Filozofická fakulta Univerzity Palackého v Olomouci

Katedra obecné lingvistiky

Evaluace RAG systémů a LLM na češtině

bakalářská diplomová práce

Autor: Tereza Brožová

Vedoucí práce: Mgr. Vladimír Matlach, Ph.D.

Olomouc

2025

Prohlášení

Prohlašuji, že jsem bakalářskou diplomovou práci „Evaluace RAG systémů a LLM na češtině“ vypracovala samostatně a uvedla jsem veškerou použitou literaturu a veškeré použité zdroje.

V Olomouci dne 15. 5. 2025 Podpis 

Poděkování

Ráda bych touto cestou vyjádřila upřímný vděk panu doktoru Matlachovi za jeho vstřícnost, čas a cenné odborné rady, které mi poskytl při zpracování této práce.

Název práce: Evaluace RAG systémů a LLM na češtině

Autor práce: Tereza Brožová

Vedoucí práce: Mgr. Vladimír Matlach, Ph.D.

Počet stran a znaků: 48 stran; 78 926 znaků

Počet příloh: 1

Abstrakt:

Práce se zabývá návrhem a evaluací systému Retrieval-Augmented Generation (RAG) pracujícího s českými texty. V teoretické části práce jsou vymezeny klíčové koncepty z oblasti zpracování přirozeného jazyka, neuronových sítí a umělé inteligence. Dále je objasněna role metod, jako je fine-tuning, promptování, chunking a využití vektorových databází v kontextu systémů RAG. Cílem experimentální části práce je otestovat schopnosti embedding modelů při vektorizaci českých textů a vyhodnotit, s jakou přesností lze ve vektorové databázi na základě sémantické podobnosti nalézat relevantní texty správně zodpovídající zadané otázky. Dva nejpřesnější modely – jeden placený a jeden bezplatný – jsou následně integrovány do systému RAG, kde jsou otestovány v kombinaci se dvěma generativními jazykovými modely. Cílem je zjistit, zda tyto modely dokážou generovat správné odpovědi na otázky na základě relevantních textů získaných z databáze, a zda může být systém RAG používající češtinu dostatečně efektivní pro praktické využití.

Klíčová slova:

Generativní jazykový model, systém RAG, velký jazykový model, embedding, vektorová databáze, prompt

Title: RAG system and LLM evaluation on Czech texts

Author: Tereza Brožová

Supervisor: Mgr. Vladimír Matlach, Ph.D.

Number of pages and characters: 48 pages; 78 926 characters

Number of appendices: 1

Abstract:

The aim of this thesis is to design and evaluate a Retrieval-Augmented Generation (RAG) system using Czech texts. The theoretical part of the thesis defines key concepts in the field of natural language processing, neural networks and artificial intelligence. Further on, the role of methods such as fine-tuning, prompting, chunking and the use of vector databases in the context of RAG systems is explained. The aim of the experimental part of the thesis is to test the capability of the embedding models in vectorizing Czech texts and to evaluate the accuracy with which relevant texts correctly answering the given questions can be found in the vector database based on semantic similarity. The two most accurate models – one paid and one free – are then integrated into the RAG system, where they are tested in combination with two generative language models. The aim is to determine whether these models can generate correct answers to questions based on the relevant texts retrieved from the database, and whether a RAG system using Czech can be effective enough for practical use.

Keywords:

Generative language model, Retrieval-Augmented Generation system, Large Language Model, embedding, vector database, prompt

Obsah

Úvod

Architektura systému Retrieval-Augmented Generation (RAG) propojuje možnosti generativní umělé inteligence s vyhledáváním relevantních textů ve vektorové databázi na základě sémantické podobnosti. Umožňuje tak velkým jazykovým modelům generovat odpovědi na základě lokálních dat, která nebyla součástí trénovacího datasetu. Výsledkem je personalizovaný generativní model schopný přesnějších, kontextově relevantnějších a aktuálnějších výstupů. Tato metoda zároveň snižuje riziko tzv. halucinací, tj. generování nepravdivých či smyšlených výstupů. Systémy RAG nacházejí praktické uplatnění zejména ve vyhledávacích systémech a konverzačních asistentech (chatbotech), kde umožňují kombinovat znalosti modelu s interními a specializovanými daty.

V bakalářské práci budou nejprve v teorii představeny klíčové koncepty v oblasti zpracování přirozeného jazyka a umělé inteligence a následně koncepty související s tvorbou systému RAG. Budou vysvětleny techniky a metody využívané při personalizaci generativních modelů a popsán samotný proces implementace systému RAG.

Cílem experimentální části je zjistit, zda lze vytvořit funkční a efektivní systém RAG v češtině a zda jej lze realizovat pouze s využitím bezplatných modelů. Experimentální část práce se skládá ze dvou fází. V první fázi bude vytvořen dataset dvaceti fiktivních novinových článků, které budou rozděleny na kratší části (chunky) o fixním počtu slov. Každému článku bude přiřazena jedna otázka, na niž lze odpovědět na základě některého z jeho chunků. Texty budou následně převedeny do číselné vektorové reprezentace prostřednictvím různých jazykových modelů a uloženy do vektorové databáze. Bude vyhodnoceno, které modely nejlépe vektorizují české texty a umožňují nalézt správné a relevantní chunky textů pomocí kosinové podobnosti.

V druhé fázi budou na základě získaných výsledků vybrány dva nejpřesnější modely – jeden bezplatný a jeden placený – a integrovány do systému RAG v kombinaci se dvěma vybranými generativními modely. V této fázi bude vytvořena šablona promptu pro generativní model, do níž budou vkládány otázky s nejpodobnějšími nalezenými texty. Prompt bude použit pro generování odpovědí pomocí čtyř různých kombinací modelů. Následně budou odpovědi kvalitativně zanalyzovány a ohodnoceny na základě předem definované bodové škály. Hodnoceno bude, do jaké míry dokážou generativní modely zodpovídat správně položené otázky a poskytovat relevantní informace. Dílčím cílem této fáze je rovněž zjistit, zda hraje rozdělení (chunking) textů roli při generování odpovědí – tedy zda je vhodnější do promptu vkládat pouze chunky, nebo celé texty.

1 Teoretický základ

1.1 Zpracování přirozeného jazyka

Zpracování přirozeného jazyka (angl. *natural language processing*, NLP) je disciplína na pomezí lingvistiky, informatiky a umělé inteligence. Věnuje se analýze přirozených jazyků (např. čeština, angličtina aj.) pomocí počítačových nástrojů, zejména programovacích jazyků a jejich knihoven. Jednou z nejdůležitějších praktických aplikací NLP je strojový překlad, výpočetně nejnáročnější pak zřejmě rozpoznávání mluvy (angl. *speech recognition*). NLP je základním pilířem v textové analýze a strojovém učení. (Sarkar 2016)

Programovací jazyk Python nabízí řadu knihoven, jako jsou spaCy či NLTK, které poskytují různé klíčové techniky NLP. Mezi nejpopulárnější techniky NLP patří například:

- tokenizace – dělení textu na jednotlivé textové jednotky, např. slova a interpunkce;
- chunking – dělení textu na úseky (chunky) o zvolené délce;
- lemmatizace – převod slov na základní slovní tvary;
- stemming – převod slov na jejich kořenový tvar;
- označování slovních druhů (angl. part-of-speech tagging, POS) – přiřazování gramatických kategorií jednotlivým slovům;
- nebo odstraňování funkčních slov – vynechání slov, která nenesou význam, jako jsou spojky či předložky.

Tyto techniky tvoří základ předzpracování textů, které je nezbytné pro další práci s texty, např. pro vektorovou reprezentaci textu nebo metody ve strojovém učení. Součástí tohoto procesu je také normalizace textů, tj. jejich převod do jednotného formátu. Ten obvykle spočívá v čištění textů odstraněním nadbytečných textových jednotek, značek, speciálních znaků, URL odkazů apod. (Sarkar 2016)

1.2 Neuronové sítě

Umělé neuronové sítě jsou důležitým nástrojem pro efektivní zpracování přirozeného jazyka, kde vykonávají úlohy, jako je strojový překlad, analýza sentimentu, klasifikace nebo sumarizace textů, rozpoznávání jmenných entit a generování textů. Neuronová síť se skládá z vrstev neuronů, z nichž každý přijímá vstupní hodnoty, které dále zpracovává pomocí aktivační funkce. Výsledkem je výstup předaný dalším neuronům v následující vrstvě. Každému neuronu náleží váhy, které určují důležitost jednotlivých vstupů pro daný neuron, a bias, který umožňuje posun aktivační funkce. (Jurafsky a Martin 2025)

Proces učení neuronové sítě spočívá v iterativní úpravě vah a biasů na základě trénovacích dat pomocí optimalizačních algoritmů. Tento proces, označovaný jako hluboké učení, umožňuje modelu hledat vzorce v datech a učit se. Neuronové sítě mají většinou velké množství vrstev. V oblasti NLP se

využívají různé typy neuronových sítí, mezi něž patří klasické dopředné neuronové sítě, rekurentní neuronové sítě, konvoluční neuronové sítě a především transformátory. (Jurafsky a Martin 2025)

1.2.1 Sémantická reprezentace textu

Jelikož počítače nerozumí přirozenému jazyku, ale pracují výhradně s číselnými hodnotami, je nezbytné předzpracované texty pro další práci vždy převést do číselné podoby – konkrétně do vektorů, které reprezentují jejich význam. (Hapke, Howard, a Lane 2019) Proces vektorizace textu spočívá v převodu slov, vět či delších textů na tzv. sémantické embeddingy (angl. *word embeddings*, v češtině nazýváno „vnoření slov“), které textové jednotky reprezentují jako body v multidimenzionálním vektorovém prostoru odvozeném na základě vztahů jednotek mezi sebou. (Jurafsky a Martin 2025) V tomto prostoru se významově podobné jednotky nacházejí blízko sebe. V jazyce Python jsou embeddingy obvykle reprezentovány jako seznamy nebo pole (např. pomocí knihovny numPy), kde každý vektor má délku *n*.

Pro srovnání lze uvést jednodušší formu vektorové reprezentace textu: *bag of words* (BoW). Tento model reprezentuje každý text jako vektor o délce celého neseřazeného slovníku. tj. počtu unikátních slov ve všech textech daného datasetu. Každá pozice (či index) ve vektoru odpovídá konkrétnímu slovu ve slovníku, přičemž hodnota udává počet jeho výskytů v daném textu. (Hapke, Howard, a Lane 2019) BoW nese informaci o tom, zda se v daném textu vyskytuje dané slovo a kolikrát, nezachycuje však pořadí slov, kontext, ani sémantickou hodnotu.

1.2.2 Tradiční sémantické embeddingy

S revolučním nápadem na převod významu slov do číselné reprezentace přišel český vědec Tomáš Mikolov (2013), který se svým týmem ve společnosti Google vyvinul jazykový model Word2vec. Tento model využívá dopřednou neuronovou síť k trénování sémantických embeddingů. Je trénován na rozsáhlém textovém korpusu, na jehož základě je vytvořen fixní globální slovník – seznam slov s přiřazenými statickými vektory o stejné dimenzi. V případě tvorby sémantického embeddingu pro delší text (větu, odstavec) lze vektory zprůměrovat do jednoho výsledného vektoru, který odráží význam celého textu.

Na stejném principu je založen i model fastText vyvinutý společností Meta. Oproti Word2vec je však přesnější, protože při trénování zohledňuje n-gramy (posloupnosti slov). (Bojanowski et al. 2017) Dalším známý modelem je GloVe, který se rychleji trénuje, využívá efektivněji RAM počítače a dosahuje dobrých výsledků i při použití menšího objemu trénovacích dat. (Hapke, Howard, a Lane 2019)

Všechny tyto modely jsou volně dostupné ke stažení zdarma ve formě předtrénovaných embeddingů bez nutnosti opětovného trénování. Práce s nimi tedy spočívá v pouhém vyhledávání vektorů odpovídajícím slovům ve statickém slovníku.

Nevýhodou tradičních sémantických embeddingů je jejich statická povaha – jelikož se jedná o statický seznam slov spojených s jedním vektorem odrážejícím význam, nemohou brát v potaz kontext a vztah slov k okolním slovům ve větě či textu. To má za následek, že například embedding pro slovo *pila* bude ve větách „*Na stole ležela pila*.“ a „*Pila jsem vodu*.“ stejný. Tento problém se vztahuje na veškerá

homonyma a polysémní slova, tj. slova s více významy. (Matlach 2023) Slova mohou mít navíc různý význam v závislosti na situaci.

1.2.3 Rekurentní neuronové sítě

Řešením dekontextualizace významu slov u tradičních embedding modelů se staly rekurentní neuronové sítě (angl. *recurrent neural network*, RNN), které při generování embeddingů zohledňují kontext. „Každý vektor slova je vložen do funkce, která ho transformuje na základě slov v okolním kontextu, čímž vzniká nová verze vektoru slova, nyní specifická pro daný token v konkrétním kontextu“. (Smith 2019, překlad vlastní) Vektory tak zachycují sémantiku textových jednotek nikoliv staticky pro celý slovník, ale dynamicky v závislosti na větě, v níž se slovo objevilo. Klíčovým prvkem jsou tzv. skryté stavy, díky nimž si sítě „pamatují“ již zpracovaná slova a jejich významy v pořadí, v jakém se v sekvenci (textu) objevily. Skrytý stav se po zpracování každého slova aktualizuje, čímž vzniká v každém kroku nová reprezentace celé věty nebo textu. (Matlach 2023)

Nevýhodou RNN je především jejich omezená schopnost pracovat s delšími sekvencemi, protože důsledkem tzv. mizejícího gradientu (angl. *vanishing gradient problem*) během trénování „zapomínají“ informace, které byly na začátku sekvence. Tento problém částečně vyřešily LSTM sítě (zkr. pro angl. *long short-term memory* – dlouhá krátkodobá paměť), které dokážou udržet relevantní informace po delší dobu zpracování sekvence a zároveň zapomínat ty informace, které již nejsou důležité. (Jurafsky a Martin 2025) Prvním významným jazykovým modelem využívajícím tuto síť byl ELMo (angl. *Embeddings from Language Models*), jenž byl představen výzkumníky z *Allen Institute for Artificial Intelligence* v roce 2018. (Peters et al. 2018)

1.2.4 Transformátor typu enkodér

V roce 2017 nastal průlom, když výzkumný tým Google Brain v článku *Pozornost je vše, co potřebujete* (angl. *Attention Is All You Need*) představil nový typ neuronové sítě zvaný transformátor (angl. *transformer*). (Vaswani et al. 2017) Transformátory se staly průlomovými díky podstatně vyšší přesnosti ve všech NLP úlohách a rychlejšímu trénování oproti předešlým neuronovým sítím. Významně se také rozšířila škála úloh, které zvládají. Patří mezi ně schopnost plynulé konverzace (chatboti), strojový překlad, generování kódu, přeformulování či shrnutí textu a další úlohy, na které původně modely ani nebyly trénovány. (Matlach 2023) Tyto modely mají především mimořádnou schopnost práce s textem.

Prvním používaným transformátorem byl maskovaný jazykový model (angl. *masked language model*, MLM) zvaný BERT (angl. *Bidirectional Encoder Representations from Transformers*), který vyvinula společnost Google. (Devlin et al. 2018) „Tento model je trénován prostřednictvím maskovaného jazykového modelování, při němž je namísto predikce následujícího slova v trénovacím textu náhodně maskováno jedno slovo. Model má za úkol uhodnout, jaké slovo zde má být v závislosti na slovech na obou stranách. Tato metoda umožňuje modelu vidět kontext zleva i zprava.“ (Jurafsky a Martin 2025, překlad vlastní) BERT je tzv. enkodér (angl. *encoder-only model*), což znamená, že text dokáže zakódovat (převést

do číselné reprezentace), pro generování nového textu jej však nelze použít. Používá se tedy primárně pro sémantickou interpretaci textů, např. pro jejich klasifikaci nebo sentimentovou analýzu. Jazykové modely schopné generování textu jsou pak typu enkodér-dekodér (angl. *encoder-decoder model*) nebo dekodér (angl. *decoder-only model*), a již nepoužívají metodu maskovaného jazykového modelování. (Jurafsky a Martin 2025)

1.2.5 Transformátor typu enkodér-dekodér

Druhým typem transformátorů jsou modely typu enkodér-dekodér, které jsou využívány především pro úlohy strojového překladu, sumarizace textu, odpovídání na otázky či vedení konverzace. Tyto modely dokážou generovat výstupní sekvence libovolné délky, které významově odpovídají kontextu vstupní sekvence. (Jurafsky a Martin 2025)

Princip jeho fungování spočívá v přijetí vstupní sekvence na straně kodéru a vytvoření její číselné reprezentace na základě kontextu. Reprezentace je následně předána dekodéru, který na jejím základě vygeneruje odpovídající sekvenci např. v jiném jazyce nebo jako odpověď na položenou otázku. (Jurafsky a Martin 2025) Příkladem takových modelů jsou např. BART (zkr. pro angl. *Bidirectional and Auto-Regressive Transformers*) vyvinutý společností Meta (Lewis et al. 2019) nebo T5 od společnosti Google (Raffel et al. 2023).

1.2.6 Transformátor typu dekodér

Třetím typem transformátorů je dekodér, který je dnes klíčový pro generativní umělou inteligenci ve velkých jazykových modelech. Zpracovává vstupní text (*prompt* uživatele) a na jeho základě generuje pokračování textu. Na rozdíl od RNN transformátory nezpracovávají sekvence tokenů jeden po druhém, ale pracují s celou sekvencí najednou, výpočty totiž provádí paralelně. V případě dlouhých sekvencí nezapomínají kontext, není-li text delší než velikost tzv. kontextového okna, které omezuje počet tokenů, jež model dokáže současně zpracovat. Aby model neztrácel informaci o kontextu a větných vazbách, používá tzv. poziční vektory, které nesou informaci o pozici každého tokenu ve větě. Poziční vektory se sčítají se sémantickými embeddingy tokenů a každý vektor tak nese informaci jak o významu, tak o pozici ve větě. Prostřednictvím klíčového sebepozornostního mechanismu (angl. *self-attention mechanism*) sleduje vztahy mezi všemi tokeny ve vstupní sekvenci (včetně vztahu každého tokenu k sobě samému) a hledá v nich vzory. Na základě těchto vzorů se učí, jaká slova jsou si sémanticky blízko a zda se vyskytují ve stejných kontextech. Model si tedy nepamatuje konkrétní trénovací texty, ale generalizuje z naučených vzorů. (Jurafsky a Martin 2025)

Dekodéry jsou trénovány na rozsáhlých korpusech textových dat. Podle naučených vzorů dokážou generovat další, koherentní a gramaticky správné texty v reakci na prompt uživatele. Na straně výstupu je tedy úkolem dekodéru generovat každý další token na základě dosavadního kontextu a pravděpodobností přiřazených jednotlivým slovům. Predikce tokenů probíhá zleva doprava (nebo

zprava doleva u jazyků, jako je např. arabština). Tento proces je nazýván autoregresivní generování. (Jurafsky a Martin 2025)

1.3 Velké jazykové modely

Generativní umělá inteligence (AI) obecně zahrnuje systémy, které dokážou generovat text, obrázky, zvuk či video. (Gheorghiu 2024) Velké jazykové modely (angl. *large language models*, LLM) spadají do skupiny dekodérů a představují jeden z nejpopulárnějších typů generativní AI. Excelují především ve zpracování a chápání přirozeného jazyka, generování koherentních, kohezních a kontextově relevantních textů vysoké kvality, řešení problémů a rozhodování se. (Yang et al. 2023)

1.3.1 Využití

Velké jazykové modely jsou trénovány na rozsáhlých textových korpusech obsahujících miliardy slov, díky čemuž se učí „porozumět" přirozenému jazyku a generovat nové, originální texty. (Gheorghiu 2024) „Tyto modely, jako například GPT-4, mají miliony až miliardy parametrů a jsou využívány v různých aplikacích, včetně chatbotů, strojového překladu a generování obsahu." (Česká asociace umělé inteligence z.ú., b.r.) Jsou rovněž využívány například ke shrnování textů, odpovídání na otázky nebo psaní kódu.

1.3.2 Známé modely

Mezi nejznámější LLM patří modely GPT (zkr. pro angl. *Generative Pre-trained Transformer*) od společnosti OpenAI, Llama (angl. *Large Language Model Meta AI*) modely od Mety, Claude modely od Anthropic a Gemini modely od Googlu. Tyto modely jsou známé pro jejich schopnost generovat různorodý, vysoce kvalitní obsah, jako je text, obrázky, videa, zvuk nebo zdrojový kód. Excelují také v chápání jazyka a instrukcí a v mnohých úlohách zpracování přirozeného jazyka. OpenAI poskytuje aktuálně nejznámějšího a hojně používaného chatbota ChatGPT, který je používán pro tvorbu obsahu. (Generative AI 2024)

1.3.3 Limity

Byť jsou LLM neuvěřitelně výkonné a schopné generovat texty běžně nerozpoznatelné od textů napsaných člověkem, mají také své limity, které je důležité zmínit. LLM nemají přístup k aktuálním datům (pokud nemají nástroj pro vyhledávání informací v reálném čase), ale pouze k datům, na nichž byly naposledy trénovány. Z toho důvodu mohou odpovídat na uživatelské prompty nepravdivě, přestože výstupy působí věrohodně. (Gheorghiu 2024) Často k tomu dochází v případech, kdy se uživatel ptá na události, které se odehrály po datu, kdy byl model naposledy trénován, nebo na informace, které model nikdy neviděl. Tato nespolehlivost LLM bývá označována jako „halucinování". A jelikož mohou tyto výstupy působit velmi přesvědčivě, uživatelé na ně mohou příliš spoléhat a věřit jim. (Yang et al. 2023) „Takové výstupy mohou mít vážné následky v mnohých doménách, jako je zdravotnictví, finančnictví nebo politika, kde je přesnost a spolehlivost informací kritická." (Yang et al. 2023, překlad vlastní) Je proto potřeba si informace vždy ověřovat i z jiných zdrojů a přistupovat k nim kriticky.

LLM navíc nezaručují faktickou správnost a logiku vygenerovaných odpovědí. Mohou také generovat toxický obsah, jelikož bývají trénovány na datech z internetu, která sama o sobě mohou obsahovat urážky či jinak problematický obsah. V neposlední řadě mohou mít LLM problém udržet kontext u velmi dlouhých vstupů. (Gheorghiu 2024) Ačkoliv tedy mají podstatně lepší schopnost práce s kontextem než například modely typu RNN, stále jsou i v tomto směru omezené.

1.3.4 Fine-tuning

Fine-tuning (česky jemné doladění) je jedním ze způsobů, jak předejít situacím, kdy velký jazykový model generuje nepřesné nebo zavádějící odpovědi, protože původně nebyl trénován na relevantních datech. Spočívá v dalším trénování již předtrénovaného modelu na specializovaném datasetu úzce zaměřeném na konkrétní obor, jazyk nebo typ úlohy, což umožňuje jeho přizpůsobení specifickým potřebám. Díky tomu má přizpůsobený model jak obecné znalosti, tak nově naučené, zaměřené na danou oblast. To je užitečné například v medicíně, právu nebo technických oborech, kde běžné jazykové modely často postrádají dostatečně přesné a relevantní znalosti. (Gheorghiu 2024)

Nevýhodami fine-tuningu jsou např. vysoké výpočetní a finanční náklady nebo potřeba nového rozsáhlého datasetu. Dotrénování na nových datech navíc trvale mění původní model, což znamená, že nové znalosti již nelze „vypnout“ nebo aktualizovat. To může vést ke snížení schopnosti modelu generovat obecnější odpovědi. (Gheorghiu 2024)

1.3.5 Prompt engineering

Prompt engineering (česky prompt inženýrství nebo promptování) je proces spočívající ve snaze ovlivnit výstup LLM tak, aby byl co nejpřesnější a odpovídal požadavkům uživatele, a to prostřednictvím pečlivě formulovaných promptů (vstupů). LLM prompt využívá jako klíčový kontext pro predikci dalšího textu. (Jurafsky a Martin 2025)

Správné promptování spočívá ve formulaci promptů tak, aby model jasně chápal záměr a požadavky uživatele a generoval co nejpřesnější, kontextově relevantní výstup. Uživatel by měl umět správně předvídat, jak různé způsoby formulace promptu ovlivní výstup. Prompt by neměl být dvojsmyslný a měl by poskytovat jasné a explicitní instrukce, aby nedošlo k nesprávné interpretaci. Je rovněž důležité specifikovat, zda je očekávána kreativní a obecná odpověď, nebo velice konkrétní a přesná. Prompt engineering lze považovat za jednodušší formu fine-tuningu, jelikož i zde je cílem přizpůsobit chování modelu specifickým požadavkům uživatele. (Gheorghiu 2024)

Typy promptů se dělí na systémové a uživatelské. Systémové určují chování modelu a jeho roli (např. „*Odpovídej jako odborník na lingvistiku.*“), zatímco uživatelské představují konkrétní dotazy, které model zpracovává (např. „*Vysvětli rozdíl mezi syntaktickou a sémantickou analýzou*“). Prompt by neměl mít příliš nejednoznačnou povahu (např. „*Napiš něco o strojovém učení.*“), jelikož by výstup byl příliš obecný, což by mohlo mít za důsledek nutnost dalšího doptání. Lépe formulovaný prompt by mohl znít

např. „*Vysvětli princip strojového učení a jeho proces a uveď příklady jeho využití v NLP.*“. Takový prompt poskytuje jasnější instrukce a očekávaný rozsah požadované odpovědi. (Gheorghiu 2024)

V neposlední řadě lze rozlišovat různé metody promptování, mezi které patří tzv. *few-shot* a *zero-shot* prompting. Few-shot prompting, někdy označovaný jako *k-shot prompting*, spočívá v poskytnutí několika příkladů, jak by měla odpověď vypadat. Zadávány jsou vzorové otázky a odpovědi a následně nová otázka, na niž model odpoví na základě poskytnutého vzoru. Zero-shot prompting naopak znamená, že model dostane pouze samotný dotaz bez jakéhokoliv příkladu, což je pro něj často obtížnější na zpracování. Další užitečnou metodou je *Chain-of-thought* (CoT) prompting, který je efektivní především u úloh vyžadujících logické a vícekrokové uvažování. Model v tomto případě rozkládá problém na menší části a ve výstupu popisuje svůj myšlenkový proces. (Gheorghiu 2024)

1.3.6 Sampling

Sampling nastavuje parametry, které určují, z kolika nejpravděpodobnějších možností model vybere další predikovaný token. To umožňuje vyvažovat kvalitu a variabilitu vygenerovaných odpovědí. Oproti prompt engineeringu není výstup ovlivňován explicitními požadavky, ale nastavením hodnot *top-k*, *top-p* a *temperature*.

Top-k sampling určuje, z kolika k nejpravděpodobnějších slov bude model vybírat další predikovaný token. Pokud je $k > 1$, může být náhodně vybráno slovo, které není nejpravděpodobnější, ale je právě mezi k nejpravděpodobnějšími. Např. pokud $k = 10$, model náhodně vybere jedno z deseti nejpravděpodobnějších slov.

Top-p sampling jako alternativa ponechává ta slova, jejichž kumulativní pravděpodobnost dosáhne hodnoty p. Např. pokud $p = 0.5$, model vybírá z prvních slov, která tvoří dohromady 50 % kumulativní pravděpodobnosti. (Jurafsky a Martin 2025)

Temperature sampling mění distribuci pravděpodobností slov. Tento parametr ovlivňuje variabilitu odpovědí – vyšší hodnota (> 1) vede ke kreativnějším a různorodějším odpovědím, potenciálně ale méně koherentním. Nižší hodnota (< 1) naopak zajišťuje, že stejná otázka bude mít téměř vždy stejnou odpověď, ale se zvýšeným rizikem halucinování modelu v případě, že nezná správnou odpověď. (El Amri 2023)

1.3.7 API

Aplikační programové rozhraní (angl. *Application Programming Interface*, API) je ve světě velkých jazykových modelů prostředek umožňující komunikaci mezi aplikací vývojáře a jazykovým modelem. Proces spočívá v odeslání uživatelského promptu generativnímu modelu prostřednictví API, který následně odešle nazpět výstup. Díky API tedy mohou vývojáři integrovat jazykové modely do vlastních aplikací. (El Amri 2023)

Při využívání API je důležité brát v úvahu cenovou dostupnost, jelikož ne všechna API jsou zdarma. Například OpenAI nabízí své modely výhradně prostřednictvím placeného API, zatímco Google poskytuje některé modely zdarma a jiné za poplatek (v obou případech závisí cena na počtu požadavků za den či minutu). Naopak např. Llama modely od Mety lze využívat zcela zdarma bez výraznějších omezení.

Společnou vlastností těchto modelů je, že jejich API je veřejně přístupné. V případě placených API se cena obvykle uvádí v dolarech za zpracované tokeny, přičemž u některých API se liší cena vstupních a výstupních tokenů. Náklady na využívání API tedy závisí jak na délce promptu, tak na délce odpovědi. Tuto skutečnost je rovněž potřeba zohlednit při prompt engineeringu, jelikož délka vstupu přímo ovlivňuje cenu jednoho volání API. V promptu lze přibližně specifikovat očekávanou délku výstupu, což umožňuje cenu kontrolovat.

1.4 Chunking

Dle definice na webu GeeksForGeeks (2024) je chunking proces segmentace textů na menší části, jako jsou například věty, odstavce nebo jednoduše části textů o předem určené délce. Chunking je využíván pro dosažení přesnějších výsledků při analýzách textů, např. při klasifikaci nebo analýze sentimentu, neboť kratší texty se snáze zpracovávají a lépe zachovávají informaci o významu při převodu na sémantické embeddingy.

1.4.1 Způsoby chunkingu

Existuje několik způsobů, jak lze texty segmentovat. Nejjednodušším z nich je segmentace podle předem určené fixní délky, tedy podle počtu znaků nebo tokenů. Dále lze texty segmentovat na věty nebo na odstavce. Segmentaci lze také provádět např. podle sémantické blízkosti nebo obsahu. V jazyce Python je k dispozici řada knihoven umožňujících provádět chunking bez nutnosti vytvářet vlastní funkce, např. LangChain nebo LlamaIndex.

1.4.2 Velikost chunků

Sémantický embedding získaný z příliš krátkého textu (chunku) nemusí vždy zachycovat kompletní relevantní informaci nebo myšlenku. Dlouhý text, např. celý dokument, naopak obsahuje velké množství informací a kontextů, a protože se mnoho různých významů při převodu slévá do jednoho embeddingu, může to mít za následek ztrátu informací. Typické velikosti chunků bývají 128, 256, 512, 1024 nebo 2048 tokenů.

Délka promptu pro generativní model má vliv jak na potřebný výpočetní výkon, tak na výši finančních nákladů v případě využívání placeného API. Pokud je tedy modelu zadán prompt obsahující např. tři chunky textů, na základě nichž má zodpovědět dotaz, může delší vstup značně zvýšit cenu. Je taktéž potřeba brát v potaz omezená kontextová okna modelů, jimž je vhodné velikost chunků přizpůsobit.

Často se u chunkingu také používá tzv. *overlap* (přesah), který určuje, do jaké míry se jednotlivé chunky překrývají. Díky tomu se při textové analýze minimalizuje riziko ztráty klíčových informací, které by se mohly nacházet na hranicích chunků. (Roie Schwaber-Cohen 2023)

1.5 Vektorové databáze

Vektorové databáze slouží k ukládání vektorových reprezentací textů, přesněji sémantických embeddingů. Na rozdíl od tradičních relačních databází, ve kterých jsou relevantní data vyhledávána na základě přesné shody (např. textového řetězce nebo číselné hodnoty), probíhá vyhledávání relevantních dat ve vektorových databázích na základě výpočtu podobnosti mezi vektory. Tento proces využívá algoritmus ANN (angl. *Approximate Nearest Neighbor*, česky přibližný nejbližší soused), který umožňuje rychlé hledání vektorů s největší sémantickou podobností k hledanému dotazu. (Moez Ali 2025)

1.5.1 Využití

Vektorové databáze jsou využívány například v doporučovacích systémech e-shopů. Díky nim může uživatel dostávat návrhy na produkty nejen na základě dřívějších nákupů, ale i na základě podobnosti vlastností produktů, chování uživatele a preferencí. Ve finanční analýze napomáhají detekovat vzorce v investičních strategiích a ve zdravotnictví umožňují přizpůsobení léčby pacienta prostřednictvím analýzy genomických sekvencí tak, aby lépe odpovídala jeho genetické výbavě. V aplikacích zpracování přirozeného jazyka slouží k vyhledávání obsahově nejpodobnějších textů převedených na sémantické embeddingy. Mezi populární vektorové databáze patří Chroma, Pinecone, Weaviate, Faiss, Qdrant, Milvus a pgvector, přičemž každá z nich je vhodná pro jiný typ aplikací. (Moez Ali 2025)

1.5.2 Mechanismus

Princip práce s vektorovou databází spočívá v několika krocích. Například v případě databáze Chroma jsou nejprve do databáze vložena textová data spolu s jejich metadaty. Tyto texty jsou následně převedeny na sémantické embeddingy pomocí funkce, v jejímž rámci je specifikováno, jaký jazykový model bude použit pro vektorizaci textu. Při zadání dotazu do databáze je tento dotaz rovněž převeden na sémantický embedding. Databáze následně vrátí sémanticky nejbližší texty seřazené podle míry podobnosti. Tento princip umožňuje efektivní vyhledávání v dokumentech na základě významu a kontextu, což je klíčové pro systém RAG, v němž je vektorová databáze jednou z nezbytných komponent.

1.6 Retrieval-Augmented Generation

Načítání rozšířené generace (angl. *Retrieval-Augmented Generation*, RAG) je metoda vyvinutá výzkumníky z Facebook AI (P. Lewis et al. 2021), která kombinuje vyhledávání relevantních dat na základě uživatelského promptu ve vektorových databázích a generování odpovědí pomocí generativního modelu. Systém RAG umožňuje jazykovému modelu pracovat s aktuálními a externími daty, což snižuje riziko

halucinací při zodpovídání dotazů, na něž model nezná odpověď, jelikož na relevantních datech nebyl trénován. (P. Lewis et al. 2021)

1.6.1 Výhody

Je-li do promptu vloženo několik celých dokumentů s požadavkem vyhledat konkrétní informaci, generativní model by ji v důsledku slévání informací do jednoho embeddingu a omezené velikosti kontextového okna nemusel najít. Systém RAG tento problém řeší využitím chunkingu, protože vektorová databáze většinou obsahuje chunky textů a nikoliv celé dokumenty. V systému RAG jsou tak do promptu vkládány jen nejrelevantnější informace z částí dokumentů. Tím je také ušetřen výpočetní výkon a čas, případně i finanční náklady, pokud je používáno placené API.

Systém RAG se nespoléhá pouze na znalosti generativního modelu, ale umožňuje práci se specificky zaměřenými informacemi, na nichž model původně nebyl trénován. Díky přístupu k externím datům relevantním k promptu je pravděpodobnější, že model odpoví správně. (Gheorghiu 2024) RAG tak lze využít například ve firemním prostředí, kde umožňuje generativnímu modelu pracovat s interními firemními daty. Teoreticky lze RAG chápat jako personalizovanou alternativu fine-tuningu, která nevyžaduje výpočetně a finančně nákladné dotrénování modelu ani nová trénovací data.

1.6.2 Limity

Přestože má systém RAG oproti fine-tuningu mnoho výhod, není bez omezení. Podstatným limitem je volba správné velikosti chunků – jak již bylo zmíněno, malé chunky textů mohou postrádat klíčové informace, zatímco velké mohou ztrácet kontext a obsahovat irelevantní části, které tvoří informační šum. Jelikož model generuje odpověď na základě poskytnutých chunků textů, neobsahují-li dostatečně relevantní a obsahově bohaté informace, nebude vygenerována ani správná odpověď. Jsou-li navíc v promptu použity například tři nejrelevantnější nalezené chunky namísto jednoho, bude kontext třikrát delší, což samo o sobě může vést ke ztrátě informace, přitom právě správná a jednoznačná formulace promptu je klíčovou pro získání žádoucího výstupu.

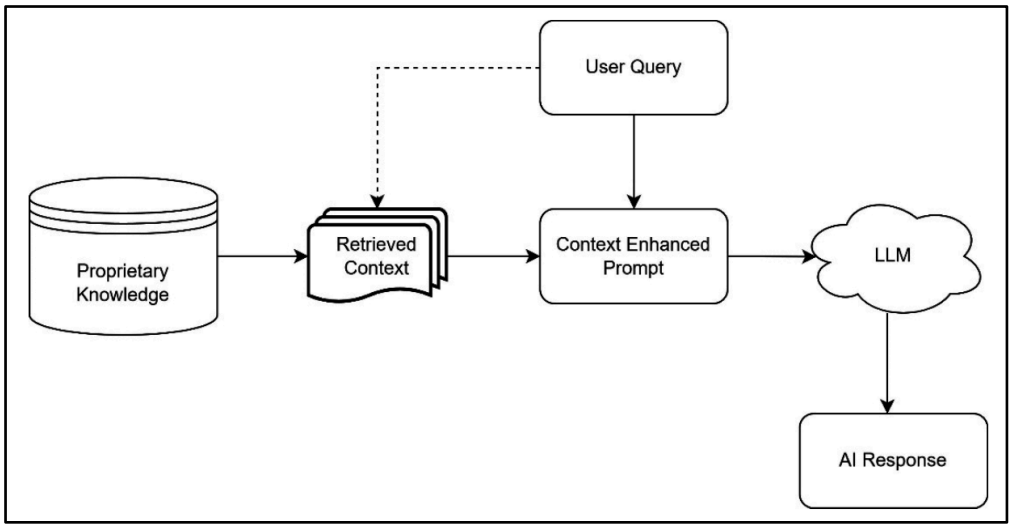
1.6.3 Implementace

Systém RAG se skládá ze tří hlavních komponent:

- *retrieval* (vyhledávání relevantních informací) – na základě uživatelského dotazu jsou ve vektorové databázi vyhledány nejrelevantnější chunky textů,
- *augmentation* (rozšíření promptu) – získané chunky jsou zkombinovány s původním dotazem,
- *generation* (generování odpovědi) – model vygeneruje odpověď na základě dotazu, instrukcí a kontextu v promptu. Neodpovídá tedy pouze na základě dat, na nichž byl trénován, ale využívá i externí informace. (CustomGPT.ai, b.r.)

Celý proces lze rozložit do následujících kroků:

1. Předzpracování textů – zejména chunking, během něhož je specifikována velikost chunků a případný overlap.
2. Převod textů na embeddingy – každý chunk je převeden na sémantický embedding pomocí zvoleného modelu.
3. Uložení do vektorové databáze – embeddingy se uloží do databáze (např. Chroma, Faiss, Qdrant), která umožňuje rychlé a efektivní vyhledávání na základě sémantické podobnosti.
4. Vyhledání relevantních textů – po zadání promptu je dotaz rovněž převeden na embedding a pomocí algoritmů ANN jsou v databázi vyhledány nejpodobnější chunky textů.
5. Předání informací generativnímu modelu – získané chunky jsou jakožto kontext spolu s dotazem vloženy do promptu.
6. Generování odpovědi – model na základě promptu (kontextu a samotného dotazu) vygeneruje odpověď.



Obrázek 1 – schéma systému RAG (Gheorghiu 2024)

1.7 Měření podobnosti textů

Podobnost textů lze měřit pomocí různých metrik, které slouží k výpočtu vzdálenosti mezi vektory, do nichž byly texty převedeny. Výsledná číselná hodnota odráží míru blízkosti vektorů a závisí na dvou faktorech – vlastnostech porovnávaných vektorů (tj. původních objektů) a zvolené metrice. (Anna Huang 2008)

1.7.1 Euklidovská vzdálenost

Euklidovská, nebo také L2 vzdálenost je standardní metrika používaná v geometrii i v oblasti strojového učení, například ve shlukovacích algoritmech jako je *k-means*. Tato vzdálenost představuje délku nejkratší možné cesty mezi dvěma body v n-dimenzionálním prostoru. Ve dvou- nebo třídimenzionálním prostoru ji lze snadno změřit pomocí pravítka jako úsečku mezi dvěma body. (Anna Huang 2008)

Pro měření vzdálenosti mezi sémantickými embeddingy se využívá méně často, protože nezohledňuje směr vektorů a úhel mezi nimi, ale pouze jejich absolutní rozdíl. Může však být vhodná, pokud jsou embeddingy předem znormalizovány tak, aby měly L2 normu (tj. délku) rovnu 1. Např. vektorová databáze Faiss tuto normalizaci umožňuje, hledání nejbližších vektorů proto vrací stejné výsledky jako při použití kosinové podobnosti. (Douze et al. 2024)

1.7.2 Kosinová podobnost

Kosinová podobnost je metrika, která určuje míru podobnosti mezi dvěma vektory na základě úhlu, který mezi sebou svírají v n-dimenzionálním prostoru. Nezohledňuje délku vektorů, ale pouze jejich směr a jak jsou orientovány vůči sobě. Hodnota podobnosti se pohybuje mezi 0 a 1: identické vektory svírají úhel 0° a mají podobnost 1; kolmé vektory svírají úhel 90° a mají podobnost 0. (El Amri 2023)

Tato metrika je běžně používaná pro měření podobnosti textů (jejich sémantických embeddingů), protože zachycuje jejich významovou blízkost podle orientace vektorů. Ve vektorových databázích je proto pro hledání nejpodobnějších textů většinou používána právě kosinová podobnost – čím vyšší je výsledná hodnota (čím blíže 1), tím podobnější je text zadanému dotazu.

2 Experimentální část: evaluace embedding modelů

Cílem první fáze experimentální části práce bylo identifikovat nejlepší jazykové modely pro vektorizaci českých textů – jeden s bezplatným a jeden s placeným API. Tyto modely byly následně použity pro testování systému RAG v druhé fázi. Po získání baseline výsledků modelu fastText bylo otestováno 21 vybraných embedding modelů s použitím vektorové databáze Chroma. Hlavním cílem testování bylo ověřit, zda nejpodobnější nalezené texty, respektive jejich chunky, poskytují správné odpovědi na předem definované otázky, které byly formulovány s ohledem na obsah textů.

Prostřednictvím ChatGPT byl vygenerován textový dataset o 20 článcích. V Pythonu byly texty následně rozděleny na chunky, na jejichž základě byly vytvořeny odpovídající otázky, které byly zaznamenány do tabulky. V rámci testování byly názvy chunků, které daný model označil jako nejpodobnější, porovnávány s názvy správných odpovídajících chunků. V případě shody názvů se zvýšilo skóre přesnosti daného modelu. Výsledná přesnost jednotlivých modelů byla v procentech zapsána do tabulky v CSV formátu a vizualizována pomocí spojnicového grafu.

2.1 Využitá data pro testování modelů a systému RAG

2.1.1 Volba typu textových dat

Zprvu by se mohlo zdát, že je vhodné použít veřejně dostupné texty obsahující fakticky správné informace, u nichž lze odpovědi spolehlivě ověřit. Je však nutné vzít v potaz, že jazykové modely jsou trénovány převážně právě na takových datech. V první fázi experimentu příliš nezáleželo na volbě modelu a textových dat, protože šlo pouze o měření kosinové podobnosti mezi embeddingy. Cílem druhé fáze experimentu však bylo zjistit, zda generativní model dokáže zodpovídat otázky na základě externích dat, nebylo by tudíž žádoucí, aby měl možnost potenciálně odpovídat na základě vlastních znalostí z trénovacích dat. Měřit podobnost textů lze tedy na téměř jakýchkoliv textech, nicméně pro následné testování systému RAG bylo vhodnější volit data, která generativní model nikdy dříve neviděl. Z tohoto důvodu byla v této práci použita data fiktivní.

2.1.2 Tvorba a uložení dat

Pro vytvoření fiktivních textů byl použit chatbot ChatGPT od OpenAI, jenž vygeneroval 20 smyšlených novinových článků s nepravdivými informacemi. Aby nebylo mezi embeddingy příliš snadné rozpoznat, který text odpovídá které otázce, kdyby se každý text týkal zcela odlišného tématu, byly vygenerovány dvě tematické sady po deseti článcích o zhruba 500 slovech (zhruba, jelikož generativní model zpravidla nedokáže přesně dodržet požadovaný rozsah). Tématy byly *„vesmírné nálezy“* a *„poznatky a novinky ve světě AI“*.

Cílem druhé fáze experimentální části práce bylo rovněž otestovat, zda bude generativní model v systému RAG správně odpovídat v situacích, kdy data v databázi odporují datům, která už model zná.

Z tohoto důvodu byly články zasazeny maximálně do roku 2021. Tato kritéria byla zvolena za předpokladu, že modely od OpenAI budou v testech dosahovat nejlepších výsledků a následná práce bude prováděna právě s nimi (byť v první fázi zaměřené na měření podobnosti textů nezáleží na tom, v jakém období se události z článků odehrály).

Při formulování promptu pro ChatGPT bylo dbáno na jednoznačnost a stručnost instrukcí. Aby byly informace ve vygenerovaném článku rovnoměrně rozloženy, bylo modelu zabráněno, aby přidával jakékoliv typy shrnutí, jejichž části by potom mohly modelům sloužit jako informačně bohatší a jednodušší na vyhledání. Prompt měl následující podobu:

> "Vygeneruj smyšlený novinový článek o 500 slovech. Článek nemá mít žádné složité formátování, nepiš úvod, závěr, ani žádný jiný typ shrnutí. Nadpis bude co nejkratší.
> Článek se bude týkat tématu ‚*XY*'. Piš o objevech, novinkách nebo událostech, které se nikdy nestaly a jsou časově zasazeny maximálně do roku 2021. Můžeš používat jména a názvy věcí, které existují. Obsah textu si můžeš vymyslet nebo ho zkombinovat s reálnými informacemi, ty však nesmí odporovat tomu, co sis vymyslela.
> Pod článek napiš otázku, na niž lze odpovědět na základě informací z tvého článku. Nepoužívej v ní nutně názvy."

Protože model často generoval texty kratší než 500 slov, bylo v některých případech potřeba zadat dodatečně prompt „*Článek musí být delší.*“, na jehož základě model vygeneroval stejný článek v delší verzi. Po každém získaném článku byl zadán prompt „*Vymysli další článek na téma ‚XY'.*“ Po deseti vytvořených článcích byla založena nová konverzace k vygenerování článků s druhým tématem. Články byly postupně ukládány do samostatných textových souborů.

2.1.3 Definice otázek a požadovaných odpovědí

V experimentech bylo možné pracovat buď s celými texty, nebo pouze s jejich částmi (chunky). Práce s celými texty by však vyžadovala expertní znalost vymyšlených článků, aby bylo možné formulovat otázky na základě celkového kontextu každého textu. Byl proto použit chunking, což umožnilo vytvoření konkrétnějších otázek a přesnější vyhodnocování shod otázek s odpověďmi.

	Výhody	Nevýhody
Chunking	Správné odpovědi lze jednoznačně přiřadit chunkům	Pro přesné měření se odpověď musí vyskytnout pouze v jednom chunku
	Konzistentní porovnávání díky stejné délce chunků	Nelze se ptát na otázky vyplývající z celého textu
	Zachování významu v embeddingách	Ztráta širšího kontextu v embeddingách
	Snazší na zpracování a analýzu	Nutnost zvolit velikost chunku
Celé texty	Zachování kontextu	Potenciální ztráta konkrétních významů
	Vhodnější pro obecné dotazy	Vyšší výpočetní náročnost
	Není potřeba dělit a odlišovat části textu	Těžší evaluace správnosti odpovědí
		Nutnost znalosti kontextu a přesného obsahu všech textů

Tabulka 1 – srovnání výhod a nevýhod chunkingu oproti používání celých textů

V Pythonu byly textové dokumenty rozděleny na chunky o předem definované velikosti 128 slov. Velikost byla zvolena s ohledem na délku textů, která se pohybovala v rozmezí zhruba 400–700 slov. Byl nastaven overlap 20 slov, aby se minimalizovalo riziko ztráty důležitých informací na hranicích chunků. Použití fixní velikosti umožnilo chunky uložit, označit unikátním názvem a identifikátorem a jednoznačně k nim přiřadit otázky. Díky tomu bylo možné automatizovaně srovnávat nalezené nejpodobnější chunky se správnou odpovědí podle shody názvů.

Chunky textů	**Počet**
Celkově	104
Průměr na jeden text	5,2

Tabulka 2 – celkový počet chunků a průměrný počet chunků pro jeden text

	Znaky	**Slova**
Průměrná délka	781,21	114,66
Medián	865	128
Směrodatná odchylka	214,84	30,4
Minimální délka	67	8
Maximální délka	984	128

Tabulka 3 – popisná statistika délek chunků (ve znacích a slovech)

Po kvalitativním zhodnocení otázek vygenerovaných pomocí ChatGPT a přečtení textů byla většina otázek přepsána, neboť se často zaměřovaly na velmi úzké části textu a chunky by na základě jejich sémantické podobnosti mohlo být příliš jednoduché najít. Některé otázky se naopak zaměřovaly na kontext celého textu a nebylo by možné je zodpovědět na základě jediného chunku a tudíž ani potvrzovat nebo vyvracet, zda je nejbližší nalezený chunk správný.

Otázky byly znovu a definitivně navrženy až po rozdělení textů na chunky. Díky tomu bylo možné jednoznačně označit, zda se v chunku vyskytuje správná odpověď, a tudíž i evaluovat přesnost daného modelu. Cílem bylo, aby nalezení správných odpovědí představovalo pro modely výzvu, ale zároveň aby jejich nalezení nebylo nemožné a bylo možné na otázky stále jednoznačně na základě chunku odpovědět. Většina otázek proto neobsahovala explicitní názvy a klíčová slova z článků. Naopak některé otázky se na názvy implicitně ptaly. Otázky byly sepsány do samostatného textového souboru a zároveň do tabulky v Excelu, kde k nim byly přiřazeny příslušné názvy chunků získané v Pythonu.

Název otázky	Název textu	Otázka	Odpovídající chunk
01VN_otazka	01VN_Neptun	Jaký byl výsledek měření teploty na povrchu objektu na okraji Sluneční soustavy a co to naznačuje?	01VN_Neptun_2
02VN_otazka	02VN_Tajemny_signal	Existují nějaké známé přírodní jevy, které by tajemný signál u Proximy Centauri mohly způsobit?	02VN_Tajemny_signal_3
03VN_otazka	03VN_Podzemni_ocean	Jaké chemické sloučeniny byly na povrchu jednoho z měsíců Sluneční soustavy detekovány?	03VN_Podzemni_ocean_1
04VN_otazka	04VN_Zablesky	Jakým teleskopem budou vědci zkoumat podrobněji světelné impulzy vycházející z oblasti černé díry v galaxii M87?	04VN_Zablesky_4
05VN_otazka	05VN_Podzemni_jezera	Byl na Marsu někdy život?	05VN_Podzemni_jezera_5

Tabulka 4 – prvních pět otázek s názvy textů, jimž otázky náleží, a názvy chunků zodpovídajících otázky

2.2 Specifikace nástrojů a metod

2.2.1 Volba vektorové databáze

Pro vyhledávání relevantních textů byla zvolena vektorová databáze ChromaDB vzhledem k její jednoduché implementaci v Pythonu. ChromaDB umožňuje snadno měnit jednotlivé embedding modely a generovat nové sémantické embeddingy. Umožňuje také přidání metadat k dokumentům, díky čemuž lze snadněji označovat správné odpovědi. Oproti např. FAISS nebo Weaviate databázím je jednodušší na správu dat a je uživatelsky více intuitivní.

2.2.2 Výběr embedding modelů

V baseline testu byl použit model fastText, následně byly testovány veřejně dostupné jazykové modely typu transformátor, jejichž tvůrci většinou poskytují přístup přes API – ať už zdarma, ve zkušební (trial) verzi, nebo za menší poplatek. Prostřednictvím těchto modelů lze převádět data do vektorové reprezentace, nelze jimi však generovat nová data. Testovány byly všechny vícejazyčné embedding modely, které lze snadno použít v ChromaDB pomocí funkce EmbeddingFunction, jež umožňuje volání API a převod textů na embeddingy, většinou bez nutnosti definovat vlastní funkce. Byly použity vícejazyčné modely od následujících vývojářů:

- Hugging Face: Společnost poskytující knihovnu SentenceTransformers[1], která obsahuje velké množství jazykových modelů vytvořených různými vývojáři, z nichž některé jsou trénovány na vícejazyčných datech. V ChromaDB byly konkrétně použity modely Distiluse a Paraphrase.
- Cohere: Společnost poskytující v bezplatné trial verzi (jejíž limity jsou uvedeny v Tabulce 5) tři vícejazyčné modely – dva ve verzi 3.0 (základní verze, nebo menší a rychlejší light verze) a jeden ve starší verzi 2.0[2].
- Jina: Společnost poskytující dva vícejazyčné modely, které jsou v případě nekomerčního použití dostupné v trial verzi zdarma do limitu 1 milion tokenů na IP adresu.[3]
- Google: Společnost poskytující platformu Google Vertex AI, která obsahuje mnoho nástrojů pro práci v oblasti datové analýzy a generativní umělé inteligence. Poskytuje dva vícejazyčné modely. Proběhl pokus o implementaci nového experimentálního modelu *gemini-embedding-exp-03-07*, který je trénován přímo na Gemini (Logan Kilpatrick, Zach Gleicher, a Parashar Shah 2025), měl však pro tuto práci příliš nízký limit požadavků za minutu v bezplatné verzi, a to i přesto, že Google uvádí vyšší. Pro experimenty byl použit model *text-multilingual-embedding-002*, k jehož implementaci byla vytvořena vlastní funkce. Ve bezplatné verzi je limitovaný viz Tabulka 5.
- OpenAI: Společnost poskytující tři vícejazyčné modely, dva ve verzi 3 a jeden starší. Embeddingy lze získat prostřednictvím placeného API po zakoupení kreditů. Cena za milion vstupních tokenů se pohybuje od 0,02 dolarů (přibližně půl koruny) do 0,13 dolarů (přibližně 3 koruny), v závislosti na konkrétním modelu.[4]

[1] Dokumentace dostupná na https://huggingface.co/sentence-transformers.

[2] Informace získané z https://docs.cohere.com/v2/docs/cohere-embed.

[3] Informace získané z https://jina.ai/.

[4] Kompletní ceník dostupný na https://platform.openai.com/docs/pricing.

API	Limity
Cohere	100 požadavků za minutu / 1 tisíc požadavků za měsíc[5]
Jina	1 milion tokenů vázaný na IP adresu, poté nutnost zakoupení dalších tokenů[6]
Google	1 500 požadavků za minutu / 20 tisíc tokenů na požadavek[7]
OpenAI	1 milion tokenů za minutu / 3 miliony tokenů za den / 3 tisíce požadavků za den, nejprve nutnost zakoupení kreditu min. za 5 dolarů (bez trial verze)[8]

Tabulka 5 – omezení tokenů a požadavků u placených modelů v trial verzi

Bylo také experimentálně testováno několik modelů trénovaných pouze na angličtině s předpokladem, že jejich výsledky na češtině nebudou příliš přesné. Jednalo se o následující modely:

- *all-MiniLM-L6-v2*, multi-qa-distilbert-cos-v1, multi-qa-MiniLM-L6-cos-v1 a multi-qa-mpnet-base-dot-v1 z knihovny SentenceTransformers,
- *text-embedding-004* od Googlu,
- *snowflake-arctic-embed* od Snowflake,
- *nomic-embed-text* od NomicAI,
- *mxbai-embed-large* od MixedBread AI,
- *distiluse-base-multilingual-cased-v1* z knihovny SentenceTransformers trénovaný na 14 jazycích, mezi něž však nepatří čeština.

2.2.3 Metoda měření výsledků

Pro hodnocení přesnosti modelů bylo nejprve nutné stanovit způsob měření podobnosti textů s otázkami. Pro tuto práci byla zvolena kosinová podobnost, která se běžně používá pro měření podobnosti textů. V baseline testu byl výpočet proveden pomocí metody z knihovny *scikit–learn*, která je využívána v aplikacích strojového učení (Pedregosa et al. 2012). Při testování dalších modelů pak byly výpočty prováděny prostřednictvím databáze ChromaDB.

V Pythonu byly vytvořeny funkce pro zaznamenání počtů shod názvů nejpodobnějších chunků s názvem správného chunku. Stoprocentní přesnost v TOP 1 by znamenala, že ve všech dvaceti případech obsahoval první nejpodobnější chunk správnou odpověď. Stoprocentní úspěšnost v TOP 5 by znamenala, že se ve všech případech vyskytla správná odpověď mezi prvními pěti nejpodobnějšími chunky. Výsledky každého modelu byly postupně zapisovány do tabulky ve formátu CSV.

[5] Informace získané z https://docs.cohere.com/v2/docs/rate-limits.

[6] Informace získané z https://jina.ai/embeddings.

[7] Informace získané z https://cloud.google.com/vertex-ai/generative-ai/docs/quotas.

[8] Informace získané z https://platform.openai.com/settings/organization/limits.

2.3 Baseline test modelu fastText

Pro stanovení baseline hranice přesnosti byl otestován jazykový model fastText, který poskytuje sémantické embeddingy českých slov. Jedná se o nejstarší z testovaných modelů, jenž je trénován prostřednictvím klasické dopředné neuronové sítě a neumí pracovat s kontextem. Test byl tedy proveden s předpokladem, že model bude nejméně přesný oproti novějším modelům typu transformátor.

2.3.1 Výsledek měření

Správný chunk se na prvním místě, tedy jakožto nejpodobnější, vyskytl dvakrát, perfektní úspěšnost modelu tedy nastala v 10 % případů. Na druhém nebo třetím místě se správná odpověď vyskytla 7krát a na čtvrtém nebo pátém se vyskytla 4krát. Celkově se mezi prvními pěti nejpodobnějšími texty vyskytla správná odpověď 13krát z dvaceti. Model fastText tak v tomto testu dosáhl kumulativní přesnosti 65 %. Stejným způsobem byly následně analyzovány i další modely.

2.4 Test modelů s vektorovou databází

Cílem testu bylo identifikovat mezi modely od SentenceTransformers, Cohere, Jina, Google Vertex AI, OpenAI, Snowflake, MixedBreadAI a NomicAI nejpřesnější model pro češtinu. Vektorová databáze byla v každém kroku naplněna textovými dokumenty, respektive jejich chunky. Databáze je následně vektorizovala pomocí příslušeného embedding modelu a vrátila *n* (v tomto případě pět) chunků nejpodobnějších ke každé zadané otázce. Výpočet přesnosti byl následně proveden stejným způsobem jako v baseline testu.

Navržený program umožňuje automatizované vyhodnocování přesnosti vyhledávání relevantních dokumentů ve vektorové databázi. Díky tomu vznikla opakovatelná metoda pro výběr embedding modelu pro libovolná textová data. Schéma níže znázorňuje příklad obsahu JSON souboru, v jakém byly ukládány výsledky pro každou otázku. Program a výsledky jsou dostupné v GitHub repozitáři uvedeném v Příloze 1.

```
[
   {
      "question_name": "01VN_otazka",
      "question": "Jaký byl výsledek měření teploty na povrchu objektu na okraji Sluneční soustavy a co to naznačuje?",
      "returned_answers": [
         {
            "rank": 1,
            "chunk_name": "10VN_Kus_zdi_2",
            "chunk_text": "co jsme dosud objevili…"
         },
         {
            "rank": 2,
            "chunk_name": "01VN_Neptun_3",
            "chunk_text": "soustavě…"
         },
      …
         }
       ]
   },
….
]
```


2.5 Výsledky testů

Testem sémantického vyhledávání ve vektorové databázi byl potvrzen předpoklad, že modely od OpenAI budou dosahovat nejvyšší přesnosti. Těsně za prvním nejpřesnějším modelem od OpenAI však zaujal druhé i čtvrté místo model od Jina. Oproti OpenAI modelům lze Jina modely do limitu 1 milionu tokenů používat bezplatně, je tedy překvapivé, že dosahují podobné přesnosti jako výhradně placené modely (byť ani modely od OpenAI nejsou finančně příliš nákladné – jedna vektorizace textů v rámci této práce stála v přepočtu na koruny asi 15 haléřů).

Žádný z modelů nedokázal ve 100 % případů identifikovat správnou odpověď jako první nejpodobnější chunk. V tom může nicméně hrát roli i skutečnost, že chunk mohl obsahovat alternativní odpověď, která by také otázku mohla zodpovídat, ač jiným způsobem. Jelikož se otázky často týkaly podobných témat nebo na ně bylo možné odpovědět alespoň částečně i pomocí jiných chunků, je očekávatelné, že správná odpověď bude například až druhá nejpodobnější, neboť může zčásti obsahovat správné informace. Je proto důležité brát ohled především na přesnost v TOP 3 nebo TOP 5, protože do promptu v systému RAG nebude vždy vkládán pouze první, nejpodobnější výsledek.

Modely *text-embedding-3-large* a *jina-embeddings-v3* identifikovaly správnou odpověď na prvním místě v 11 a 12 případech z 20, zároveň však měly 100% úspěšnost v TOP 5. Model *text-embedding-3-large* měl navíc 100% přesnost i v TOP 3. Nejpřesnějším bezplatným modelem byl *paraphrase-multilingual-mpnet-base-v2* s přesností 70 % v TOP 5, který identifikoval správnou odpověď jako první nejpodobnější pouze ve 4 případech. Je překvapivé, že se svou přesností blížil modelu fastText, jenž poskytuje statické embeddingy. Oba tyto modely jsou oficiálně trénované na češtině, přičemž Paraphrase model je typu transformátor, bylo by tedy očekávatelné, že dosáhne vyšší přesnosti.

Tabulka 6 prezentuje výsledky přesnosti všech testovaných modelů. Je seřazena podle hodnot TOP 5, při shodě hodnot pokračuje řazení podle hodnot TOP 3, případně pak hodnot TOP 1.

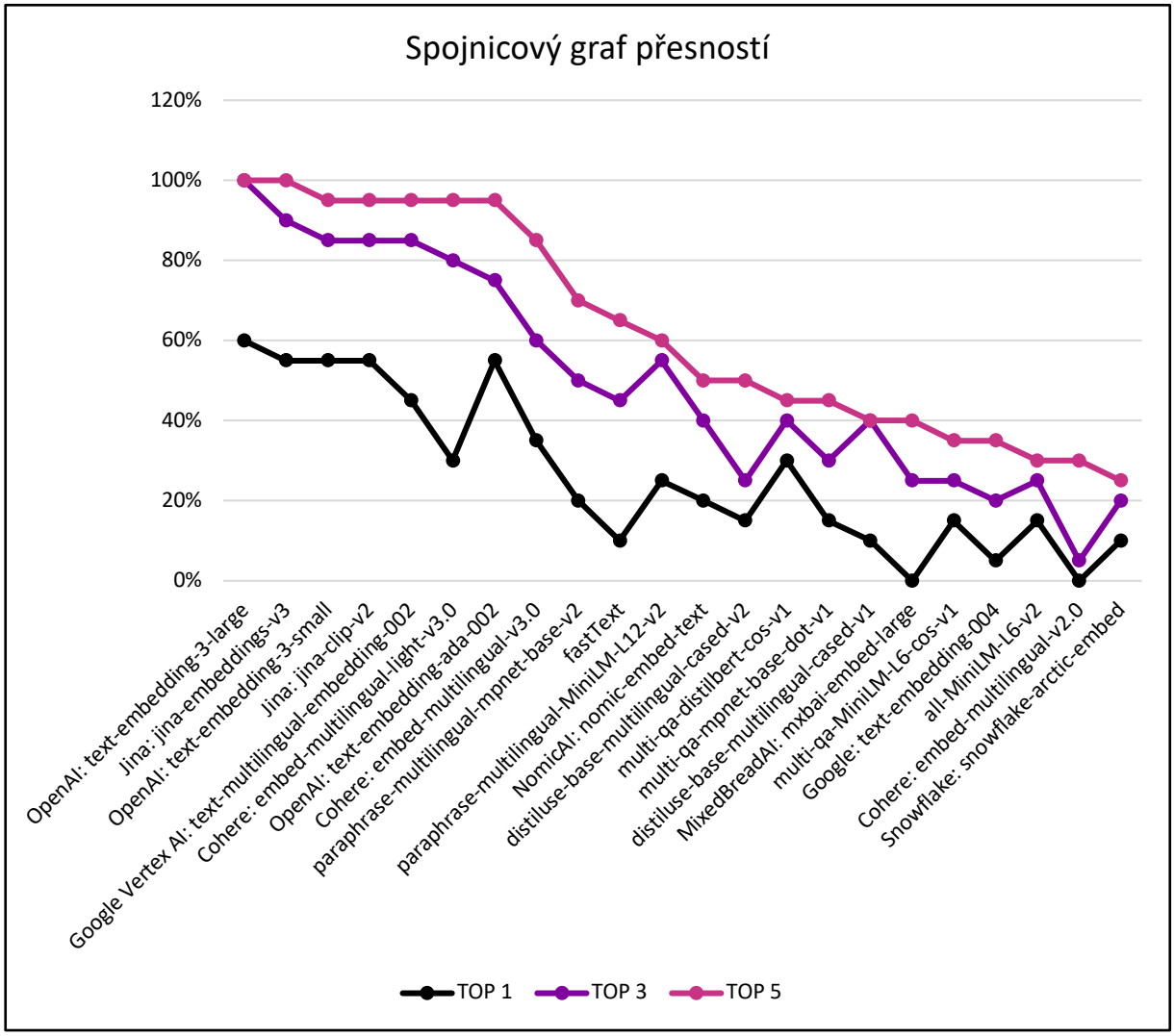
Zdarma	Model	Dimenze	TOP 1	TOP 3	TOP 5
	OpenAI: text-embedding-3-large	3072	60%	100%	**100%**
	Jina: jina-embeddings-v3	1024	55%	90%	**100%**
	OpenAI: text-embedding-3-small	1536	55%	85%	**95%**
	Jina: jina-clip-v2	1024	55%	85%	**95%**
	Google Vertex AI: text-multilingual-embedding-002	3072	45%	85%	**95%**
	Cohere: embed-multilingual-light-v3.0	384	30%	80%	**95%**
	OpenAI: text-embedding-ada-002	1536	55%	75%	**95%**
	Cohere: embed-multilingual-v3.0	1024	35%	60%	**85%**
✓	**paraphrase-multilingual-mpnet-base-v2**	768	20%	50%	**70%**
✓	**fastText**	300	10%	45%	**65%**
✓	**paraphrase-multilingual-MiniLM-L12-v2**	384	25%	55%	**60%**
✓	**NomicAI: nomic-embed-text**	768	20%	40%	**50%**
✓	**distiluse-base-multilingual-cased-v2**	512	15%	25%	**50%**
✓	**multi-qa-distilbert-cos-v1**	768	30%	40%	**45%**
✓	**multi-qa-mpnet-base-dot-v1**	768	15%	30%	**45%**
✓	**distiluse-base-multilingual-cased-v1**	512	10%	40%	**40%**
✓	**MixedBreadAI: mxbai-embed-large**	512	0%	25%	**40%**
✓	**multi-qa-MiniLM-L6-cos-v1**	384	15%	25%	**35%**
	Google: text-embedding-004	768	5%	20%	**35%**
✓	**all-MiniLM-L6-v2**	384	15%	25%	**30%**
	Cohere: embed-multilingual-v2.0	768	0%	5%	**30%**
✓	**Snowflake: snowflake-arctic-embed**	1024	10%	20%	**25%**

Tabulka 6 – výsledky měření přesnosti modelů

Spojnicový Graf 1 znázorňuje konzistentní přesnost placených modelů a následný výrazný propad u přesnosti bezplatných. Tento trend naznačuje výraznou souvislost mezi přesností a cenou modelů. V křivce hodnot TOP 1 lze navíc pozorovat postupný propad i u placených modelů, avšak s výjimkou modelu od OpenAI, který svou přesností opět vyniká nad ostatními.

Je důležité zdůraznit, že model fastText, jenž poskytuje pouze statické sémantické embeddingy, které byly pro věty zprůměrovány, dosáhl vyšší přesnosti než většina bezplatných modelů typu

transformátor, včetně těch vícejazyčných. Tento výsledek ukazuje, že novější a pokročilejší technologie automaticky nezaručují lepší výsledky, alespoň co se týče vícejazyčných modelů použitých na češtině. Použitý fastText model byl trénován pouze na češtině, což naznačuje, že model jakéhokoliv typu musí být důkladně natrénován na konkrétním jazyce, aby dosahoval uspokojivé přesnosti.



Graf 1 – graf přesností všech modelů v kategoriích TOP 1, 3 a 5

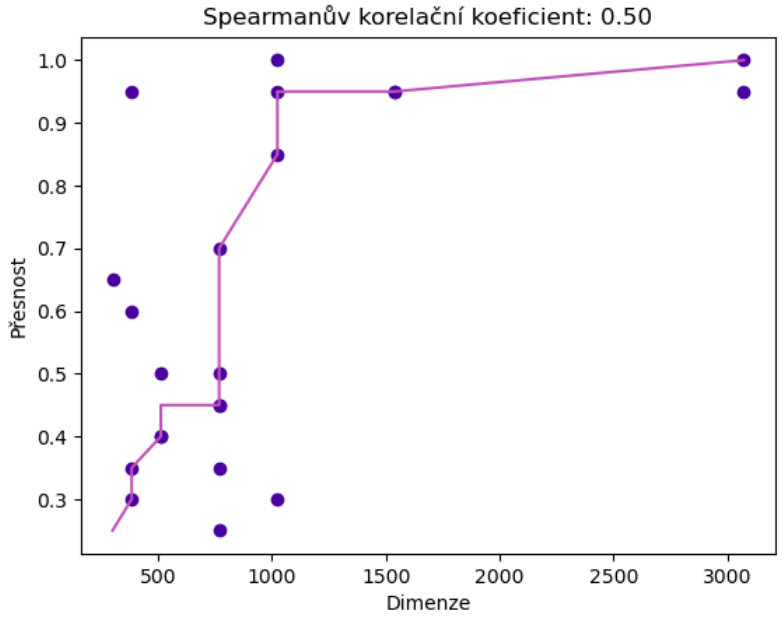
2.5.1 Korelace mezi počtem dimenzí a přesností

Vztah mezi přesností modelů a dimenzemi vektorů či sémantických embeddingů, které produkují, byl změřen pomocí Spearmanova koeficientu, jenž měří monotónní vztah mezi proměnnými[9]. Výsledná hodnota r = 0.5 naznačuje středně silnou pozitivní závislost, kdy se s rostoucím počtem dimenzí zvyšuje i přesnost modelů. Tento vztah však není zcela konzistentní ani lineární – v grafu lze pozorovat, že pouze

[9] K měření byla použita Python knihovna sciPy, viz funkce https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.spearmanr.html.

dva velmi přesné modely měly největší vektory, jedná se tedy o odlehlé hodnoty. V případě odstranění těchto dvou modelů by korelace byla slabší, konkrétně s hodnotou r = 0.36.

Z grafu a Tabulky 6 vyplývá, že bezplatné modely, které obecně dosahovaly nižší přesnosti, mají zároveň menší vektory. Placené modely produkují až dvojnásobně velké vektory. Není však pravidlem, že by větší počet dimenzí znamenal vyšší přesnost. Přesnost modelů se v souvislosti s počtem dimenzí jeví do dosažení přibližně 1 tisíce dimenzí jako poměrně náhodná, je však velmi vysoká u největších vektorů.



Graf 2 – Spearmanova korelace přesností modelů a velikostí jejich vektorů

2.5.2 Limity modelů

Žádný z nejpřesnějších modelů není zcela zdarma a každý má určitá omezení, která již byla uvedena v Tabulce 5. V případě potřeby použití zcela bezplatného a nijak omezeného modelu v praxi by bylo nutné zvolit modely ze skupiny SentenceTransformers, případně model fastText, které však v kategorii TOP 5 dosáhly maximálně přesnosti 70 %. Tato skutečnost již naznačuje, že v případě implementace systému RAG v praxi může být problematické pro češtinu volit bezplatné modely, pokud by byla žádoucí vysoká přesnost. Většina jazykových modelů není trénována na češtině do takové míry, aby modely byly mimořádně přesné, a to především ty bezplatné.

3 Experimentální část: testování a evaluace systému RAG

3.1 Specifikace nástrojů a metod

V první fázi experimentální části práce byly získány výsledky 22 testovaných embedding modelů[10], na jejichž základě byly pro následující testy systému RAG vybrány modely *text-embedding-3-large* a *paraphrase-multilingual-mpnet-base-v2*. V druhé fázi proběhlo propojení těchto modelů s generativními v systému RAG. Vybrány byly dva generativní LLM – placený GPT-4o mini a bezplatný Llama 3.3 70B Instruct.

Sampling parametry pro generování odpovědí byly ponechány na výchozích hodnotách. Byla vytvořena šablona promptu pro vkládání otázek a tří nejpodobnějších chunků nalezených v databázi (*augmentation* část procesu). Každá kombinace modelů byla otestována na dvaceti otázkách a vygenerované odpovědi byly zaznamenány do souborů ve formátu JSON a tabulky v Excelu. Následně bylo na základě předem stanovené bodové škály kvalitativně zhodnoceno, zda vygenerované odpovědi správně zodpovídají otázky na základě příslušných textů. Přesnější embedding model byl následně otestován také na celých textech, čímž bylo zjištěno, zda je pro dosažení vyšší přesnosti potřeba texty rozdělovat do chunků, nebo je dostačující používat celé texty.

3.1.1 Výběr modelů pro retrieval komponentu

Jakožto nejpřesnější embedding model byl vybrán *text-embedding-3-large* od OpenAI, který je přístupný prostřednictvím placeného API bez možnosti trial verze. Skutečnost, že měl v kategorii TOP 3 100% přesnost, znamená, že při vyhledávání relevantních textů vždy nalezl mezi třemi nejpodobnějšími chunky správnou odpověď. To zaručilo, že generativnímu modelu budou vždy poskytnuty správné informace (kromě jiných). Navíc se pouze ve 4 z 20 případů v prvních třech nejpodobnějších chuncích objevil chunk náležející jinému textu než tomu, v němž se vyskytuje správná odpověď. To rovněž zajistilo vyšší pravděpodobnost správné odpovědi, pokud by nedošlo k halucinování modelu či vynechání informací.

Jako bezplatná alternativa byl použit vícejazyčný model *paraphrase-multilingual-mpnet-base*-v2 trénovaný také na češtině, jehož přesnost však byla podstatně nižší s pouhými 50 % v kategorii TOP 3, byť byl v kategorii TOP 5 prvním nejpřesnějším mezi bezplatnými modely.

3.1.2 Výběr modelů pro generation komponentu

Jako placený generativní model byl zvolen GPT-4o mini od OpenAI dostupný prostřednictvím API, který je finančně méně nákladný ve srovnání s dalšími GPT modely. Vybrán byl za předpokladu, že stejně jako měl embedding model od OpenAI nejlepší výsledky, bude i generativní model od OpenAI velmi přesný.

[10] Pro přehlednost a odlišení modelů pro *retrieval* a *generation* komponenty jsou modely použité pro vektorizaci textu v práci označovány jako embedding modely a modely pro generování textu jako generativní modely či LLM.

Model je trénován na více než 50 jazycích a oficiálně podporuje češtinu. Jeho přesné parametry OpenAI neuvádí.

Jako bezplatná alternativa byl zvolen model Llama 3.3 70B Instruct Turbo od společnosti Meta, u nějž – na rozdíl například od modelů Gemini od Googlu – existuje větší jistota, že zůstane bezplatný i do budoucna. Model je oficiálně trénován na osmi jazycích, mezi něž však nepatří čeština. Llama modely lze volně stáhnout a spustit lokálně bez potřeby API klíče, vzhledem k vysokým výpočetním nárokům 70miliardové verze modelu jej však nebylo možné lokálně spustit. Bylo proto využito inferenční API, které umožňuje přístup k modelu prostřednictvím vzdáleného serveru. V rámci této práce byl zvolen poskytovatel Together, který k datu 10. dubna 2025 poskytoval bezplatný přístup k modelu[11] s limitem 60 požadavků za minutu, což bylo pro prováděné testy dostačující.

	Llama3.3 70B Instruct Turbo[12]	**GPT-4o-mini[13]**
Společnost	Meta	OpenAI
Počet podporovaných jazyků	8 (nezahrnují češtinu)	50+
Cena (v dolarech)	Zdarma	15/60 centů za 1 mil. vstupních/výstupních tokenů
Kontextové okno	128 tisíc tokenů	128 tisíc tokenů
Počet parametrů (vah)	70 miliard	neuvedeno
Velikost trénovacích dat	Přibližně 15 trilionů tokenů	neuvedeno
Požadavky na hardware	Dostatečná kapacita GPU VRAM	Připojení přes API
Velikost při stažení	43 GB	Připojení přes API
MMLU[14]	79,1 %	76,7 %
Hranice znalostí	Prosinec 2023	1. 10. 2023

Tabulka 7 – srovnání použitých generativních modelů Llama a GPT

[11] Informace získané z https://www.together.ai/models/llama-3-3-70b-free.

[12] Informace ve sloupci získané z webu Hugging Face https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct.

[13] Informace ve sloupci získané z webu OpenAI https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

[14] Výsledky získané 5. dubna 2025 z oficiálního žebříčku od Center for Research on Foundation Models (CRFM) dostupného na https://crfm.stanford.edu/helm/mmlu/latest/#/leaderboard.

3.1.3 Prompt

Při tvorbě šablony promptu bylo dbáno na to, aby byl explicitní, jednoznačný a podával jasné instrukce. Použitý prompt viditelný níže specifikuje, že model má vygenerovat odpověď výhradně na základě kontextu získaného z vektorové databáze. Do kontextu byly v aplikaci vloženy vždy první tři nejpodobnější texty. Použití pouze prvního nalezeného textu by nebylo vhodné, často se totiž i u modelu *text-embedding-3-large* vyskytovala správná odpověď až na třetím místě.

Slovo „pouze“ modelu říká, že nemá používat své vlastní znalosti, díky čemuž by neměl halucinovat a používat vlastní znalosti. Délka odpovědi byla určena, aby nebyly generovány odpovědi s informacemi, které nejsou klíčové, protože při prvním pokusu o generování odpovědí byly některé odpovědi v rozsahu jedné kratší věty, zatímco jiné byly zbytečně rozsáhlé. Zároveň byla modelu zadána tzv. negativní instrukce, podle níž by neměl odpovídat, pokud odpověď v poskytnutém kontextu nenašel.

```
Pouze na základě následujícího kontextu odpověz na otázku. Pokud jsi nenašel odpověď, napiš
'Nevím.'
Kontext:
{texts}
Otázka: {question}
```

3.1.4 Metodologie hodnocení odpovědí

Hodnocení vygenerovaných odpovědí bylo provedeno na základě bodové škály uvedené v Tabulce 8. Nejprve byly v chuncích se správnou odpovědí vyznačeny růžově informace, jež lze považovat za klíčové pro zodpovězení otázky. Fialově byly zvýrazněny názvy, pokud se na ně otázka explicitně či implicitně ptá. Všechny získané vygenerované odpovědi byly vloženy do tabulky v Excelu. Analýza odpovědí proběhla podle následujících kategorií:

- Obsah (růžová): v odpovědích byly růžově zvýrazněny všechny pasáže odpovídající růžovým pasážím ve správných chuncích. Hodnocení bylo provedeno na základě toho, do jaké míry od 0 do 1 byly tyto pasáže shodné se správnými především doslovně, ale i významově, a nakolik správné informace naopak chyběly.
- Název a Implicitní název (fialová): v případě, kdy se otázka ptala pouze na název, bylo hodnocení za název (nebo více názvů dohromady) 1 bod. V případě, kdy se otázka ptala na název pouze nepřímo, tzn. např. název v otázce byl nahrazen slovem „objekt“ a v odpovědi se potom vyskytl název objektu, bylo přičteno 0,1 bodu.
- Bonus (oranžová): v odpovědích byly oranžově zvýrazněny pasáže, které sloužily buďto jako užitečné doplňující informace, nebo jako dostatečně uspokojivá alternativa za správné (růžové) odpovědi v případech, kdy správný chunk nebyl k dispozici. V případě výskytu pasáží, které nebyly obsaženy ve správném chunku, byly ověřeny ostatní nejbližší chunky nalezené daným

modelem, aby bylo zjištěno, jaké informace byly v promptu obsaženy a zda je lze považovat za správné – respektive zda se vztahují ke správnému textu.

- Chyba (červená): jedná se o penalizaci a odečtení maximálně 0,5 bodu v případě, kdy pasáž narušila správnost odpovědi například tím, že odpověď vycházela ze špatného textu (ověřeno přečtením příslušných textů), nebo došlo k chybné interpretaci části textu.

Název kategorie	Specifikace kategorie	Maximum
Obsah	Správné odpovědi	1
Bonus	Správné odpovědi z jiného chunku	0,5
Název	Explicitní otázka na název	1
Implicitní název	Implicitní otázka na název	0,1
Chyba (záporné)	Chybná odpověď	0,5

Tabulka 8 – bodová škála pro hodnocení vygenerovaných odpovědí

3.2 Průběh analýzy

U takového typu analýzy by nebylo možné, aby odpovědi posuzoval automatizovaně počítač a vytvořil jednotnou škálu bodování, hodnocení každé kategorie proto proběhlo relativně subjektivně. Odpovědi bylo potřeba přečíst a lidsky zhodnotit, do jaké míry obsahují požadované informace, které další informace lze bonusově považovat za přínosné a které by v odpovědi naopak být neměly.

Analýza proběhla vzájemným srovnáním šesti odpovědí pro každou otázku tak, aby měly spravedlivé hodnocení především vůči sobě, protože nebylo možné hodnotit odpovědi každé otázky podle totožných měřítek. Díky takovému postupu bylo celkové skóre pro každý model férové vůči sobě, i když se hodnocení některých otázek mohlo mírně lišit.

U otázek, které se explicitně ptaly pouze na název, bylo maximální možné skóre 1 a u otázek, kde byla otázka na název podána implicitně, bylo maximální dosažitelné skóre 1,6. Pro získání celkového skóre v procentech byly kategorie pro všechny otázky na konci sečteny a vyděleny maximálním možným počtem bodů. Za celkové maximální možné skóre bylo považováno 20 bodů, i když s bonusovými body bylo možné dosáhnout až 29,3 bodů. Modely, které by opravdu vynikaly, by tak mohly dosáhnout přesnosti vyšší než 100 %.

3.3 Vyhodnocení experimentu

V GitHub repozitáři dostupném v Příloze 1 lze vidět způsob, průběh a výsledky analýzy odpovědí v Excel tabulce. Repozitář rovněž obsahuje Python Notebooky s testy embedding modelů a systému RAG s příslušnými modely. Obrázek 2 prezentuje několik příkladů analýzy a hodnocení odpovědí.

	Embedding	Paraphrase							
Otázka	**Chunky obsahující správnou odpověď**	**Llama**				**GPT**			
Jaký byl výsledek měření teploty na povrchu objektu na okraji Sluneční soustavy a co to naznačuje?	mohlo jít o těleso zachycené ze vzdálenější oblasti vesmíru, například z Oortova oblaku. Jiní nevylučují ani exotická vysvětlení, včetně hypotézy, že jde o fragment dávné mimozemské sondy. Další pozorování objektu přinesla ještě větší překvapení. Infračervené senzory sondy New Horizons, která se k objektu přiblížila na 400 milionů kilometrů, zjistily, že X7-N1 vykazuje nepřirozené výkyvy teploty. "Teplota na povrchu by měla být relativně stabilní, ale v průběhu několika hodin jsme zaznamenali nárůst o několik stupňů, což v takové vzdálenosti od Slunce nedává smysl," řekla doktorka Aisha Panelová z Harvardovy univerzity. Tento jev může naznačovat, že objekt obsahuje aktivní geologické procesy nebo dokonce neznámý zdroj energie. Pokud se tyto domněnky potvrdí, mohlo by jít o první objekt ve vnější Sluneční soustavě s aktivním vnitřním ohřevem, což by zpochybnilo současné modely	"Měření teploty na povrchu objektu X7-N1 ukázalo nepřirozené výkyvy teploty, kdy teplota stoupla o několik stupňů během několika hodin, což je neobvyklé pro objekt ve vzdálenosti 400 milionů kilometrů od Slunce. To naznačuje, že objekt může obsahovat aktivní geologické procesy nebo neznámý zdroj energie."				"Výsledky měření teploty na povrchu objektu X7-N1 ukázaly nepřirozené výkyvy, kdy došlo k nárůstu teploty o několik stupňů během několika hodin, což na takovou vzdálenost od Slunce nedává smysl. To naznačuje, že objekt může obsahovat aktivní geologické procesy nebo neznámý zdroj energie, což by mohlo znamenat, že se jedná o první objekt ve vnější Sluneční soustavě s aktivním vnitřním ohřevem."			
		0,7	0,1	0,0	-0,2	1,0	0,1	0,0	0,0
Jaký teleskop bude vědcům umožňovat analýzu atmosféry planety Gliese 486 b?	v mnoha ohledech podobná Zemi, přičemž její umístění v obyvatelné zóně je klíčovým faktorem pro možné hledání života," řekl doktor Isaac Thompson z Harvard-Smithsonian Center for Astrophysics. "Pokud se nám podaří analyzovat její atmosféru, mohli bychom získat cenné informace o tom, jak podobné planety mohou podporovat život, a to jak na základě chemických složek, tak podmínek pro vznik vody." Díky pokročilým technologiím, jako je James Webb Space Telescope, budou vědci schopni zkoumat atmosféru této planety s mnohem vyšší přesností než dříve. JWST je vybaven spektrografy, které dokážou detekovat přítomnost chemických prvků a molekul ve vzdálených atmosférách. První analýzy, které byly již provedeny, naznačují, že atmosféra Gliese 486 b by mohla obsahovat známky oxidu uhličitého, vodní páry a možná i metanu, což jsou klíčové složky, které mohou naznačovat příznivé	Kosmický teleskop James Webb (JWST) a Keplerův teleskop budou použity k analýze atmosféry planety Gliese 486 b.				Vědcům umožní analýzu atmosféry planety Gliese 486 b Kosmický teleskop James Webb.			
		0,0	1,0	0,0	-0,5	0,0	1,0	0,0	0,0
Jakým způsobem by mohl objev jiného životního cyklu hvězd změnit naše metody studia vesmíru v budoucnosti?	dokážou absorbovat zbytkový plyn z okolí a přetvářet ho na novou formu energie, což způsobuje obnovu jejich vývojového cyklu. Tento jev naznačuje, že hvězdy v Orionu mají neuvěřitelnou schopnost regenerace, která může trvat mnohem déle, než jsme si dříve mysleli. „Tento objev nás může přimět přehodnotit naše představy o vývoji hvězd a výbuchu supernov, stejně jako o formování planetárních systémů," říká Dr. Michael Johnson, astrofyzik z NASA. „Dosud jsme si mysleli, že vývoj hvězd je lineární proces, ale to, co nyní vidíme, je cyklický proces, který se může opakovat, což otevírá nové možnosti pro pochopení vzniku planetárních systémů a života na exoplanetách." Vědci nyní plánují provést podrobné analýzy, aby ověřili, zda tento cyklický proces není omezen pouze na Orionovu mlhovinu, nebo zda je tento jev běžný i v	"Objev jiného životního cyklu hvězd by mohl změnit naše metody studia vesmíru tím, že by nás vedl k přehodnocení tradičních modelů vývoje hvězd a jejich životních cyklů. To by mohlo mít zásadní důsledky pro naše chápání vzniku a zániku hvězd, a mohlo by také ovlivnit naše hledání exoplanet a životu ve vesmíru. Pokud by se prokázalo, že stabilní cyklický proces vzniku hvězd vede k vytvoření stabilních planetárních systémů, mohlo by to mít zásadní vliv na naše schopnosti hledat život ve vesmíru."				"Objev jiného životního cyklu hvězd by mohl změnit naše metody studia vesmíru tím, že přesměruje pozornost na složitější procesy vzniku a vývoje hvězd, což by vedlo k novým teoriím a modelům. Především by to mohlo ovlivnit naše pochopení stabilních planetárních systémů a hledání exoplanet, protože nám umožní lépe předvídat podmínky potřebné pro vznik a udržení života v různých hvězdných systémech."			
		0,2	0,0	0,3	0,0	0,1	0,0	0,3	0,0
Jakým způsobem analyzuje revoluční AI změny v čase?	Revoluční AI od Googlu předpovídá budoucnost s 87% úspěšností Tým výzkumníků z DeepMind, dceřiné společnosti Googlu, oznámil průlom v oblasti prediktivní umělé inteligence. Nový model s názvem Chronos-21 dokáže na základě analýzy historických dat, ekonomických trendů a sociálních interakcí předpovídat události s ohromující 87% úspěšností. Podle hlavního výzkumníka Dr. Eliase Morina se Chronos-21 liší od předchozích modelů tím, že využívá nejen tradiční neuronové sítě, ale i experimentální přístup nazvaný „časové rekurentní predikce". Tento nový algoritmus analyzuje změny v čase podobně jako šachový velmistr plánuje tahy dopředu – hledá pravděpodobné scénáře vývoje situací a zpětně analyzuje, které faktory k nim vedly. „Nejde o věštění budoucnosti," vysvětluje Dr. Morin. „Model neříká přesně, co se stane, ale dokáže velmi přesně určit pravděpodobnost různých událostí. Například v oblasti financí se nám podařilo	"Revoluční AI, konkrétně model Chronos-21, analyzuje změny v čase pomocí experimentálního přístupu nazvaného „časové rekurentní predikce". Tento algoritmus analyzuje změny v čase podobně jako šachový velmistr plánuje tahy dopředu – hledá pravděpodobné scénáře vývoje situací a zpětně analyzuje, které faktory k nim vedly."				"Revoluční AI Chronos-21 analyzuje změny v čase pomocí experimentálního přístupu nazvaného „časové rekurentní predikce", který plánuje pravděpodobné scénáře vývoje situací a zpětně analyzuje faktory, které k nim vedly, podobně jako šachový velmistr plánuje své tahy dopředu."			
		1,0	0,1	0,0	0,0	1,0	0,1	0,0	0,0

Obrázek 2 – příklady ohodnocených vygenerovaných odpovědí u Paraphrase embedding modelu a Llama a GPT generativních modelů

3.3.1 Výsledky pro chunky textů

Výsledky uvedené v Tabulce 9 potvrzují, že GPT model dosahoval v kombinaci s OpenAI embedding modely nejvyšší přesnosti. V kombinaci s bezplatnými Paraphrase embeddingy byl GPT model však slabší než bezplatný Llama model. Obecně lze říci, že z hlediska přesnosti byl použitý Llama model srovnatelný s GPT modelem, což je pozoruhodné vzhledem k tomu, že Llama modely nejsou trénované na češtině a jsou navíc dostupné zdarma, zatímco GPT modely češtinu oficiálně podporují a jsou placené.

Přestože Paraphrase model dosahoval pouze 50% přesnosti v kategorii TOP 3 – a tudíž v polovině případů generativnímu modelu vůbec neposkytl chunk se správnou odpovědí –, obě jeho skóre byly obstojné, protože generativní model dokázal správně využít i informace z jiných chunků. I když měl tento embedding model zhruba o 20 % nižší přesnost, kvalitativní přečtení jednotlivých odpovědí ukazuje, že jsou obsahově uspokojivé a poskytují přínosné informace zejména z pohledu uživatele, který by neměl povědomí o existenci správnějších informací v textech.

Embedding	LLM	Obsah	Název	Bonus	Chyba	Celkem	Skóre
OpenAI	**GPT**	14,3	3,0	1,4	-0,1	18,6	**89,42 %**
OpenAI	**Llama**	11,9	3,5	2,5	0,0	17,9	**86,06 %**
Paraphrase	**Llama**	9,7	2,0	3,0	-0,7	14,0	**67,31 %**
Paraphrase	**GPT**	9,5	2,0	2,6	-0,9	13,2	**63,46 %**

Tabulka 9 – výsledky analýzy vygenerovaných odpovědí u použití chunků

3.3.2 Výsledky pro celé texty

Férové ohodnocení odpovědí získaných na základě kontextu obsahujícího tři celé texty bylo obtížnější, jelikož odpovědi byly komplexnější a čerpaly informace z většího množství textových dat. V některých případech modely také odpovídaly i na základě jiných textů, poskytnuté informace bylo však možné v bonusové kategorii považovat za správné.

Při použití celých textů bylo kontextové okno generativních modelů podstatně plnější, což mohlo způsobit ztrátu nebo misinterpretaci informací. K tomu došlo dvakrát u modelu GPT, který nesprávně interpretoval informace z poskytnutých textů. Tyto pasáže se sice na první pohled téměř shodovaly se správnými odpověďmi, nesprávným použitím nebo vynecháním slov a slovosledem však změnily význam. Navíc model v odpovědích neposkytoval názvy v takové míře jako Llama model, ani nevyužíval další informace jako bonusové. Llama model v tomto testu naopak nechyboval vůbec a často poskytnul navíc užitečné doplňující informace včetně relevantních názvů.

Embedding	LLM	Obsah	Název	Bonus	Chyba	Celkem	Skóre
OpenAI	**Llama**	12,7	3,6	2,1	0,0	18,4	**88,46 %**
	GPT	12,4	3,0	1,5	-0,5	16,4	**78,85 %**

Tabulka 10 – výsledky analýzy vygenerovaných odpovědí u použití celých textů

3.4 Problematické otázky

Byl na Marsu někdy život?

Na této otázce je zajímavé, že na ni oba generativní modely ve všech případech odpověděly větou „Nevím." Nejedná se přitom o otázku, na kterou by bylo obtížné na základě textů odpovědět. Dostačující a uspokojivou odpovědí na základě textu by mohlo být například „Pod Marsovým povrchem možná existují podmínky pro život, což by znamenalo, že na planetě vznikl život v minulosti." Oba embedding modely relevantní chunk nebo text generativnímu modelu vždy poskytly, generativní modely však informace nedokázaly využít.

Jaká objevená látka má potenciál v boji proti virům?

Celá ideální odpověď na otázku se nacházela v chunku 16AI_Vyvoj_leku_3, samotný název látky se však nacházel na úplném konci předcházejícího chunku textu. Ten se v poskytnutých textech běžně objevoval, bylo však potřeba brát v potaz, že v tomto případě je odpověď skutečně rozprostřena do dvou chunků a nelze na otázku odpovědět pouze na základě jednoho. Právě díky třem použitým textům namísto jednoho však odpovědi byly do vysoké míry správné.

Může AI v budoucnosti přebírat roli vědce a přinášet nová vědecká poznání?

Na tuto otázku nebylo potřeba komplikovaně odpovídat a modely na ni odpovídaly dobře. Problém v hodnocení nastal u odpovědi od GPT modelu na základě celých textů, protože model odpověděl na základě tří odlišných textů, nikoliv tedy chunků. V tomto případě mu nakonec byly připsány bonusové body, jelikož informace lze považovat za užitečné, byť se týkají jiných textů.

4 Diskuze nad výsledky

4.1 Shrnutí zjištění ve vztahu k cílům práce

Důležitým zjištěním této práce je skutečnost, že pro velké jazykové modely obecně není čeština jednoduchý jazyk. Ať už jsou modely na češtině trénovány či nikoliv, většinou nedosahují nejvyšší přesnosti. Na základě výsledků této práce fungují pro češtinu nejpřesněji placené embedding modely. O výkonu generativních modelů by se dalo dále polemizovat, protože lze obecně říci, že modely GPT a Llama si byly rovnocenné. Rozšířením výzkumu o další generativní modely by však mohlo být zjištěno, které generativní modely by mohly skutečně vynikat nad ostatními.

4.1.1 Srovnání embedding modelů

Na základě výsledků první fáze se mohlo zdát, že efektivní použití bezplatných modelů je pro češtinu téměř nemožné, neboť v kategorii TOP 3 dosahovaly maximálně 50% přesnosti a byly tudíž v polovině případů nespolehlivé. Výsledky druhé fáze nicméně ukázaly, že lze relativně efektivně systém RAG pro češtinu vytvořit zdarma, byť za cenu zvýšeného rizika improvizace modelu v případech, kdy mu relevantní texty nejsou vůbec poskytnuty. Záleží samozřejmě na tom, jakou spolehlivost uživatel vyžaduje. V kritických infrastrukturách, jako je lékařství či právo, by mohlo být použití bezplatných embedding modelů velmi riskantní, jelikož právě tyto modely mají zajistit předání správné informace generativnímu modelu. Uživatel by si výstupy z takového systému musel vždy pečlivě ověřit, jelikož hrozí velké riziko chybné interpretace modelem nebo absence klíčových informací v odpovědi. Tyto aspekty je nutné brát v potaz při případném využití systému v praxi.

Při použití placeného embedding modelu od OpenAI se výsledky zhruba o 20 % zlepšily. Příčinou je zřejmě především skutečnost, že se ve třech nejpodobnějších chuncích nalezených v databázi vyskytovala správná odpověď ve 100 % případů. Navíc se v nich v 16 z 20 případů zároveň vyskytovaly pouze chunky textů náležející textu se správnou odpovědí.

Je nicméně vhodné zdůraznit, že ani v případě použití bezplatného embedding modelu nebyly odpovědi úplně chybné, a naopak, že generativní modely v několika případech chybovaly i při použití placeného embedding modelu. To znamená, že kvalita odpovědí závisí stejnou měrou jak na přesnosti embedding modelů, tak na schopnostech generativních modelů.

4.1.2 Srovnání generativních modelů

Je-li systém RAG ve zmíněném stavu improvizace v důsledku nižší přesnosti embedding modelu, nelze přesně hodnotit výkon a roli generativního modelu samotného, jelikož se neposkytnutí správných informací při retrievalu odráží v celém procesu.

Z hlediska gramatiky a koherence textu byly modely rovnocenné, a to navzdory skutečnosti, že model Llama oficiálně češtinu nepodporuje. To je pozoruhodné vzhledem k tomu, že bezplatný model,

o němž je uváděno, že by neměl být používán pro jiné jazyky než oficiálně podporovaných osm[15], je rovnocenný placenému GPT modelu, který češtinu podporuje. Na druhou stranu, tento Llama model má vysoké výpočetní nároky a nelze jej používat bez lokálního spuštění nebo inferenčního API.

4.1.3 Chunking textů

V případě použití embedding modelu od OpenAI můžeme při srovnání Tabulek 9 a 10 pozorovat, že GPT model dosahoval při použití chunkingu až o 10 % vyšší přesnosti, lze tedy říci, že je výhodnější texty rozdělovat do chunků. Naopak u Llama modelu bylo skóre velmi podobné, v jeho případě je chunking tedy vhodné uvážit po posouzení jiných faktorů, např. ceně vstupních tokenů. Je důležité podotknout, že Llama model dosahoval téměř stejného skóre při použití celých textů, jakého dosahoval GPT model při použití chunků. Na základě získaných výsledků tedy nelze jednoznačně říci, zda je pro kvalitní výsledky výhodnější používat chunking či celé texty.

4.2 Možné příčiny a důsledky

Velké jazykové modely tokenizují české texty jiným způsobem než například texty anglické. Problematická tokenizace a vektorizace textů může být důsledkem flektivní povahy češtiny, která má bohatou morfologii. Zároveň má čeština diakritiku, která může být problematická zejména pro vícejazyčné modely, které nebyly na češtině trénovány dostatečně.

Přestože model fastText oproti ostatním modelům pouze průměruje statické embeddingy slov ve větách a neumí zohledňovat kontext, byl přesnější než většina bezplatných modelů typu transformátor. K tomuto jevu může docházet pravděpodobně právě důsledkem nesprávné a složité tokenizace prováděné transformátory.

V současné době zřejmě neexistuje embedding model, který by byl pro češtinu dostatečně přesný a zároveň plně zdarma. Je rovněž vhodné připomenout, že neexistuje silný vztah mezi počtem dimenzí vektorů, které embedding modely produkují, a přesností modelů. To znamená, že i když může vyšší počet dimenzí vyvolávat dojem, že budou modely dosahovat lepších výsledků než menší modely, nemusí to být pravidlem a v přesnosti modelů tudíž hrají roli i jiné aspekty.

4.3 Limity výzkumu

Objektivní posouzení správnosti textů nalezených na základě sémantické podobnosti s promptem, stejně jako správnosti vygenerovaných odpovědí, není jednoduché. Vyžadovalo by totiž detailní obsahovou znalost všech textů. I za takových podmínek by ale hodnocení zůstalo do určité míry subjektivní. Subjektivita tudíž hrála roli jak při tvorbě otázek, tak při hodnocení odpovědí a výsledky práce by se mohly lišit při použití jiných textových dat nebo jiných otázek. Vygenerované odpovědi se navíc lišily délkou.

[15] Informace poskytnuté na https://ollama.com/library/llama3.3.

Z těchto důvodů je potřeba výsledky interpretovat s určitou rezervou, základní představu o přesnosti velkých jazykových modelů a jejich schopnosti práce s češtinou nicméně poskytují.

Mezi další limity patřil výpočetní výkon, finanční prostředky i čas. Výzkum by mohl být podstatně rozsáhlejší s větším textovým datasetem, vyšším výpočetním výkonem pro otestování většího množství modelů, včetně těch opravdu velkých. Takový výzkum by zároveň vyžadoval vyšší finanční náklady.

4.4 Praktické dopady a aplikace

V praktickém využití by pro co nejvyšší přesnost bylo jednoznačně spolehlivější použít embedding model od OpenAI, zde však stále vyvstává otázka, zda je výhodnější používat chunking. Podle toho by potom byl zvolen GPT nebo Llama model. V každém případě je však nutné – stejně jako v běžné konverzaci s chatbotem – interpretovat výstupy kriticky a ověřovat si získané informace.

Systém RAG zprostředkovává velmi jednoduchou formu personalizace generativního LLM a umožňuje získání odpovědí na otázky, na něž na základě trénovacích dat nemůže znát odpověď, zároveň však nezaručuje, že výsledky budou vždy pravdivé a správné. I výstupy nejpřesnějších modelů je tak nadále nutné ověřovat. Z dlouhodobého hlediska je v praktickém využití obecně lepší implementovat systém RAG v češtině s použitím placených modelů. V případě potřeby použití výhradně bezplatných modelů je pak nutné si informace pečlivěji ověřovat, protože během retrievalu nemusí vždy naleznout správné texty.

4.5 Rozšíření pro budoucí výzkum

Experimentálně by bylo možné testovat jiné způsoby promptování a upravovat sampling parametry modelů, odpovědi by se však – vzhledem k délce textů a krátkým vygenerovaným odpovědím – možná příliš nelišily. Výzkum byl omezen jak finančně, tak nižším výpočetním výkonem, bylo by tedy možné jej rozšířit o testování větších či dražších LLM a o větší textový dataset.

Závěr

Práce se zabývala návrhem a implementací systému Retrieval-Augmented Generation (RAG) v češtině, jehož cílem bylo vyhledávání relevantních informací na základě dotazů a jejich následné využití při generování odpovědí. První fáze výzkumu byla zaměřena na určení nejpřesnějších embedding modelů pro vektorizaci textů, které umožňují efektivní vyhledání relevantních textů ve vektorové databázi. V druhé fázi byla realizována samotná implementace systému RAG a otestování, za jakých podmínek a s použitím kterých kombinací modelů jsou generovány nejlepší odpovědi na dotazy. Bylo rovněž zkoumáno, zda je výhodnější použití chunkingu či celých textů.

V teoretické části byly popsány klíčové koncepty v oblasti umělé inteligence a implementace systému RAG. V první fázi experimentální části byl vytvořen program pro měření přesnosti a evaluaci embedding modelů ve vektorové databázi, jehož prostřednictvím lze automatizovaně testovat jednotlivé modely na libovolných textových datech. Na základě výsledků byly vybrány dva nejpřesnější modely – bezplatný a placený –, které byly v druhé fázi výzkumu integrovány do systému RAG v kombinaci se dvěma vybranými generativními modely, taktéž jedním bezplatným a jedním placeným. Vygenerované odpovědi na otázky byly následně kvalitativně zanalyzovány a bylo ohodnoceno, do jaké míry odpovídají očekávaným správným odpovědím. Tímto způsobem byl získán základní přehled o přesnosti a výkonu velkých jazykových modelů na češtině.

Výsledky této práce souhrnně ukazují, že lze systém RAG efektivně implementovat i pro češtinu. Získané odpovědi by však bylo vhodné vždy pečlivě ověřovat, zejména při použití bezplatných embedding modelů, jejichž přesnost na češtině není příliš vysoká a nelze jejich prostřednictvím vždy nalézat správné texty. Z tohoto důvodu je výhodnější používat placené modely dosahující vyšší přesnosti. Díky metodám použitým v této práci se navíc podařilo ve všech případech zabránit generativním modelům halucinovat, lze proto takový systém považovat za poměrně spolehlivý. Obecně lze na základě kvalitativní analýzy všech vygenerovaných odpovědí říci, že byly převážně uspokojivé a vyhovující pro další použití systému RAG v dalších aplikacích.

Seznam použitých zdrojů a literatury

Bojanowski, Piotr, Edouard Grave, Armand Joulin, a Tomas Mikolov. 2017. „Enriching Word Vectors with Subword Information". arXiv. https://doi.org/10.48550/arXiv.1607.04606.

CustomGPT.ai. b.r. „The Key Components of a RAG System: A Technical Deep Dive". *CustomGPT.ai* (blog). Viděno 5. duben 2025. https://customgpt.ai/components-of-a-rag-system/.

Česká asociace umělé inteligence z.ú. b.r. „Slovník pojmů". *Česká asociace umělé inteligence z.ú.* (blog). https://asociace.ai/slovnik-pojmu/.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, a Kristina Toutanova. 2018. „BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". arXiv. https://doi.org/10.48550/ARXIV.1810.04805.

Douze, Matthijs, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, a Hervé Jégou. 2024. „The Faiss library". arXiv. https://doi.org/10.48550/ARXIV.2401.08281.

El Amri, Aymen. 2023. *LLM Prompt Engineering for Developers: The Art and Science of Unlocking LLMs' True Potential*. Erscheinungsort nicht ermittelbar: FAUN.

GeeksforGeeks. 2024. „How to Chunk Text Data: A Comparative Analysis". *GeeksforGeeks* (blog). 2. srpen 2024. https://www.geeksforgeeks.org/how-to-chunk-text-data-a-comparative-analysis/.

Generative AI. 2024. „Best LLM 2024: Top Models for Speed, Accuracy, and Price". *LinkedIn* (blog). 5. listopad 2024. https://www.linkedin.com/pulse/best-llm-2024-top-models-speed-accuracy-price-genai-works-qe49f/.

Gheorghiu, Andrei. 2024. *Building Data-Driven Applications with LlamaIndex: A Practical Guide to Retrieval-Augmented Generation (RAG) to Enhance LLM Applications*. 1. vyd. Birmingham: Packt Publishing Limited.

Hapke, Hannes Max, Cole Howard, a Hobson Lane. 2019. *Natural Language Processing in Action: Understanding, Analyzing, and Generating Text with Python*. Shelter Island, N.Y: Manning Publications Co.

Huang, Anna. 2008. „Similarity measures for text document clustering", Proceedings of the 6th New Zealand Computer Science Research Student Conference, , leden, 49–56.

Jurafsky, Dan, a James H. Martin. 2025. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3. vyd. https://web.stanford.edu/~jurafsky/slp3.

Lewis, Mike, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, a Luke Zettlemoyer. 2019. „BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". arXiv.

https://doi.org/10.48550/arXiv.1910.13461.

Lewis, Patrick, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, et al. 2021. „Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". arXiv. https://doi.org/10.48550/arXiv.2005.11401.

Logan Kilpatrick, Zach Gleicher, a Parashar Shah. 2025. „State-of-the-art text embedding via the Gemini API". *Google for Developers* (blog). 7. březen 2025. https://developers.googleblog.com/en/gemini-embedding-text-model-now-available-gemini-api/.

Matlach, Vladimír. 2023. *Úvod do zpracování dat 1*. 1. vyd. Olomouc: Univerzita Palackého v Olomouci.

Mikolov, Tomas, Kai Chen, Greg Corrado, a Jeffrey Dean. 2013. „Efficient Estimation of Word Representations in Vector Space". arXiv. https://doi.org/10.48550/ARXIV.1301.3781.

Moez Ali. 2025. „The Top 7 Vector Databases in 2025". *DataCamp* (blog). 18. leden 2025. https://www.datacamp.com/blog/the-top-5-vector-databases.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, et al. 2012. „Scikit-learn: Machine Learning in Python". https://doi.org/10.48550/ARXIV.1201.0490.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, a Luke Zettlemoyer. 2018. „Deep contextualized word representations". arXiv. https://doi.org/10.48550/arXiv.1802.05365.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, a Peter J. Liu. 2023. „Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". arXiv. https://doi.org/10.48550/arXiv.1910.10683.

Roie Schwaber-Cohen. 2023. „Chunking Strategies for LLM Applications". *Pinecone* (blog). 30. leden 2023. https://www.pinecone.io/learn/chunking-strategies/.

Sarkar, Dipanjan. 2016. *Text Analytics with Python*. Berkeley, CA: Apress. https://doi.org/10.1007/978-1-4842-2388-8.

Smith, Noah A. 2019. „Contextual Word Representations: A Contextual Introduction". arXiv. https://doi.org/10.48550/ARXIV.1902.06006.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, a Illia Polosukhin. 2017. „Attention Is All You Need". arXiv. https://doi.org/10.48550/ARXIV.1706.03762.

Yang, Jingfeng, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, a Xia Hu. 2023. „Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond". arXiv. https://doi.org/10.48550/arXiv.2304.13712.

Seznamy tabulek, grafů a obrázků

Přílohy

Příloha 1: GitHub repozitář dostupný na https://github.com/sladkydrevo/RAG-testing/.

1. Složka *dataset*
    - složka *questions* obsahující všechny otázky
    - složka *texts* obsahující všechny texty
    - soubor *all_chunks.txt* obsahující chunky s jejich názvy
    - soubor *openai_results.json* obsahující pro každou otázku 3 nejpodobnější nalezené chunky získané pomocí text-embedding-3-large modelu
    - soubor *paraphrase_results.json* obsahující pro každou otázku 3 nejpodobnější nalezené chunky získané pomocí paraphrase-multilingual-mpnet-base-v2 modelu
    - soubor *right_answers.txt* obsahující názvy chunků se správnými odpověďmi
    - soubor *right_chunks.txt* obsahující chunky se správnými odpověďmi
    - soubor *text_names.txt* obsahující názvy všech textů
2. Složka *generated_answers_chunks* obsahující vygenerované odpovědi ve formátu JSON pro každou kombinaci modelů na základě chunků
3. Složka *generated_answers_whole_texts* obsahující vygenerované odpovědi ve formátu JSON pro každou kombinaci modelů na základě celých textů
4. Složka *tabulky* obsahující Excel tabulku s celou kvalitativní analýzou a hodnocením vygenerovaných odpovědí
5. Skript *analyze_chunk_stats.py* – výpočet hodnot v Tabulkách 2 a 3
6. Python Notebook *pt1_baseline.ipynb* – testování fastText modelu
7. Python Notebook *pt2_db.ipynb* – testování 21 embedding modelů
8. Python Notebook *pt3_rag_v2_final.ipynb* – test 4 kombinací modelů pro vygenerování odpovědí s použitím chunků
9. Python Notebook *pt3_rag_v2_whole_texts.ipynb* – test 2 kombinací modelů, modifikovaný pro vygenerování odpovědí s použitím celých textů
10. Skript *rag_functions.py* – základní funkce používané napříč všemi Python Notebooky