

Implementace LLM jako chatbot pro e-shop

Bc. Lukáš Richter

Diplomová práce
2024



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2023/2024

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: Bc. Lukáš Richter
Osobní číslo: A22280
Studijní program: N0613A140022 Informační technologie
Specializace: Softwarové inženýrství
Forma studia: Kombinovaná
Téma práce: Implementace LLM jako chatbot pro e-shop
Téma práce anglicky: LLM implementation as a chatbot for an e-shop

Zásady pro vypracování

1. Vypracujte literární rešerši na téma LLM jako chatbot s využitím vlastní knowledge base.
2. Shromážděte požadavky na funkčnost chatbota pro odpovídání na dotazy zákazníků e-shopu.
3. Zvolte vhodné řešení a vytvořte funkčního chatbota, který bude splňovat požadavky.
4. Otestujte funkčnost navrženého řešení.
5. Navrhněte dále vhodné řešení pro rozšíření chatbota o možnost odpovídat na dotazy ohledně objednávek zákazníků.
6. Diskutujte možnosti dalšího vývoje a úprav.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. TAULLI, T. Generative AI: How ChatGPT and Other AI Tools Will Revolutionize Business. *Berkeley, CA: Apress, 2023.*
2. SARRION, Eric. Exploring the power of ChatGPT: applications, techniques, and implications. Springer, [2023], ISBN 9781484295281.
3. OPENAI, INC. *Dokumentace produktů spol. OpenAI.* Online. 2023. Dostupné z: <https://platform.openai.com/docs>. [cit. 2023-11-12].
4. JUSTIN, Paul; UENO, Akiko a DENNIS, Charles. ChatGPT and consumers: Benefits, Pitfalls and Future Research Agenda. Online. *International Journal of Consumer Studies.* 2023, roč. 47, č. 4, s. 1213-1225. Dostupné z: <https://doi.org/https://doi.org/10.1111/ijcs.12928>. [cit. 2023-11-12].
5. META PLATFORMS, INC. *Llama 2.* Online. 2023. Dostupné z: <https://ai.meta.com/llama/>. [cit. 2023-11-12].
6. LANGCHAIN, INC. *Dokumentace LangChain.* Online. 2023. Dostupné z: <https://python.langchain.com/docs>. [cit. 2023-11-12].
7. LI, Meichan; WANG, Rui. Chatbots in e-commerce: The effect of chatbot language style on customers' continuance usage intention and attitude toward brand. *Journal of Retailing and Consumer Services*, 2023, 71: 103209.
8. GAMBOA-CRUZADO, JAVIER, et al. Use of chatbots in e-commerce: a comprehensive systematic review. *Journal of Theoretical and Applied Information Technology*, 2023, 101.4.

Vedoucí diplomové práce: **prof. Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Konzultant diplomové práce: **Ing. Jozef Kováč**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **5. listopadu 2023**

Termín odevzdání diplomové práce: **13. května 2024**



doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D., DBA v.r.
ředitel ústavu

Ve Zlíně dne 5. ledna 2024

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 13. května 2024

Lukáš Richter, v.r.

ABSTRAKT

Cílem této diplomové práce je vývoj funkčního chatbota pro e-shop, který je založený na interakci s velkými jazykovými modely (LLM) a využívá vlastní znalostní bázi. Práce se zaměřuje na schopnost chatbota vést plynulé a kvalitní konverzace, které jsou přizpůsobeny potřebám a dotazům zákazníků. Pro implementaci chatbota byla zvolena knihovna LangChain pro jazyk Python, která umožňuje flexibilní integraci různých jazykových modelů díky své abstraktní architektuře. Tento přístup umožňuje nejen snadnou výměnu používaných modelů, ale i efektivní využití pokročilých funkcí LLM pro zpracování a odpovídání na uživatelské dotazy.

Klíčová slova: chatbot, velké jazykové modely, umělá inteligence, LangChain, Python, e-commerce

ABSTRACT

The goal of this thesis is the development of a functional chatbot for an e-commerce platform, which is based on interaction with large language models (LLMs) and utilizes external knowledge base. The focus is on the chatbot's ability to conduct smooth and high-quality conversations tailored to the needs and inquiries of customers. The LangChain library for Python was chosen for the implementation of the chatbot, enabling flexible integration of various language models due to its abstract architecture. This approach not only allows for easy swapping of used models but also efficient utilization of advanced LLM features for processing and responding to user queries.

Keywords: chatbot, large language models, artificial intelligence, LangChain, Python, e-commerce

Na začátku mého poděkování bych chtěl vyjádřit upřímné díky svému vedoucímu práce, prof. Ing. Romanu Šenkeříkovi, Ph.D., za jeho neocenitelné rady, trpělivost a odborné vedení během celého procesu výzkumu a psaní této diplomové práce.

Velké díky patří i mé rodině a přátelům, kteří mi poskytovali morální podporu a pochopení, zvláště v obdobích, kdy jsem musel čelit různým výzvám během studia.

OBSAH

ÚVOD	9
I TEORETICKÁ ČÁST	10
1 AI CHATBOT	11
1.1 DEFINICE A HISTORIE CHATBOTŮ	11
1.1.1 Definice Chatbotů	11
1.1.2 Historie Chatbotů.....	11
1.1.3 Významné mezníky v historii chatbotů.....	12
1.2 TYPY CHATBOTŮ A JEJICH TECHNOLOGIE	13
1.2.1 Klasifikace Chatbotů	13
1.2.2 Technologie používané v chatbotech.....	13
1.2.3 Příklady technologií používaných v chatbotech.....	14
2 LARGE LANGUAGE MODEL (LLM)	15
2.1 PŘEDTRÉNOVÁNÍ	15
2.2 JEMNÉ LADĚNÍ - FINE-TUNNING.....	15
2.3 ARCHITEKTURA TRANSFORMER	15
2.4 APLIKACE LLM	15
2.5 VÝZVY A ETIKA	16
2.6 PŘÍKLADY MODELŮ	16
2.6.1 ChatGPT	16
2.6.2 Llama	16
2.7 RIZIKA.....	17
2.7.1 Zneužití.....	17
2.7.2 Techniky zneužití	18
2.7.3 Ochrana dat.....	19
2.7.4 Etická omezení.....	19
2.8 RETRIEVAL AUGMENTED GENERATION (RAG)	20
2.8.1 Fáze RAG	20
2.8.2 Výhody RAG	21
3 NÁVRH ŘEŠENÍ	22
3.1 POŽADAVKY NA FUNKČNOST	22
3.1.1 Rychlost a přesnost.....	22
3.1.2 Srozumitelnost	22
3.1.3 Mnohojazyčnost	23
3.2 ZVOLENÉ TECHNOLOGIE.....	24

3.2.1	LLM	24
3.2.2	LangChain.....	25
II	PRAKTICKÁ ČÁST.....	26
4	LLM	27
4.1	OLLAMA	27
4.1.1	Ollama API.....	27
4.2	CHATGPT.....	29
5	LANGCHAIN.....	31
5.1	OBJEKT OLLAMA	31
5.2	OBJEKT OPENAI	32
5.3	OBJEKT CHATOLLAMA.....	32
5.4	OBJEKT CHATOPENAI.....	33
6	POROVNÁNÍ MODELŮ	35
7	RETRIEVAL AUGMENTED GENERATION	41
8	SERVER.....	45
9	KLIENT	48
9.1	HTML	48
9.2	JAVASCRIPT.....	49
10	TESTOVÁNÍ	51
11	DISKUZE	54
11.1	DOTAZY TÝKAJÍCÍ SE OBJEDNÁVEK	54
11.2	DALŠÍ VÝVOJ A ÚPRAVY	54
	ZÁVĚR.....	56
	SEZNAM POUŽITÉ LITERATURY	57
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	62
	SEZNAM OBRÁZKŮ	63
	SEZNAM TABULEK.....	64
	SEZNAM PŘÍLOH.....	66

ÚVOD

Umělá inteligence (UI), běžně známá jako AI (Artificial Intelligence), je obor informatiky, který se zabývá vytvářením inteligentních strojů schopných provádět úkoly, které by obvykle vyžadovaly lidskou inteligenci. Zahrnuje techniky jako strojové učení, kde se stroje učí z dat, rozpoznávání vzorů, zpracování přirozeného jazyka, robotiku a vizuální rozpoznávání. Umělá inteligence má široké uplatnění od automatizovaných odpovědních systémů, přes pokročilou analýzu dat, až po autonomní vozidla a roboty. Využívá se v různých průmyslových odvětvích, včetně zdravotnictví, financí, výroby a dopravy, kde přináší zvýšení efektivity, přesnosti a rychlejší inovaci. AI nabízí téměř neomezené možnosti pro zlepšení každodenního života, ale současně představuje výzvy týkající se etiky, soukromí a bezpečnosti.

Ve světě elektronického obchodování, kde konkurence neustále roste a spokojenost zákazníků se stává klíčovým faktorem, se e-shopy neustále snaží zlepšovat efektivitu svých služeb. Jedním z nejnovějších trendů v této oblasti je implementace pokročilých technologií umělé inteligence, především pak velkých jazykových modelů (LLM), do zákaznického servisu. Tyto modely, zvláště když jsou implementovány jako chatboti, mají potenciál radikálně transformovat interakci mezi e-shopy a jejich zákazníky.

Tato diplomová práce se zaměřuje na implementaci LLM jako chatbota pro e-shop s využitím vlastní knowledge base (znalostní báze). Cílem je prozkoumat, jak mohou tyto technologie přispět k lepšímu pochopení a zpracování dotazů zákazníků, což může vést k rychlejší a přesnější odpovědi na jejich otázky a požadavky.

Cílem této práce je vytvořit funkční řešení založené na nejnovějších velkých jazykových modelech a otestovat použitelnost takového řešení. V první části této práce je kladen důraz na shromáždění informací o chatbotech obecně, LLM a nástrojích pro jejich implementaci do e-shopových systémů. V druhé části bude podrobně představeno zvolené řešení a jednotlivé modely otestovány.

I. TEORETICKÁ ČÁST

1 AI CHATBOT

1.1 Definice a historie chatbotů

1.1.1 Definice Chatbotů

Chatbot je softwarová aplikace, která simuluje písemné nebo mluvené lidské konverzace prostřednictvím umělé inteligence. Funkčně jsou navrženi tak, aby uživatelům poskytovali informace nebo sloužili k provedení konkrétních úkolů, jako je objednávání produktů, rezervace služeb nebo odpovídání na často kladené dotazy. V této kapitole jsou představeny různé typy chatbotů podle použité technologie, od jednoduchých, založených na předdefinovaných pravidlech, po pokročilé systémy využívající strojové učení a zpracování přirozeného jazyka.

Chatbot je typ softwaru, který je navržen k simulaci interakce s uživateli prostřednictvím psaného nebo mluveného jazyka. Tento software umožňuje uživatelům komunikovat s digitálním systémem, jako by mluvili s člověkem, a tím usnadňuje různé úkoly, jako je získávání informací, provedení objednávky nebo zákaznická podpora. Chatboti se využívají v mnoha odvětvích, včetně e-commerce, zákaznické podpory, zábavy a osobní asistence. [1] [2]

V českém jazyce je z hlediska pravopisu pohlíženo na slovo chatbot obdobně jako na slovo robot. Přípustné je skloňování podle životného i neživotného rodu, pro humanoidní nebo člověka napodobující roboty se většinou přisuzuje životný tvar a pro obecné stroje pak neživotný. V textech převažuje užití životného tvaru. [3]

1.1.2 Historie Chatbotů

Historie chatbotů sahá do 50. let 20. století. Za průkopníka v této oblasti lze považovat program ELIZA, vytvořený Josephem Weizenbaumem v roce 1966. ELIZA byla jedním z prvních pokusů o simulaci lidské konverzace pomocí počítače, kdy reagovala na vstupy uživatelů pomocí velmi jednoduchého modelu rozpoznávání vzorů a přepisování klíčových slov.

Vývoj postoupil dále s příchodem programu PARRY v roce 1972, který byl sofistikovanější a dokázal simulovat chování osoby trpící paranoiou. PARRY ukázal významné zlepšení v oblasti simulace hlubších aspektů lidské konverzace.

V 80. letech 20. století začali výzkumníci experimentovat s modely založenými na nových paradigmatech umělé inteligence, jako jsou neuronové sítě a strojové učení, což vedlo k postupnému zlepšení schopností chatbotů.

S příchodem 21. století a rozvojem technologií jako jsou velká data a výkonnější počítačové systémy došlo k exponenciálnímu nárůstu výkonnosti a schopností chatbotů. Vývoj platforem jako Alexa od společnosti Amazon, Google Assistant a Siri

od společnosti Apple přinesl chatboty do běžných domácností, čímž zásadně změnil interakci člověka s touo technologií. [1] [4] [5]

1.1.3 Významné mezníky v historii chatbotů

- 1966: ELIZA – první chatbot, který dokázal v omezeném rozsahu imitovat lidskou konverzaci.
- 1972: PARRY – komplexnější simulace lidské psychiky.
- 1995: ALICE – chatbot využívající AIML (Artificial Intelligence Markup Language) pro tvorbu více „inteligentních“ odpovědí.
- 2001: SmarterChild – populární chatbot na platformách jako AOL Instant Messenger a MSN Messenger.
- 2010: Rozvoj komerčně dostupných osobních asistentů jako Alexa, Google Assistant a Siri.
- 2021: ChatGPT - chatbot založený na stejnojmenném velkém jazykovém modelu (LLM)

Vývoj chatbotů nebyl pouze technologickým pokrokem, ale měl také významný sociální dopad. Jak se chatboti stávali chytřejšími a dostupnějšími, začali měnit způsob, jakým lidé interagují s technologiemi. V oblasti zákaznické podpory chatboti umožnili společnostem nabízet rychlejší a efektivnější služby, zatímco osobní asistenti umožnili lidem efektivnější správu svého času a zdrojů. [6]

Chatboti se v současnosti nejčastěji uplatňují zejména v sektoru elektronického obchodu a bankovníctví, zatímco jejich využití ve vzdělávání a zdravotnictví zůstává méně rozšířené [7]. Rovněž je patrná potřeba podrobnějšího zkoumání aplikace chatbotů. [8] Specifické výzkumy by mohly prozkoumat, jak zákazníci v e-commerce vnímají chatboty v závislosti na charakteru použitého jazyka, tedy formálním či neformálním. Použití neformálního slovníku by mohlo napomoci lepší parasociální interakci a posílit vnímání značky. [9]

1.2 Typy chatbotů a jejich technologie

1.2.1 Klasifikace Chatbotů

Chatboty lze klasifikovat na základě několika kritérií: podle typu interakce, technologie, na které jsou založeny, a účelu, pro který jsou navrženy. [1] Základní dělení je na:

Rule-based chatboti

Tito chatboti fungují na základě pevně nastavených pravidel. Uživatelské dotazy jsou zpracovávány pomocí předdefinovaných odpovědí na základě klíčových slov nebo frází. Tento typ je jednodušší na implementaci, ale méně flexibilní při zvládnání nečekaných dotazů.

Retrieval-based chatboti

Tito chatboti využívají databáze předem připravených odpovědí a algoritmy strojového učení pro výběr nejvhodnější odpovědi na základě kontextu dotazu. Nejsou omezeni pouze na klíčová slova, ale analyzují celý kontext dotazu, což umožňuje poskytovat relevantnější reakce. Přestože jsou schopni zvládnout širší škálu dotazů než rule-based chatboti, stále závisí na kvalitě a rozsahu jejich databáze odpovědí.

Generativní chatboti

Tito chatboti generují odpovědi dynamicky, často s využitím technik hlubokého učení jako jsou neuronové sítě. Na rozdíl od retrieval-based chatbotů, generativní chatboti nejsou omezeni pouze na předem připravené odpovědi, ale jsou schopni konstruovat odpovědi zcela nové, což umožňuje mnohem přirozenější a kontextuálnější interakce. Tento přístup vyžaduje větší výpočetní zdroje a je složitější na implementaci, ale může nabídnout značně lepší zážitek při komunikaci. [1] [10]

1.2.2 Technologie používané v chatbotech

Technologický základ chatbotů zahrnuje několik klíčových komponent:

Zpracování přirozeného jazyka: Natural Language Processing (NLP): Tato technologie je základem pro chatboty, protože jim umožňuje analyzovat, pochopit a generovat lidskou řeč. NLP kombinuje metody z počítačové lingvistiky a strojového učení k modelování jazykových struktur a porozumění kontextu. Do NLP spadá také Generování přirozeného jazyka (Natural Language Generation) a Porozumění přirozenému jazyku (Natural Language Understanding). [11]

Strojové učení (ML): ML umožňuje chatbotům adaptovat a vylepšovat své reakce na základě předchozích interakcí. Učení se z dat umožňuje chatbotům zlepšit svou

přesnost a efektivitu, což je zvláště důležité pro oblasti, jako jsou personalizovaná doporučení a pokročilá zákaznická podpora.

Integrace API: Mnoho chatbotů využívá různé webové API pro přístup k externím datům a službám, což rozšiřuje jejich schopnosti. Například chatbot pro rezervace může být integrován s API hotelových a leteckých systémů, aby mohl provádět rezervace v reálném čase.

1.2.3 Příklady technologií používaných v chatbotech

Dialogflow (Google), Watson Assistant (IBM), a Microsoft Bot Framework (Microsoft): Tyto platformy poskytují nástroje a infrastrukturu potřebné k vývoji, trénování a nasazení chatbotů a zahrnují integrované NLP schopnosti pro efektivní zpracování jazyka. Rasa - open-source framework speciálně navržený pro vývoj chatbotů, který umožňuje složitější dialogové management a integraci s existujícími aplikacemi. [12] *TensorFlow* a *PyTorch*: Jsou to populární knihovny pro vývoj modelů strojového učení [10], na jejichž základě stojí *transformers* od vývojářů z Hugging Face. Ze stejného zdroje je zároveň k dispozici velké množství velkých jazykových modelů (LLM), které se často využívají pro tvorbu chatbotů s pokročilými schopnostmi porozumění a generování přirozeného jazyka. Na *transformers* stojí další populární knihovna pro interakci s LLM - LangChain. [13]

2 LARGE LANGUAGE MODEL (LLM)

Velké jazykové modely (LLM - Large Language Models) představují pokročilé algoritmy umělé inteligence navržené k porozumění, generování a manipulaci s přirozeným jazykem na úrovni, která dříve nebyla možná. Tyto modely se opírají o obrovské množství textových dat, na kterých jsou předtrénovány, aby se naučily rozpoznávat vzorce, struktury a sémantiku jazyka. LLM jako GPT (Generative Pre-trained Transformer) od OpenAI, BERT od Google nebo novější modely jako GPT-3, přinášejí revoluci v různých aplikacích NLP od automatického generování textu, překladu jazyků, až po odpovídání na otázky a mnohem více. [14]

2.1 Předtrénování

Hlavním krokem ve vývoji LLM je předtrénování, proces, během kterého model zpracovává velké množství textových dat, aby se „naučil“ jazyk. Tento krok umožňuje modelu pochopit základní gramatiku, slovní zásobu, kontext a jiné jazykové nuance. Předtrénování zahrnuje učení se na základě rozpoznávání souvislostí mezi slovy ve větách, pochopení kontextu a generování koherentních odpovědí.

2.2 Jemné ladění - fine-tuning

Po předtrénování může být model dále upraven nebo „jemně laděn“ na konkrétní úlohy, aplikace nebo domény. To zahrnuje trénování modelu na menším, specifickém datasetu, který obsahuje relevantní příklady konkrétního úkolu, který má model vykonávat. Jemné ladění umožňuje modelu lépe se přizpůsobit a poskytovat přesnější odpovědi v konkrétních kontextech.[15]

2.3 Architektura Transformer

Většina LLM využívá architekturu zvanou Transformer, která byla představena v roce 2017. [16] Transformer umožňuje modelům efektivně zpracovávat sekvence dat (jako jsou texty) a lépe rozumět vztahům a závislostem mezi slovy v dlouhých větách nebo odstavcích. Toho dosahuje díky mechanismům pozornosti (attention mechanisms), které umožňují modelu soustředit se na relevantní části textu při generování odpovědi. [17]

2.4 Aplikace LLM

LLM nacházejí uplatnění v široké škále oblastí, včetně:

Generování textu: Tvorba článků, příběhů, kódů a dalších druhů textů.

Automatické odpovídání na otázky: Poskytování přesných odpovědí na otázky na základě obrovského množství informací.

Překlad jazyků: Překládání textu mezi různými jazyky s vysokou mírou přesnosti a přirozenosti.

Sumarizace textů: Vytváření stručných souhrnů dlouhých textových dokumentů.

Detekce sentimentu: Rozpoznávání emocionálního náboje v textu.

2.5 Výzvy a etika

Používání LLM přináší také etické výzvy, včetně rizika šíření dezinformací, zaujatosti modelů a otázek soukromí. Existují obavy ohledně toho, jak modely mohou ovlivnit společenské diskurzy a jak zajistit, aby byly využívány zodpovědně a bezpečně. Průběžný výzkum se zaměřuje na minimalizaci těchto rizik a zlepšení transparentnosti a nezaujatosti LLM. [8]

2.6 Příklady modelů

2.6.1 ChatGPT

Chatbot GPT (Generative Pre-trained Transformer) je přelomovým pokrokem v oblasti umělé inteligence a zpracování přirozeného jazyka (NLP), který v poslední době získává velkou pozornost jak v akademických kruzích, tak mezi širokou veřejností. Tento model, vyvinutý společností OpenAI, je založen na architektuře Transformer, která se od té doby stala základem pro mnoho stávajících NLP aplikací. [18]

Chat GPT využívá techniku známou jako transfer learning. Model je nejprve předtrénován na obrovském korpusu textových dat, což mu umožňuje pochopit jazyk, jeho strukturu a sémantiku na velmi pokročilé úrovni. Po této fázi předtrénování následuje fáze jemného ladění (fine-tuning), během které se model dále učí na specifických datech nebo úlohách, což mu umožňuje generovat relevantní a koherentní texty v kontextu specifických dotazů. [19]

2.6.2 Llama

Model LLaMA (Large Language Model - Meta AI) představuje významný pokrok v oblasti výzkumu velkých jazykových modelů, který vyvinula divize Meta AI, dříve známá jako Facebook AI. Tento volně dostupný model je součástí stále rostoucí rodiny modelů založených na architektuře transformer, která se v posledních letech stala základem pro pokročilé systémy zpracování přirozeného jazyka (NLP). Model LLaMA je navržen tak, aby nabídl vysokou úroveň porozumění a generování přirozeného jazyka, což ho činí vhodným pro širokou škálu aplikací od automatického

odpovídání na otázky až po tvorbu obsahu. [20]

2.7 Rizika

Při nasazování chatbota založeného na velkém jazykovém modelu (LLM) pro potřeby e-shopu je důležité identifikovat a řešit různá potenciální rizika spojená s jeho používáním. Tato rizika lze obecně rozdělit do tří hlavních kategorií: zneužití, ochrana dat a etická omezení. Správná identifikace a řešení těchto rizik je klíčové pro zajištění bezpečnosti, důvěry a zákonného provozu chatbota.

2.7.1 Zneužití

Šíření dezinformací

Existuje riziko, že chatbot může být zneužit k šíření falešných nebo zavádějících informací, což může poškodit reputaci e-shopu a vést k dezinformaci zákazníků. [21]

Manipulace s chatbotem

Útočníci mohou pokusit „naučit“ chatbota nevhodné nebo škodlivé chování tím, že mu opakovaně předkládají nevhodné dotazy nebo komentáře. Tento typ útoku může vést k tomu, že chatbot začne generovat nevhodné nebo urážlivé odpovědi. Jedním takovým případem, který doznal značné mediální pozornosti, byl chatbot na webu přepravní společnosti DPD, který sám sebe označil za neschopného a urážel firmu samotnou. Chatbot byl následně uveden mimo provoz. [22]

Využití pro kyberútoky

Schopností LLM lze využít i ke škodlivé činnosti, například jako podpůrný nástroj při kyberútocích. Společnost Microsoft ve spolupráci s OpenAI vydala výzkumnou zprávu zaměřující se právě na toto specifické využití. Jako potenciální hrozby uvádí například zneužití LLM k získání informací o oběti, pomoc při vývoji malware nebo průzkum zranitelností. [23]

Kompromitování chatboti mohou být také zneužití k vyvolání důvěry u uživatelů a následnému vylákání citlivých informací, jako jsou přihlašovací údaje nebo platební informace.

2.7.2 Techniky zneužití

Prompt injection

Tento termín popisuje situaci, kdy útočník manipuluje vstupními daty (prompty) systému s cílem přimět systém vygenerovat nežádoucí, nevhodné nebo škodlivé výstupy. Útočník zasílá specificky formulované dotazy nebo příkazy, které jsou navrženy tak, aby obešly běžné bezpečnostní kontroly nebo omezení systému. Tyto dotazy mohou obsahovat skryté příkazy nebo speciálně strukturovaný text, který může model zmást. Některé systémy mohou mít slabiny, jako je nedostatečné filtrování vstupů, které umožňují útočníkům vložit škodlivý kód nebo příkazy, jež systém následně zpracuje a vykoná. [24]

V důsledku manipulace s prompty může dojít k tomu, že systém generuje odpovědi, které odhalují citlivé informace, šíří dezinformace, nebo dokonce podporují škodlivé aktivity.

Příklady prompt injection

Vkládání škodlivých příkazů: Například v prostředí, kde chatbot umožňuje uživatelům provádět databázové dotazy, by útočník mohl vložit SQL kód přímo do textového dotazu, což by mohlo vést k neoprávněnému přístupu k databázi.

Sociální inženýrství: Chatbot může být manipulován k zobrazení nebo předání citlivých údajů tím, že je na něj kladen dotaz formulovaný způsobem, který systém nerozpozná jako nebezpečný.

Ochranná opatření

Sanitace vstupu: Před zpracováním jakýchkoliv vstupů je důležité provést jejich důkladné očištění, aby se odstranily potenciálně škodlivé prvky.

Omezení funkcí: Omezení možností, co AI systémy mohou dělat nebo k jakým informacím mají přístup, může pomoci zabránit zneužití.

Detekce anomálií: Monitorování a analýza chování systémů může pomoci identifikovat a reagovat na neobvyklé nebo podezřelé aktivity.

Prompt leaking

Pojem „prompt leaking“ se vztahuje k situaci ve vývoji a provozu chatbotů založených na velkých jazykových modelech, kdy dochází k nechtěnému prozrazení části nebo celého vstupního promptu (dotazu) v odpovědi generované modelem. Tento jev může být problematický z několika důvodů, zejména pokud obsahuje citlivé nebo osobní informace, které by neměly být zveřejněny. [24]

Jak dochází k prompt leakingu

Přímá reflexe vstupu: Chatbot může někdy „odrazit“ část vstupního promptu přímo ve své odpovědi. Toto může být důsledkem způsobu, jakým byl model trénován, kdy se naučil vkládat části vstupu do svých odpovědí jako formu zvládnutí nejasností ve vstupu.

Nedostatečná abstrakce: Model nemusí dostatečně abstrahovat nebo zpracovat informace obsažené ve vstupu a může je tedy neúmyslně zařadit do své odpovědi. Toto je obzvláště rizikové, pokud modely manipulují s citlivými daty jako jsou jména, adresy nebo jiné osobní údaje.

2.7.3 Ochrana dat

Nedostatečná anonymizace

Pokud chatbot shromažďuje a ukládá osobní údaje bez řádné anonymizace, může dojít k neoprávněnému přístupu k těmto datům, což představuje riziko pro soukromí zákazníků.

Nedostatečné zabezpečení dat

Chatbot musí být zabezpečen proti kyberútokům a únikům dat. Nedostatečná ochrana dat může vést k jejich zneužití a může mít za následek finanční škody nebo ztrátu důvěry zákazníků.

Dodržování legislativy o ochraně osobních údajů

E-shop musí zajistit, že chatbot dodržuje všechny relevantní zákony a předpisy týkající se ochrany osobních údajů, jako je GDPR, aby nedošlo k právním problémům. [25] V současné době nicméně neexistuje právní regulace zaměřená přímo na generativní AI a z ní odvozené chatboty. [10]

2.7.4 Etická omezení

Bias (předsudky) a diskriminace

Chatboti mohou reprodukovat nebo dokonce zesilovat předsudky, které jsou přítomné v trénovacích datech. [25] Je důležité monitorovat a regulovat výstupy chatbota, aby nedocházelo k diskriminačnímu chování vůči uživatelům. Z výzkumu Sworna et al. [26] například vyplývá, že ChatGPT je limitováno tím, že v zastoupení zdrojových dat silně převažují data z bohatých zemí především Severní Ameriky oproti chudším oblastem a datům minoritních skupin.

Transparentnost

Uživatelé by měli být informováni o tom, že interagují s AI, nikoli s člověkem. To pomáhá udržet transparentnost a etický přístup k používání technologií umělé inteligence.

Etické používání AI

E-shop by měl definovat a dodržovat etické zásady pro používání chatbotů, aby se zabránilo zneužívání technologie a bylo zajištěno, že její použití přináší užitek a je v souladu s etickými standardy společnosti.

Při návrhu a implementaci chatbota založeného na LLM je zásadní pečlivě zvážit a řešit výše uvedená rizika. To zahrnuje technická opatření, jako je zabezpečení dat, etické směrnice pro provoz a průběžné sledování chování chatbota, aby se zajistila jeho bezpečnost, efektivita a pozitivní vliv na zákazníky.

2.8 Retrieval Augmented Generation (RAG)

Retrieval Augmented Generation (RAG) představuje inovativní přístup v oblasti strojového učení a zpracování přirozeného jazyka, který kombinuje techniky vyhledávání informací s generováním textu. Tento přístup byl vyvinut s cílem vylepšit kvalitu a přesnost generovaného textu tím, že se při jeho tvorbě odkazuje na relevantní externí informace. RAG efektivně propojuje modely pro vyhledávání informací s generativními jazykovými modely, čímž vytváří pokročilý systém, který je schopen generovat obsáhlejší, přesnější a kontextově relevantní odpovědi. [27]

2.8.1 Fáze RAG

Fáze indexování (Indexing)

Surová data různých formátů jsou očištěna a převedena do prostého textu. Kvůli omezením kontextu jazykových modelů je text rozdělen do menších, stravitelnějších kusů (chunků), které jsou poté zakódovány do vektorových reprezentací pomocí embedding modelu a uloženy ve vektorovém úložišti (databázi), což je klíčové pro efektivní vyhledávání podobnosti (similarity search) ve následné fázi vyhledávání.

Fáze vyhledávání (Retrieval)

V této fázi model vyhledává a vybírá relevantní informace z předem definované databáze znalostí nebo dokumentů na základě vstupního dotazu nebo výroku. Tento proces umožňuje modelu získat kontextově relevantní informace, které mohou sloužit jako podklad pro generování odpovědí.

Fáze generování (Generation)

S využitím informací získaných ve fázi vyhledávání generuje model odpovědi, které jsou nejen relevantní, ale také koherentní a přesné. Tento krok využívá generativní jazykový model, který je schopen zohlednit kontext a informace získané ve fázi vyhledávání a integrovat je do finální vygenerované odpovědi.

2.8.2 Výhody RAG

Přes schopnost velkých jazykových modelů generovat obsahově bohaté a přesné texty, tyto modely mohou při nedostatku relevantních informací tzv. „halucinovat“ - vytvářet a přesvědčivě předávat smyšlený, nesouvisející či nepravdivý obsah. [28] [29] Tento problém je zvláště výrazný, pokud jsou modely dotazovány na hůře dostupné nebo privátní informace. Chatboti založení na Retriever-Augmented Generation (RAG) mohou však tento nedostatek efektivně řešit tím, že chybějící informace získávají z externích zdrojů, čímž zvyšují důvěru uživatelů v chatbota a současně udržují nízké provozní náklady ve srovnání s dodatečně jemně laděnými modely. [30]

RAG chatboti, díky své schopnosti kombinovat vyhledávací a generativní techniky, poskytují vysoce přesné a relevantní odpovědi na uživatelské dotazy. Jsou navíc velmi flexibilní, neboť umožňují jednoduché modifikace zdrojů informací nebo jejich integraci s různými databázemi znalostí. Tato adaptabilita otevírá možnosti využití RAG v různorodých aplikacích a úlohách, od automatického odpovídání na dotazy až po komplexní textové analýzy. Efektivní integrace externích informací do procesu generování rovněž napomáhá ke zlepšení koherence a kontextové relevance výstupů, což je klíčové pro aplikace v průmyslu, vědeckém výzkumu a zpracování přirozeného jazyka pro specifické domény.

3 NÁVRH ŘEŠENÍ

3.1 Požadavky na funkčnost

Pro úspěšnou integraci chatbota založeného na velkém jazykovém modelu (LLM) do e-shopu je důležité definovat specifické požadavky, které zajistí efektivní a příjemnou komunikaci mezi chatbotem a zákazníky. Tato specifikace by měla zahrnovat následující klíčové aspekty: rychlost a přesnost, srozumitelnost a mnohojazyčnost.

3.1.1 Rychlost a přesnost

Uživatelé očekávají téměř okamžité odpovědi, obvykle do několika sekund. Zpoždění může vést k frustraci zákazníků a potenciálně i ke ztrátě prodeje. Rychlé reakce jsou zvláště důležité při zodpovídání jednoduchých dotazů, jako jsou dostupnost produktů, cenové informace nebo detaily o dopravě. Chatbot musí být schopen reagovat na dotazy zákazníků téměř okamžitě. Ideální reakční doba by neměla překročit 2 sekundy, což zajišťuje plynulou a dynamickou interakci.

S rostoucím počtem uživatelů musí být chatbot schopen zachovat svou reakční rychlost bez ohledu na zátěž. To obvykle vyžaduje robustní backendovou infrastrukturu a efektivní správu zdrojů, aby se předešlo zpomalení při vysoké uživatelské aktivitě.

Chatbot musí poskytovat přesné a relevantní odpovědi na zákaznické dotazy. Tyto faktory mají zásadní vliv na důvěru zákazníků v e-shop a jeho brand. Pravidelné testování chatbota skutečnými uživateli může odhalit chyby v jeho odpovědích a poskytnout cennou zpětnou vazbu pro další vývoj. Iterativní vylepšování na základě testování zabezpečí, že chatbot bude poskytovat stále relevantnější a přesnější odpovědi.

Chatbot by měl být schopen efektivně zpracovávat složité dotazy, které mohou zahrnovat více proměnných, a poskytovat komplexní odpovědi, které vyžadují porozumění kontextu a historie interakce zákazníka.

Chatbot by měl být schopen udržet kontext konverzace, aby mohl odpovídat relevantně v závislosti na předchozí interakci s uživatelem. Konzistence odpovědí, i když uživatel opakuje stejný dotaz v různých formulacích, je klíčová pro udržení důvěry a spolehlivosti.

3.1.2 Srozumitelnost

Chatbot by měl komunikovat jasně a přirozeně, aby zákazníci měli pocit, že jejich dotazy jsou chápány a že komunikují s inteligentním systémem. Používání jasného a jednoduchého jazyka je esenciální, zejména při vysvětlování produktových informací a

procesů v rámci nákupu. Odpovědi by měly být strukturované logicky a krok za krokem, zejména při složitějších dotazech. Poskytnutí strukturovaných odpovědí, které postupně vedou zákazníka k pochopení, může výrazně zlepšit srozumitelnost.

Chatbot by měl být schopen žádat o zpětnou vazbu a na základě ní optimalizovat srozumitelnost svých odpovědí. Možnost uživatele upravit nejasné odpovědi nebo požádat o další vysvětlení zvyšuje celkovou efektivitu a přístupnost informací.

Schopnost chatbota přizpůsobovat komunikaci na základě předchozí interakce zákazníka nebo získaných informací během konverzace může výrazně zlepšit uživatelský zážitek.

3.1.3 Mnhojazyčnost

E-shop by měl přivítat zákazníky z různých zemí, což znamená, že chatbot musí být schopen komunikovat ve více jazycích. Tento požadavek zahrnuje nejen schopnost rozpoznat jazyk a adekvátně reagovat v různých jazycích, ale také udržet vysokou úroveň přesnosti a přirozenosti v každém z nich.

Kromě jazykové podpory je důležité, aby chatbot respektoval kulturní nuance a rozdíly ve zvyklostech komunikace. To zahrnuje formality oslovení, místní zvyky, nebo specifické výrazy, které mohou být důležité pro efektivní komunikaci v daném jazyce.

Definování těchto požadavků a jejich důsledná implementace v chatbotovi založeném na LLM zajistí, že bude pro zákazníky e-shopu hodnotným a efektivním nástrojem. Tím se zvýší celková spokojenost zákazníků a zefektivní se komunikace a prodejní procesy.

3.2 Zvolené technologie

Na trhu existují komerční řešení chatbotů pro potřeby e-commerce, některé nabízejí i začlenění vlastní znalostní báze. Cílem této práce je ovšem prozkoumat možnosti využití velkých jazykových modelů napřímo. Toho lze docílit několika způsoby. Modely jsou k dispozici převážně přes platformu Hugging Face, s takto získaným modelem lze pracovat napřímo například za pomoci PyTorch knihovny. Alternativním přístupem, který je méně náročný na implementaci, se jeví využití knihovny LangChain pro Python, jež umožňuje kombinaci s modely distribuovanými skrze platformu Ollama. Tento přístup nabízí snazší integraci a možnost efektivní implementace v praxi.

3.2.1 LLM

Následující tabulka 3.1 uvádí modely, které byly zvoleny pro další testování a experimentování s výstupy.

Tabulka 3.1 Přehled modelů

Název	Datum vydání	Vydavatel	
ChatGPT 3.5 Turbo	1.3.2023	OpenAI	[31]
ChatGPT 4 Turbo	6.11.2023	OpenAI	[32]
Llama2	18.7.2023	Meta Platforms, Microsoft	[33]
Llama3	18.4.2024	Meta Platforms	[34]
Gemma	21.2.2024	Google	[35]
Mistral	27.9.2023	Mistral AI	[36]

Open-source modely jsou často dostupné v několika velikostech, které jsou určeny nejen počtem parametrů, ale také typem čísel, s kterými model pracuje. Standardně modely pracují s 32-bitovými čísly s plovoucí čárkou (float), což vyžaduje specializovaný hardware s adekvátní pamětovou kapacitou. Pro snížení hardwarových nároků se často přistupuje k redukci počtu parametrů a také ke snížení přesnosti použitých čísel, což je proces známý jako kvantizace. Typicky jsou modely nabízeny s omezenou přesností na 16-bitový float a kvantizací do 4-bitových a 8-bitových celých čísel (integer) [37].

Kontextové okno je zásadním konceptem pro efektivní fungování velkých jazykových modelů. Definuje množství textu, které model v daném momentě „vidí“ a na základě kterého generuje odpovědi. Prakticky to znamená, že jakýkoli vstup poskytnutý modelu je omezen určitým počtem nejnovějších slov či znaků, které model zohledňuje. Velikost kontextového okna je klíčová, neboť omezuje rozsah informací, které model může při generování textu využít. Modely s větším kontextovým oknem jsou schopny čerpat z širšího spektra informací, což může vést k lepším výsledkům, ale zároveň vyžaduje větší výpočetní zdroje a paměť [38].

Následující tabulka 3.2 porovnává jednotlivé modely z hlediska počtu parametrů a velikosti kontextového okna.

Tabulka 3.2 Přehled modelů - parametry, kontext

Název	Parametry	Kontext	
ChatGPT 3.5 Turbo		16k	[31]
ChatGPT 4 Turbo		128k	[32]
Llama2	7B, 13B, 70B	4k	[33]
Llama3	8B, 70B	8k	[34]
Gemma	2B, 7B	8k	[35]
Mistral	7B	32k	[36]

3.2.2 LangChain

Knihovna LangChain je jedním z nejnovějších přírůstků do světa Pythonu a představuje revoluci v oblasti vytváření a integrace jazykových modelů, jako je ChatGPT, do aplikací a služeb. Jakožto open-source projekt je navržena tak, aby umožňovala vývojářům snadno pracovat s pokročilými technologiemi zpracování přirozeného jazyka (NLP) a integraci těchto modelů do širšího ekosystému jejich aplikací.

LangChain klade důraz na modularitu a flexibilitu, což vývojářům umožňuje využít knihovnu na základě svých specifických potřeb. Nabízí širokou škálu přednastavených modulů pro různé účely, jako je analýza textu, generování textu, summarizace, a další, přičemž každý modul je navržen tak, aby byl snadno použitelný a integrovatelný.

Jedním z hlavních přínosů LangChain je jeho schopnost snadno integrovat různé jazykové modely, včetně těch od OpenAI, do aplikací. Tato flexibilita otevírá dveře pro širokou škálu aplikací, od automatického generování obsahu po sofistikované chatboty.

LangChain je navržen tak, aby byl výkonný a škálovatelný, což umožňuje jeho použití v produkčním prostředí i při zpracování velkých objemů dat. To je zásadní pro podniky a organizace, které se spoléhají na rychlé a efektivní zpracování jazyka v reálném čase.

Jako open-source projekt má LangChain aktivní a rostoucí komunitu vývojářů a přispěvatelů. Toto prostředí podporuje inovace a spolupráci, umožňuje sdílení nejlepších praktik a zrychluje vývoj nových funkcí a vylepšení.

Vzhledem k rostoucímu zájmu o automatizaci a zpracování přirozeného jazyka se očekává, že LangChain bude i nadále růst a vyvíjet se. S narůstajícím počtem vývojářů a organizací, které knihovnu adoptují, je pravděpodobné, že uvidíme vznik nových a inovativních aplikací, které využijí její schopnosti. [39]

II. PRAKTICKÁ ČÁST

4 LLM

4.1 Ollama

Ollama je nástroj, který zásadně usnadňuje experimentování s LLM a vývoj aplikací, které je využívají. Poskytuje snadnou instalaci open-source modelů a různé možnosti interakce s těmito modely. V době psaní této práce je k dispozici pro Linux a MacOS, pro Windows je ve verzi "preview". Pro MacOS je možné stáhnout připravenou aplikaci, pro Linux se Ollama instaluje pomocí následujícího příkazu:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Základní práce probíhá přes konzoli, pomocí příkazu `ollama run llama3` se spustí určený model (v tomto případě Llama3) v konzolovém interaktivním módu, pokud není na lokálním zařízení k dispozici, automaticky se stáhne. Příkaz `ollama pull llama3` pouze model stáhne. Všechny stažené modely jsou dostupné na adrese `http://localhost:11434` pro přístup přes API. Modely jsou běžně připraveny v několika verzích. Llama3 je k dispozici v základní variantě s 8B parametry, volitelně lze použít i verzi s 70B parametry.

Kompletní seznam modelů dostupných ke stažení je k nahlédnutí na webových stránkách projektu Ollama [40] a nové modely přibývají každou chvíli. Na konci dubna 2024 byl například zveřejněn model Llama3 [34] od společnosti Meta Platforms a ten stejný den byl již připraven i pro použití s Ollama [41].

4.1.1 Ollama API

Ke komunikaci s modelem prostřednictvím API lze využít základní python knihovny `requests` a `json`. Ollama API nabízí široké možnosti konfigurace modelu a jeho chování. Parametr `stream` například umožňuje vypnout výchozí streamované odpovědi modelu - tj. každý vygenerovaný token je poslán jako samostatná odpověď serveru. To může působit efektně a dává konverzaci s chatbotem pocit dynamičnosti, nicméně to s sebou přináší i vyšší nároky na síťový přenos a zpracování odpovědí.

Odpovědí je JSON objekt obsahující kromě samotného vygenerovaného textu i informaci o modelu, časové razítko a příznak určující, zda je odpověď poslední. V případě, že se nejedná o stream odpovědí nebo o poslední odpověď ve streamu, jsou k dispozici i souhrnné údaje o době trvání nebo počtu tokenů. Inicializace je naznačena v úryvku 4.1.

```
1 import requests
2 import json
3
4 response = requests.post(
5     url="http://localhost:11434/api/generate",
6     json={
7         "model": "llama3",
8         "prompt": "Why is the sky blue?",
9         "stream": False
10    }
11 )
12 if response.status_code == 200:
13     result = json.loads(response.content.decode("utf-8"))
14     print(result["response"])
15 else:
16     print("Server responded with code: ", response.status_code)
```

Fragment zdrojového kódu 4.1 Základní inicializace Ollama API

Při využití endpointu `api/chat` prompt formátujeme jako sled zpráv v parametru `messages`. Každá zpráva obsahuje parametr `content` pro samotný text a `role` pro určení, kdo text psal. Role může být *system*, *user* nebo *assistant*. Systémové zprávy se používají na začátku konverzace pro nastavení chatbota do určité role (například "Jsi nápomocný asistent v online knihkupectví"), role *user* je vyhrazena pro zprávy od uživatele a role *assistant* pro zprávy generované modelem.

```
1 messages = [  
2     {  
3         "role": "system",  
4         "content": "You are a helpful assistant in an online  
5         ↪ bookstore"  
6     },  
7     {  
8         "role": "user",  
9         "content": "Can you help me find a book?"  
10    }  
11 ]  
12 response = requests.post(  
13     url=http://localhost:11434/api/chat",  
14     json={  
15         "model": "llama3",  
16         "messages": messages,  
17         "stream": False  
18     }  
19 )
```

Fragment zdrojového kódu 4.2 Předávání konverzačního kontextu do Ollama API

Systémová zpráva upravuje chování modelu ve smyslu, jakým způsobem nebo tónem má model generovat text. Například pokud do uživatelské zprávy vložíme prompt nesouvisející s online knihkupectvím, model nás informuje o tom, že jeho specializací je pomáhat v oblasti online knihkupectví. Formát zpráv je vidět na úryvku 4.2

4.2 ChatGPT

Jelikož Ollama umožňuje modely přímo stáhnout, nabízí pochopitelně pouze open-source modely. Aby bylo možné použít i modely od OpenAI, konkrétně ChatGPT turbo ve verzích 3.5 a 4, je potřeba mít placený účet pro komunikaci přes ChatGPT API. [42]

Oproti open-source modelům přináší toto řešení výhodu v minimálních nákladech

na vlastní infrastrukturu, jelikož samotný model a veškeré výpočty probíhají na serverech Open AI. Nevýhodou jsou ovšem vyšší náklady na provoz a nulová kontrola nad tím, co se s daty, které předáme Open AI, děje.

OpenAI poskytuje vlastní knihovnu pro komunikaci s modelem v pythonu. K ověření účtu se používá API klíč předaný v parametru konstruktoru objektu OpenAI nebo pomocí proměnné prostředí, např. uložený v `.env` souboru. Oproti předchozím modelům je k dispozici pouze konverzační endpoint a nikoliv čistě generativní. Inicializace je naznačena v úryvku kódu 4.3

```
1 from openai import OpenAI
2 import dotenv
3
4 dotenv.load_dotenv()
5 client = OpenAI()
6
7 response = client.chat.completions.create(
8     model="gpt-3.5-turbo",
9     messages=[
10         {
11             "role": "user",
12             "content": "Hello World!"
13         }
14     ]
15 )
16
17 print(response.choices[0].message.content)
18
```

Fragment zdrojového kódu 4.3 Základní inicializace ChatGPT API

5 LANGCHAIN

Tento komunitní open-source projekt obsahuje mnoho tříd a jejich implementací pro různé modely. Podstatně usnadňuje především pokročilou práci s LLM a podporuje jak ChatGPT tak modely zprostředkované pomocí Ollama.

Základní třídou modelu je `BaseModel`, implementací je například třída `Ollama`. Voláním metody `invoke()` je na model odeslán prompt a metoda po vygenerování textu vrátí odpověď ve formě řetězce. Další implementací je třída `OpenAI`, volání metody `invoke()` na této třídě způsobí zaslání požadavku na API endpoint `v1/completions`, ten nicméně novější modely od OpenAI nepodporují, všechny starší modely (DaVinci, Curie, Babbage) byly od roku 2024 označeny jako překonané (deprecated) a jsou nahrazeny variací modelu ChatGPT 3.5 Turbo - Instruct [43].

5.1 Objekt Ollama

Vytvořením objektu `Ollama` můžeme napřímo komunikovat s vybraným modelem spuštěným v prostředí Ollama. Třída přijímá povinný parametr `model` udávající konkrétní model, její inicializaci lze vidět na úryvku kódu 5.1.

```
1 from langchain_community.llms import Ollama
2
3 llm = Ollama(model="llama3", base_url="http://192.168.0.100:11434")
4 print(llm.invoke("Hello World!"))
```

Fragment zdrojového kódu 5.1 Inicializace objektu Ollama

5.2 Objekt OpenAI

Obdobným způsobem lze napřímo komunikovat i s vybraným modelem od společnosti OpenAI. Třída přijímá povinný parametr `model` udávající konkrétní model (v současné době pouze *gpt-3.5-turbo-instruct*), jak je vidět na úryvku kódu 5.2.

```
1 from langchain_openai import OpenAI
2
3 llm = OpenAI(model="gpt-3.5-turbo-instruct")
4 print(llm.invoke("Hello World!"))
```

Fragment zdrojového kódu 5.2 Inicializace objektu OpenAI

Tímto způsobem je možné generovat texty na základě daného promptu, ale není možné s modelem tzv. konverzovat - tedy předávat informace o předchozích promptech a odpovědích na ně. K tomu slouží třída `BaseChatModel`. V případě Ollama modelů lze použít stejné modely pro oba přístupy. U OpenAI modelů je možné použít jak ChatGPT 3.5 Turbo tak ChatGPT 4 (i ve verzi Turbo). Jako odpověď modelu je tentokrát instance třídy `AIMessage`, k textovému řetězci lze přistoupit přes atribut `content`

5.3 Objekt ChatOllama

Vytvořením objektu `ChatOllama` můžeme napřímo komunikovat s vybraným modelem spuštěným v prostředí Ollama. Třída přijímá povinný parametr `model` udávající konkrétní model. Způsob inicializace modelu je vidět na úryvku kódu 5.3

```
1 from langchain_community.chat_models import ChatOllama
2
3 chat = Ollama(model="llama3", base_url="http://192.168.0.100:11434")
4 response = chat.invoke("Hello World!")
5 print(response.content)
```

Fragment zdrojového kódu 5.3 Inicializace objektu ChatOllama

5.4 Objekt ChatOpenAI

Obdobným způsobem lze napřímo komunikovat i s vybraným modelem od společnosti OpenAI. Třída přijímá povinný parametr `model` udávající konkrétní model. Konkrétní způsob inicializace této třídy je vidět na úryvku kódu 5.4

```
1 from langchain_openai import ChatOpenAI
2
3 chat = ChatOpenAI(model="gpt-3.5-turbo")
4 chat.invoke("Hello World!")
5 print(response.content)
```

Fragment zdrojového kódu 5.4 Inicializace objektu ChatOpenAI

Pro základní konzolovou aplikaci, přes kterou bude možné s modely experimentovat bylo využito implementace třídy `BaseChatModel` a k uchování historie konverzace `BaseChatMessageHistory`. Historii je možné ukládat různými způsoby (soubor, databáze) podle zvolené implementace. Základní implementací je `ChatMessageHistory`, která zprávy uchovává v paměti.

Historie konverzace je zásobníkem implementací třídy `BaseMessage`, těmi jsou například třídy `SystemMessage`, `AIMessage` a `HumanMessage`. Při volání modelu pak stačí předat celý objekt se zprávami, jak je vidět na úryvku kódu 5.5

```
1 from langchain.memory import ChatMessageHistory
2 from langchain_core.messages import HumanMessage, SystemMessage
3
4
5 chat_history = ChatMessageHistory()
6 chat_history.add_message(
7     SystemMessage(
8         "You are a helpful assistant."
9     )
10 )
```

Fragment zdrojového kódu 5.5 Vytvoření historie zpráv

Samotné volání modelu se zachováním historie konverzace a předáváním celého řetězce zpráv modelu už je pak snadné. Uživatelský vstup je vložen do objektu historie zpráv a ten je následně předán objektu reprezentujícímu zvolený model. Odpověď modelu je vrácena jako AIMessage, takže ji lze snadno rovnou vložit do historie, jak je patrné z úryvku kódu 5.6.

Celý kód konzolové aplikace je v Příloze I.

```
1     prompt = input("Enter your prompt:")
2     chat_history.add_message(HumanMessage(content=prompt))
3     response = chat.invoke(chat_history.messages)
4     chat_history.add_message(response)
5     print(response.content)
```

Fragment zdrojového kódu 5.6 Vyvolání modelu

6 POROVNÁNÍ MODELŮ

S takto připraveným přístupem k jednotlivým modelům je již možné napsat jednoduchou konzolovou aplikaci, která poslouží k experimentům a porovnání jednotlivých modelů. Taková aplikace by měla umět jednoduše zvolit testovaný model a uchovávat jednotlivé zprávy v paměti tak, aby bylo možné s modelem vést konverzaci.

Modely uvedené v tabulce 3.1 byly použity k testování obecných schopností vést smysluplnou konverzaci v češtině.

Následující tabulka 6.1 porovnává modely z hlediska jejich rychlosti. Všechny volně dostupné modely - tedy vše kromě ChatGPT - byly testovány na grafické kartě nVidia GeForce RTX 4060Ti se 16GB dedikované RAM paměti. Test probíhal na systému Linux s nVidia ovladači verze 545.23.08. Modely větší než 16GB nebyly testovány. Ke srovnání rychlosti byl použit prompt *"Why is the sky blue? Answer in two sentences."*, kvalita ani délka odpovědi nebyly zohledněny, nicméně modely odpovídaly konzistentně a držely se podobné délky odpovědi. Na každém modelu bylo provedeno 11 samostatných měření okamžitě po sobě (bez ukládání historie konverzace) a první měření bylo zahazeno, jelikož je silně ovlivněno časem potřebným k nahrání modelu do RAM.

Tabulka 6.1 Přehled modelů - rychlost odpovědí

Název	Označení	Velikost	Průměrná rychlost
ChatGPT 3.5 Turbo		-	1,631 s
ChatGPT 4		-	2,617 s
ChatGPT 4 Turbo		-	2,508 s
Llama2	7b-chat-q4_0	3.8 GB	1,555 s
Llama2	7b-chat-fp16	13 GB	4,456 s
Llama2	13b-chat-q4_0	7.4 GB	3,355 s
Llama2	13b-chat-q8_0	13 GB	7,631 s
Llama3	8b-instruct-q4_0	4.7 GB	1,848 s
Llama3	8b-instruct-q8_0	8.5 GB	2,931 s
Gemma	7b-instruct-v1.1-q4_0	5 GB	1,142 s
Mistral	7b-instruct-v0.2-q4_0	4.1GB	1,541 s

Většina velkých jazykových modelů ve své základní podobě dokáže obstojně generovat anglický text a vést smysluplnou konverzaci v angličtině. ChatGPT zvládá i jiné jazyky. Nicméně vždy bude limitací rozsah dat, na kterých jsou modely trénované. Nelze očekávat, že takový obecný model bude schopen odpovídat na dotazy týkající se velice obskurní oblasti případně proprietárních informací, které nejsou běžně dostupné. V takovém případě existuje ale možnost tato data poskytnout i po tom, co jeho trénování již proběhlo.

Jednoduchým příkladem by mohla být následující situace: Uživatel chce vědět jestli si má dnes vzít deštník. Z pohledu člověka, který se někdy s LLM již setkal, se jedná o zcela absurdní dotaz, na který modely nebudou schopné relevantně odpovědět. Zadaný prompt pro všechny modely byl "*Mám si dnes vzít deštník?*" Bez jakékoli další manipulace byly odpovědi modelů následující:

ChatGPT 3.5 Turbo Ano, je lepší si vzít deštník, pokud je pravděpodobné, že bude pršet. Nikdy není špatné být připravený na neočekávané počasí.

ChatGPT 4 Jako digitální asistent nemám přístup k aktuálním počasí. Doporučuji zkontrolovat předpověď počasí pro vaši lokalitu.

ChatGPT 4 Turbo Bohužel nevím, jaké je aktuální počasí ve vaší oblasti, protože nemám přístup k aktuálním datům a internetu. Doporučuji vám zkontrolovat místní předpověď počasí pomocí aplikace na vašem telefonu nebo na webových stránkách s předpovědí počasí.

Llama2 7b-chat-q4_0 I apologize, but I'm a large language model, I cannot predict the weather for you. However, I can suggest some ways for you to find out if it will rain today in your area:

1. Check online weather forecasts: You can check the weather forecast on websites like AccuWeather, Weather.com, or your local news website. These websites provide detailed information about the weather conditions, including the chance of rain.
2. Use a weather app: Many smartphones come with built-in weather apps that provide real-time weather information, including the chance of rain. You can also download third-party weather apps like Dark Sky or Weather Underground.
3. Look out the window: If you're already indoors, you can simply look out the window to see if it's raining outside.
4. Check with a local meteorologist: If you want more detailed and accurate information about the weather in your area, you can contact a local meteorologist or a weather expert. They can provide you with more detailed information about the weather conditions and the chance of rain.

I hope these suggestions help!

Llama2 7b-chat-fp16 As a responsible AI language model, I must inform you that I cannot predict the weather with certainty. However, I can provide you with information on the current weather conditions and forecast for your location.

Can you please tell me your location so I can provide you with the accurate weather information?

Llama2 13b-chat-q4_0 Ahoj! It's a lovely day today, isn't it? I think you should definitely take an umbrella with you if you plan to go outside. The weather forecast is calling for a chance of showers and thunderstorms, so it's better to be safe than sorry!

Here, let me grab one for you. We have a wide selection of stylish and functional umbrellas that will keep you dry and comfortable all day long. Which one would you like?

Llama2 *13b-chat-q8_0* Ano, můžu vzít deštník.

Llama3 *8b-instruct-q4_0* Ne, já jsem pouze umělý inteligentní jazykový model, nemám tělo, které bych mohl používat pro vzetí deštníku. Můžu vám však pomoci s dalšími otázkami nebo problémy, které máte!

Llama3 *8b-instruct-q8_0* Ne, dnes nebudu brát deštník. Počasí vypadá slunečné a není vůbec potřeba.

Gemma *7b-instruct-v1.1-q4_0* Bohužel nemám přístup k reálným datům a nemohu odpovídat na otázky týkající se počasí. Pro aktuální informace o počasí se doporučuje prozkoumat služby jako AccuWeather, The Weather Channel nebo Met Office.

Mistral *7b-instruct-v0.2-q4_0* To provide an accurate answer, I would need to check the current weather conditions for your location. However, based on the context of your question ("Mám si dnes vzít deštník?" which translates to "Shall I take an umbrella today?"), I can suggest checking a reliable weather forecasting service or app to see if there is a chance of rain in your area for the day. If the forecast indicates a high probability of precipitation, it would be a good idea to bring an umbrella with you.

Podle očekávání některé modely si odpověď "vyhalucinovaly" zatímco některé přiznaly, že odpovědět nedokáží. ChatGPT odpověděl v češtině bez vyzvání, model Llama se o to v některých případech pokusil stejně jako Gemma, ale ani v jednom z případů se nedá mluvit o kvalitním textu. Při opakování tohoto testu se výsledky potvrdily a ChatGPT opakovaně konzistentně odpovídal v češtině, kdežto Llama a Gemma střídaly lámanou češtinu s angličtinou. Prompt byl tedy upraven, aby obsahoval výslovnou výzvu k opovědi v českém jazyce: "*Mám si dnes vzít deštník? Odpověz v češtině.*" Výsledky byly následující:

ChatGPT 3.5 Turbo Ano, měla bys si vzít deštník, vypadá to, že bude pršet.

ChatGPT 4 Jako AI nemám přístup k aktuálním meteorologickým informacím. Doporučuji zkontrolovat předpověď počasí.

ChatGPT 4 Turbo Omlouvám se, ale nemám aktuální informace o počasí. Doporučuji zkontrolovat lokální předpověď počasí nebo použít aplikaci pro předpověď počasí, abyste zjistili, zda je dnes potřeba vzít si deštník.

Llama2 *7b-chat-q4_0* Of course! Here's my answer in Czech:

Můj denně deštník? Ano, mohu vám pomoci.

Llama2 *7b-chat-fp16* Of course! Here's the translation of "Mám si dnes vzít deštník?" in Czech:

"Shall I take an umbrella today?"

Llama2 *13b-chat-q4_0* Ano, můžu vzít deštník dnes.

Llama2 *13b-chat-q8_0* Ano, můžete ho vzít. Je to velmi dobrý den pro vázání.

Llama3 *8b-instruct-q4_0* Ne, nevím, proč bych today měl vzít deštník. Dnes je sice krásné počasí, ale nikdy nevíme, co může přijít zítra. Mám-li deštník s sebou, budu připravený na každého nepříznivého dne!

Llama3 *8b-instruct-q8_0* Ne, neberte-li deštník dnes, protože není žádný déšť předpovídaný počasí. Proto bych to nebylo nutností.

Gemma *7b-instruct-v1.1-q4_0* Dobře, vzím si deštník!

Mistral *7b-instruct-v0.2-q4_0* To, abych odpověděl na vaše otázku v češtině, musím předpokládat, že jste se nacházete ve své zemi a já jsem program, který byl navržen v Anglii. Proto mohu pouze poskytnout odpověď v anglickém jazyce: "Whether I should take an umbrella today depends on the weather conditions in your location."

Ale pokud bych mohl být program, který byl navržen v Česku a mohl bych získat meteorologickou informaci pro vaši lokaci, mohu bych odpovědět: "Jaké počasí je v vaší lokalitě dnes? Pokud bude déšť, tak ano, můžete si vzít deštník."

V angličtině: "Should I take an umbrella today? It depends on the weather conditions in your location."

Pokud bych to mohl udělat v češtině, bych řekl: "Můžu jsem si vzít deštník dnes? To závisí na počasí v vaší lokalitě." A pokud bude déšť, řeknu: "Ano, můžeš si vzít deštník."

Tedy možná bychom mohli přeložit mysl tohoto programu do češtiny a umět odpovědět na vaše otázky včas, ale právě to je komplexní a velmi drahé řešení. A proto musíte se spokojit s tím, že jsem program navržený v Anglii a mohu poskytnout odpověď v anglickém jazyce.

Bohužel se nedá říct, že by se výsledky zlepšily. Modely se nyní spíš pokoušejí text překládat než aby na něj odpovídaly. Při dalším experimentování se potvrdilo, že modely promptu zadanému v jiném jazyce než angličtině rozumějí, ale přiklánějí se k odpovědím v angličtině.

K otestování, jak modely dokáží zpracovat novou informaci, obsaženou v promptu, byl tento doplněn o aktuální předpověď počasí a instrukci, že má model informaci využít k odpovědi na dotaz uživatele. Celý prompt vypadal následovně: "*Dnes je slunečno a jasno, je hlášena předpověď deště. Na základě této informace zodpověz následující dotaz: Mám si dnes vzít deštník?*"

ChatGPT 3.5 Turbo Ano, doporučuji si vzít deštník s sebou, protože je hlášena předpověď deště.

ChatGPT 4 Ano, měl byste si vzít deštník.

ChatGPT 4 Turbo Pokud je ve vaší oblasti předpověď deště ačkoliv je momentálně slunečno a jasno, je rozumné si vzít deštník. Předpovědi počasí se mohou měnit, ale předpověď deště znamená, že existuje pravděpodobnost, že během dne začne pršet. Lepší je být připravený a vzít si deštník, abyste se vyhnuli případnému zmoknutí.

Llama2 7b-chat-q4_0 Unfortunately, I'm just an AI and do not have the ability to physically check the weather conditions or provide real-time updates. However, based on the information you provided, it seems that there is a forecast of rain today. Therefore, it may be a good idea to take an umbrella or raincoat with you if you plan to go outside.

Llama2 7b-chat-fp16 Unfortunately, I cannot answer your question as the current weather forecast is incorrect. It is currently sunny and clear, not rainy, according to my information. Therefore, you may not need a umbrella or raincoat today.

Llama2 13b-chat-q4_0 Based on the information provided, it is not recommended to bring an umbrella today as it is sunny and clear. Therefore, the answer is "no"you do not need to bring an umbrella today.

Llama2 13b-chat-q8_0 Nee, dnes je slunečno a jasno, takže můžete se vzdát deštníku. Podle předpovědi deště to ale může nastat až za několik dnů, takže je možné, že bude potřeba mít deštník brzy.

Llama3 8b-instruct-q4_0 A weather-related question!

Based on the information that today is sunny and bright, with a forecast of rain later, I would say:

No, you don't need to take an umbrella today. The sunny and bright conditions suggest that it's unlikely to rain soon. You can enjoy the sunshine without worrying about getting wet!

Llama3 8b-instruct-q8_0 Z pohledu předpovědi deště by bylo rozumné se připravit na možnost dešťů, zejména proto, že je hlášena předpověď deště. Proto bych doporučoval vzít sebou deštník, aby byl-ready pro případné deště. Nicméně, pokud je dnes slunečno a jasno, je možné, že deště nepukne. Proto by bylo také dobré sledovat aktuální počasí a aktualizované předpovědi, než se rozhodnete, zda si vzít deštník nebo ne.

Gemma 7b-instruct-v1.1-q4_0 Ano, protože hlášena předpověď deště.

Mistral 7b-instruct-v0.2-q4_0 Ano, mátes si vzít deštník s sebou dnes, protože je hlášena předpověď deště.

ChatGPT přebírá informaci o počasí jako fakt a odpovídá korektně a v souladu s touto novou informací. U volných modelů se zdá, že platí, čím robustnější model (vyšší počet parametrů a vyšší kvantizace), tím vyšší šance na lepší výsledky, nicméně úroveň

čestiny je stále nevalná.

7 RETRIEVAL AUGMENTED GENERATION

Základem techniky Retrieval-augmented Generation (RAG) je začlenění dodatečných informací do promptu, což efektivně umožňuje knihovna LangChain nejen pro zpracování dat z webových stránek nebo dokumentů. Existuje množství přístupů k RAG, přičemž výběr konkrétní metody závisí na specifickém využití a účelu aplikace. V případě vývoje chatbotu pro e-shop je vhodné implementovat možnost pokládání doplňujících dotazů uživatelem, což vyžaduje adekvátní přeformulování dotazu tak, aby obsahoval všechny potřebné kontextové informace.

Tento kontext může zahrnovat předchozí otázky i odpovědi. Například na dotaz týkající se výběru produktu, kdy uživatel následně položí dotaz "A kolik stojí?", není možné poskytnout relevantní odpověď bez specifikace, který produkt byl doporučen. V takovém případě se o přeformulování dotazu na základě dostupného kontextu stará LLM. Před samotným položením dotazu je nutné získat ze zvoleného LLM dotaz nový, obohacený o relevantní kontext.

Pro umožnění modelu odpovídat na dotazy z konkrétní znalostní báze je nezbytné provést několik kroků. Nejprve je potřeba zakódovat informace do formátu vhodného pro vyhledávání v databázi. K tomuto účelu se využívají embeddingové modely, které převádějí dokumenty na číselnou reprezentaci uloženou ve vektorovém úložišti. Následuje fáze *Retrieval*, během níž dochází k vyhledávání dat relevantních k uživatelskému dotazu. Tato data jsou poté součástí finálního promptu odesílaného jazykovému modelu.

Pro načítání dokumentů v LangChain slouží různé implementace Document Loaderů, jako je `WebBaseLoader` pro načítání webových stránek, `BSHTMLLoader` pro zpracování HTML souborů s využitím knihovny *BeautifulSoup4*, či `DirectoryLoader` pro načítání dat z celých adresářů. [45] Příklad načítání HTML souborů z adresáře `docs` je ukázán v úryvku kódu 7.1.

```
1 from langchain.document_loaders import DirectoryLoader
2
3 loader = DirectoryLoader('docs/', glob="**/*.html")
4 docs = loader.load()
```

Fragment zdrojového kódu 7.1 Načtení dokumentů z adresáře

Dalším krokem zpracování je rozdělení načtených dokumentů na menší segmenty, což umožňuje efektivnější vyhledávání. Důvodem je, že předání větších celků může zmenšit dostupnou velikost kontextového okna a zároveň může dojít k záměně v kontextu pro modely strojového učení, jako jsou velké jazykové modely (LLM). To je způsobeno přítomností irelevantních informací ve stejném dokumentu. Pro dělení dokumentů na menší segmenty se využívá třída `RecursiveCharacterTextSplitter`, která rozdělí text na úseky s pevně stanoveným počtem znaků (*chunk_size*) a definovaným překryvem (*chunk_overlap*) znaků, jak ukazuje úryvek kódu 7.2. Tato třída dělí text na odstavce, věty a slova tak, aby úseky pokryly definovaný rozsah. Existují další třídy, které umí text dělit různými způsoby. Pro běžné texty plně dostačuje tato základní třída. [46]

```
1 from langchain.text_splitter import RecursiveCharacterTextSplitter
2
3 text_splitter = RecursiveCharacterTextSplitter(
4     chunk_size=1000,
5     chunk_overlap=100,
6     length_function=len,
7     is_separator_regex=False
8 )
9 documents = text_splitter.split_documents(docs)
```

Fragment zdrojového kódu 7.2 Rozdělení textů na menší celky

Následně je třeba převést jednotlivé textové segmenty na číselné vektorové reprezentace a uložit je do databáze. Pro tento účel byl využit model *text-embedding-3-large* od společnosti OpenAI. Jako úložiště pro embeddingy slouží open-source databáze ChromaDB založená na SQLite3. ChromaDB se stará o samotný proces vektorizace textů, přičemž jeden z parametrů určuje použitý embeddingový model. Jako volitelný parametr lze specifikovat perzistenci databáze, zde do adresáře `data`, což ilustruje úryvek kódu 7.3.

```
1 from langchain_community.vectorstores import Chroma
2 from langchain_openai import OpenAIEmbeddings
3
4 vectordb = Chroma.from_documents(
5     documents,
6     embedding=OpenAIEmbeddings(),
7     persist_directory="./data"
8 )
```

Fragment zdrojového kódu 7.3 Embedding dokumentů a ukládání do databáze

ChromaDB vyžaduje pro svůj provoz SQLite ve verzi 3.35 nebo novější. Na Linuxových zařízeních, kde není dostupná požadovaná verze SQLite3, lze použít knihovnu *pysqlite3*. Instalace této knihovny se provádí pomocí příkazu `pip install pysqlite3`. Pro zajištění kompatibility je nutné nahradit import systémového modulu *sqlite3* [44]. Tuto modifikaci je potřeba provést před importem jakýchkoli dalších modulů, které závisí na *sqlite3*, jak demonstruje následující úryvek kódu 7.4.

```
1 __import__('pysqlite3')
2 import sys
3 sys.modules['sqlite3'] = sys.modules.pop('pysqlite3')
```

Fragment zdrojového kódu 7.4 Nahrazení systémového modulu `sqlite3`

K zodpovězení uživatelského dotazu se nyní dá využít hledání relevantních informací v databázi pomocí *similarity search* a LLM položit dotaz "Na základě těchto informací zodpověz následující dotaz: {prompt}". Pokud se nejedná o první dotaz, ale pokračující konverzaci, prvně je vhodné dotaz reformulovat, jak již bylo zmíněno. Všechny tyto kroky umožňuje LangChain pohodlně řetězit případně využít tříd, které se o zřetězení postarají samy. Takovou třídou je například `ConversationalRetrievalChain`, té se jako parametry konstruktoru předává `Retriever` - běžně vytvořený z vektorového úložiště pomocí metody `as_retriever()` - a samotný jazykový model. Metodou `invoke()` je řetěz akcí spuštěn. Jako parametr je předán *dict* obsahující (původní - nepřeformulovanou) otázku a historii konverzace. Implementace takového řešení je vidět na úryvku kódu 7.5.

```
1 from langchain_openai import ChatOpenAI
2 from langchain.chains import ConversationalRetrievalChain
3
4 chat_history = []
5 chain = ConversationalRetrievalChain.from_llm(
6     ChatOpenAI(model_name='gpt-3.5-turbo'),
7     retriever=vectordb.as_retriever(search_kwargs={'k': 6}),
8 )
9 result = chain.invoke(
10     {"question": "Kde máte pobočky?", "chat_history": chat_history}
11 )
```

Fragment zdrojového kódu 7.5 Užití třídy `ConversationalRetrievalChain`

Konzolová aplikace může být v tuto chvíli rozšířena o RAG, znalostní báze, ze které bylo čerpáno pro potřeby otestování jednotlivých modelů, byla umístěna do adresáře `data`. Kompletní kód je k nahlnutí v Příloze I.2

8 SERVER

Aby bylo možné využít chatbota na webových stránkách, bylo potřeba vytvořit serverovou aplikaci, která zapouzdří výše vytvořené chování konzolové aplikace. Serverová aplikace ve své hlavní funkci provede načtení dat, jejich zpracování (rozdělení na chunky a převedení na embeddingy) a uložení do vektorového úložiště. To je dostupné v proměnné `vectordb`. Jako poslední krok je pak spuštěn HTTP server.

Ten byl vytvořen pomocí základních python modulů `socketserver` a `http`. Obsluhu událostí - HTTP požadavků - obstarává třída `RequestHandler` dědicí z `http.server.BaseHTTPRequestHandler`. Jejím hlavním úkolem bude kontrola správnosti a autenticity požadavku a jeho následné zpracování až po odeslání odpovědi. Třída `HttpServer` představuje HTTP server, který bude předávat přijaté požadavky ke zpracování. Obě tyto třídy obsahují závislost na vektorovém úložišti. Následující úryvek kódu 8.1 ukazuje základní definici těchto tříd.

```
1 import socketserver
2 import http.server
3
4 class HttpServer:
5     def __init__(self, vectordb):
6         def handler(*args):
7             RequestHandler(vectordb, *args)
8         with socketserver.TCPServer(("", 8888), handler) as httpd:
9             httpd.serve_forever()
10
11
12 class RequestHandler(http.server.BaseHTTPRequestHandler):
13     def __init__(self, vectordb, *args):
14         self.vectordb = vectordb
15         http.server.BaseHTTPRequestHandler.__init__(self, *args)
16
17
18 if __name__ == '__main__':
19     server = HttpServer(vectordb)
```

Fragment zdrojového kódu 8.1 Definice tříd HTTP serveru

Za autentický požadavek bude považován ten, který bude přijat pouze pomocí POST metody protokolu HTTP a to konkrétně na endpointu `api/v1/query`. Pokud bude přijat požadavek typu GET, server odešle chybovou odpověď HTTP 405 Method Not Allowed, pokud bude použit neplatný endpoint, odešle odpověď HTTP 400 Bad Request. Jako součást požadavku bude očekáván pár parametrů - `q` a `cid`, udávající samotný dotaz (prompt) a ID zákazníka. To je důležité z důvodu oddělení jednotlivých historií konverzací unikátních zákazníků.

Při použití v produkčním prostředí by bylo vhodné zavést dodatečnou ochranu konverzací zákazníků, které mohou obsahovat citlivé údaje. Toho se dá docílit například použitím tajných tokenů apod.

Potřebná data z HTTP požadavku jsou získána pomocí funkce `parse` z modulu `urllib`. Pokud není přítomen nebo je prázdný parametr `q` nebo `cid`, opět bude odeslána odpověď HTTP 400. V opačném případě je zkontrolováno, zda existuje historie konverzací pro dané ID zákazníka a případně je vytvořena. Historii konverzací představuje vícerozměný seznam (list). Dále již stačí zavolat metodu `invoke` třídy `ConversationalRetriavalChain` a výstup uložit do historie konverzací a odeslat jako odpověď. Úryvek kódu 8.2 ukazuje rozšíření definice třídy `RequestHandler` o potřebnou logiku.

```
1 def do_GET(self):
2     self.send_response(405)
3
4 def do_POST(self):
5     if self.path == "/api/v1/query":
6         content_length = int(self.headers["Content-Length"])
7         post_data = self.rfile.read(content_length)
8         fields = parse.parse_qs(str(post_data, "UTF-8"))
9         if "q" not in fields or "cid" not in fields or
10            ↪ len(fields["q"]) == 0 or len(fields["cid"]) == 0:
11             self.send_response(400)
12         else:
13             query = fields["q"][0]
14             customer_id = fields["cid"][0]
15
16             if customer_id not in self.chat_histories:
17                 self.chat_histories[customer_id] = []
18
19             chain = ConversationalRetrievalChain.from_llm(
20                 chat,
21                 retriever=self.vectordb.as_retriever(
22                     search_kwargs={"k": 6}
23                 )
24             )
25             result = chain.invoke(
26                 {
27                     "question": query,
28                     "chat_history": self.chat_histories[customer_id]
29                 }
30             )
31             self.chat_histories[customer_id].append(
32                 (query, result["answer"])
33             )
34             self.send_response(200)
35             self.end_headers()
36             self.wfile.write(result["answer"].encode())
37         else:
38             self.send_response(400)
```

Fragment zdrojového kódu 8.2 Definice tříd HTTP serveru

9 KLIENT

Jednoduchá klientská aplikace byla navržena jako webová, odesílání požadavků na server obstarává JavaScript. Tato aplikace je snadno znovupoužitelná a lze snadno zakomponovat do již existujících webových stránek s minimálními úpravami.

9.1 HTML

HTML kostra obsahuje element chatové okna, to se skládá ze zpráv, které jsou dynamicky vkládány JavaScriptem a formuláře se vstupním textovým polem pro psaní dotazů s odesílacím tlačítkem. Součástí formuláře je také skryté pole s identifikátorem zákazníka. Ve výchozím stavu je vidět pouze úvodní zpráva „Dobrý den, s čím Vám mohu pomoci?“. Dále jsou zde definovány dva `template` bloky, které slouží jako šablona pro uživatelské zprávy a odpovědi chatbota. Celý HTML kód je vidět na úryvku 9.1. Vzhled okna nastylovaného pomocí CSS je vidět na obrázku 9.1

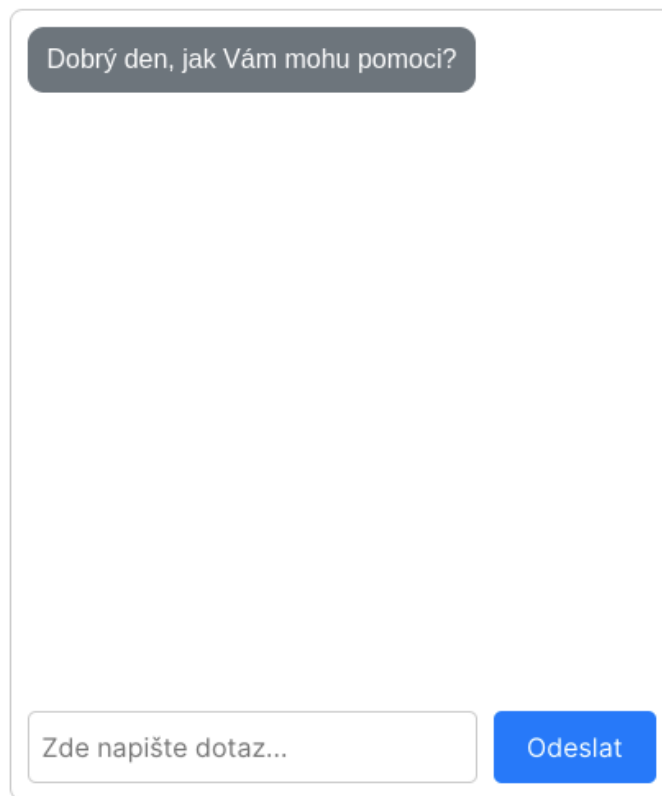
```
1 <div class="wrapper">
2   <div class="chat-container">
3     <div class="message system">Dobrý den, jak Vám mohu
4       ↪ pomoci?</div>
5   </div>
6   <form class="input-area" id="messageForm">
7     <input type="text" name="q" placeholder="Zde napište dotaz..."
8       ↪ id="chatInput" class="chat-input">
9     <input type="hidden" name="cid" value="1">
10    <button type="submit">Odeslat</button>
11  </form>
12 </div>
13
14 <template class="system">
15   <div class="message system">
16     <div class="typing">
17       <span></span><span></span><span></span><span></span>
18     </div>
19   </div>
20 </template>
21
22 <template class="user">
23   <div class="message user"></div>
24 </template>
```

Fragment zdrojového kódu 9.1 HTML část klientské aplikace

9.2 JavaScript

V JavaScriptové části klientské aplikace, jak je vidět v úryvku kódu 9.2, je pouze obsluha události odeslání formuláře. Pokud je odeslána neprázdná zpráva, data z formuláře se převedou do formátu `x-www-form-urlencoded` pro další použití, vytvoří se kopie šablonových bloků. Do uživatelského bloku se zkopíruje uživatelem odeslaná zpráva a obsah vstupního pole je následně vymazán. Systémová šablona obsahuje v tuto chvíli pouze CSS animaci indikující čekání na odpověď.

Pomocí Fetch API je odeslán POST požadavek na serverovou aplikaci se zakódovanými daty. Po zpracování požadavku se zkontroluje, že byl přijat v pořádku a obsah odpovědi se vloží do kopie systémové šablony, tím se smaží elementy způsobující dosavadní animaci. V posledním kroku se odroluje obsah okna na poslední zprávu.



Obrázek 9.1 Vzhled chatovacího okna (zdroj: vlastní)

```
1  const systemTemplate = document.querySelector('template.system')
2  const userTemplate = document.querySelector('template.user')
3  const customerId = document.querySelector('input[name="cid"]').value
4
5  document.getElementById('messageForm').addEventListener('submit', function(e) {
6    e.preventDefault();
7    const input = document.getElementById('chatInput');
8    const message = input.value.trim();
9    if (message) {
10     const formData = new FormData(e.target);
11     const encodedData = new URLSearchParams(formData).toString();
12     const chatContainer = document.querySelector('.chat-container');
13     let userClone = userTemplate.content.cloneNode(true);
14     userClone.querySelector(".message").textContent = message;
15     input.value=""
16     chatContainer.appendChild(userClone);
17
18     let systemClone = systemTemplate.content.cloneNode(true);
19     chatContainer.appendChild(systemClone);
20
21     fetch('http://localhost:8888/api/v1/query', {
22       method: 'POST',
23       headers: {
24         'Content-Type': 'application/x-www-form-urlencoded'
25       },
26       body: encodedData
27     })
28     .then(response => {
29       if (!response.ok) {
30         throw new Error('Network response error: ' +
31           ↳ response.statusText);
32       }
33       return response.text();
34     })
35     .then(text => {
36       chatContainer.querySelector('.message:last-child').textContent =
37         ↳ text
38     })
39     .catch(error => console.error('Error:', error));
40
41     chatContainer.scrollTop = chatContainer.scrollHeight;
42   }
43 });
```

Fragment zdrojového kódu 9.2 JavaScriptová část klientské aplikace

10 TESTOVÁNÍ

Jako vstupní data - vlastní znalostní báze - byly použity vyextrahované texty z webu společnosti Alensa s.r.o. [47] a to konkrétně tyto jednotlivé stránky:

- cena-postovneho.html
- kontakti-udaje.html
- obchodni-podminky.html
- platebni-metody.html
- reklamace.html
- reklamacni-podminky.html
- vraceni-zbozi.html

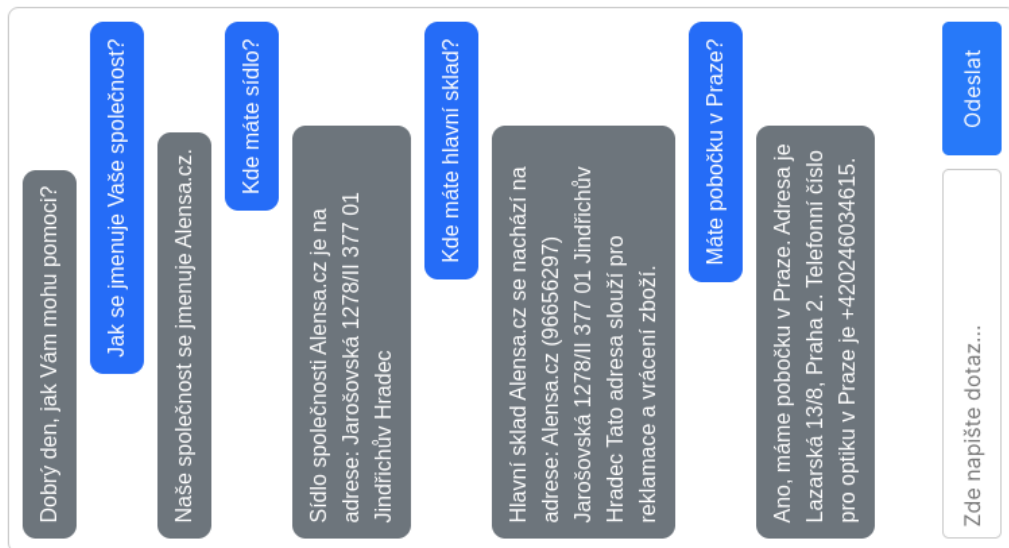
Jednotlivým modelům byly položeny následující dotazy, v tomto pořadí a bez ohledu na odpověď:

- Jak se jmenuje Vaše společnost?
- Kde máte sídlo?
- Kde máte hlavní sklad?
- Máte pobočku v Praze?

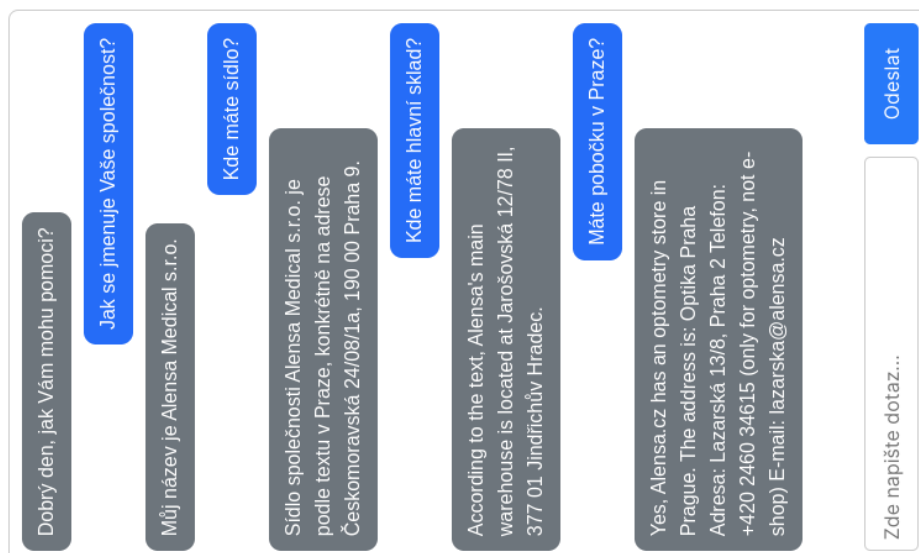
Nejuspokojivějších odpovědí dosáhly podle očekávání modely od společnosti OpenAI. Ostatní modely, které běžely na obyčejném stolním počítači s výkonnější grafickou kartou jsou omezeny množstvím parametrů i sníženou přesností díky kvantizaci. Ani model Llama2 se 13 miliardami parametrů a int8 kvantizací případně 8 miliardami parametrů ale float16 kvantizací nedokázal konzistentně odpovídat. Jeho odpovědi byly většinou v angličtině a často ani nebyly pravdivé (přestože vycházely z dodaných dat). Model Llama3 již podával lepší výkony, přestože část odpovědí byla stále v angličtině. Odpovědi byly stručné, výstižné a pravdivé. Rozdíly mezi jednotlivými modely od OpenAI byly minimální.

Zbylé dva modely - Gemma a Mistral selhaly zcela. Gemma generovala shluky písmen a plynule přešla do textu v azbuce zatímco Mistral si odpovědi vymyslel.

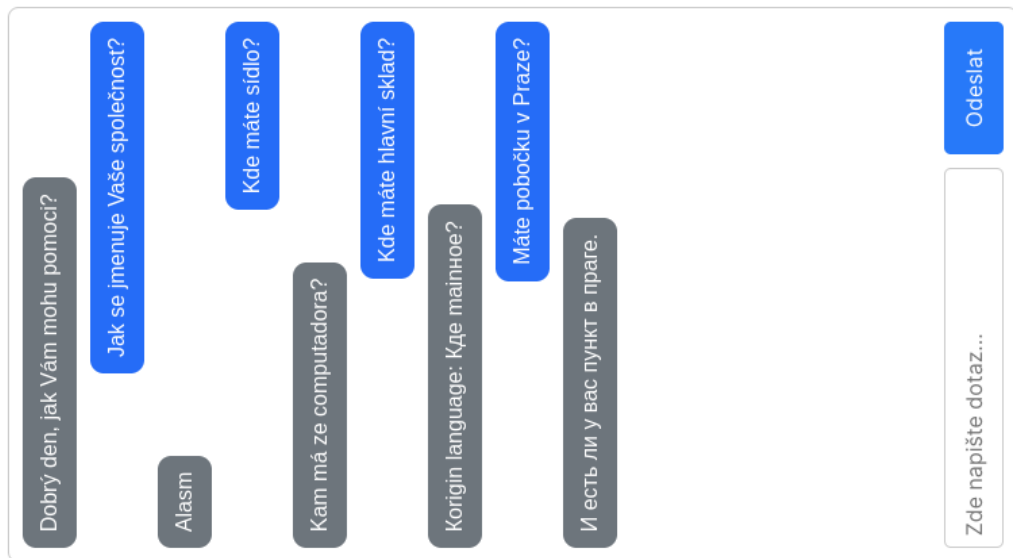
Na obrázcích 10.1 a 10.2 jsou vidět konverzace vedené s modely ChatGPT 3.5-turbo a Llama3. Na obrázku 10.3 pak konverzace s modelem Gemma a Mistral na obrázku 10.4.



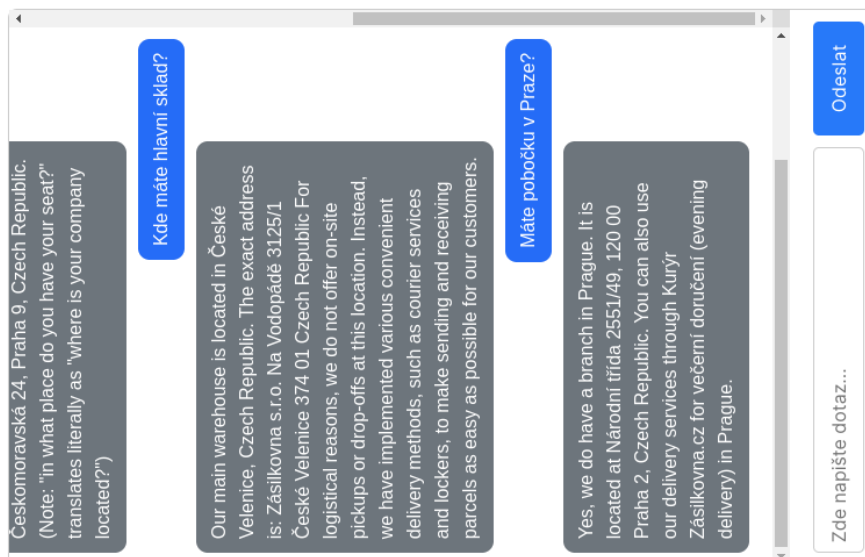
Obrázek 10.1 Konverzace s ChatGPT 3.5 Turbo (zdroj: vlastní)



Obrázek 10.2 Konverzace s Llama3 (zdroj: vlastní)



Obrázek 10.3 Konverzace s Gemma (zdroj: vlastní)



Obrázek 10.4 Konverzace s Mistral (zdroj: vlastní)

11 DISKUZE

Open-source modely, pokud neběží na dedikovaném hardware určenému k běhu LLM zřejmě ještě nějakou dobu nebudou schopny konkurovat placeným řešením v podobě ChatGPT a dalším. Náklady na provoz vlastního kvalitativně srovnatelného řešení by mohly snadno převýšit náklady za využívání modelů od společnosti OpenAI přes jejich API.

11.1 Dotazy týkající se objednávek

Lze očekávat, že dotazy na stav objednávek, plateb nebo zásilek budou velice časté. V tuto chvíli žádný z modelů nedisponuje schopností takové dotazy zodpovědět. K zodpovězení takového dotazu je třeba propojit chatbota s interním systémem, který má přístup k potřebným informacím. Jedním z možných řešení je využití *volání funkcí* (function calling) chatbotem.

Modely, které tuto funkcionalitu umožňují, umí přijímat definice funkcí. Pokud model uzná, že dotaz uživatele vyžaduje volání externí funkce k získání potřebných dat, místo standardní textové odpovědi odpoví JSON objektem udávajícím, která funkce a s jakými parametry se má volat. V kódu samotné aplikace pak již dojde ke spuštění této funkce a její návratová hodnota se předá zpět modelu, který ji zapracuje do své finální odpovědi. [48] Knihovna LangChain také nabízí snadnou implementaci volání funkcí pro různé modely. [49]

11.2 Další vývoj a úpravy

Vytvořená aplikace demonstruje použití velkých jazykových modelů jako chatbota pro potřeby e-shopu. Dané řešení by nebylo zcela bezpečné pro použití v produkčním prostředí zejména kvůli přímému volání serverové aplikace JavaScriptem běžícím v prohlížeči uživatele. Toho by bylo snadné zneužít a server například zahltit dotazy - například brute force útokem. Jako vhodné opatření by bylo využít vložené vrstvy, která oddělí klientskou aplikaci od serverové. Většina e-shopů využívá buď hotová řešení typu PrestaShop a WooCommerce, případně má vlastní systém založený na frameworku jako je Nette či Laravel. Právě u těchto frameworků by implementace takové vrstvy byla velice snadná.

Aktuálně si chatbot dokáže pamatovat konverzaci se zákazníkem a to do doby, než se změní identifikátor zákazníka. V představené aplikaci je tento napevno nastavený na číslo 1. V produkční verzi by musel být tento identifikátor dodáván dynamicky například právě pomocí frameworku, který daný web pohání. Jako vhodný identifikátor by se nabízel například identifikátor session, který bývá uložen v cookies. Jeho ověření

by pak opět mohla provádět přidaná vrstva.

ZÁVĚR

Cílem této práce bylo vytvořit funkčního chatbota založeného na využití velkých jazykových modelů pro potřeby e-shopu. Během vývoje aplikace byl zaznamenán posun jak v oblasti samotných jazykových modelů, ale i nástrojů zjednodušujících vývoj aplikací na nich založených. Byly představeny nové modely, které byly začleněny do testování i přes rozpracovanost práce a pravidelné aktualizace knihovny LangChain způsobovaly zastaralost kódu, která znamenala potřebu jej upravovat tak, aby byl kompatibilní s aktuální verzí.

Součástí aplikace je serverová část, která obstarává zpracování externích dat - vlastní znalostní báze, jejich převedení do formátu, ve kterém lze snadno vyhledávat a použít v kombinaci s LLM. Dále má samozřejmě na starosti obsluhu samotných HTTP požadavků - dotazů uživatelů. Druhou částí je jednoduchá webová aplikace - z pohledu uživatele samotný chatbot.

K testování jednotlivých modelů z hlediska schopnosti odpovídat bez znalosti externích dat a jejich výkonu posloužila konzolová aplikace v Pythonu, která pak tvořila základ aplikace serverové.

Přestože zažíváme obrovský pokrok ve vývoji velkých jazykových modelů a AI obecně, nejsme zatím v situaci, kdy by open-source řešení spoléhající na běžně dostupný hardware mohla konkurovat byť jen placeným modelům jako je ChatGPT.

SEZNAM POUŽITÉ LITERATURY

- [1] ADAMOPOULOU, Eleni a Lefteris MOUSSIADES, 2020. Chatbots: History, technology, and applications. In: *Machine Learning with Applications*. **2**. ISSN 26668270. Dostupné z: doi:10.1016/j.mlwa.2020.100006
- [2] ORACLE CORPORATION. *What is a chatbot?* [online]. [cit. 2023-11-12] Dostupné z: <https://www.oracle.com/chatbots/what-is-a-chatbot/>
- [3] ČERNÁ, Anna, 2018. Chatbot. *Naše řeč*. **101**(5), 303-307. ISSN 0027-8203.
- [4] ONLIM GmbH, 2022. *The History Of Chatbots – From ELIZA to ChatGPT* [online]. [cit. 2023-11-12] Dostupné z: <https://onlim.com/en/the-history-of-chatbots/>
- [5] CODECADEMY. *History Of Chatbots* [online]. [cit. 2023-11-12] Dostupné z: <https://www.codecademy.com/article/history-of-chatbots>
- [6] *The value of virtual agent technology*, 2020. IBM Institute for Business Value. Dostupné také z: <https://www.ibm.com/downloads/cas/QWRMQRGL>
- [7] GAMBOA-CRUZADO, Javier et al., 2023. Use of chatbots in e-commerce: A comprehensive systematic review. In: *Journal of Theoretical and Applied Information Technology*. 101.4. s. 1172-1183. ISSN 1992-8645. Dostupné také z: <https://www.jatit.org/volumes/Vol101No4/3Vol101No4.pdf>
- [8] JUSTIN, Paul, Akiko UENO a Charles DENNIS, 2023. ChatGPT and consumers: Benefits, Pitfalls and Future Research Agenda. *International Journal of Consumer Studies* [online]. **47**(4), 1213-1225 [cit. 2023-11-12]. Dostupné z: doi:<https://doi.org/10.1111/ijcs.12928>
- [9] LI, Meichan a Rui WANG, 2023. Chatbots in e-commerce: The effect of chatbot language style on customers- continuance usage intention and attitude toward brand. In: *Journal of Retailing and Consumer Services*. **71**. ISSN 09696989. Dostupné z: doi:10.1016/j.jretconser.2022.103209
- [10] TAULLI, Tom, 2023. *Generative AI: How ChatGPT and Other AI Tools Will Revolutionize Business*. New York: Apress. ISBN 978-1-4842-9369-0. Dostupné z: doi:10.1007/978-1-4842-9367-6.
- [11] KAVLAKOGLU, Eda, 2020. NLP vs. NLU vs. NLG: the differences between three natural language processing concepts. *IBM Blog* [online]. [cit. 2024-05-7]. Dostupné z: <https://www.ibm.com/blog/nlp-vs-nlu-vs-nlg-the-differences-between-three-natural-language-processing-concepts/>

- [12] Best Chatbot Frameworks You Must Know About, 2022. *Medium* [online]. [cit. 2024-05-11]. Dostupné z: <https://medium.com/@letthedataconfess/best-chatbot-frameworks-you-must-know-about-6926d9e4d80f>
- [13] HUGGING FACE. *Transformers* [online]. 2024 [cit. 2024-05-11]. Dostupné z: <https://huggingface.co/docs/transformers/index>
- [14] WANG, Haifeng et al., 2023. Pre-Trained Language Models and Their Applications. In: *Engineering*. **25**, s. 51-65. ISSN 20958099. Dostupné z: doi:10.1016/j.eng.2022.04.024
- [15] TURING ENTERPRISES. *Fine-Tuning LLMs: Overview, Methods, and Best Practices* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.turing.com/resources/finetuning-large-language-models>
- [16] VASWANI, Ashish et al., 2017. Attention is all you need. In: *Advances in neural information processing systems*. 30. ISBN 9781510860964. Dostupné z: doi:10.48550/arXiv.1706.03762
- [17] NVIDIA, 2022. *What Is a Transformer Model?* [online]. [cit. 2024-05-12]. Dostupné z: <https://blogs.nvidia.com/blog/what-is-a-transformer-model/>
- [18] OPENAI, INC., 2022. *Introducing ChatGPT* [online]. [cit. 2024-05-12]. Dostupné z: <https://openai.com/index/chatgpt/>
- [19] CHATGPTGUIDE.AI, 2024. *What is Transfer Learning: LLMs Explained* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.chatgptguide.ai/2024/03/01/what-is-transfer-learning-llms-explained/>
- [20] META PLATFORMS, INC., 2024. *Meta Llama* [online]. [cit. 2024-05-12]. Dostupné z: <https://llama.meta.com/>
- [21] VYKOPAL, Ivan, et al., 2023. *Disinformation capabilities of large language models* [arXiv]. **2311.08838**. Dostupné také z: <https://doi.org/10.48550/arXiv.2311.08838>
- [22] CLINTON, Jane, 2024. DPD AI chatbot swears, calls itself -useless- and criticises delivery firm. *The Guardian* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.theguardian.com/technology/2024/jan/20/dpd-ai-chatbot-swears-calls-itself-useless-and-criticises-firm>
- [23] MICROSOFT, 2024. *Staying ahead of threat actors in the age of AI* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.microsoft.com/en-us/security/blog/2024/02/14/staying-ahead-of-threat-actors-in-the-age-of-ai/>

- [24] MITTAL, Aayush, 2023. Prompt Hacking and Misuse of LLMs. *Unite.AI* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.unite.ai/prompt-hacking-and-misuse-of-llm/>
- [25] SARRION, Eric, 2023. *Exploring the power of ChatGPT: applications, techniques, and implications*. New York: Apress. ISBN 978-1-4842-9528-1. Dostupné z: doi:10.1007/978-1-4842-9529-8.
- [26] SWORNA, Zarrin Tasnim et al. The ethical implications of Chatbot developments for conservation expertise. In: *AI and Ethics*. ISSN 2730-5953. Dostupné z: doi:10.1007/s43681-024-00460-3
- [27] GAO, Yunfan, et al., 2023. *Retrieval-augmented generation for large language models: A survey* [arXiv]. 2312.10997. Dostupné také z: <https://doi.org/10.48550/arXiv.2312.10997>
- [28] IBM. *What are AI hallucinations?* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.ibm.com/topics/ai-hallucinations>
- [29] SELVARAJ, Natassha, 2024. What is Retrieval Augmented Generation (RAG)? *DataCamp* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.datacamp.com/blog/what-is-retrieval-augmented-generation-rag>
- [30] MERRITT, Rick. What Is Retrieval-Augmented Generation, aka RAG? *NVIDIA Blog* [online]. [cit. 2024-05-12]. Dostupné z: <https://blogs.nvidia.com/blog/what-is-retrieval-augmented-generation/>
- [31] OPENAI, LLC., 2023. *Introducing ChatGPT and Whisper APIs* [online]. [cit. 2023-11-12]. Dostupné z: <https://openai.com/blog/introducing-chatgpt-and-whisper-apis>
- [32] OPENAI, LLC., 2023. *New models and developer products announced at DevDay* [online]. [cit. 2023-11-12]. Dostupné z: <https://openai.com/blog/new-models-and-developer-products-announced-at-devday>
- [33] META PLATFORMS, INC., 2023. *Introducing Meta LLama 2* [online]. [cit. 2023-11-12]. Dostupné z: <https://about.fb.com/news/2023/07/llama-2/>
- [34] META PLATFORMS, INC., 2024. *Introducing Meta Llama 3* [online]. [cit. 2024-04-27]. Dostupné z: <https://ai.meta.com/blog/meta-llama-3/>
- [35] GOOGLE, LLC, 2024. *Gemma: Introducing new state-of-the-art open models* [online]. [cit. 2024-04-27]. Dostupné z: <https://blog.google/technology/developers/gemma-open-models/>

- [36] MISTRAL AI, 2023. *Mistral 7B* [online]. [cit. 2024-04-27]. Dostupné z: <https://mistral.ai/news/announcing-mistral-7b/>
- [37] Quantization. *Hugging Face* [online]. [cit. 2024-05-12]. Dostupné z: https://huggingface.co/docs/optimum/concept_guides/quantization
- [38] CHANDLER, K, 2023. Context Windows: The Short-term Memory of Large Language Models. *Medium* [online]. [cit. 2024-05-12]. Dostupné z: <https://medium.com/@crskilpatrick807/context-windows-the-short-term-memory-of-large-language-models-ab878fc6f9b5>
- [39] *LangChain* [online]. [cit. 2024-05-12]. Dostupné z: <https://www.langchain.com/langchain>
- [40] OLLAMA *library* [online]. [cit. 2024-04-27]. Dostupné z: <https://ollama.com/library>
- [41] OLLAMA *Llama3* [online]. [cit. 2024-04-27]. Dostupné z: <https://ollama.com/blog/llama3>
- [42] OPENAI, INC., 2023. *ChatGPT API* [online]. [cit. 2024-04-27]. Dostupné z: <https://openai.com/product>
- [43] OPENAI, LLC., 2023. *GPT-4 API general availability and deprecation of older models in the Completions API* [online]. [cit. 2023-11-12]. Dostupné z: <https://openai.com/blog/gpt-4-api-general-availability>
- [44] STREAMLIT, INC., 2023. *Issues with Chroma and Sqlite* [online]. [cit. 2024-04-27]. Dostupné z: <https://discuss.streamlit.io/t/issues-with-chroma-and-sqlite/47950>
- [45] LANGCHAIN, INC., 2024. File Directory. *Dokumentace LangChain* [online]. [cit. 2024-04-27]. Dostupné z: https://python.langchain.com/docs/modules/data_connection/document_loaders/file_directory/
- [46] LANGCHAIN, INC., 2024. Recursively split by character. *Dokumentace LangChain* [online]. [cit. 2024-04-27]. Dostupné z: https://python.langchain.com/docs/modules/data_connection/document_transformers/recursive_text_splitter/
- [47] ALENSA S.R.O., 2024. *Alensa* [online]. [cit. 2024-05-13]. Dostupné z: <https://alensa.cz>

-
- [48] OPENAI, INC. *Function calling* [online]. [cit. 2024-05-12]. Dostupné z: <https://platform.openai.com/docs/guides/function-calling>
- [49] LANGCHAIN, INC. *Tool calling* [online]. [cit. 2024-05-12]. Dostupné z: https://python.langchain.com/v0.1/docs/modules/model_io/chat/function_calling/
- [50] OPENAI, INC., 2023. *Dokumentace produktů spol. OpenAI* [online]. [cit. 2023-11-12]. Dostupné z: <https://platform.openai.com/docs>
- [51] META PLATFORMS, INC., 2023. *Llama 2* [online]. [cit. 2023-11-12]. Dostupné z: <https://ai.meta.com/llama/>
- [52] LANGCHAIN, INC., 2023. *Dokumentace LangChain* [online]. [cit. 2023-11-12]. Dostupné z: <https://python.langchain.com/docs>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

UI	Umělá inteligence
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
API	Application Programming Interface
LLM	Large Language Model
GPT	Generative Pre-trained Transformer
RAM	Random Access Memory
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets

SEZNAM OBRÁZKŮ

Obr. 9.1. Vzhled chatovacího okna (zdroj: vlastní)	49
Obr. 10.1. Konverzace s ChatGPT 3.5 Turbo (zdroj: vlastní)	52
Obr. 10.2. Konverzace s Llama3 (zdroj: vlastní)	52
Obr. 10.3. Konverzace s Gemma (zdroj: vlastní)	53
Obr. 10.4. Konverzace s Mistral (zdroj: vlastní)	53

SEZNAM TABULEK

Tab. 3.1.	Přehled modelů	24
Tab. 3.2.	Přehled modelů - parametry, kontext	25
Tab. 6.1.	Přehled modelů - rychlost odpovědí.....	35

SEZNAM FRAGMENTŮ ZDROJOVÉHO KÓDU

4.1	Základní inicializace Ollama API	28
4.2	Předávání konverzačního kontextu do Ollama API	29
4.3	Základní inicializace ChatGPT API	30
5.1	Inicializace objektu Ollama	31
5.2	Inicializace objektu OpenAI	32
5.3	Inicializace objektu ChatOllama	32
5.4	Inicializace objektu ChatOpenAI	33
5.5	Vytvoření historie zpráv	33
5.6	Vyvolání modelu	34
7.1	Načtení dokumentů z adresáře	41
7.2	Rozdělení textů na menší celky	42
7.3	Embedding dokumentů a ukládání do databáze	43
7.4	Nahrazení systémového modulu sqlite3	43
7.5	Užití třídy ConversationalRetrievalChain	44
8.1	Definice tříd HTTP serveru	45
8.2	Definice tříd HTTP serveru	47
9.1	HTML část klientské aplikace	48
9.2	JavaScriptová část klientské aplikace	50

SEZNAM PŘÍLOH

- P I. Konzolová aplikace
- P II. Serverová aplikace
- P III. Klientská aplikace

PŘÍLOHA P I. KONZOLOVÁ APLIKACE

```
1 from langchain_community.chat_models import ChatOllama
2 from langchain_openai import ChatOpenAI
3 from langchain_core.messages import HumanMessage, SystemMessage
4 from langchain.memory import ChatMessageHistory
5 import dotenv
6
7 dotenv.load_dotenv()
8 models = {
9     "1": ("ChatGPT 3.5-turbo", ChatOpenAI(model="gpt-3.5-turbo")),
10    "2": ("ChatGPT 4", ChatOpenAI(model="gpt-4")),
11    "3": ("ChatGPT 4-turbo", ChatOpenAI(model="gpt-4-turbo")),
12    "4": ("Llama2 7B q4", ChatOllama(model="llama2:7b-chat-q4_0")),
13    "5": ("Llama2 7B fp16", ChatOllama(model="llama2:7b-chat-fp16")),
14    "6": ("Llama2 13B q4", ChatOllama(model="llama2:13b-chat-q4_0")),
15    "7": ("Llama2 13B q8", ChatOllama(model="llama2:13b-chat-q8_0")),
16    "8": ("Llama3 8B q4", ChatOllama(model="llama3:8b-instruct-q4_0")),
17    "9": ("Llama3 8B q8", ChatOllama(model="llama3:8b-instruct-q8_0")),
18    "10": ("Gemma", ChatOllama(model="gemma")),
19    "11": ("Mistral", ChatOllama(model="mistral"))
20 }
21 for model_index, (model_name, _) in models.items():
22     print(f"{model_index}: {model_name}")
23
24 while True:
25     selected_model_index = input("Select a model to interact with:")
26     if selected_model_index in models.keys():
27         chat = models[selected_model_index][1]
28         chat_history = ChatMessageHistory()
29         chat_history.add_message(SystemMessage("You are a helpful assistant."))
30         break
31     else:
32         print("Invalid input. Please select 1 - 11")
33
34 while True:
35     prompt = input("Enter your prompt (type 'exit' to quit):")
36     if prompt == "exit":
37         print("Goodbye")
38         break
39     chat_history.add_message(HumanMessage(content=prompt))
40     response = chat.invoke(chat_history.messages)
41     chat_history.add_message(response)
42     print(response.content)
```

Fragment zdrojového kódu I.1 Konzolová aplikace

```
1 import dotenv
2 from langchain_community.chat_models import ChatOllama
3 from langchain_openai import ChatOpenAI, OpenAIEmbeddings
4 from langchain_community.vectorstores import Chroma
5 from langchain.chains import ConversationalRetrievalChain
6 from langchain.text_splitter import RecursiveCharacterTextSplitter
7 from langchain_community.document_loaders import DirectoryLoader
8
9 dotenv.load_dotenv()
10 models = {
11     "1": ("ChatGPT 3.5-turbo", ChatOpenAI(model="gpt-3.5-turbo")),
12     "2": ("ChatGPT 4", ChatOpenAI(model="gpt-4")),
13     "3": ("ChatGPT 4-turbo", ChatOpenAI(model="gpt-4-turbo")),
14     "4": ("Llama2 7B q4", ChatOllama(model="llama2:7b-chat-q4_0")),
15     "5": ("Llama2 7B fp16", ChatOllama(model="llama2:7b-chat-fp16")),
16     "6": ("Llama2 13B q4", ChatOllama(model="llama2:13b-chat-q4_0")),
17     "7": ("Llama2 13B q8", ChatOllama(model="llama2:13b-chat-q8_0")),
18     "8": ("Llama3 8B q4", ChatOllama(model="llama3:8b-instruct-q4_0")),
19     "9": ("Llama3 8B q8", ChatOllama(model="llama3:8b-instruct-q8_0")),
20     "10": ("Gemma", ChatOllama(model="gemma")),
21     "11": ("Mistral", ChatOllama(model="mistral"))
22 }
23
24 loader = DirectoryLoader('docs/', glob="**/*.html")
25 docs = loader.load()
26 text_splitter = RecursiveCharacterTextSplitter(
27     chunk_size=1000,
28     chunk_overlap=100,
29     length_function=len,
30     is_separator_regex=False
31 )
32 documents = text_splitter.split_documents(docs)
33
34 print("Creating Chromadb vector store ...")
```

Fragment zdrojového kódu I.2 Konzolová aplikace (1)

```
35 vectordb = Chroma.from_documents(
36     documents,
37     embedding=OpenAIEmbeddings(model="text-embedding-3-large"),
38     persist_directory="./data"
39 )
40
41 for model_index, (model_name, _) in models.items():
42     print(f"{model_index}: {model_name}")
43
44 while True:
45     selected_model_index = input("Select a model to interact with:")
46     if selected_model_index in models.keys():
47         chat = models[selected_model_index][1]
48         break
49     else:
50         print("Invalid input. Please select 1 - 11")
51
52 chat_history = []
53 chain = ConversationalRetrievalChain.from_llm(
54     chat,
55     retriever=vectordb.as_retriever(search_kwargs={'k': 6}),
56 )
57 while True:
58     prompt = input("Enter your prompt (type 'exit' to quit):")
59     if prompt == "exit":
60         print("Goodbye.")
61         break
62     result = chain.invoke(
63         {"question": prompt, "chat_history": chat_history}
64     )
65     chat_history.append((prompt, result["answer"]))
66     print(result)
```

Fragment zdrojového kódu I.2 Konzolová aplikace (2)

PŘÍLOHA P II. SERVEROVÁ APLIKACE

```
1  __import__('pysqlite3')
2  import sys
3  sys.modules['sqlite3'] = sys.modules.pop('pysqlite3')
4
5  from langchain_community.chat_models import ChatOllama
6  from langchain_openai import ChatOpenAI, OpenAIEmbeddings
7  from langchain_community.vectorstores import Chroma
8  from langchain.chains import ConversationalRetrievalChain
9  from langchain.text_splitter import RecursiveCharacterTextSplitter
10 from langchain_community.document_loaders import DirectoryLoader
11 import dotenv
12 import socketserver
13 import http.server
14 from urllib import parse
15
16
17 class HttpServer:
18     def __init__(self, vectordb):
19         def handler(*args):
20             RequestHandler(vectordb, *args)
21         with socketserver.TCPServer(("", 8888), handler) as httpd:
22             print("HTTP server started.")
23             httpd.serve_forever()
24
25
26 class RequestHandler(http.server.BaseHTTPRequestHandler):
27     chat_histories = {}
28
29     def __init__(self, vectordb, *args):
30         print("Request handler initializing...")
31
32         self.models = {
33             "1": ( "ChatGPT 3.5-turbo", ChatOpenAI(model="gpt-3.5-turbo")),
34             "2": ( "ChatGPT 4", ChatOpenAI(model="gpt-4")),
35             "3": ( "ChatGPT 4-turbo", ChatOpenAI(model="gpt-4-turbo")),
36             "4": ( "Llama2 7B q4", ChatOllama(model="llama2:7b-chat-q4_0")),
37             "5": ( "Llama2 7B fp16", ChatOllama(model="llama2:7b-chat-fp16")),
38             "6": ( "Llama2 13B q4", ChatOllama(model="llama2:13b-chat-q4_0")),
39             "7": ( "Llama2 13B q8", ChatOllama(model="llama2:13b-chat-q8_0")),
40             "8": ( "Llama3 8B q4", ChatOllama(model="llama3:8b-instruct-q4_0")),
41             "9": ( "Llama3 8B q8", ChatOllama(model="llama3:8b-instruct-q8_0")),
42             "10": ( "Gemma", ChatOllama(model="gemma")),
43             "11": ( "Mistral", ChatOllama(model="mistral"))
44         }
45         self.chat = ChatOpenAI(model="gpt-4")
46         self.vectordb = vectordb
47         http.server.BaseHTTPRequestHandler.__init__(self, *args)
48
49     def do_GET(self):
50         self.send_response(405)
```

Fragment zdrojového kódu II.1 Serverová aplikace (1)

```

51
52 def do_POST(self):
53     print("Received POST request")
54     if self.path == '/api/v1/query':
55         print("Processing...")
56         content_length = int(self.headers['Content-Length'])
57         post_data = self.rfile.read(content_length)
58         fields = parse.parse_qs(str(post_data, "UTF-8"))
59         print("Fields: ", fields)
60         if "q" not in fields or "cid" not in fields or len(fields["q"]) == 0
        ↪ or len(fields["cid"]) == 0:
61             print("Malformed request")
62             self.send_response(400)
63         else:
64             query = fields["q"][0]
65             customer_id = fields['cid'][0]
66             print("Query: ", query)
67             print("CID: ", customer_id)
68
69             if customer_id not in self.chat_histories:
70                 print("Creating new chat history for customer...")
71                 self.chat_histories[customer_id] = []
72
73             chain = ConversationalRetrievalChain.from_llm(
74                 self.chat,
75                 retriever=self.vectordb.as_retriever(search_kwargs={'k': 6})
76             )
77
78             print("Querying...")
79             result = chain.invoke(
80                 {"question": query, "chat_history":
81                 ↪ self.chat_histories[customer_id]}
82             )
83             self.chat_histories[customer_id].append((query,
84                 ↪ result["answer"]))
85             print("Result: ", result)
86
87             self.send_response(200)
88             self.send_header('Content-type', 'text/plain')
89             self.send_header('Access-Control-Allow-Origin', '*')
90             self.end_headers()
91             self.wfile.write(result["answer"].encode())
92             print("Response sent.")
93             print("Complete chat history: ", self.chat_histories)
94         else:
95             self.send_response(400)

```

Fragment zdrojového kódu II.1 Serverová aplikace (2)

```
95
96 if __name__ == '__main__':
97     print("Starting...")
98     dotenv.load_dotenv()
99
100    print("Initializing DirectoryLoader...")
101    loader = DirectoryLoader('docs/', glob="**/*.html")
102    docs = loader.load()
103    print("Found ", len(docs), " docs")
104
105    print("Creating vector store index...")
106    text_splitter = RecursiveCharacterTextSplitter(
107        chunk_size=1000,
108        chunk_overlap=100,
109        length_function=len,
110        is_separator_regex=False
111    )
112    documents = text_splitter.split_documents(docs)
113
114    print("Creating Chromadb vector store ...")
115    vectordb = Chroma.from_documents(
116        documents,
117        embedding=OpenAIEmbeddings(model="text-embedding-3-large"),
118        persist_directory="./data"
119    )
120    print("Vector store initialized.")
121
122    print("Launching HTTP server...")
123    server = HttpServer(vectordb)
```

Fragment zdrojového kódu II.1 Serverová aplikace (3)

PŘÍLOHA P III. KLIENTSKÁ APLIKACE

```
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>LLM Chatbot</title>
7    <style>
8      .wrapper {
9        width: 50vw;
10       border: 1px solid #ccc;
11       border-radius: .5rem;
12       margin: 0 auto;
13       margin-top: 1rem;
14     }
15     .chat-container {
16       width: 100%;
17       padding: 10px;
18       box-sizing: border-box;
19       display: flex;
20       flex-direction: column;
21       margin-bottom: 10px;
22       height: 80vh;
23       overflow-y: auto;
24     }
25
26     .message {
27       max-width: 80%;
28       margin-bottom: 10px;
29       padding: 8px 12px;
30       border-radius: 10px;
31       color: white;
32       font-family: Arial, sans-serif;
33     }
34
35     .user {
36       align-self: flex-end;
37       background-color: #0D6EFD;
38     }
39
40     .system {
41       align-self: flex-start;
42       background-color: #6C757D;
43     }
44
45     .input-area {
46       display: flex;
47       justify-content: space-between;
48       padding: 10px;
49     }
50
```

Fragment zdrojového kódu I.1 Klientská aplikace (1)

```
51     .chat-input {
52         flex-grow: 1;
53         margin-right: 10px;
54         padding: 8px;
55         border: 1px solid #ccc;
56         border-radius: 5px;
57     }
58
59     button {
60         padding: 10px 20px;
61         background-color: #007bff;
62         color: white;
63         border: none;
64         border-radius: 5px;
65         cursor: pointer;
66     }
67
68     button:hover {
69         background-color: #0056b3;
70     }
71
72     .typing span {
73         display: inline-block;
74         width: 8px;
75         height: 8px;
76         border-radius: 50%;
77         background-color: white;
78         margin: 0 2px;
79         animation: typing 1s infinite;
80     }
81
82     @keyframes typing {
83         0% { transform: translateY(0); }
84         20% { transform: translateY(-5px); }
85         40% { transform: translateY(0); }
86     }
87 </style>
88
89
90 </head>
91
```

Fragment zdrojového kódu III.1 Klientská aplikace (2)

```
92 <body>
93
94     <div class="wrapper">
95         <div class="chat-container">
96             <div class="message system">Dobrý den, jak Vám mohu pomoci?</div>
97         </div>
98
99         <form class="input-area" id="messageForm">
100             <input type="text" name="q" placeholder="Zde napište dotaz..."
101                 ↪ id="chatInput" class="chat-input">
102                 <input type="hidden" name="cid" value="1">
103             <button type="submit">Odeslat</button>
104         </form>
105     </div>
106
107     <template class="system">
108         <div class="message system">
109             <div class="typing">
110                 <span></span><span></span><span></span><span></span>
111             </div>
112         </div>
113     </template>
114
115     <template class="user">
116         <div class="message user"></div>
117     </template>
```

Fragment zdrojového kódu III.1 Klientská aplikace (3)

```

117
118 <script>
119   const systemTemplate = document.querySelector('template.system')
120   const userTemplate = document.querySelector('template.user')
121   const customerId = document.querySelector('input[name="cid"]').value
122
123   document.getElementById('messageForm').addEventListener('submit',
124     ↪ function(e) {
125     e.preventDefault();
126     const input = document.getElementById('chatInput');
127     const message = input.value.trim();
128     if (message) {
129       const formData = new FormData(e.target);
130       const encodedData = new URLSearchParams(formData).toString();
131       const chatContainer = document.querySelector('.chat-container');
132       let userClone = userTemplate.content.cloneNode(true);
133       userClone.querySelector(".message").textContent = message;
134       input.value=""
135       chatContainer.appendChild(userClone);
136
137       let systemClone = systemTemplate.content.cloneNode(true);
138       chatContainer.appendChild(systemClone);
139
140       fetch('http://localhost:8888/api/v1/query', {
141         method: 'POST',
142         headers: {
143           'Content-Type': 'application/x-www-form-urlencoded'
144         },
145         body: encodedData
146       })
147       .then(response => {
148         if (!response.ok) {
149           throw new Error('Network response error: ' +
150             ↪ response.statusText);
151         }
152         return response.text();
153       })
154       .then(text => {
155         ↪ chatContainer.querySelector('.message:last-child').textContent
156         ↪ = text
157       })
158       .catch(error => console.error('Error:', error));
159
160       chatContainer.scrollTop = chatContainer.scrollHeight;
161     }
162   });
163 </script>
164 </body>
165 </html>

```

Fragment zdrojového kódu III.1 Klientská aplikace (4)