třetí strany). V případě citlivých dat by byla vhodnější self-hosted varianta (jako n8n), aby organizace měla plnou kontrolu (Infragistics, 2025).
- **Méně možností ladění a testování:** Přestože moderní no-code nástroje nabízejí slušné možnosti logování a testování, stále to nedosahuje detailu, jaký má vývojář s debuggerem a jednotkovými testy v kódu. Složitější scénáře mohou být obtížněji čitelné, pokud jsou znázorněny pouze vizuálně. Např. modul selže a dá jen obecnou hlášku.

U vlastního kódu má vývojář možnost lépe ošetřit výjimky a logovat detailní informace (Parimi, 2025).

- **Náklady a škálování:** Na malých objemech dat jsou mnohé no-code platformy zdarma nebo levné. Ale pro produkční nasazení ve velkém měřítku mohou náklady

narůstat. Zapier i Make mají modely zpoplatnění často podle počtu tzv. operací (každý krok/scénář běhu) nebo podle počtu uživatelů. Intenzivní scraper, který projde tisíce stránek denně, by mohl rychle vyčerpat dostupný limit operací. Klasické skriptovací řešení běžící na vlastním serveru může být v takovém případě ekonomicky výhodnější. Škálování no-code řešení může znamenat nutnost přejít na dražší tarif, zatímco u vlastního řešení přidáme výkonnější server nebo optimalizujeme kód (Yan, 2021).

Z výše uvedeného plyne, že no-code přístup je ideální pro rychlý vývoj a středně velké úlohy, zejména pokud existuje mnoho integrací, které chceme propojit. U velmi specifických nebo rozsáhlých projektů se ale může stát, že kombinace vlastního kódu a no-code (či úplný přechod na kód) bude nutná.

2 RAG databáze a velké jazykové modely (LLM)

V této kapitole bude představen koncept Retrieval-Augmented Generation (RAG) a jeho propojení s velkými jazykovými modely LLM. Princip vyhledávání znalostí a generování odpovědí pomocí LLM, včetně hlavních přínosů RAG oproti čistě generativním modelům a ukázek využití (část 2.1). Následné porovnání možnosti uložení dat v tradičních databázích oproti vektorovým databázím, přičemž zaměření bude na nástroj Pinecone použitý v práci. Popsány budou husté embeddingy (dense embeddings), využití modelu text-embedding-ada-002 pro vytváření vektorových reprezentací a práce s dokumenty získanými web scrapingem (data ze Steamu přes platformu Apify, z IGDB API a z PDF reportu). Vše v kontextu vektorové databáze (část 2.2). Dále bude shrnut fungování velkých jazykových modelů: architekturu transformerů, přístupy zero-shot a few-shot promptování, a porovnání špičkových komerčních modelů GPT-4 od OpenAI s open-source alternativami (např. LLaMA, Mistral), včetně výhod GPT-4 při generování odpovědí s externím kontextem (část 2.3). V závěru kapitoly bude navržena jednoduchá struktura databáze v rámci práce. Konkrétně vektorové databáze s jednorázově nahranými daty, embeddingy vytvořenými modelem ada-002 a jednoduchými metadaty. Uvedeny budou důvody této zvolené struktury a její vhodnost pro vyhodnocení pomocí metrik, jako je Mean Reciprocal Rank (MRR) a Precision@K (část 5.1.3)

2.1 Co je Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) je způsob, který kombinuje modely pro vyhledávání informací s generativními modely pro tvorbu odpovědí. Principem RAG je obohacení vstupu do velkého jazykového modelu o dodatečné informace získané z externí znalostní báze před tím, než model vygeneruje odpověď. Prakticky to znamená, že k uživatelskému dotazu nejprve vyhledáme relevantní dokumenty nebo údaje z databáze znalostí (např. textové dokumenty, články, databázové záznamy) a teprve poté je společně s původním dotazem předáme jazykovému modelu k zodpovězení. Tím se model připraví o kontextové znalosti, které nemusí být součástí jeho parametrické paměti (Kenneweg et al., 2024). RAG tedy rozděluje úlohu na dvě fáze: vyhledání (retrieval) relevantních znalostí a generování finální odpovědi modelu za využití nalezených znalostí (generation) (Kenneweg et al., 2024).

Hlavním důvodem vzniku RAG je omezená znalost uložená v parametrech LLM a nutnost pracovat s aktuálními či specializovanými informacemi. Velké jazykové modely jako GPT-3 nebo GPT-4 jsou trénovány na obrovských korpusech textů, takže v nich jsou velké množství obecných znalostí (Lewis et al., 2020). Tyto znalosti, ale končí časově i obsahově tam, kde končila trénovací data modelu, a není snadné je dodatečně rozšířit o nové informace. (Petroni et al., 2019) například ukázali, že předtrénované jazykové modely dokážou ve svých parametrech uchovávat fakta podobně jako znalostní báze, ačkoli přístup k těmto faktům je nespolehlivý a model občas „zapomíná“ či zkresluje znalosti. Navíc LLM neobsahují informace, které nebyly v trénovacích datech. Typicky novinky po

datu posledního trénování, privátní uživatelská data nebo specifické doménové znalosti. RAG tento problém řeší tím, že model nemusí spoléhat pouze na svou parametrovou paměť, ale může získat aktuální a relevantní informace z vnějšího zdroje těsně před generováním odpovědi (Lewis et al., 2020). Tím lze do systému začlenit nové poznatky bez nutnosti přetrénování samotného jazykového modelu.

Přínosy RAG oproti čistě generativním LLM se projevují zejména ve dvou oblastech: (a) přesnost a faktická správnost generovaných odpovědí a (b) rozšíření domény znalostí modelu. Vzhledem k tomu, že LLM mohou při samostatném generování trpět tzv. halucinacemi (hallucinations), tj. generováním textu, který zní věrohodně, ale je nesmyslný nebo fakticky neodpovídá zdrojovým informacím. RAG umožňuje, jak toto riziko snížit (Lewis et al., 2020). Poskytnutím ověřených znalostí z externí databáze můžeme model usměrnit, aby se držel faktů a kontextu nalezených dokumentů, čímž se snižuje pravděpodobnost vzniku halucinací (Chang et al., 2023). Uživatelé tak mohou dosáhnout vyšší věcné správnosti výsledků tím, že model má k dispozici relevantní podklady a „nevymýšlí si“ vlastní odpovědi mimo znalosti v promptu (Lewis et al., 2020). Druhým přínosem je, že LLM rozšířený o retrieval umí zodpovídat i dotazy, které by jinak kvůli omezenému doménovému oboru nezvládl. Například dotaz na velmi specifická data jako jsou technické parametry určitého produktu, citace z odborného článku nebo aktuální statistiky, lze pomocí RAG zodpovědět tak, že se příslušná informace nejprve vyhledá ve specializované databázi či na webu a model ji pak zakomponuje do své odpovědi.

Příklady využití RAG sahají napříč různými úlohami zpracování přirozeného jazyka, zvláště tam, kde je vyžadována práce s externími znalostmi. RAG se úspěšně aplikuje v otevřeném doménovém zodpovídání dotazů (open-domain question answering), kde systém hledá odpovědi například ve Wikipedii či jiných znalostních zdrojích a následně formuluje odpověď vlastními slovy (Zhu et al., 2021). Právě v této oblasti dosáhl model RAG od Facebook AI špičkových výsledků s kombinací vyhledávače (dense retriever) a generativního modelu se podařilo překonat dřívější systémy v znalostně náročných úlohách otázek a odpovědí (Lewis et al., 2020). Další oblastí jsou dialogové systémy a chatboti, kde se RAG používá k obohacení odpovědi o aktuální informace. Weston et al. (2018) a Dinan et al. (2019) například ukazují, že do konverzačních agentů lze zapojit modul, který k danému tématu vyhledá relevantní věty z dokumentů (tzv. knowledge retrieval) a model je pak zapracuje do své odpovědi, čímž konverzace působí informovaněji. Podobně ve strojovém překladu byl testován přístup, kdy překladový systém při překladu věty nejprve vyhledá podobné věty nebo fráze v obrovském vícejazyčném textovém korpusu a využije je jako kontext pro vytvoření přesnějšího překladu (Gu et al., 2018). Obecně se dá říct, že RAG nachází uplatnění všude tam, kde čistě generativní model nemusí mít dostatečné informace. Typicky v doménově-specifických asistentech (např. lékařský nebo právní asistent hledající v odborné literatuře), v systémech otázka-odpověď nad podnikovými dokumenty, nebo v aplikacích jako jsou vyhledávače kombinující klasické fulltextové vyhledávání s jazykovým modelem (např. Bing Chat apod.). RAG celkově představuje flexibilní rámec, který spojuje schopnost najít relevantní data v rozsáhlém korpusu schopnost plynule formulovat lidsky srozumitelnou odpověď. Díky tomu dokáže uživatelům poskytnout přesnější a aktuálnější odpovědi, než jaké by generoval samotný model bez podpory externích znalostí (Lewis et al., 2020).

2.2 Možnosti uložení dat: tradiční vs. vektorové databáze

Aby mohl systém typu RAG fungovat, je zapotřebí vhodně uložit znalostní bázi (dokumenty či data, ze kterých se vyhledává). Nabízejí se v zásadě dvě hlavní možnosti: použití tradičního databázového systému (případně fulltextový vyhledávač) nebo moderní vektorovou databázi optimalizovanou pro ukládání embeddingů. V podkapitole jsou popsány rozdíly mezi oběma přístupy a objasněno, proč byl v projektu využit vektorové úložiště Pinecone spolu s hustými embeddingy z modelu ada-002.

2.2.1 Tradiční databáze vs. vektorové databáze

Tradiční databáze (relační SQL databáze typu MySQL, PostgreSQL apod., případně NoSQL dokumentové databáze) jsou navrženy pro ukládání a dotazování nad strukturovanými daty. Typicky číselnými hodnotami a texty, které lze indexovat pomocí klíčů, filtrů nebo fulltextově (Mittal et al., 2024). Vyhledávání informací v textových datech zde obvykle probíhá na základě klíčových slov. Například fulltextový index najde záznamy obsahující určité slovo či frázi. Tento přístup je vhodný pro přesné dotazy, ale jeho schopnost zachytit sémantickou podobnost dvou textů je omezená. Celkově platí, že tradiční databáze pracují s daty v původní podobě (slova, čísla) a porovnávají je na základě přesné shody nebo jednoduchých operací (řetězcové porovnání, numerické operátory) (Mittal et al., 2024).

Oproti tomu vektorové databáze jsou specializované systémy pro ukládání a vyhledávání dat reprezentovaných jako vícerozměrné vektory (číselná pole). Myšlenkou je, že nestrukturovaný obsah (text, obrázek, audio...) se před uložením převede pomocí strojově naučeného modelu do formy embeddingu. Tedy matematického vektoru v desítkách až stovkách rozměrů, který nese sémantickou informaci o obsahu. Podobné položky, jako jsou věty s podobným významem, jsou v takovém vektorovém prostoru reprezentovány body blízko sebe (Belagatti, 2024). Vektorová databáze pak neukládá text přímo, ale jako vektorové reprezentace a umožňuje efektivně vyhledávat nejbližší vektory k zadanému dotazu, který je rovněž převeden na vektor. Namísto hledání shody slov tak probíhá hledání na základě matematické podobnosti vektorů. Často se využívá metrika kosinusové podobnosti nebo eukleidovské vzdálenosti mezi vektory. Výsledkem dotazu je seznam nejvíce podobných položek z celé databáze (tzv. *nearest neighbors search*) (Bruch, 2024).

Z technického hlediska představují vektorové databáze zásadní odklon od architektury tradičních databází. Musí být navrženy tak, aby zvládly efektivně indexovat a prohledávat velkorozměrná data. Vektor v např. 1000rozměrném prostoru nelze indexovat klasickými B-stromy či hash tabulkami tak, jako jednorozměrné číslo nebo textový klíč. Proto se používají algoritmy pro přibližné vyhledávání nejbližších sousedů (*Approximate Nearest Neighbors*, ANN) (Bruch, 2024), jako jsou grafové algoritmy (např. HNSW – Hierarchical Navigable Small World) nebo kvantizační metody, které dokážou výrazně urychlit vyhledávání podobných vektorů i v miliardách bodů. Vektorová databáze často interně kombinuje takový ANN index s tradičními databázovými prvky pro uložení samotných vektorů a jejich metadata. Oproti pouhým knihovnám pro ANN (jako je například Faiss od Facebook AI) (Schwaber-Cohen, 2023). Plnohodnotná vektorová databáze poskytuje i

funkce databázového systému, jako je perzistence dat na disku, škálování na více uzlů, správa více kolekcí dat, aktualizace a mazání záznamů či možnost filtrovat výsledky podle metadat (Schwaber-Cohen, 2023).

Klíčovou výhodou vektorových databází je schopnost porovnat dotaz s uloženými položkami na bázi významu, nikoli jen povrchu textu. Umožňují tak implementovat sémantické vyhledávání, např. najít dokumenty, které odpovídají tématu dotazu, i když nepoužívají stejná slova. To je zásadní pro aplikace s LLM, protože model může potřebovat najít informace vyjádřené různými slovy, než jaká použil uživatel v dotazu. Vektorové databáze jsou proto vhodné pro AI aplikace, kde data mají povahu embeddingů: vyhledávání podobných obrázků, doporučovací systémy (hledání podobných uživatelských preferencí), rozpoznávání hlasu či sémantické prohledávání textů (Belagatti, 2024). Na druhou stranu, tradiční databáze zůstávají lepší volbou pro úlohy, kde je potřeba precizních dotazů nad strukturovanými poli (např. „najdi všechny uživatele s věkem 25–30 z Prahy“). Zde vektorová reprezentace nedává smysl. V praxi se tedy často kombinují: například dokumenty se ukládají v dokumentové databázi či souborovém úložišti pro následné zobrazení, ale paralelně se jejich embeddingy ukládají do vektorového úložiště pro účely vyhledávání.

Tradiční databáze ukládá data ve formě blízké jejich původní struktuře a umožňuje dotazování nad přesnými hodnotami nebo vzory (v případě textu spíše lexikální shoda), zatímco vektorová databáze ukládá odvozené reprezentace (embeddingy) a umožňuje dotazy založené na významové podobnosti. V kontextu RAG, kde chceme najít dokumenty relevantní k uživatelskému dotazu dle obsahu, poskytuje vektorová DB mnohem lepší výkon v tom smyslu, že dokáže najít i implicitně související znalosti.

2.2.2 Uložení znalostí ve vektorové DB (Pinecone) a embeddingy ada-002

V tomto projektu byla využita vektorová databáze Pinecone jako úložiště pro znalostní data ke generování odpovědí. Pinecone je plně spravovaná cloudová vektorová databáze, která nabízí jednoduché API pro vložení vektorů a vyhledávání nejbližších sousedů ve velkém měřítku (škálovatelně až na miliony a více vektorů) („[Vaccination 2021] The Pinecone Vector Database System (Edo Liberty)“, 2021). Hlavním důvodem volby Pinecone byla její schopnost rychle zpracovávat dotazy podobnosti a integrace s moderními embedding modely. Pinecone pod kapotou kombinuje metody vektorové indexace s cloudovou infrastrukturou, takže uživatel se nemusí starat o detaily implementace ANN algoritmů nebo distribuce dat. Jednoduše vloží vektory s přidruženými metadaty a později k nim může provádět dotazy pomocí vektoru dotazu. Pinecone vrátí ID nejbližších vektorů a jejich skóre podobnosti.

Dense embeddings neboli husté vektorové reprezentace textu hrají důležitou roli v tom, jak jsou data do Pinecone ukládána a jak se v nich vyhledává. Pro převod textových dokumentů do vektorové podoby využíváme jazykový model pro embeddingy od OpenAI s názvem *text-embedding-ada-002*. Jde o model, který na vstupu přijme libovolný text (větu, odstavec, celý dokument až do délky cca 8 tisíc tokenů) a převede jej na 1536-rozměrný číselný vektor (Greene et al., 2024). Tento model byl představen koncem roku 2022 jako výrazně vylepšená generace embedding modelů dosahuje lepších výsledků

v úlohách vyhledávání a sémantické podobnosti než předchozí větší model (davinci) a přitom je mnohem rychlejší a levnější na použití (Greene et al., 2024). Výstupem modelu ada-002 je tzv. hustý vektor o délce 1536, což znamená, že hodnoty jsou rozprostřeny napříč všemi rozměry (na rozdíl od řídkých vektorů, jako je třeba klasický TF-IDF, kde by mnohé položky byly nulové). Hustota zaručuje, že model zakódoval informaci o vstupním textu distribuovaně. Každá část vstupu ovlivní většinu komponent vektoru. Výhodou je vysoká vyjadřovací schopnost: vektor zachycuje význam slov, kontext, styl a další aspekty textu v numerické formě, kterou lze snadno porovnávat výpočtem vzdálenosti.

V rámci projektu byla shromážděna data z více zdrojů pomocí web scrapingu a API, konkrétně: Z webu Steam (popisy, recenze či další informace o produktech/hrách). Extrakce proběhla přes platformu Apify, která umožňuje automatizované scrapovací scénáře. Z IGDB API (Internet Games Database), což je otevřené API nabízející strukturovaná data o počítačových hrách, např. žánry, vydavatele, hodnocení atd. Z PDF reportu (např. analyzujícího trh videoher či jiný relevantní dokument). Tyto zdrojové dokumenty pokrývají jak strukturované informace (z IGDB např. konkrétní pole s daty), tak nestrukturované texty (popisy ze Steamu, text zprávy v PDF). Před uložením v databázi Pinecone bylo potřeba všechny tyto heterogenní obsahové jednotky převést do jednotného vektorového tvaru pomocí zmíněného modelu ada-002. Každý dokument či textová pasáž tak dostal svůj 1536dimenzionální embedding. Ten byl následně nahrán do Pinecone spolu s jednoduchým metadatem typu text. U každého vektoru jsou uchovány informace jako zdroj pomocí prefixu (např. „Steam“, „IGDB“, „Report“). Tato metadata umožňují případně filtrovat vyhledávání (např. hledat odpovědi jen v určitém zdroji) a také slouží pro orientaci při výpisu výsledků.

Pinecone jako úložiště vektorů se v projektu stará o rychlé nalezení těch nejvíce podobných embeddingů k embeddingu dotazu. Když uživatel položí dotaz, systém: (1) vezme text dotazu a spočítá jeho embedding (opět modelem ada-002), (2) pošle tento vektor jako dotaz do Pinecone, (3) Pinecone provede vyhledání nejbližších vektorů v databázi a vrátí např. top K nejpodobnějších spolu s jejich ID a metadaty. (4) Systém poté získá původní texty dokumentů odpovídajících těmto ID (ty může uchovávat buď přímo v Pinecone jako součást metadat, nebo v samostatném úložišti, v tomto případě pro zjednodušení byly ukládány krátké texty přímo do Pinecone). Tyto nalezené texty pak tvoří znalostní kontext, který je předán LLM pro finální vygenerování odpovědi. Celý proces vyhledání je díky vektorové databázi velmi rychlý. Pinecone je navržen na latence v řádu desítek milisekund i při kolekcích obsahujících miliony vektorů, a navíc se automaticky škáluje, takže performance netrpí ani s rostoucím objemem dat (Schwaber-Cohen, 2023).

Zvolená kombinace Pinecone a ada-002 přináší několik výhod. Jednoduchost integrace s existujícími klientskými knihovnami pro Pinecone i OpenAI API, vysoká kvalita embeddingů ada-002 zajišťuje, že podobné texty budou blízko sebe ve vektorovém prostoru a možnost sjednotit velmi různorodá data strukturovaná i nestrukturovaná do jedné vyhledávací indexové struktury. Alternativou by bylo použít klasický fulltextový vyhledávač (např. Elasticsearch) nad texty. Ten však neposkytuje tak jemnou sémantickou shodu a museli bychom se spoléhat na ručně zvolená klíčová slova nebo booleovské filtry, což je méně flexibilní. Proto je vektorová databáze v kontextu RAG téměř nezbytností u moderních systémů pracujících s nehomogenními textovými znalostmi.

2.3 Principy fungování velkých jazykových modelů (LLM)

Velké jazykové modely tvoří jádro generativní části systému RAG. Abychom porozuměli jejich schopnostem a limitacím, bude přiblížen princip jejich architektury a způsob, jakým se učí z promptů. Zaměření bude na rozdíly mezi špičkovými uzavřenými modely konkrétněji GPT-4 od OpenAI a open-source alternativami jako jsou modely rodiny LLaMA či model Mistral 7B. Pozornost věnujeme tomu, jak si tyto modely vedou při využití externího kontextu, a jaké výhody v tomto ohledu nabízí GPT-4.

2.3.1 Architektura transformerů jako základ LLM

Moderní velké jazykové modely jsou téměř výhradně založeny na architektuře Transformer, kterou představili (Vaswani et al., 2017). Transformer způsobil revoluci ve zpracování přirozeného jazyka tím, že nahradil dříve dominující rekurentní neurální sítě a sekvenční modely efektivnější attention mechanismem. Základním stavebním kamenem transformera je mechanismus self-attention, který umožňuje modelu vážit význam jednotlivých slov (tokenů) v kontextu všech ostatních slov ve větě, nebo obecně v sekvenci vstupních tokenů. Model se tedy učí dívat na vstupní text „všemi směry“: každé slovo může přímo navázat pozornost na jiná relevantní slova, bez ohledu na jejich vzdálenost.

Transformer se skládá z vrstvených bloků, které obsahují multi-head attention a feed-forward sítě. Vícehlavá pozornost umožňuje modelu nahlížet na vstup v několika projekcích zároveň. Každá hlava pozornosti se může soustředit na jiný aspekt vztahů mezi slovy (např. jedna hlava může zachycovat vztah podmět-sloveso, jiná hlava vztah slovesa a předmětu, další např. anaforické vztahy atd.) (Gloeckle et al., 2024). Tím model získává bohatší reprezentaci kontextu. Následuje plně propojená neuronová síť v každém bloku, která zpracuje výstupy z attention a provádí nelineární transformaci. Tyto bloky se skládají na sebe do značné hloubky (u velkých modelů i 50–100 vrstev). Transformer může být dekodér (generativní model, který na základě kontextu předchozích tokenů predikuje další token. To je případ GPT modelů, enkodér tedy analytický model, který zpracuje celý vstup a produkuje jeho vektorové reprezentace používají např. BERT modely pro porozumění textu nebo kombinace enkodér–dekodér) (Devlin et al., 2019). U strojového překladu, kde enkodér zpracuje vstupní větu a dekodér generuje výstupní větu. LLM typu GPT jsou dekodérové transformery, trénované predikovat pokračování textu.

2.3.2 Zero-shot a few-shot prompting

Specifickou vlastností velkých jazykových modelů, objevenou zvláště s nástupem modelu GPT-3 (175B parametrů) od OpenAI, je schopnost Zero-shot a Few-shot learning čistě pomocí vstupního promptu. Brown et al. (2020) ukázali, že dostatečně velký model dokáže řešit nové úlohy bez jakéhokoli dalšího trénování, pouze z textového zadání, které popíše, co má model dělat (*zero-shot*). Například pokud modelu předložíme dotaz ve stylu „*Přelož následující větu do francouzštiny: Text...*“, model to dokáže provést, přestože se explicitně neučil pravidla francouzštiny, obecné znalosti jazyka a kontext z instrukce mu dostačují k pochopení úkolu. Zero-shot prompting tedy znamená, že model je instruován k určitému

úkolů pouze slovním popisem nebo otázkou, bez příkladů. Model využije svou vnitřní generalizovanou znalost.

Ještě lepších výsledků lze často dosáhnout, když je použit Few-shot prompting poskytneme modelu i několik ukázkových vstupů a správných výstupů (tzv. *prompt examples* nebo *demonstrations*), na základě, kterých model pochopí, jaký je formát a požadovaný styl odpovědi. Například modelu je předložen prompt: „*Převeď číselky na slova. Příklad: 123 -> sto dvacet tři. 45 -> čtyřicet pět. Nyní odpověz: 678 ->*“. Tímto promptem jsou předány dvě tréninkové dvojice (few-shot) a model s velkou pravděpodobností správně napíše „šest set sedmdesát osm“. Few-shot učení v promptu umožňuje modelu lépe se naladit na konkrétní úkol a výstupní formát bez nutnosti skutečného gradientního trénování. Vnitřně model využije schopnost *in-context learning*, tedy poučení se z kontextových ukázek. Tento jev byl jedním z hlavních překvapivých objevů u GPT-3 (Brown et al., 2020). I bez explicitního ladění na úlohu model dosahoval velmi dobrých výkonů napříč mnoha typy úloh, jen na základě chytrého zformulování vstupu.

V praxi to znamená, že při použití LLM v našem systému je možné promptovat model tak, aby správně využil dodaný kontext z RAG. Možné je například napsat instrukci: „*Na základě uvedených dokumentů odpověz na dotaz...*“ a model (pokud je dostatečně schopný) pochopí, že má čerpat informace hlavně z poskytnutých dokumentů. Je-li požadováno, do promptu lze napsat: „*Dokument: ... Otázka: ... Odpověď: ...*“ jako demonstraci formátu. Tím je představen koncept prompt engineering. Schopnost navrhnout prompt tak, aby model podal požadovaný výkon. Zero-shot a few-shot jsou dvě z hlavních strategií. Zero-shot je jednodušší (kratší prompt), few-shot bývá spolehlivější, ale na úkor délky promptu (zabírá místo v kontextu). U nejnovějších modelů (včetně GPT-4) se často využívá i tzv. chain-of-thought prompting (Wei et al., 2023), kde se model nabádá, aby nejprve krok za krokem vysvětlil své uvažování, než dá finální odpověď. To může zlepšit přesnost v komplexních úlohách. Tyto techniky však spadají spíše do prompt engineering. Pro RAG je důležité, že model dokáže zpracovat dodaný kontext (nalezené dokumenty) a odpovědět na otázku, aniž by bylo nutného implicitního programování pro práci s dokumenty, kde stačí správně formulovat pokyn v promptu.

2.3.3 GPT-4 vs. open-source LLM (LLaMA, Mistral) a práce s externím kontextem

GPT-4 od OpenAI (uvedený v březnu 2023) představuje špičku mezi jazykovými modely z hlediska porozumění a generování textu. Jde o proprietární model s odhadovanou velikostí stovek miliard parametrů, trénovaný na velkém množství dat a doladěný pomocí RLHF pro vyšší kvalitu a bezpečnost výstupů. Alternativou jsou open-source modely jako LLaMA 2 (Touvron et al., 2023) nebo Mistral 7B (Jiang et al., 2023), které lze provozovat a upravovat lokálně. Mají menší velikost (např. 7B, 13B, 70B parametrů) a nižší nároky na výpočetní výkon, přičemž díky efektivní architektuře dokážou někdy konkurovat i větším modelům (např. Mistral 7B překonává LLaMA 13B v některých úlohách).

Rozdíly mezi GPT-4 a open-source modely se projevují v několika rovinách:

- **Výkonnost:** GPT-4 dosahuje špičkových výsledků v benchmarkových testech, zatímco menší modely mohou mít omezenou schopnost logického uvažování či výkon ve specifických doménách.
- **Kontext:** GPT-4 zvládne vstupy o délce až 32 000 tokenů, což umožňuje zpracování rozsáhlých dokumentů. Oproti tomu LLaMA 2 a Mistral 7B běžně pracují s ~4 000 tokeny.
Porozumění instrukcím: GPT-4 díky RLHF lépe dodržuje pokyny a přizpůsobuje se stylu. Menší modely často vyžadují přesnější promptování a mohou mít tendenci ignorovat pokyny, pokud nejsou dostatečně doladěné (OpenAI, 2023).

V kontextu RAG má GPT-4 výhodu v tom, že dokáže přesně zpracovat externí dokumenty a odpovědět na základě jejich obsahu. Je méně náchylný k halucinacím a lépe pracuje s odborným textem (OpenAI, 2023). Open-source modely mohou v těchto případech selhávat, ale jejich výkon lze zlepšit pomocí fine-tuningu (např. LLaMA 2-Chat 70B poskytuje solidní výstupy, pokud je správně použit).

Další rozdíl je v dostupnosti. GPT-4 je přístupný pouze přes API a podléhá omezením (platby, limity, nutnost připojení). Oproti tomu open-source modely lze provozovat lokálně, což zajišťuje plnou kontrolu nad daty a modelem.

GPT-4 obecně nabízí lepší výsledky v náročných dotazech a s komplexním kontextem, zatímco open-source modely jsou vhodné tam, kde je nutný lokální provoz.

2.4 Návrh struktury vektorové databáze pro projekt

Vzhledem k výše popsaným možnostem je zvolen pro práci no-code webscraper s RAG jednoduchá a přímočará struktura vektorové databáze. Cílem je ukládat data tak, aby byla snadno použitelná pro vyhledávání relevantních informací k dotazu a zároveň aby nastavení systému bylo přehledné pro účely experimentů a vyhodnocení. Klíčové prvky návrhu zahrnují: jednorázovou inicializaci databáze se statickým souborem dat (tj. data se do DB nahrají jednorázově po jejich scrapování, nikoli průběžně během dotazování), vytváření embeddingů všech položek předem modelem ada-002, a ukládání jen nezbytných metadat u každé položky. Tato sekce popisuje konkrétní strukturu tabulek/kolekcí a zdůvodňuje, proč je zvolené řešení dostatečné a vhodné pro hodnocení systému pomocí metrik jako MRR a Precision@K.

2.4.1 Struktura uložených dat a metadat

Vektorová databáze Pinecone je organizována do tzv. indexů (indexes), které je chápáno jako ekvivalent tabulky nebo kolekce dokumentů. Pro projekt postačí jeden index, v němž budou uloženy všechny znalostní položky (dokumenty) napříč zdroji. Každý záznam v indexu obsahuje následující složky:

V projektu byla zvolena minimalistická, ale dostatečně flexibilní struktura vektorové databáze, která odpovídá charakteru získaných dat a požadavkům systému RAG. Každý dokument v databázi reprezentuje jednu znalostní položku (např. popis hry, recenzi, část článku) a obsahuje tři klíčové komponenty:

- **ID:** unikátní identifikátor dokumentu ve formátu list3_game_123, list4_game_045 apod., který zároveň nese informaci o původu (např. list3, list4, BAIN).
- **Embedding:** numerická reprezentace textu ve formě vektoru délky 1536, vytvořená modelem text-embedding-ada-002. Tento embedding slouží jako základ pro porovnávání podobnosti mezi dotazem a uloženými dokumenty.
- **Text:** samotný obsah položky, který může zahrnovat název hry, popis, žánr, cenu, recenzi apod. Text je uložen jako jeden celek ve formě řetězce. Vzhledem k omezenému objemu dat si můžeme dovolit uchovávat celý text přímo ve struktuře dokumentu, což zjednodušuje proces získání zpětné odpovědi.

Ukázka záznamu v indexu Pinecone:

Record ×

namespace: (Default)

ID: list4_game_50

values: [0.00589520065, -0.0268163681, -0.00335528166, -0.0148318326, -0.0127804...

metadata:

text: "Title: Kingdom Come: Deliverance II\nDeveloper: Warhorse Studios\nPublisher: Deep Silver\nFeatures: Single-player, Steam Achievements, Steam Trading Cards, Steam Cloud, Family Sharing\nEarly Access: False\nComing Soon: False\nRelease Date: Feb 4, 2025\nGenre: Action, Adventure, RPG \nPrice: \$59.99\nOS: windows \nUrl: https://store.steampowered.com/app/1771300/Kingdom_Come_Deliverance_II/?snr=1_4_4__tab-TopGrossing\nAll Reviews: Very Positive\nRecent Reviews: Very Positive\nSALE: False\nSale Percentage: \nSale Until: \nLanguages: \nDescription: A thrilling story-driven action RPG, with a rich open world, set in 15th century Medieval Europe. Experience the ultimate medieval adventure - through the eyes of young Henry - as you embark on a journey of epic proportions.\nTags: RPG, Medieval, Open World, Singleplayer, Realistic, Realistic, First-Person, Story Rich, Adventure, Action, Combat, Historical, Political, Swordplay, Immersive, Alternate History, Choices Matter, Atmospheric, LGBTQ+, Action-Adventure, 3D"

Zdroj: vlastní zpracování v rámci Pinecone.io

Volitelně by šlo strukturu rozšířit o další metadata, např.:

- **source:** označení původu dat (např. „Steam“, „IGDB“, „BAIN report“),
- **type:** druh obsahu (např. „description“, „review“, „article“),
- **scrape_date:** datum získání dat pro účely filtrování či sledování aktuálnosti.

Každý dokument tedy při nahrání do DB projde pipeline: text -> embedding (ada-002) -> uložení (ID, embedding, metadata). Po nahrání všech dat (z různých zdrojů) je připravena znalostní báze pro dotazy.

Jednoduchá struktura bez složitých relací či více indexů byla zvolena z několika důvodů. Zaprvé, objem dat není extrémně velký (řádově tisíců položek), takže není třeba řešit horizontální škálování nebo rozdělovat data dle zdroje. Jeden index je dostačující. Zadruhé, sjednocení dat do jednoho indexu nám umožňuje dotazovat napříč všemi zdroji zároveň a nechat vyhledávací algoritmus určit, které položky jsou nejrelevantnější. Alternativou by bylo nejprve detekovat, ze kterého zdroje asi odpověď bude, a dotazovat jen ten. To by ale znesnadnilo proces a mohlo vést k přehlédnutí relevantní informace jinde. Jednoduchá struktura také usnadňuje ladění. Umožňuje snadno měnitelné parametry vyhledávání (např. kolik výsledků se vrací, jaké skóre podobnosti je ještě přijatelné) a vše se aplikuje na jednotné množině dat.

3 Návrh a implementace systému

3.1 Architektura systému

Retrieval Augmented Generation (RAG) je architektura, která kombinuje velký jazykový model (LLM) s externí znalostní databází. Cílem je poskytnout LLM aktuální a relevantní informace v průběhu generování odpovědi a tím zvýšit přesnost výsledků (Lewis et al., 2021). V našem systému je RAG realizován propojením tří hlavních komponent:

- **Make.com** – no-code automatizační platforma použitá jako web scraper a integrační prostředí pro orchestraci datových toků.
- **Pinecone** – cloudová vektorová databáze určená k ukládání a vyhledávání vektorových reprezentací (embeddingů) textových dokumentů.
- **OpenAI GPT-4** – velký jazykový model (přes OpenAI API) využitý k porozumění dotazům a generování konverzačních odpovědí s využitím nalezeného kontextu.

Tato architektura umožňuje překonat omezení statického tréninku LLM tím, že při dotazu dokáže do promptu dynamicky vložit relevantní znalosti z externích zdrojů (Proser, 2023). Jinými slovy, pokud uživatel položí otázku, systém nejprve vyhledá související informace ve vlastní znalostní bázi (uložené ve Pinecone) a tyto informace předá modelu GPT-4. Model pak generuje odpověď, která je „augmented“ (obohacená) o aktuální a doménově specifická data.

1. **Budování znalostní báze:** Nejprve proběhla jednorázová extrakce znalostí z externích zdrojů. Data z PDF, CSV apod. se nejdříve chunkují, transformují na vektory (embeddingy) pomocí embedovacího modelu (např. OpenAI *text-embedding-ada-002*). Takto získané vektory se následně indexují ve Pinecone, tj. ukládají se metodou upsert do indexu. Pokud index již obsahuje nějaká data, nová jsou přidána a existující aktualizována. Výsledkem je vektorový index naplněný „znalostmi“ ve formě vektorů (Walsh, 2024).
2. **Dotazování a generování odpovědi:** Při každém dotazu uživatele chatbot nejprve zpracuje dotaz a vygeneruje embedding dotazu stejným modelem jako v předchozím kroku. Tento vektor je pak použit pro dotaz do Pinecone (Proser, 2023). Databáze vrátí např. nejbližších k uložených vektorů spolu s jejich původním textem. Nalezené relevantní textové pasáže jsou následně předány GPT-4 jako součást promptu (doplnění dotazu uživatele o kontext). Model GPT-4 poté provede generování odpovědi, který zakomponuje poskytnuté kontextové informace do své odpovědi uživateli (Sasín, 2021).

Tímto způsobem systém využívá aktuální data (herní report, databáze her) uložená ve vektorové databázi ke zodpovídání dotazů, na které by samotný LLM neznal odpověď.

RAG architektura tak zlepšuje relevanci a faktickou správnost odpovědí a zároveň umožňuje dohledat zdroje, z nichž informace pocházejí (Proser, 2023).

3.2 Možnosti scrapování dat

Existuje několik přístupů, jak získat data pro účely RAG systému, pokud nejsou k dispozici již hotové datasety či API. Tou metodou je web scraping čili extrakce dat z webových stránek. Web scraping lze provádět programátorsky, a to pomocí skriptů a knihoven nebo v rámci práce za využití specializovaných nástrojů s minimálním programováním (tzv. *no-code/low-code* nástroje).

3.3 Zvolené postupy pro získání dat v práci

Apify Steam Actor: Platforma Apify nabízí předpřipravené Actors pro scrapování webů bez nutnosti programování. Konkrétně byl využit Steam Store Scraper, který prochází webový obchod Steam a extrahuje z každé stránky hry strukturovaná data (Moucka, 2022). Tento actor dokáže získat informace jako název hry, popis, datum vydání, žánry, tagy, podporované platformy, cenu či souhrnné hodnocení uživatelů. Výhodou použití Apify je, že výsledná data jsou automaticky uložena v datasetu a lze je exportovat v různých formátech (JSON, CSV, Excel) (Moucka, 2022). Steam samotný sice nabízí některá data přes vlastní API, ale ne v takovém rozsahu (např. detailní popisy her či uživatelské tagy nejsou veřejně dostupné přes oficiální API). Scrapování webu Steam proto poskytuje bohatý zdroj informací přímo z aktuálních stránek obchodu. Zvolení hotového Apify scraperu zároveň ušetřilo čas oproti vývoji vlastního crawleru a minimalizovalo riziko blokování ze strany Steamu.

IGDB API: internet Game Database (IGDB) je rozsáhlá online databáze videoher provozovaná společností Twitch (Amazon (Hong, 2023)). Obsahuje strukturované údaje o tisících hrách, například datum vydání, platformy, žánry, vývojáře, stručné popisy atd. (Hong, 2023). IGDB poskytuje oficiální API, které umožňuje vývojářům tyto údaje dotazovat. V této práci bylo API využito pro získání ověřených informací o hrách z autoritativního zdroje. Důvodem volby IGDB je B (jde o komunitně spravovanou databázi s jednotnou strukturou záznamů), a také to, že použití API je oficiální a přímočaré pro větší množství dotazů (na rozdíl od scrapování jiných webů, které by mohly porušovat podmínky užití). IGDB API tak doplnilo data ze Steamu o další atributy či hry, které na Steamu nemusely být snadno dohledatelné. Výhodou také je, že data z API jsou vrácena ve strukturovaném formátu (JSON), což usnadňuje následné zpracování.

PDF report od Bain & Company: Zdrojem dat byl oborový report stažený ve formátu PDF od poradenské společnosti Bain & Company. Konkrétně šlo o výroční studii herního trhu (tzv. *Video Game Consumption Survey*), která obsahuje agregované statistiky a poznatky o chování hráčů a trendech v herním průmyslu. Tato data jsou odlišná povahy než údaje o jednotlivých hrách ze Steamu/IGDB. Jde spíše o analytické informace, např. kolik hodin týdně průměrný hráč hraje, jaké žánry jsou nejpopulárnější, demografické rozložení hráčské komunity apod. Zařazení tohoto dokumentu do znalostní databáze

umožňuje chatbotovi odpovídat i na obecnější dotazy o herním odvětví (např. globální trendy nebo statistiky), které by samotné databáze her nepokryly. Bain & Company je renomovaný zdroj průzkumů trhu, a proto byly informace z tohoto reportu považovány za důvěryhodné. PDF soubor byl získán manuálně (stažením z webu) a představuje ukázkou statického zdroje dat doplňujícího dynamického zdroje (Steam, IGDB).

Zvolené tři zdroje dat se vhodně doplňují: Steam (Apify) poskytl aktuální uživatelsky orientované informace o hrách (popisy, hodnocení, tagy), IGDB dodal strukturovaná fakta o hrách (datum vydání, žánr, tvůrci) a Bainův report přidal kontextuální znalosti o celém odvětví. Alternativními možnostmi mohlo být např. použití jiných API (např. OpenCritic, RAWG) či scrapování Wikipedie nebo Metacritic. Autor však vyhodnotil, že kombinace Steam + IGDB pokryje potřebné informace o jednotlivých hrách a Bain poskytne širší pohled, což dohromady dostačuje pro zamýšleného chatbota.

3.4 Výzvy spojené s extrakcí a strukturováním dat

Při práci s výše uvedenými zdroji nastalo několik výzev v oblasti extrakce a strukturování dat. Každý zdroj měl totiž odlišnou formu, omezení a vyžadoval specifické zpracování, aby bylo možné data sjednotit do jedné znalostní báze použitelné pro RAG.

Omezení a specifika API IGDB: Ačkoli IGDB API poskytuje bohatá data, je nutné počítat s jeho omezeními. Pro využití je nejprve vyžadována registrace vývojářského účtu u Twitch a získání API klíče (OAuth tokenu), který má omezenou platnost a musí se pravidelně obnovovat. Z hlediska výkonu má API stanovený rate limit. Podle dokumentace je možné posílat maximálně 4 požadavky za sekundu, jinak server vrací chybu 429 (Too Many Requests) (CorerMaximus, 2020). Toto omezení (spolu s dřívějším limitem 50 000 požadavků za měsíc) klade důraz na to, aby aplikace dotazování optimalizovala a případně kešovala výsledky. V praxi bylo třeba implementovat zpoždění mezi sekvenčními dotazy a rozdělit stahování většího množství herních záznamů do delšího časového období, aby nedošlo k překročení kvót. Pro jednoduchost a úsporu kvóty bylo zvoleno stažení pouze nejdůležitějších polí (např. jméno hry, datum vydání, žánry jako text) přímo, aby nebylo nutné kombinovat více dotazů. Strukturovaná data z IGDB (ve formátu JSON) pak bylo nutné převést do jednotného tabulkového formátu (CSV) s definovanými sloupci, což obnášelo i rozhodnutí, které atributy jsou pro chatbota relevantní.

Zpracování dat ze Steamu přes Apify: Použití Apify actoru značně usnadnilo extrakci, nicméně i zde byla strukturalizace dat výzvou. Výstupní dataset ze Steam Store Scraperu obsahoval jeden objekt (řádek) na každou nalezenou hru (Moucka, 2022). Tyto objekty obsahovaly řadu polí (název, popis, cena, tagy, jazyky atd., viz ukázka JSON struktury). Před použitím v RAG bylo potřeba data očistit a případně transformovat. Také bylo nutné zvážit, které informace ze Steamu mají smysl pro chatbota. Některé detailní údaje jako seznam podporovaných jazyků či systémové požadavky nebyly pro odpovídání na obecné dotazy relevantní, a proto mohly být vypuštěny. Naopak uživatelské tagy nebo žánry byly užitečné pro dotazy typu „jaký žánr je tato hra“. Proto byla pro každý záznam hry vytvořena konsolidovaná textová reprezentace zahrnující klíčové informace (např. "The Stanley Parable: Ultra Deluxe – adventura, indie; vydáno 2022; popis: ...; uživatelské

hodnocení: velmi pozitivní...). Tato reprezentace kombinovala data z IGDB i Steamu (aby se eliminovala redundance a využilo se to nejlepší z obou zdrojů). Výzvou bylo zajistit, aby sloučené informace byly konzistentní (např. shoda názvů her z obou zdrojů). Bylo nutné párovat záznamy podle názvu či ID a řešit případné odlišnosti (např. „Game XYZ“ vs. „Game XYZ: Ultimate Edition“).

```
"Title: Sekiro™: Shadows Die Twice - GOTY Edition\nDeveloper: FromSoftware,\nInc.\nPublisher: Activision (Excluding Japan and Asia)\nFeatures: Single-player, Steam\nAchievements, Steam Trading Cards, Steam Cloud, Remote Play on Tablet, Remote Play on\nTV\nEarly Access: False\nComing Soon: False\nRelease Date: Mar 21, 2019\nGenre: Action,\nAdventure\nPrice: $29.99\nOS: windows\nUrl:\nhttps://store.steampowered.com/app/814380/Sekiro_Shadows_Die_Twice__GOTY_Edition/?snr=1_4_\n4_tab-Specials\nAll Reviews: Very Positive\nRecent Reviews: Very Positive\nSALE:\nTrue\nSale Percentage: -0.5\nSale Until: SPECIAL PROMOTION! Offer ends March\n20\nLanguages: \nDescription: Game of the Year - The Game Awards 2019 Best Action Game of\n2019 - IGN Carve your own clever path to vengeance in the award winning adventure from\ndeveloper FromSoftware, creators of Bloodborne and the Dark Souls series. Take Revenge.\nRestore Your Honor. Kill Ingeniously.\nTags: Souls-like, Difficult, Action, Singleplayer,\nNinja, Ninja, Stealth, Adventure, Third Person, Open World, Story Rich, Violent,\nAtmospheric, Assassin, Dark Fantasy, Hack and Slash, RPG, Gore, Great Soundtrack, Rhythm,\nDark",
```

Ukázka jednoho objektu v JSON připravená k embeddování

Extrakce informací z PDF: Bainův report byl dodán jako PDF dokument, který nebyl strukturovaný pro strojové čtení. Extrahovat text z PDF představuje specifický problém. PDF soubory často obsahují sloupce, grafy, obrázky a nestandardní formátování textu. V tomto případě bylo naštěstí možné získat hlavní text a nadpisy reportu relativně přímo (převodem PDF do prostého textu pomocí nástroje), nicméně i tak bylo třeba text manuálně procházet a kontrolovat. Výzvou bylo rozdělení obsahu na logické celky. Report obsahoval kapitoly o preferencích hráčů, trendech atd., a pro účely RAG bylo vhodné tyto kapitoly rozdělit na menší části (např. jednotlivé odstavce nebo sekce s jednou statistikou). To souvisí s problematikou chunkování (viz níže). Dále bylo nutné přidat metadatové označení, odkud daný text pochází (např. "Bain report 2023") pro případnou identifikaci zdroje odpovědi. PDF dokument také mohl obsahovat speciální znaky či odrážky, které bylo potřeba vyčistit, aby nedělaly problémy při embeddování.

Potřeba vhodné chunkovací strategie: Jednou z nejdůležitějších výzev napříč všemi zdroji dat bylo rozhodnutí, jak velké úseky textu (chunks) ukládat do vektorové databáze. Každý záznam (či dokument) mohl mít různou délku textu. Od jednovětvých údajů (např. žánry, datum vydání) až po celé odstavce (popisy her nebo části reportu). Pro RAG je obecně nevhodné ukládat celý rozsáhlý dokument jako jediný embedding, jelikož by to snižovalo přesnost vyhledávání relevantních informací (Wang et al., 2024). Pokud jsou chunky příliš velké, mohou obsahovat mnoho nesouvisejících informací a vektorová podobnost pak hůře vystihne relevanci k dotazu. Naopak příliš malé chunky (např. jednotlivé věty) mohou ztrácet kontext a také vést k vynechání důležitých souvislostí (Wang et al., 2024). Optimalizace velikosti chunku je tedy kritická pro úspěch RAG systému (Choi et al., 2025). V této práci bylo potřeba empiricky nalézt vhodný kompromis:

pro popisy her ze Steamu/IGDB se jako jednotka chunku nabízel popis jedné hry. Bylo rozhodnuto ponechat jeden chunk = jedna hra, obsahující sloučený popis a klíčové údaje o hře, protože typické herní popisy byly v řádu stovek slov, což je zvládnutelné. Naproti tomu Bainův report bylo nutné rozsegmentovat na více chunků, zde byly chunky tvořeny jednotlivými odstavci či bullet-pointy z reportu, obvykle o délce několika vět. Při chunkování se také přihlíželo ke kontextové kapacitě modelu GPT-4, který v základní verzi pojme vstup o délce cca 8 192 tokenů (Lewis et al., 2020). Cílem bylo, aby i v nejhorším případě (kdy by bylo k odpovědi přiloženo více chunků) zůstala výsledná prompt pod tímto limitem. Zvolená strategie proto typicky používala chunky o délce 50–200 slov pro Bainův report a 100–150 slov pro popisy her. Tím se zajistilo, že jednotlivé chunky pokrývají ucelenou informaci (např. popis hry, nebo jednu statistiku z reportu), a zároveň že vyhledávání podle embeddingů bude dostatečně granulární (najde konkrétní relevantní hru nebo fakt z textu). Tento postup odpovídá doporučením v literatuře, která zdůrazňuje význam vhodné granularity znalostní báze pro dosažení co nejpřesnější shody dotazu s dokumentem (Wang et al., 2024).

3.4.1 Zpracování a uložení dat pro potřeby chatbota

Po získání a očištění všech dat následoval proces přípravy znalostní databáze pro integraci do RAG pipeline chatbota. Cílem bylo převést různorodá data do jednotného formátu, vytvořit jejich vektorové reprezentace (embeddings) a uložit je do vektorové databáze, odkud mohou být efektivně vyhledávány. Kromě toho bylo nutné napojit tuto databázi na samotného chatbota způsobem, který nevyžaduje zásah do jeho modelu (LLM), ale využívá jeho schopnost pracovat s poskytnutým kontextem.

Embedování přes model ADA-002: Ke konverzi textových dat do vektorové podoby byl použit embedovací model text-embedding-ada-002 od OpenAI. Jde o pokročilý model neuronové sítě trénovaný speciálně pro získávání semantických embeddingů textu. Ačkoli starší a lehce dražší na tokeny. V rámci tohoto projektu nepředstavoval embedovací model veliké rozdíly ve výstupech a zachycení vektorové podobnosti.

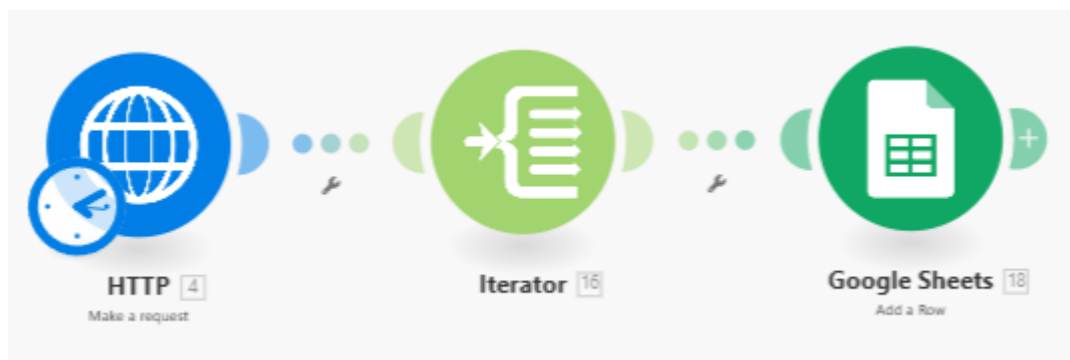
Jeho výstupem je vektor o 1536 rozměrech, který zachycuje význam vstupního textu (Wang et al., 2024). Model ADA-002 byl zvolen z několika důvodů: (1) Je dostatečně pokročilý z dostupných embedovacích modelech z hlediska kvality vektorových reprezentací (ve srovnání s předchozími verzemi dosahuje vyšší přesnosti při vyhledávání významově podobných textů), (2) je optimalizován i pro větší vstupy (má kontextový limit v řádu tisíců tokenů, takže si poradí i s delšími popisy her) a (3) je dostupný přes API za přijatelných podmínek, což umožnilo zpracovat stovky až tisíce chunků. Pro každý připravený chunk textu (řádek v tabulce) byl tedy voláním OpenAI API vygenerován embedding vektor. Tento proces byl automatizován vlastním skriptem, který iteroval přes všechny záznamy, posílal jejich text modelu ADA-002 a ukládal získané vektory. K embeddingu bylo případně doplněno i základní metadata, např. identifikátor hry nebo části dokumentu, abychom neztratili vazbu mezi vektorem a původním textem.

Uložení embeddingů do Pinecone: Výsledné embeddingy bylo nutné uložit způsobem, který umožní efektivní podobnostní vyhledávání. K tomu byla využita cloudová vektorová databáze Pinecone. Pinecone je specializovaná služba určená pro vyhledávání

nejbližších sousedů (nearest neighbor search) ve velkých množinách vektorů, optimalizovaná pro rychlé dotazy na podobnost (CorerMaximus, 2020). V kontextu RAG Pinecone slouží jako úložiště znalostí: každému uloženému vektoru (embeddingu chunku textu) jsou přiřazena metadata (např. ID, název hry či zdroj dokumentu) a Pinecone nad těmito vektory umožňuje provádět dotazy typu „najdi n vektorů nejblížeš zadanému dotazu“. V praxi byly embeddingy nahrány do indexu v Pinecone pomocí API této služby. Skript po získání embeddingu rovnou volal Pinecone API pro upsert (vložit/aktualizovat) záznam s daným vektorem a příslušnými metadaty. Po dokončení tohoto procesu obsahovala vektorová databáze Pinecone všechny znalostní chunky připravené k vyhledávání. Databáze byla konfigurována tak, aby využívala metriky kosinové podobnosti mezi vektory (což je běžná volba pro porovnávání embeddingů významu) a aby uchovávala i zmíněná metadata pro vrácení textu.

Výsledkem výše popsaného procesu je znalostní databáze optimalizovaná pro RAG: obsahuje různorodé informace (jak narativní texty z reportu, tak základní data o hrách) převedené do jednotné vektorové reprezentace. Při dotazu uživatele chatbot nejprve použije retriever nad touto databází a dotaz je rovněž převeden na embedding a jsou vyhledány nejblíže podobnost sémantického významu. Tyto nalezené pasáže (typicky 3–5 nejrelevantnějších segmentů) jsou pak předány jazykovému modelu společně s dotazem, čímž model získá aktuální znalosti potřebné k formulování odpovědi. Tím se výrazně zvyšuje faktická správnost a aktuálnost odpovědi oproti stavu, kdy by model spoléhal jen na své parametry (Lewis et al., 2020). Důležité je, že kvalita tohoto procesu závisí na kvalitě a struktuře dat v databázi a pokud by data byla nesprávně naformátována nebo by chyběl podstatný kontext, retriever by nemusel najít relevantní informace. V této kapitole je popsáno, jak byly data pečlivě získány, vyčištěny a uloženy tak, aby retriever dokázal efektivně plnit svou úlohu. Díky tomu může výsledný chatbot kombinovat to nejlepší z obou světů: rozsáhlé jazykové schopnosti LLM a přesné znalosti z aktuální databáze.

Ilustrační ukázka IGDB scrapingu:



Zdroj: vlastní zpracování v rámci Make.com

http modul s url nastavený na: <https://api.igdb.com/v4/games> s metodou POST se autorizuje s API klíčem dostupný na webových stránkách twitch developeru po založení vývojářského účtu. V header modulu bylo nastaveno také, že se jedná o JSON formát a v request body bylo zadáno dostupné informace, které API poskytuje v podobné formě jako: *fields name, summary, rating, platforms.name, first_release_date, genres.name, cover.url; where rating > 80; limit 250;*

3.5 Možnosti propojení Make.com s RAG databází a chatbotem

Velkou výhodou platformy Make.com je možnost integrovat různé služby bez nutnosti programování. A to formou vizuálního workflow (*scenario*). V rámci návrhu našeho systému byl využit Make.com k automatizaci všech kroků RAG pipeline. Jednotlivé moduly Make scénáře zajišťují sekvenční provedení úloh od získání dat až po volání API služeb a zpracování odpovědí.

Integrace Pinecone pomocí Make: Make.com nabízí jak dedikovaný modul pro Pinecone, tak univerzální HTTP modul. Pro flexibilitu byly využity převážně HTTP moduly, které umožňují ručně volat REST API Pinecone pro operace upsert a query. Bylo potřeba správně formátovat data ve formátu JSON, aby Pinecone API porozumělo požadavkům. Zejména v případě vektorových embeddingů (pole desítek či stovek čísel) bylo nutné převést interní reprezentaci pole na JSON string. K tomu Make poskytuje transformační modul `JSON > Text`, který zajistí escapování speciálních znaků a správné vložení pole do textového JSON formátu (*Pinecone Api Call, Embedding Is a Array but When Put into Http It's a String - How To*, 2025). Například embedding o dimenzi 1536 (typické pro model *ada-002*) byl nejprve pomocí modulu *Transform to JSON* převeden na text, který pak tvořil hodnotu atributu "values" při volání Pinecone upsert API. Díky tomu Pinecone obdržel celý vektor a mohl jej uložit.

Propojení chatbota (OpenAI) přes Make: Samotný chatbot byl realizován opět pomocí Make scénáře, který čekal na příchozí dotaz (např. modul *Webhooks* přijímající požadavky z webové aplikace chatbota). Jakmile dotaz dorazil, scénář spustil modul OpenAI (Create embedding) přes http požadavek k převedení uživatelského dotazu na embedding. Tento vektor (přenesený jako JSON pole) byl vložen do dotazu modulu HTTP (Pinecone Query), který volal Pinecone API pro vyhledání nejbližších vektorů (např. s parametrem `topK=5` pro získání 5 nejrelevantnějších úryvků). Pinecone vrátil seznam výsledků v JSON formátu, který obsahoval původní uložené texty a jejich skóre podobnosti. Následoval klíčový transformační krok: výstup Pinecone bylo třeba připravit pro jazykový model. K tomu Make využil modul *Iterator* nebo *Array aggregator* k sesbírání top výsledků a složení textového kontextu. Z výsledků se typicky sestaví prompt, který poskytuje samotný dotaz a získaný kontext z vektorové databáze se samotným jednoduchým promptem, který říká, co má chatbot udělat s kontextem. Takový prompt byl následně předán modulu OpenAI (Chat completion), který prostřednictvím GPT-4 vygeneroval finální odpověď pro uživatele.

- Pomocí předpřipravených modulů dokázal komunikovat s OpenAI i Pinecone Případně přes obecné HTTP moduly tam, kde specializovaný modul nebyl k dispozici nebo nevyhovoval.
- Umožnil aplikovat vlastní logiku formátování dat (skládání JSON dotazů, úprava textů) pomocí vizuálních bloků pro transformace.
- Zajistil, že výstupy z jednoho kroku (embedding, vyhledané pasáže) se správně přenesou jako vstupy do dalšího kroku.

Díky tomuto no-code přístupu bylo možné celý RAG systém integrovat bez implementace backendového serveru. Make.com scénář zastoupil roli aplikační logiky. To výrazně urychlilo vývoj prototypu. Zároveň je řešení modulární a lze snadno měnit parametry (např. jiný gpt model, prompt či hodnotu *topK*) přímo v rozhraní Make. Nevýhodou může být omezená flexibilita oproti plnému programování (např. složitější logika by byla v Make komplikovaná) a závislost na externích službách (Make, Pinecone, OpenAI). Nicméně v kontextu této práce bylo cílem ukázat možnost rychlé integrace RAG principu s minimem kódování.

3.5.1 Konkrétní řešení systému

Navržený chatbot využívá architekturu konverzačního systému s rozšířenou znalostní základnou (RAG). Uživatel komunikuje s chatbotem formou přirozeného jazyka, přičemž každému dotazu je přiřazena sekvence operací popsanych výše. Chatbot tak představuje nadstavbu nad vektorovou databází:

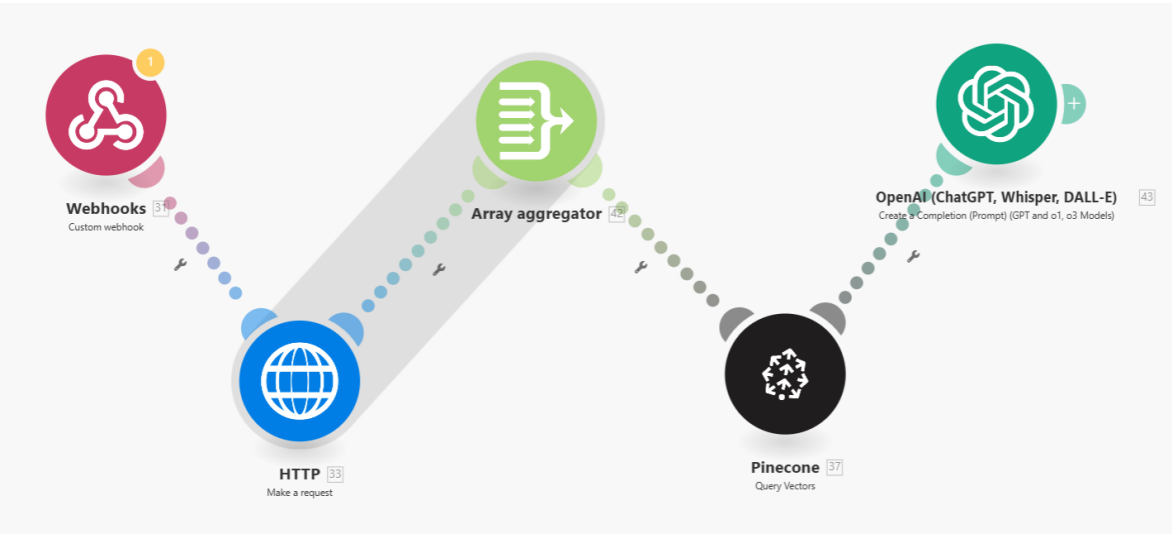
1. Přijme dotaz od uživatele (textovou otázku nebo větu).
2. Vytvoří embedding dotazu stejným modelem využitým na znalostní bázi.
3. Embedding dotazu se odešle do Pinecone a získá nejpodobnější uložené chunky.
4. Sestaví rozšířený prompt obsahující nalezené informace jako kontext.
5. Vygeneruje odpověď pomocí LLM (GPT-4) a předá ji uživateli.

Embedovací model a indexace ve Pinecone: Pro převod textů na vektory byl použit model *text-embedding-ada-002* od OpenAI, který generuje 1536rozměrné husté vektory reprezentující význam textu (Walsh, 2024). Tento model je trénovaný na obecném jazyce a poskytuje kvalitní semantické embeddingy, vhodné pro vyhledávání podobností.

Vektorová databáze Pinecone byla nastavena s metrikou podobnosti cosine a dimenzí indexu 1536 tak, aby odpovídala rozměru používaných embeddingů. Při vkládání dat byl každému vektoru přiřazen identifikátor a metadata označující zdroj (Bain report, IGDB, Steam), což nám při vyhodnocování umožnilo ověřit, odkud čerpal poskytnutý kontext.

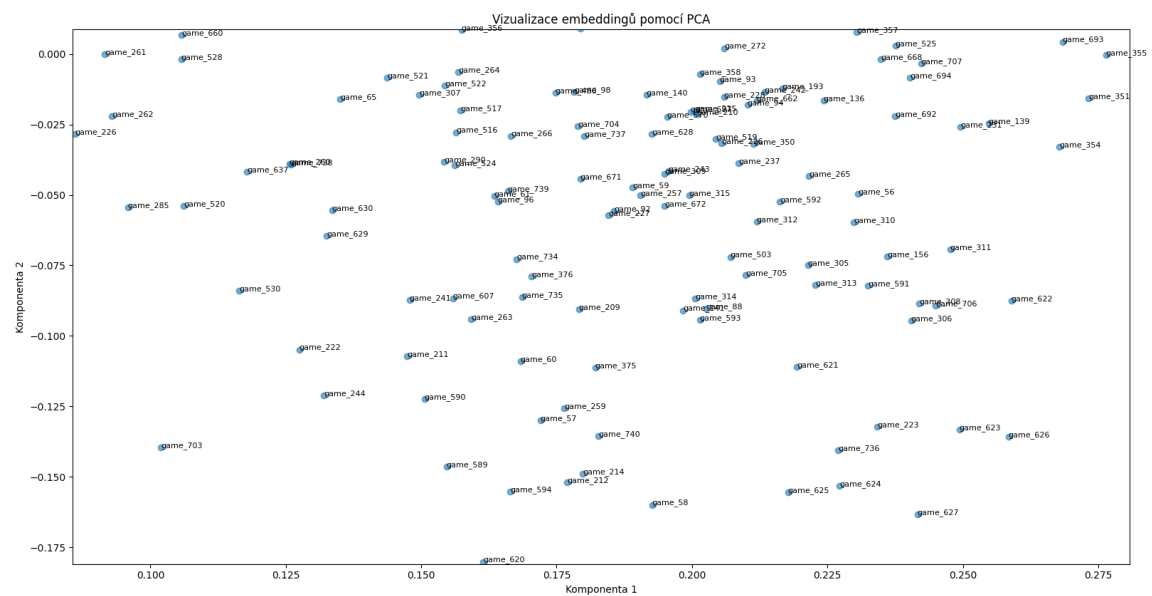
Při dotazování byl např. zvolen $k = 5$ (vracet 5 nejbližších pasáží) jako kompromis mezi pokrytím relevantních informací a stručností kontextu. Embedding model dokáže zachytit i jemnou podobnost, proto i pokud dotaz uživatele nebyl formulován úplně přesně, vektorové vyhledávání mělo šanci najít související obsah.

Finální řešení systému v Make.com:



Zdroj: vlastní zpracování v rámci Make.com

Ukázková vizualizace kosinusové podobnosti záznamů z her:



Zdroj: vlastní zpracování

4 Měření relevance odpovědí

Retrieval-Augmented Generation (RAG) kombinuje vyhledávání relevantních podkladů s generováním odpovědí pomocí jazykového modelu. V našem projektu, který propojuje web scraping (pomocí platformy Make.com) s dotazováním LLM chatbota, je klíčové hodnotit, nakolik jsou generované odpovědi relevantní a správné vzhledem k uživatelskému dotazu i dostupným podkladům. Tato kapitola přehledně popisuje metriky pro dvě hlavní fáze RAG systému. Fázi vyhledávání (*retrieval*) a fázi generování odpovědi (*generation*). Dále představuje metodiku vyhodnocení použité v projektu. Veškeré metriky jsou vysvětleny s ohledem na jejich výpočet, smysl a limity v kontextu vyhledávání pomocí vektorových embeddingů ve vektorové databázi typu Pinecone a generování odpovědí jazykovým modelem.

4.1.1 Testovací sada dotazů

Testovací sada obsahuje celkem 40 dotazů za účelem otestování relevance odpovědí. Tyto dotazy byly rovnoměrně rozděleny do dvou skupin po 20. První skupinu tvoří obecné dotazy v přirozeném anglickém jazyce. V druhé skupině se v dotazích nachází klíčová slova, která může umožnit embeddingové podobnosti odvozené z chunků. Každý dotaz byl navržen tak, aby existovala určitá „správná“ odpověď. Ne nutně jednovětná, může jít i o celý kontextový odstavec informací. Tyto očekávané odpovědi byly připraveny jako referenční odpovědi (ground truth) pro účely automatizovaného vyhodnocení. V praxi referenční odpověď představuje buď extrakt z příslušného dokumentu, nebo rukou formulovanou správnou odpověď na danou otázku na základě dostupných dat.

Rozdělení na dvě skupiny dotazů nám při vyhodnocení umožní zjistit, zda RAG kontext pomáhá více u určitého typu dotazů. Předpoklad byl, že klíčová slova (druhá skupina) budou bez poskytnutého kontextu pro GPT-4 obtížná a odpovědi mohou být nepřesné nebo obecné, zatímco s kontextem se značně zlepší. Naproti tomu obecné dotazy mohla GPT-4 do určité míry zodpovědět i samostatně, ale i tak je očekáváno, že kontext zaručí přesnost tam, kde by model mohl halucinovat nebo být nejistý. Cílem rozdělení dotazů do dvou skupin bylo zajistit spravedlivé vyhodnocení z důvodu případu užití této RAG pipeline a povahu dat, které jsou spíše strukturované povahy.

4.1.2 Struktura dotazů a kategorizace (obecné vs. klíčová slova)

Při přípravě testovacích dotazů rozdělení proběhlo na dvě hlavní kategorie, pro něž mohly platit odlišné optimální přístupy promptování:

- **Obecné dotazy v přirozeném jazyce:** Sem patří plnohodnotné otázky nebo požadavky formulované běžným jazykem, často v tázacím tvaru. Například: „Nalezni mi hru s akčním příběhem a s pozitivním hodnocením?“ nebo „Najdi mi všechny Simulator hry, které mají metacritic hodnocení 90?“. Tyto dotazy jsou blízké reálným uživatelským dotazům. U nich se očekává, že GPT-4 bez kontextu

odpoví podle svých znalostí, či vyhledá s funkcí websearch, případně že s dodaným kontextem přímo využije relevantní informace. Promptování u těchto dotazů bylo přímočaré. Buď byla zadána otázka rovnou modelu, nebo před ni připojeny nalezené texty.

- **Dotazy tvořené klíčovými slovy v „chuncích“:** Do druhé kategorie byly zařazeny dotazy, které nebyly formulovány jako dotaz běžného uživatele, ale spíše jako uživatele, který zná obsah databáze a vyhledává upřesnění. Například mohlo jít o dotaz typu: „*What is the VR puzzle game set on a mysterious island called Myst, reimagined by Cyan for both VR and flatscreen PC?*“ nebo „*Which PlayStation platformer features Claymation characters fighting Skullmonkeys to stop the evil Klogg?*“. Tyto dotazy nejsou něco, co by běžný uživatel položil, ale slouží k otestování, jak systém (zejména část vyhledávací) pracuje, pokud je dotaz přesně navázán na obsah chunků. V podstatě simulují scénář, kdy uživatel zadá velmi stručný dotaz (třeba pomocí několika klíčových slov). Pro GPT-4 bez kontextu jsou takové dotazy komplikovanější a model nemusí správně porozumět dotazu. Dá se očekávat, že bez dodatečných instrukcí by model mohl generovat obecnou odpověď nebo tápat. Naopak s RAG kontextem, pokud vyhledávání vrátí relevantní chunk, dokáže model z kontextu pochopit dotaz a zformulovat odpověď.

Rozdělení dotazů na obecné a klíčová slova umožňuje vyhodnotit, zda RAG přístup přináší větší zlepšení u určitého typu dotazů a správný případ užití, v našem případě spíše vyhledávání z databáze. Předpokládá se, že dotazy tvořené pouze klíčovými slovy budou výrazně lépe zodpovězeny s pomocí RAG (kdy kontext „rozvine“ zadání), zatímco u obecných dotazů může být rozdíl menší. Toto rozlišení také reflektuje dva typy vyhledávacích scénářů: jeden, kdy uživatel formuluje dotaz sám (přirozeně), a druhý, kdy dotaz může pocházet ze systému (např. interní dotazy generované pro potřeby prohledání databáze).

Shrnutí metod promptování: V experimentu tedy je použita konzistentní dvojice promptů pro každý dotaz: jeden bez kontextu (pouze dotaz na GPT-4) a druhý s kontextem (dotaz + nalezené chunky). Dotazy samotné jsou dvou typů (obecné vs. klíčová slova), což je zohledněno při interpretaci výsledků. Taková konfigurace poskytne široký pohled na výkon systému. Jak si GPT-4 vede samostatně, jak s podporou znalostní databáze, a to v různých situacích formulace dotazu.

První sada (Obecné dotazy)	Druhá sada (Klíčová slova v dotazu)
Find me games that have 'Overwhelmingly Positive' reviews on Steam and cost less than 10 USD.	Which game features Tilo, a mouse minstrel, sneaking through Dwindling Heights Keep using stealth mechanics?
Which game is available on Oculus VR?	What is the pixel-style turn-based dungeon crawler where players attack by rolling dice and can choose from 35 unique classes?
Find me games that have a 'Mixed' review status on Steam and are priced around 70 USD.	What World War II strategy game by Paradox Interactive features cross-platform multiplayer and grand strategy mechanics?
Find the most expensive games in the database that have a mixed review status.	What anime RPG features a magical girl named Konoha entering the world of "Grimoire" to save it, released in March 2025?

Ilustrační ukázky dotazů první a druhé sady použité k vyhodnocení RAG systému

4.1.3 Metriky pro hodnocení fáze vyhledávání (retrieval)

Ve fázi vyhledávání jde o to, jak dobře systém dokáže z množiny dokumentů či úryvků získat ty, které obsahují odpověď na daný dotaz. K hodnocení se používají klasické metriky informačního vyhledávání, přizpůsobené pro hodnocení top-K výsledků. Níže je uvedeno tři často používané metriky: Precision@K, Recall @K, Hit Rate@K a Mean Reciprocal Rank (MRR).

- **Precision@K** vyjadřuje podíl relevantních dokumentů mezi prvními K výsledky vyhledávání. Formálně se Precision@K počítá jako počet relevantních dokumentů v top K dělený číslem K. Například Precision@10 = 0,7 znamená, že ze 10 vrácených dokumentů je 7 relevantních. Tato metrika hodnotí, jak přesné je vyhledávání. Tedy jak velký podíl výsledků je skutečně správný. Precision@K je velmi užitečná při posuzování „čistoty“ výsledků: vysoká hodnota signalizuje, že mezi vrácenými dokumenty je málo nerelevantních položek, což je zvláště důležité u generativních systémů, kde šum v kontextu může negativně ovlivnit kvalitu

odpovědi. V systémech embed-based retrieval, jako je použitý v této práci, vyžaduje Precision@K pevně definované binární označení dokumentů jako relevantních či nerelevantních, což může být v praxi komplikované, pokud více dokumentů obsahuje různé správné informace. (Samuylova, 2025).

- **Recall@K** měří, jak velkou část všech existujících relevantních dokumentů se podařilo systému najít mezi prvními K výsledky. Výpočet je založen na počtu nalezených relevantních dokumentů v top K dělený celkovým počtem existujících relevantních dokumentů pro daný dotaz. Například pokud existují tři relevantní dokumenty a systém najde v top-5 dva z nich, Recall@5 je $2/3$ neboli 0,67. Recall@K tedy hodnotí úplnost vyhledávání a ukazuje, zda systém zachytil většinu nebo všechny správné odpovědi mezi vrácenými výsledky. Výhodou Recall@K je jeho citlivost (Diaz et al., 2024) na rozdíl od binárních metrik, jako je Hit Rate, rozlišuje mezi částečným a úplným pokrytím relevantních dokumentů. Nevýhodou je, že Recall@K neposuzuje pořadí nalezených výsledků. Zda byly relevantní dokumenty vráceny na prvních místech, nebo až na konci. Pokud má dotaz pouze jednu správnou odpověď, hodnoty Recall@K a Hit Rate@K budou numericky stejné. Ve složitějších případech, kdy existuje více správných dokumentů, je však Recall@K daleko informativnější. V embed-based retrieval systémech je Recall@K zvláště důležitý, protože vyšší Recall@K znamená, že generativní model dostane dostatek správných podkladů pro kvalitní odpověď. (Samuylova, 2025).
- **Hit Rate@K** hodnotí, zda se v top K výsledcích vůbec nějaký relevantní dokument vyskytuje. Je to binární metrika. Buď ano, nebo ne (Wu et al., 2024). Výsledná hodnota představuje podíl dotazů, kde byl zásah zaznamenán. Tato metrika je jednoduchá a užitečná při nastavování optimální hodnoty K. Nevýhodou je, že neposuzuje, kolik relevantních dokumentů bylo nalezeno ani jejich pořadí. Výhodou Hit Rate je jednoduchost a přehlednost. Dává jedno číslo označující úspěšnost vyhledávání. Lze ji využít např. při nastavování, kolik dokumentů K předávat do modelu. Je sledováno, pro jaké K už dosahuje téměř 100% pokrytí všech dotazů, aniž by bylo přidáváno zbytečně mnoho nerelevantních vstupů (Wu et al., 2024). Nevýhodou je, že nepostihuje kvalitu řazení ani počet relevantních výsledků. Pouze zda nastal alespoň jeden „zásah“ (Yu et al., 2025). V embed-based vyhledávání navíc nastává otázka, co je považováno za relevantní: pokud má dotaz více možných správných odpovědí (z různých dokumentů), Hit Rate je schopna to zohlednit (stačí jakákoliv z nich v top-K), avšak vyžaduje předem definovanou množinu relevantních dokumentů pro každý dotaz.
- **Mean Reciprocal Rank (MRR)**: MRR je metrika zaměřená na pozici první správné odpovědi. Mean Reciprocal Rank se definuje jako průměr převrácených pořadí prvního relevantního dokumentu napříč všemi dotazy (Sadri, 2022). MRR měří průměrné pořadí první relevantní odpovědi. Reciproční hodnota pořadí se počítá tak, že pokud je správná odpověď na první pozici, hodnota je 1, pokud na druhé pozici, hodnota je 0,5, na třetí pozici 0,33 a tak dále. Výsledná MRR je průměr těchto hodnot napříč všemi dotazy. Výhodou MRR je, že odměňuje systémy, které umísťují relevantní dokumenty na přední místa. Naopak jeho omezením je, že bere v úvahu pouze první nalezenou relevantní položku a ignoruje

další relevantní odpovědi, což může být nevýhodné v případě dotazů, kde je správných odpovědí více. Výhodou MRR je, že silně penalizuje případy, kdy je správná odpověď hluboko v seznamu (např. rank 10 dá příspěvek jen 0,1), a odměňuje systémy, které umísťují relevantní informaci co nejvýše (Sadri, 2022). Tím dobře reflektuje uživatelskou zkušenost v QA úlohách, kde typicky se očekává správná odpověď na prvních místech. Omezením MRR je, že bere v potaz jen první nalezenou relevantní položku a ignoruje případné další relevantní dokumenty. Pokud tedy systém najde více správných podkladů, MRR to neoceňuje (počítá se stále jen rank toho prvního). V kontextu RAG může nastat, že model využije i dokument na druhé či třetí pozici. MRR však hodnotí pouze pozici prvního relevantního. Proto MRR nemusí přímo korelovat s kvalitou finální odpovědi v případě, že se využívá více podkladů. Důležité spíše je, aby alespoň jeden z top-K dokumentů obsahoval požadovanou informaci. Z tohoto důvodu se MRR často používá spolu s metrikou typu Hit Rate@K (která ověřuje právě „alespoň jeden“ správný dokument v top-K). V praxi otevřených QA úloh (open-domain QA) je MRR standardní metrikou – např. v benchmarku MS MARCO (Sadri, 2022).

Limity hodnocení retrievalu v embed-based vyhledávání: Uvedené metriky předpokládají binární rozhodnutí o relevanci dokumentu (relevantní vs. nerelevantní) a existenci „ideální“ množiny relevantních výsledků pro každý dotaz. To může být v praxi obtížné přesně určit. Zvláště pokud model pracuje se vektorovými reprezentacemi a nachází i nepřesná, avšak použitelná spojení. Navíc kvalitativní aspekty jako částečná relevance (např. dokument obsahuje jen dílčí informaci) nebo informační redundance nejsou těmito metrikami zachyceny. Pro pokročilejší hodnocení relevance se někdy používají odstupňované metriky jako NDCG (Normalized Discounted Cumulative Gain) (Yu et al., 2025), které umí rozlišit úroveň relevance jednotlivých dokumentů a zohlednit jejich pořadí. V našem kontextu (jedna správná odpověď na dotaz) bylo však hodnocení zjednodušen na binární relevanci a metriky výše nám poskytují dostačující obraz o výkonu vyhledávání.

4.1.4 Vyhodnocení retrieval fáze pomocí metrik

Pro ověření účinnosti vyhledávací komponenty systému RAG byla provedena kvantitativní analýza na dvou testovacích sadách dotazů (každá obsahovala 20 dotazů). Cílem bylo zjistit, jak dobře dokáže embedovací vyhledávač (na základě vektorové podobnosti) nalézt úryvky textu, které obsahují správnou odpověď. Pro každý dotaz byly předem definovány správné dokumenty (tzv. *ground truth*) a systém vrací seznam pěti nejpodobnějších dokumentů.

Vyhodnocení probíhalo pomocí následujících metrik:

- **Precision@5** – podíl relevantních dokumentů mezi pěti vrácenými,
- **Recall@5** – podíl nalezených relevantních dokumentů vůči všem existujícím relevantním,
- **MRR** (Mean Reciprocal Rank) – vážené skóre podle pozice první správné odpovědi,

- **Hit@1 / Hit@3 / Hit@5** – binární metriky určující, zda se mezi prvních 1/3/5 vyskytla správná odpověď.

Metrika	První sada (1-20)	Druhá sada (21-40)	Celkový průměr
Precision@5	0.38	0.86	0.62
Recall@5	0.40	1.00	0.70
MRR	0.32	1.00	0.66
Hit@1	0.20	1.00	0.60
Hit@3	0.45	1.00	0.73
Hit@5	0.50	1.00	0.75

Z výsledků je patrné, že retrieval systém fungoval velmi dobře u druhé sady dotazů, zatímco v první sadě byly výsledky výrazně slabší. Důvodem může být rozdíl v typu dotazů:

- Druhá sada obsahovala konkrétní a dobře definovatelné dotazy (např. název hry, přesný žánr, známé herní postavy nebo specifická fakta), což vyhledávači usnadnilo najít odpovídající dokumenty.
- První sada byla tvořena obecnějšími nebo vícekritériálními dotazy (např. „najdi hry s mixed recenzemi a cenou kolem 70 USD“), u kterých je správná odpověď obtížnější zachytit pomocí embeddingové podobnosti.

Navzdory nižšímu výkonu v první sadě byly i zde zaznamenány případy úspěšného vyhledání. Například dotaz na konkrétní VR hru či titul s postavou jménem Henry byl zodpovězen přesně. Vyšší průměrná hodnota Hit@5 (0.75) napovídá, že ve třech ze čtyř případů byl alespoň jeden správný dokument mezi pěti nalezenými, což je pro RAG systémy zásadní, protože jazykový model dokáže následně odpověď správně složit i z jednoho relevantního úryvku.

4.2 Metriky pro hodnocení fáze generování odpovědi

Hodnocení kvality samotné generované odpovědi je ještě náročnější úkol, protože je třeba posuzovat správnost a relevance volného textu. Existuje celá řada metrik pro porovnání generované odpovědi s referenční (očekávanou) odpovědí. Některé pocházejí z oblasti strojového překladu a měří shodu na úrovni slovních n-gramů (např. BLEU, METEOR), jiné využívají sémantické modely k posouzení významové blízkosti (např. BERTScore), případně zapojují přímo LLM model jako hodnotitele (LLM-as-a-Judge). Dále existují specializované rámce pro hodnocení RAG odpovědí, které kombinují více kritérií (např. framework RAGAS). Níže charakterizovány vybrané metriky a vysvětleny, proč se některé

hodí spíše pro úlohy jako strojový překlad, zatímco jiné lépe měří sémantickou shodu odpovědi s pravdou. Zaměření bude na metriky použité v naší práci: BERTScore a hodnocení pomocí LLM (LLM-as-a-Judge).

- **BLEU** (Bilingual Evaluation Understudy): BLEU je tradiční metrika pro automatické hodnocení kvality strojového překladu. Využívá n -gramovou shodu mezi generovaným textem a jedním či více referenčními texty (Herreros et al., 2023). Počítá, jaký podíl n -gramů (sekvencí slov) z překladu se vyskytuje v referenci, přičemž kombinuje skóre pro různá n (typicky 1–4) a zavádí penalizaci za příliš krátké výstupy (brevity penalty) (Yu et al., 2025). BLEU se udává v rozmezí 0 až 1 (často v procentech 0–100). Vyšší hodnoty značí větší podobnost s referencí. Výhodou BLEU je jeho jednoduchost a nezávislost na lidském zásahu, díky čemuž se stal oblíbeným pro rychlé porovnání modelů. Nevýhodou je, že vysoké skóre neznamená bezpodmínečně správnou či plynulou odpověď. BLEU například nebere v potaz jazykovou plynulost ani gramatickou správnost (Yu et al., 2025). Kriticky také nerozlišuje význam, pouze shodu slovních sekvencí. V kontextu RAG QA to znamená, že model může dostat nízké BLEU skóre, i když odpoví správně, pokud použije jiné formulace než reference. Studie ukázaly, že korelace BLEU s lidským hodnocením odpovědí na otázky je často nízká (Herreros et al., 2023). Proto se BLEU hodí spíše pro úlohy, kde existuje jediná kanonická formulace (strojový překlad, strukturálně definované výstupy), ale pro hodnocení otevřených odpovědí v QA má omezenou vypovídací hodnotu.
- **METEOR** (Metric for Evaluation of Translation with Explicit ORdering): METEOR je metrika navržená jako reakce na nedostatky BLEU, primárně pro hodnocení strojového překladu. Využívá rovněž porovnání generovaného a referenčního textu na úrovni slov, avšak počítá harmonický průměr přesnosti a úplnosti (precision a recall) namísto čisté přesnosti jako BLEU (Herreros et al., 2023). Zároveň METEOR zohledňuje synonyma a ohýbání slov (stemming), tedy dokáže rozpoznat shodu, i pokud model použil jiné tvary či synonyma slov z reference. Dále zavádí penalizaci za nesouvislé úseky (fragmentace), čímž zohlední i slovosled (Herreros et al., 2023). Výsledné skóre METEOR je normalizováno opět do intervalu 0–1. METEOR typicky dosahuje vyšší korelace s lidským hodnocením než BLEU (Herreros et al., 2023), protože je citlivější na významovou ekvivalenci (díky synonymům) a méně penalizuje odlišnou formulaci. Přesto i METEOR trpí omezeními n -gramových metrik obecně. Například nedokáže plně postihnout celkovou sémantickou správnost textu. Pro RAG odpovědi, kde jedno a totéž sdělení může být vyjádřeno různými slovy, poskytne METEOR o něco shovívavější hodnocení než BLEU, ale stále nemusí odhalit případné faktické nesprávnosti či halucinace v textu (Balaji & Kariaden Parambath, 2025).
- **BERTScore**: BERTScore představuje novější přístup k hodnocení generovaného textu, který využívá modely typu BERT k posouzení sémantické podobnosti mezi odpovědí a referencí (Yu et al., 2025). Na rozdíl od n -gram metrik se BERTScore nedívá jen na přesnou shodu slov, ale převádí věty do kontextových embeddingů (vektorových reprezentací) pomocí trénovaného Transformer modelu (např. BERT). Následně pro každý token (slovo) v odpovědi hledá nejbližší podobný

token v referenci (na základě kosinusové podobnosti vektorů) a vypočítá průměrnou podobnost. To lze chápat jako recall generované věty vůči referenci v prostoru významu (Herreros et al., 2023). Analogicky lze spočítat i „přesnost“ a kombinací získat F1 skóre. Typicky se však uvádí přímo BERTScore F1. Klíčové je, že BERTScore zachycuje význam slov v kontextu. Je odolné vůči parafrázím a změně slovosledu (Yu et al., 2025). Výsledkem je skóre 0–1, které lépe koresponduje s tím, zda dvě věty nesou stejný význam, i když nejsou lexikálně totožné. Studie potvrzují, že BERTScore dosahuje vyšší shody s lidskými hodnoceními než BLEU u různých generativních úloh (Yu et al., 2025). V našem hodnocení RAG odpovědí využíváme BERTScore k měření, nakolik se odpověď modelu shoduje s referenční správnou odpovědí obsahově. Referencí může být například text z dokumentu, z něhož pochází ground truth informace. BERTScore umožňuje odhalit, zda model do své odpovědi zahrnul správná fakta. Vysoký BERTScore indikuje, že se odpověď významově velmi kryje s referencí (buť jinými slovy), zatímco nízký může značit vynechání podstatných informací či přidání irelevantního obsahu.

- **RAGAS: Retrieval-Augmented Generation Assessment** (RAGAS) je rámec navržený speciálně pro hodnocení systémů RAG. Jedná se o sadu metrik a postupů, které kombinují jak automatizované porovnávání textů, tak vyhodnocení pomocí LLM, s cílem komplexně posoudit kvalitu odpovědi z hlediska správnosti a využití (Yu et al., 2025). Autoři RAGAS (Es et al., 2023) definovali několik dílčích kritérií, zejména: (1) Answer Relevance – relevanci odpovědi vzhledem k otázce, (2) Context Precision/Recall – kolik relevantních informací z podkladů bylo použito, resp. zda odpovědi nechybí něco důležitého z podkladů, a (3) Faithfulness – faktickou správnost a ukotvenost odpovědi v podkladech (tj. míru „halucinací“) (Es et al., 2023). K vyhodnocení těchto aspektů používá RAGAS předpřipravené prompty, kterými se LLM dotazuje na danou odpověď a její vztah k podkladům. Například LLM může být tážán: „Je odpověď plně podložena uvedeným kontextem?“ pro hodnocení faithfulness. Kombinací odpovědí modelu na sadu takových otázek vzniká profil kvality RAG odpovědi. Výhodou RAGAS je komplexnost, který nespolehá jen na jednu metriku, ale posuzuje více rozměrů kvality odpovědi. Navíc využití LLM umožňuje zahrnout i určité „porozumění“ obsahu, což čistě statistické metriky nedokážou. RAGAS byl jedním z prvních rámců pro automatizované hodnocení RAG a byl využit např. v technických QA doménách (Yu et al., 2025). Nevýhodou je složitost a výpočetní náročnost. Vyhodnocení jedné odpovědi může vyžadovat mnoho dotazů do velkého modelu, což je pomalé a nákladné (Yu et al., 2025).

4.3 Metodologie evaluace v projektu

Pro zhodnocení přínosu RAG přístupu byly připraveny testovací sady dotazů a pro každou porovnán výkon systému s kontextem a bez kontextu. Konkrétně byly sestaveny dvě množiny po 20 dotazech reprezentujících typické otázky, na které má systém odpovídat. Každý dotaz byl formulován přirozeným jazykem (anglicky) a byl navržen tak, aby odpověď bylo možné dohledat ve znalostní bázi získané web scrapingem. U každého

dotazu byl manuálně určen ground truth. Tedy který dokument/chunk, který se považuje za správnou odpověď na dotaz. Tento ground truth měl v práci dvě formy. Jedna je retrieved chunk, dle zadané ID v Pinecone indexu. Druhý ground truth byl formou krátké věty jako odpověď na dotaz, který nám poslouží jako reference pro metriku BERTScore. Tyto ground truths sloužily jednak pro vyhodnocení retrievalu, jednak pro sestavení referenčních odpovědí při hodnocení generování.

Postup testování byl následující: Pro každý dotaz systém nejprve pomocí embedding-based vyhledávání v Pinecone získal top-K nejpodobnějších úryvků (vybráno bylo K=5). Nalezené úryvky byly předány LLM chatbotovi jako dodatečný kontext a vygenerována odpověď. Kterou je považována za RAG odpověď. Zároveň byl položen tentýž dotaz modelu bez poskytnutí kontextu, čímž byla získána odpověď bez RAG (model se snažil odpovědět z vlastních znalostí čili často obecněji nebo s rizikem halucinace). Všechny tyto výsledky byly uloženy ve formátu JSON pro následnou analýzu. Každý záznam obsahoval:

- **dotaz** (query),
- **ground truth ID** (identifikátor dokumentu/úryvku považovaného za správný zdroj odpovědi),
- **retrieved_chunks** (seznam vyhledaných úryvků s jejich ID či názvem dokumentu, včetně ranku),
- **odpověď_RAG** (text modelové odpovědi s kontextem),
- **odpověď_bez_RAG** (text odpovědi modelu bez kontextu).

Na takto nasbíraných datech (celkem 40 dotazů, 40 odpovědí s RAG a 40 bez RAG) bylo provedeno vyhodnocení.

Hodnocení generované odpovědi: Pro vyhodnocení kvality odpovědi byla použita kombinace automatické metriky a posouzení LLM. Automaticky byla každá RAG odpověď porovnána s referenční odpovědí (extrahovanou z ground truth dokumentu) pomocí BERTScore. Ten udal skóre sémantické shody odpovědi s očekávaným obsahem. Zároveň byla spočítán BERTScore i pro odpověď bez RAG vůči téže referenci, aby byl rozdíl kvantifikován. Ve výsledku RAG odpovědi dosahovaly podstatně vyšších BERTScore (blížících se 1) než odpovědi bez kontextu, které často obsahovaly nepřesnosti či vágní tvrzení (BERTScore výrazně nižší).

Dále byla využita metoda LLM-as-a-Judge. GPT-4 modelu byla předložena dvojice odpovědí (s kontextem vs. bez) a požádán o verdikt, která je lepší. Tento kvalitativní přístup potvrdil kvantitativní měření. Ve většině případů GPT-4 jednoznačně preferoval odpověď s doplněným kontextem, často s komentářem, že obsahuje konkrétnější či správnější informace. V několika málo případech, kde rozdíl nebyl zřejmý (např. dotaz byl triviální, a tudíž i model bez kontextu odpověděl správně), GPT-4 hodnotil odpovědi jako rovnocenné.

Celkově metodologie evaluace kombinovala objektivní metriky (procenta úspěšnosti vyhledání, skóre podobnosti odpovědí) se subjektivně-objektivním posouzením pomocí LLM. Tento vícevrstvý přístup je v souladu s doporučeními v literatuře: jelikož hodnocení

generovaného textu je mnohorozměrné, je vhodné zkombinovat několik kritérií (Yu et al., 2025). Náš postup umožnil jednak změřit zlepšení, jednak ilustrovat konkrétními příklady, jak RAG kontext pomohl odpovědět na otázky, které by čistý model nezvládl nebo zodpověděl hůře.

4.3.1 Využití LLM-as-a-Judge pro hodnocení relevance a správnosti

Kromě automatického porovnání s referencemi bylo nasazen také velký jazykový model v roli hodnotitele odpovědí. Přístup také známý jako *LLM-as-a-Judge*. Jde o relativně nový, ale rychle se rozvíjející způsob hodnocení, kdy se silnému modelu (typicky stejnému nebo lepšímu, než který generoval odpovědi) předloží dotaz a vygenerované odpovědi a model je instruován, aby je zhodnotil podle zadaných kritérií (např. relevance k dotazu, faktická přesnost, úplnost, stylistická kvalita apod.) (Samuylova, 2025). Výhodou tohoto přístupu je, že není potřeba mít pro každou otázku předem připravenou referenční odpověď. Model/hodnotitel posoudí odpověď podobně jako by to udělal lidský expert.

Tři klíčová kritéria využitá v hodnocení:

- **Relevance k dotazu (relevance):** zda odpověď skutečně odpovídá na položenou otázku a drží se tématu.
- **Faktická správnost (correctness):** zda jsou uvedené informace správné podle známých faktů či poskytnutého kontextu (pokud byl k dispozici).
- **Informativnost a úplnost (completeness):** zda odpověď pokrývá to podstatné, co by měla, a zda poskytuje dostatek detailů (aniž by odbíhala).

Pro každý dotaz tak LLM (např. znovu GPT-4 nebo jiný výkonný model) porovná dvě odpovědi (s kontextem vs. bez kontextu) a případně i referenční text či poznámky k správné odpovědi, pokud je to vhodné. Modelu bude zadán speciální prompt, který ho instruuje v roli hodnotícího. Může to vypadat například takto (parafrázováno do češtiny pro potřeby práce): „Posud’ následující dvě odpovědi na danou otázku. Kritéria: (1) relevance k otázce, (2) faktická správnost, (3) úplnost odpovědi. U každé odpovědi uveď krátký komentář, která z odpovědí je lepší.“. Je možné, že hodnocení provedeme formou škál (např. ohodnotit každé kritérium na stupnici 1–5 pro každou odpověď), nebo formou přímého slovního zhodnocení a výběru lepší odpovědi.

Tímto postupem získáme pro každou dvojici odpovědí kvalitativní hodnocení. Očekáváme, že ve většině případů LLM-hodnotitel označí odpověď s RAG kontextem za lepší či přesnější, zejména u těch dotazů, kde původní GPT-4 bez kontextu tápe. Nicméně důležitější, než potvrzení této hypotézy je odhalit důvody případných selhání. LLM-as-a-Judge nám totiž může v odůvodnění popsat, co konkrétně je špatně. Např. „Odpověď A opomíná zmínit rok konference, který byl v otázce tázán“ nebo „Odpověď B uvádí nepravdivé tvrzení, které není podloženo žádným zdrojem“. Taková zpětná vazba je velmi cenná pro analýzu chování systému.

Je na místě zmínit spolehlivost LLM jako soudce. Nedávný výzkum ukazuje, že silné modely jako GPT-4 dokážou hodnocení odpovědí provádět překvapivě konzistentně a v

souladu s lidskými hodnoceními. Uvádí se, že GPT-4 jako hodnotitel dosahuje shody s lidskými posuzovateli v ~80 % případů (Jung et al., 2024), což je srovnatelné s mírou shody mezi různými lidskými hodnotiteli. To naznačuje, že takové automatizované hodnocení může být dostatečně důvěryhodné pro účely práce. Samozřejmě budeme brát v potaz, že model-hodnotitel může být zaujatý (např. preferovat detailnější odpovědi) a tomu přizpůsobíme interpretaci výsledků.

```
You are evaluating two answers to the same question.
Respond ONLY in valid JSON. Do not write any explanations outside the JSON block.
Instructions:
- Score each answer from 1 to 5 based on correctness, relevance, and completeness.
- Then explain briefly which answer is better and why.
Use this exact JSON format (do not add commentary):
{
  "score_rag": [number from 1 to 5],
  "score_non_rag": [number from 1 to 5],
  "comment": "Your explanation here"
}
```

Konkrétní prompt pro LLM-as-a-Judge použitý v práci

4.4 Srovnání kvality odpovědí s RAG a bez něj

Pro ověření přínosu využití externího kontextu při generování odpovědí byly vyhodnoceny dvě varianty odpovědí na totožné dotazy:

1. **Odpovědi s RAG** – tedy s poskytnutým kontextem získaným pomocí vyhledávání ve vektorové databázi,
2. **Odpovědi bez RAG** – generované čistě modelem bez jakýchkoliv doplňujících podkladů.

Každý dotaz byl zpracován dvěma cestami a výsledné odpovědi byly porovnány pomocí dvou metod:

- **BERTScore** – měřil sémantickou podobnost odpovědi vůči referenčnímu textu. Hodnota podobnosti 0-1.
- **LLM-as-a-Judge** – model GPT-4o fungoval jako hodnotitel kvality, který měl za úkol určit, která odpověď lépe odpovídá položené otázce. Hodnocení probíhalo na škále 1-5.

Metrika	První sada otázek (1-20)	Druhá sada (21-40)
LLM-as-a-Judge (RAG)	2.95	4.95
LLM-as-a-Judge (bez RAG)	3.60	3.80
BERTScore (RAG)	0.897	0.983
BERTScore (bez RAG)	0.840	0.920

Hodnoty LLM-as-a-Judge a BERTScore pro první a druhou sadu dotazů

Metrika	Odpovědi s RAG	Odpovědi bez RAG
LLM-as-a-Judge (1–5)	3.95	3.70
BERTScore (0–1)	0.940	0.880

Průměrné hodnoty LLM-as-a-Judge a BERTScore

Výsledky ukazují, že odpovědi generované s kontextem dosahují výrazně **vyšší** sémantické shody s referenční odpovědí (dle BERTScore) i vyššího subjektivního hodnocení kvality (dle GPT-4o). Průměrné skóre LLM-as-a-Judge se u RAG varianty blíží horní hranici škály, zatímco bez kontextu model častěji odpovídal vágně nebo s menší faktickou přesností.

Z dat rovněž vyplývá, že:

- RAG odpovědi byly konkrétnější a bohatší na fakta obsažená v retrievaných dokumentech,
- Odpovědi bez RAG častěji halucinovaly nebo se opíraly o obecné znalosti bez ověření v datech,
- V několika případech byla odpověď bez RAG sice věcně správná, ale neobsahovala dostatečné detaily, a tím získala nižší hodnocení.
- V první sadě dotazů byl přínos RAG nižší – často šlo o obecnější nebo obtížněji specifikovatelné otázky, na které i samotné LLM odpovídalo relativně dobře.
- Ve druhé sadě dotazů došlo k výraznému zlepšení při použití RAG. Hodnocení pomocí LLM i BERTScore výrazně stoupl. To ukazuje, že při dotazech s konkrétními detaily nebo méně známými fakty hraje kvalitní retrieval a kontext zásadní roli.

Na dotaz: "Find me games that have 'Overwhelmingly Positive' reviews on Steam and cost less than 10 USD."

Odpověď s RAG: "The game \"Slay the Spire\" fits your criteria. It has \"Overwhelmingly Positive\" reviews on Steam and is currently priced at \$8.49."

Odpověď bez RAG: "Certainly! Here's a list of some popular games with \"Overwhelmingly Positive\" reviews on Steam that generally cost less than \$10. Keep in mind that prices and ratings can fluctuate, so it's always a good idea to check Steam for the most up-to-date information.\n\n1. **Celeste** - While sometimes priced slightly above \$10, it frequently goes on sale for under \$10, and it has received widespread acclaim for its platforming mechanics and emotional storytelling.\n\n2. **Stardew Valley** - This charming farming simulation RPG offers a relaxing yet immersive experience and often goes on sale within your budget.\n\n3. **Undertale** - Known for its unique combat system and engaging narrative, Undertale is a classic indie game beloved by many players.\n\n4. **Hollow Knight** - This metroidvania action-adventure game offers beautiful hand-drawn art and a challenging experience, and it frequently comes on sale for under \$10.\n\n5. **OneShot** - A puzzle-adventure game with an engaging story and creative gameplay elements that frequently appears in sales for less than \$10.\n\n6. **A Short Hike** - A relaxing exploration game with charming visuals and an open world to explore at your own pace, often available for under \$10.\n\n7. **Hotline Miami** - Known for its intense action and stylish presentation, this game often goes on sale for less than \$10.\n\n8. **The Binding of Isaac: Rebirth** - This roguelike offers immense replayability and frequently appears on sale under the \$10 mark.\n\nTo find the current prices and deals, you might want to keep an eye on Steam sales or third-party sites that track discounts."

Ilustrační příklad odpovědi s přidáním kontextem a bez kontextu

Ačkoli obě odpovědi byly formálně správné, první verze byla podrobnější a přesnější, Tím pádem získala vyšší BERTScore i vyšší hodnocení od LLM. Hodnocení u LLM-as-a-Judge také zohledňuje skutečnost, že se uživatel doptává na základě obohacením kontextu. LLM-as-a-Judge na ilustračním příkladě hodnotí odpověď s RAG 4 a odpověď bez ní obržela 3.

"llm_judge_comment": "Answer A correctly identifies 'Slay the Spire' as a game with 'Overwhelmingly Positive' reviews and a price under \$10, which matches the ground truth context. However, it only provides one game, which limits its completeness. Answer B provides a broader list of games that generally fit the criteria, but it lacks specific confirmation from the ground truth context that these games are currently under \$10 and have 'Overwhelmingly Positive' reviews. Some games listed may not consistently meet the price criterion without sales, making it less precise. Therefore, Answer A scores higher for correctness and relevance, while Answer B scores lower due to lack of specific verification and potential inaccuracies in pricing."

"bert_score_rag": 0.9147,

"bert_score_non_rag": 0.8198

Komentář LLM-as-a-Judge a hodnoty sémantické podobnosti BERTScore

5 Diskuse a limity

Výsledky experimentu ukazují, že úspěšnost vyhledávání relevantních dokumentů i kvalita generovaných odpovědí se výrazně liší podle typu dotazů. Fáze vyhledávání dosáhla v první testovací sadě průměrné přesnosti precision@5 jen 38 %. Zatímco v druhé sadě dotazů to bylo až 86 %. To znamená, že zejména u komplexnějších dotazů našel systém mezi prvními pěti výsledky jen zlomek relevantních dokumentů. Například dotaz typu „Find me games that have a 'Mixed' review status on Steam and are priced around 70 USD.“ nedokázal systém správně zodpovědět. Precision@5 zde byla 0 %. Tedy ani jeden z pěti vrácených dokumentů nepatřil mezi správné odpovědi. Naopak u jednoznačně formulovaných dotazů, které obsahovaly unikátní informace obsažené v databázi, byla úspěšnost vysoká. Z toho vyplývá, že vektorové vyhledávání dobře funguje pro dotazy popisující specifický obsah obsažený v datech, ale selhává u dotazů vyžadujících kombinaci více filtrů či číselných podmínek. Embeddingový model nedokázal dostatečně zachytit atributy jako cena, datum či vydavatel, takže dotazy typu „hra s Mixed hodnocením kolem 70 USD“ nebo filtry na žánr a vydavatele (např. hry od Electronic Arts, které nejsou střilečky) (viz. Příloha C) nevrátily relevantní výsledky.

Z experimentálních výsledků vyplývá, že úspěšnost systému je silně závislá na povaze dotazů a typu zpracovaných dat. V rámci této práce byl systém navržen primárně jako nástroj pro dotazování nad semi-strukturovanou databází herních titulů, kde každý záznam obsahoval jasně oddělené atributy (např. název, hodnocení, cena, žánr). V takovém prostředí se RAG model choval spíše jako vyhledávací a odpovídací systém nad databází než jako generativní chatbot s obecným porozuměním textu. Výsledky tedy ukazují, že systém fungoval lépe u dotazů obsahujících klíčová slova odpovídající strukturovaným atributům (např. názvy her, recenzní hodnocení, cena, vývojář, popisek).

Naopak u obecných nebo popisných dotazů by systém potřeboval velmi rozsáhlý a různorodý korpus ne-strukturovaných textových dat, aby mohl rozpoznat význam a poskytnout relevantní odpověď. V takových případech by pomohlo například jemné doladění embedding modelu na doménový text, nebo kombinace RAG s metadaty a filtračními pravidly, které by pomohly omezit rozsah vyhledávání na základě známých atributů (např. žánr = „adventure“, platforma = „android“). Současné řešení však takové metody nevyužívá, a proto u těchto komplexnějších scénářů selhávalo.

Odpovědi modelu, kterému byly poskytnuty nalezené kontextové informace, dosahovaly vyšší sémantické shody s referenčními odpověďmi než odpovědi modelu bez kontextu. Průměrný BERTScore generovaných odpovědí s RAG byl ~0,94, zatímco bez RAG ~0,88. Model s přístupem RAG tedy ve většině případů zahrnul do odpovědi relevantní fakta z nalezených dokumentů a eliminoval některé halucinace, které se objevovaly u modelu bez externí znalostní základny. Například u dotazu na „narativně laděnou hru o příbězích členů rodiny Finchových“ dokázal systém s RAG správně identifikovat hru *What Remains of Edith Finch*, zatímco samotný LLM bez kontextu tipoval nesprávnou odpověď (hra *Myst*). Navzdory těmto zlepšením však generátor odpovědí ne vždy využil poskytnutý

kontext optimálně. V některých případech byly odpovědi nekompletní. Model opomněl zmínit všechny relevantní informace, i když je měl k dispozici. Například na dotaz hledající hry s „Mixed“ hodnocením a cenou kolem 70 USD byla v databázi přítomna jak hra *Monster Hunter Wilds*, tak i *PGA TOUR 2K25*, avšak retriever v rámci nastaveného top-k vrátil pouze první z nich. Model tak neměl přístup k druhé správné informaci, i když byla ve vektorové databázi uložena. Tento nedostatek ukazuje na limitace fáze retrievalu při souhrnném vyhledávání více relevantních dokumentů. Zároveň potvrzuje, že efektivita celého RAG řešení závisí nejen na přítomnosti správných dat, ale také na schopnosti retrieveru je identifikovat a předat modelu k dalšímu zpracování.

Zmíněné nedostatky úzce souvisejí s limity navrženého řešení a použitých dat. Kvalita a konzistence herního datasetu představuje limit. Data pocházejí z více zdrojů (Steam, IGDB) s různou strukturou a úplností. Popisy her mají různou délku i formát, chybí jednotný styl a některé položky (např. uživatelské hodnocení, datum vydání či žánrové tagy) nebyly u všech her dostupné nebo aktuální. To vše ztěžuje vytvoření univerzální reprezentace vhodné pro vektorové vyhledávání. Dalším limitem byly technické výzvy při samotném webscrapingu. Kvůli omezením API a kvótám na straně cílových služeb bylo nutné omezit rozsah sběru dat a operací. Výsledná databáze her tak nezahrnuje všechny možné tituly či nejnovější aktualizace, což může vést k tomu, že některé dotazy nemají v datech odpověď, nebo se odpověď opírá o neaktuální informace.

Další oblast limitací spočívá v použitých metodách hodnocení a rozsahu testování. Metriky jako Precision@5 a Recall@5 hodnotí kvalitu vyhledávání pouze z kvantitativního hlediska a na omezeném počtu výsledků. Pokud měl dotaz více než pět správných odpovědí v databázi, Precision@5 ani Recall@5 nepostihnou, zda by se relevantní dokument objevil třeba na šesté či desáté pozici. Také BERTScore, použitý pro hodnocení generovaných odpovědí, měří sémantickou podobnost s referencí, ale nepostihne například stylistické nedostatky nebo částečně chybějící informace v odpovědi. Dvě odpovědi mohou dosáhnout vysokého BERTScore, přestože jen jedna z nich obsahuje úplná fakta (Chen et al., 2022). *LLM-as-a-judge* hodnocení poskytlo užitečný kvalitativní pohled, ale nese s sebou subjektivitu a případnou nekonzistentnost automatického posuzování. V rámci práce byla testována sada 40 dotazů, která pokrývá jen omezené scénáře použití. Chybí rozsáhlejší uživatelská evaluace a porovnání s jinými systémy, takže závěry o obecné platnosti mají své rezervy. Navíc pro mnoho dotazů neexistuje sada „správných“ odpovědí. Referenční odpovědi bylo nutné odvodit z nascrapovaných dat a odborných znalostí. Tato absence nezávislého referenčního datasetu limituje ověření správnosti a nelze jednoznačně určit, zda by jiný systém či člověk poskytl lepší odpověď, protože chybí standard pro srovnání.

Identifikované limity naznačují prostor pro zlepšení navrženého řešení. V rámci diskuse vyplynulo, že kvalita odpovědí závisí na kvalitě dat a vyhledávání. Pokud nejsou data úplná a relevantní dokument se nedostane mezi top výsledky. Současně bylo ověřeno, že no-code přístup (Make.com) může sloužit k rychlé prototypizaci. Následující závěrečná kapitola shrnuje dosažené cíle práce a navrhuje konkrétní směry, jak těmto limitům čelit a dále rozvíjet vzniklé řešení.

Závěr

Cílem práce bylo ověřit možnost propojení no-code platformy pro webscraping s LLM chatbotem pomocí RAG (Retrieval-Augmented Generation) přístupu a zhodnotit, nakolik takové řešení zlepší odpovědi na dotazy uživatelů. Těchto cílů se podařilo dosáhnout. Byl navržen a implementován systém využívající Make.com pro automatizované získávání dat o videohrách a jejich uložení ve vektorové databázi, s následnou integrací s velkým jazykovým modelem pro dotazování. Na implementovaném prototypu proběhlo experimentální vyhodnocení kvality odpovědí. Zároveň se potvrdilo, že kvalita generovaných odpovědí silně závisí na kvalitě vstupních dat a efektivitě vyhledání relevantních informací. Pokud dotazovaná znalost v databázi chybí nebo není nalezena, model odpovídá neúplně. Tím se projevil hlavní limit celého řešení.

Bakalářská práce přinesla praktický důkaz konceptu propojení no-code nástroje a LLM v oblasti herních informací. Autor demonstroval, že i bez hlubšího programování lze získat a využít data k rozšíření znalostí jazykového modelu a zvýšení relevance jeho odpovědí. Současně byly identifikovány hranice tohoto přístupu, vyplývající z datových omezení i z vlastností použitých metod. Tyto poznatky otevírají několik směrů pro budoucí rozvoj. Za prvé, bylo by vhodné podstatně rozšířit databázi her. Zahrnout více zdrojů (např. další herní databáze) a udržovat data aktuální, aby systém pokrýval širší spektrum dotazů včetně nejnovějších informací. Za druhé, nabízí se nasazení pokročilejších metod vyhledávání. Kombinace vektorového vyhledávání s klasickým fulltextem či filtrováním metadat by mohla lépe obsloužit dotazy kombinující více kritérií (například číselné rozsahy cen, konkrétní vydavatele apod.). Případně lze experimentovat s jemným doladěním embedding modelu na doménová data, aby lépe rozlišoval specifické herní pojmy a atributy. Takové fine-tuning by mohlo zvýšit přesnost vyhledávání relevantních dokumentů. Dále by bylo přínosné zapojit více uživatelů do testování chatbotu a sběru zpětné vazby. Uživatelská evaluace (např. formou scénářů otázek a hodnocení spokojenosti s odpověďmi) by poskytla cenný pohled na praktickou použitelnost systému a odhalila další slabiny, které automatické metriky nemusí odhalit. V neposlední řadě, budoucí práce může zkoumat integraci výkonnějších LLM, případně optimalizaci promptů a řetězení dotazů tak, aby chatbot efektivněji využíval poskytnutý kontext. Realizace těchto vylepšení by mohla posunout navržené řešení blíže k nasazení v praxi a umožnila poskytovat uživatelům přesnější a informačně bohatší odpovědi.

Použitá literatura

Belagatti, P. (2024, prosinec 6). *A Deep Dive Into Vector Databases | SingleStoreDB for Vectors*. SingleStore. <https://www.singlestore.com/blog/a-complete-guide-to-vector-databases/>

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* (No. arXiv:1810.04805). arXiv. <https://doi.org/10.48550/arXiv.1810.04805>

Zanini, A. (2025). *Best Headless Browsers for Scraping and Testing*. Bright Data. <https://brightdata.com/blog/web-data/best-headless-browsers>

Zanini, A. (2025). *Best Python Web Scraping Libraries of 2025*. Bright Data. <https://brightdata.com/blog/web-data/python-web-scraping-libraries>

Gloeckle, F., Idrissi, B. Y., Rozière, B., Lopez-Paz, D., & Synnaeve, G. (2024). *Better & Faster Large Language Models via Multi-token Prediction* (No. arXiv:2404.19737). arXiv. <https://doi.org/10.48550/arXiv.2404.19737>

Choi, K., Jeong, M., Shin, Y., Ma, J., Kim, K., & Kim, H. (2025). *Construct-Rag: Contrastive Sentence Training & Retrieval Using Chunk Block-Based Text for Rag* (SSRN Scholarly Paper No. 5205959). Social Science Research Network. <https://doi.org/10.2139/ssrn.5205959>

Yu, H., Hua, J., & Julien, C. (2021). Dataset: Analysis of IFTTT Recipes to Study How Humans Use Internet-of-Things (IoT) Devices. *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, 537–541. <https://doi.org/10.1145/3485730.3494115>

Balaji, M., & Kariaden Parambath, B. (2025, březn 6). *Evaluate RAG responses with Amazon Bedrock, LlamaIndex and RAGAS | AWS Machine Learning Blog*. <https://aws.amazon.com/blogs/machine-learning/evaluate-rag-responses-with-amazon-bedrock-llamaindex-and-ragas/>

Sadri, N. (2022). *Evaluating Dense Passage Retrieval using Transformers* (No. arXiv:2208.06959). arXiv. <https://doi.org/10.48550/arXiv.2208.06959>

Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., & Liu, Z. (2025). *Evaluation of Retrieval-Augmented Generation: A Survey* (Roč. 2301, s. 102–120). https://doi.org/10.1007/978-981-96-1024-2_8

Yashaswini, S. (2024). *EXPLORING WEB SCRAPING TECHNIQUES FOR DATA EXTRACTION IN E-COMMERCE PLATFORMS: A COMPREHENSIVE ANALYSIS*. *International Research Journal of Modernization in Engineering Technology and Science*. <https://doi.org/10.56726/IRJMETS50725>

- Bruch, S. (2024). *Foundations of Vector Retrieval*. <https://doi.org/10.1007/978-3-031-55182-6>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., & Zhou, D. (2023). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models* (No. arXiv:2201.11903). arXiv. <https://doi.org/10.48550/arXiv.2201.11903>
- CorerMaximus. (2020, červen 10). *IGDB by Twitch API hourly/minute rate limit?* [Forum post]. Stack Overflow. <https://stackoverflow.com/q/62314675>
- Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., ... Amodei, D. (2020). *Language Models are Few-Shot Learners* (No. arXiv:2005.14165). arXiv. <https://doi.org/10.48550/arXiv.2005.14165>
- Samuylova, E. (2025). *LLM evaluation metrics and methods*. <https://www.evidentlyai.com/llm-guide/llm-evaluation-metrics>
- Kumaresh. (2024, září 18). *Make.com—The Game-Changing Automation Tool Anyone Can Master. Dialed In AI*. <https://medium.com/dialed-in-ai/make-com-the-game-changing-automation-tool-anyone-can-master-613a99ca89c7>
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. de las, Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., & Sayed, W. E. (2023). *Mistral 7B* (No. arXiv:2310.06825). arXiv. <https://doi.org/10.48550/arXiv.2310.06825>
- Chen, Y., Belouadi, J., & Eger, S. (2022). *Reproducibility Issues for BERT-based Evaluation Metrics* (No. arXiv:2204.00004). arXiv. <https://doi.org/10.48550/arXiv.2204.00004>
- Greene, R., Sanders, T., Weng, L., & Neelakantan, A. (2024, březn 13). *New and improved embedding model*. <https://openai.com/index/new-and-improved-embedding-model/>
- Walsh, D. (2024). *Overview of RAG Systems for AI Automation | Coconote*. <https://coconote.app/notes/613e4eb2-2501-4fdf-bd21-a16ad47fb291>
- Parimi, S. (2025, únor 13). *(PDF) Impact of Low-Code/No-Code Platforms*. ResearchGate. <https://doi.org/10.13140/RG.2.2.34792.79366>
- Herreros, Q., Veasey, T., & Papaoikonomou, T. (2023, prosinec 1). *RAG evaluation metrics: UniEval, BLEU, ROUGE & more*. Elasticsearch Labs. <https://www.elastic.co/search-labs/blog/evaluating-rag-metrics>
- Es, S., Jithin, J., Espinosa-Anke, L., & Schockaert, S. (2023). *RAGAS: Automated Evaluation of Retrieval Augmented Generation*. *arXiv.Org*. <https://www.proquest.com/docview/2869798598/AC68701C10684BCCPQ/1?sourcetype=Working%20Papers>

- Diaz, F., Ekstrand, M. D., & Mitra, B. (2024). *Recall, Robustness, and Lexicographic Evaluation* (No. arXiv:2302.11370). arXiv. <https://doi.org/10.48550/arXiv.2302.11370>
- Proser, Z. (2023). *Retrieval Augmented Generation (RAG) | Pinecone*. <https://www.pinecone.io/learn/retrieval-augmented-generation/>
- Kenneweg, T., Kenneweg, P., & Hammer, B. (2024). *Retrieval Augmented Generation Systems: Automatic Dataset Creation, Evaluation and Boolean Agent Setup* (No. arXiv:2403.00820). arXiv. <https://doi.org/10.48550/arXiv.2403.00820>
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *ArXiv, abs/2005.11401*. <https://consensus.app/papers/retrievalaugmented-generation-for-knowledgeintensive-lewis-perez/fbc9d8d6f6de501987cc8c3afa034696/>
- Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., & Chua, T.-S. (2021). *Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering* (No. arXiv:2101.00774). arXiv. <https://doi.org/10.48550/arXiv.2101.00774>
- Wewerka, J., & Reichert, M. (2020). *Robotic Process Automation—A Systematic Literature Review and Assessment Framework* (No. arXiv:2012.11951). arXiv. <https://doi.org/10.48550/arXiv.2012.11951>
- Wu, S., Li, Y., Zhu, K., Zhang, G., Liang, Y., Ma, K., Xiao, C., Zhang, H., Yang, B., Chen, W., Huang, W., Moubayed, N. A., Fu, J., & Lin, C. (2024). *SciMMIR: Benchmarking Scientific Multi-modal Information Retrieval* (No. arXiv:2401.13478). arXiv. <https://doi.org/10.48550/arXiv.2401.13478>
- Wang, X., Wang, Z., Gao, X., Zhang, F., Wu, Y., Xu, Z., Shi, T., Wang, Z., Li, S., Qian, Q., Yin, R., Lv, C., Zheng, X., & Huang, X. (2024). *Searching for Best Practices in Retrieval-Augmented Generation* (No. arXiv:2407.01219). arXiv. <https://doi.org/10.48550/arXiv.2407.01219>
- Mittal, A., Bheemreddy, A., & Tao, H. (2024). *Semantic SQL -- Combining and optimizing semantic predicates in SQL* (No. arXiv:2404.03880). arXiv. <https://doi.org/10.48550/arXiv.2404.03880>
- Moucka, L. (2022, květen 16). *Steam Store Scraper*. Apify. <https://apify.com/m0uka/steam-store-scraper>
- Sasín, J. (2021). *Strojové učení pro odpovídání na otázky v přirozeném jazyce* [Bakalářská práce, Vysoké učení technické v Brně]. <https://theses.cz/id/odfo2z/>
- Yan, Z. (2021). *The Impacts of Low/No-Code Development on Digital Transformation and Software Development* (No. arXiv:2112.14073). arXiv. <https://doi.org/10.48550/arXiv.2112.14073>

- Jung, J., Brahman, F., & Choi, Y. (2024). *Trust or Escalate: LLM Judges with Provable Guarantees for Human Agreement* (No. arXiv:2407.18370). arXiv. <https://doi.org/10.48550/arXiv.2407.18370>
- Hong, Z. (2023, listopad 30). Unlock Efficient Gaming Data Extraction: A Comprehensive Guide to IGDB, Postman, and Twitch.... *Medium*. <https://medium.com/@zhonghong9998/unlock-efficient-gaming-data-extraction-a-comprehensive-guide-to-igdb-postman-and-twitch-9fe44e4d1586>
- Infragistics. (2025, leden 22). *Vendor Lock-In Risks: Why Low-Code Platforms Must Prioritize Freedom*. App Builder. <https://www.appbuilder.dev/blog/vendor-lock-in>
- Tal. (2025). *Web Scraping vs API: What You Need to Know*. Bright Data. <https://brightdata.com/blog/web-data/web-scraping-vs-api>
- Schwaber-Cohen, R. (2023). *What is a Vector Database & How Does it Work? Use Cases + Examples* | Pinecone. <https://www.pinecone.io/learn/vector-database/>
- Kumparak, G. (2020, březn 9). Zapier CEO Wade Foster on scaling a remote team up to 300 employees. *TechCrunch*. <https://techcrunch.com/2020/03/09/zapier-ceo-wade-foster-on-scaling-a-remote-team-up-to-300-employees/>
- Make. (2025). *Automation in Schools: Introducing the Make Academic Alliance*. Make. <https://www.make.com/en/blog/introducing-make-academic-alliance>
- An evaluation of the impact of Twitch's content classification labelling*. (2024, únor 9). Ww.ofcom.Org.Uk. <https://www.ofcom.org.uk/online-safety/illegal-and-harmful-content/an-evaluation-of-the-impact-of-twitchs-content-classification-labelling/>
- Apify: Full-stack web scraping and data extraction platform*. (2025). Apify. <https://apify.com>
- IGDB: Video Game Database API*. (b.r.). IGDB.Com. Získáno 21. duben 2025, z <https://www.igdb.com/api>
- LLM-as-a-judge: A complete guide to using LLMs for evaluations*. (b.r.). Získáno 28. duben 2025, z <https://www.evidentlyai.com/llm-guide/llm-as-a-judge>
- n8n.io—A powerful workflow automation tool*. (2025). <https://n8n.io/>
- Pinecone api call, embedding is a array but when put into http it's a string—How To*. (2025, březn 15). Make Community. <https://community.make.com/t/pinecone-api-call-embedding-is-a-array-but-when-put-into-http-its-a-string/75198>
- [Vaccination 2021] *The Pinecone Vector Database System* (Edo Liberty). (b.r.). *Carnegie Mellon Database Group*. Získáno 21. duben 2025, z <https://db.cs.cmu.edu/events/vaccination-2021-the-pinecone-vector-database-system-edo-liberty/>

Dinan, E., Roller, S., Shuster, K., Fan, A., Auli, M., & Weston, J. (2019). *Wizard of Wikipedia: Knowledge-Powered Conversational agents* (No. arXiv:1811.01241). arXiv. <https://doi.org/10.48550/arXiv.1811.01241>

An evaluation of the impact of Twitch's content classification labelling. (2024, únor 9). Wwww.Ofcom.Org.Uk. <https://www.ofcom.org.uk/online-safety/illegal-and-harmful-content/an-evaluation-of-the-impact-of-twitchs-content-classification-labelling/>

Caballar, R. D. (2020). *Programming Without Code: The Rise of No-Code Software Development - IEEE Spectrum.* <https://spectrum.ieee.org/programming-without-code-no-code-software-development>

Přílohy

Příloha A: První sada dotazů

Obsaženo v souboru evaluation_queries.json.

Příloha B: Druhá sada dotazů

Obsaženo v souboru evaluation_queries2.json.

Příloha C: Evaluace retrievalu

Soubor obsahuje jednotlivé výsledky vyhledávání dokumentů z databáze na každý dotaz.

Obsaženo v RetrievalEval.txt

Příloha D: JSON soubor s plnou evaluací

Soubor obsahuje hodnocení LLM-as-a-Judge pro RAG a bez RAG odpověď. S BERTScore a ground truth a dotazy.

Obsaženo v souboru evaluation_llm_with_bertscore.json.

Příloha E: Připravené textové soubory pro každý záznam hry

Obsažené v souborech: chunked_igdb_list4, chunked_igdb_list3_cleanE, BAIN-report