



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ

DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ VELKÝCH PŘEDTRÉNOVANÝCH JAZYKOVÝCH MODELŮ PRO KONFIGURACI A PODPORU KLINICKÉHO INFORMAČNÍHO SYSTÉMU

UTILISING LARGE PRETRAINED LANGUAGE MODELS FOR CONFIGURATION AND SUPPORT OF A CLINICAL INFORMATION SYSTEM

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

MICHAL SOVA

VEDOUCÍ PRÁCE

SUPERVISOR

RNDr. MAREK RYCHLÝ, Ph.D.

BRNO 2024

Zadání diplomové práce



153714

Ústav: Ústav informačních systémů (UIFS)
Student: **Sova Michal, Bc.**
Program: Informační technologie a umělá inteligence
Specializace: Inteligentní systémy
Název: **Využití velkých předtrénovaných jazykových modelů pro konfiguraci a podporu klinického informačního systému**
Kategorie: Umělá inteligence
Akademický rok: 2023/24

Zadání:

1. Seznamte se s podstatou a možným použitím velkých předtrénovaných jazykových modelů, prozkoumejte a porovnejte různé typy těchto modelů (LLaMA, Alpaca) a související služby (např. Open AI API, Azure Open AI). Seznamte se s možnostmi konfigurace vybraného klinického informačního systému a možnostmi jeho adaptace na konkrétní prostředí zákazníků.
2. Zvolte vhodný typ a nastavení velkého předtrénovaného jazykového modelu, postupy trénování a trénovací datové sady a natrénujte vybraný model pro dotazy nad strukturou entit (datový model), uživatelskou a programovou dokumentaci a řešenými požadavky uživatelské podpory. Zvolte vhodný způsob měření použitelnosti natrénovaného modelu a proveďte vyhodnocení.
3. Navrhněte způsob integrace jazykového modelu do klinického informačního systému jako služby poskytující informace konzultantům a správcům při testování, konfiguraci, nasazení a podpoře systému.
4. Po konzultaci s vedoucím implementujte navrženou službu integrujte do klinického informačního systému a výsledek důkladně otestujte.
5. Vyhodnoťte dosažené výsledky a diskutujte použitelnost konceptu a řešení při testování, konfiguraci, nasazení a podpoře klinického informačního systému.

Literatura:

- ZHAO, Wayne Xin, et al. A survey of large language models. arXiv preprint arXiv:2303.18223, 2023. Available at: <https://arxiv.org/abs/2303.18223>
- dokumentace vybraného klinického informačního systému

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

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

Vedoucí práce: **Rychlý Marek, RNDr., Ph.D.**
Vedoucí ústavu: Kolář Dušan, doc. Dr. Ing.
Datum zadání: 1.11.2023
Termín pro odevzdání: 17.5.2024
Datum schválení: 30.10.2023

Abstrakt

Cílem této práce je seznámení se s podstatou a možným použitím velkých předtrénovaných jazykových modelů, seznámení se s možnostmi konfigurace klinického informačního systému FONS Enterprise a možnost jeho adaptace na konkrétní prostředí zákazníků. Práce nejprve představuje velké předtrénované jazykové modely a informační systém FONS Enterprise. Následně se zaměřuje na možnosti dotrénování modelů a implementaci metody RAG na datech z klinického systému. Implementace RAG architektury je realizována pomocí nástroje LangChain a LlamaIndex. Výsledky ukazují, že metoda RAG s modelem Gemma a embedding modelem bge-m3 poskytuje nejrelevantnější odpovědi, ale má potíže s porozuměním složitějších otázek. Metoda dotrénování modelu nepřináší očekávané výsledky, a to ani po úpravách parametrů trénování.

Abstract

The aim of this work is to get acquainted with the essence and use of large pre-trained language models, to get acquainted with the configuration options of the clinical information system FONS Enterprise and the possibility of its adaptation to the specific environment of customers. The work first presents large pre-trained language models and the FONS Enterprise clinical information system. This work examines possibilities of training models and implementing RAG methods on data from the clinical system. The implementation of the RAG architecture is supported by the tools LangChain and LlamaIndex. The results show that the RAG method with the Gemma model and the bge-m3 embedding model provides the most relevant answers on basic questions, but struggles to understand more complex questions. The method of pre-training the model does not produce the expected results, even after adjusting the training parameters.

Klíčová slova

klinický informační systém, velké jazykové modely, velké předtrénované jazykové modely, architektura RAG, trénování jazykových modelů, Llama, Gemma, Mistral, LangChain, LlamaIndex

Keywords

clinical information system, large language models, RAG architecture, pre-training, Llama, Gemma, Mistral, LangChain, LlamaIndex

Citace

SOVA, Michal. *Využití velkých předtrénovaných jazykových modelů pro konfiguraci a podporu klinického informačního systému*. Brno, 2024. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

Využití velkých předtrénovaných jazykových modelů pro konfiguraci a podporu klinického informačního systému

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana RNDr. Marka Rychlého, Ph.D. Další informace mi poskytl Mgr. Jan Ptašek a společnost STAPRO s.r.o. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Michal Sova
16. května 2024

Poděkování

Chtěl bych poděkovat vedoucímu práce a Mgr. Janu Ptaškovi za poskytnutí odborné pomoci. Dále bych chtěl poděkovat MetaCentru za poskytnutí výpočetních zdrojů.

Obsah

1	Úvod	4
2	Velké předtrénované sítě	6
2.1	Nejvýznamnější pokroky modelů	7
2.1.1	ChatGPT a jeho vývoj	7
2.1.2	Modely LLaMA	8
2.2	Služby a nástroje	9
3	Klinický systém FONS Enterprise	11
3.1	Konfigurace systému FONS Enterprise	12
4	Možnosti trénování	13
4.1	Ladění instrukcí	13
4.2	Ladění chování	14
4.3	Metody efektivního ladění parametrů	15
4.4	Trénování	15
4.4.1	Data	16
4.4.2	Předzpracování dat	16
4.4.3	Architektura	18
4.4.4	Optimalizace	18
5	Získávání kontextu pro přesnou odpověď	20
5.1	RAG	20
5.2	Architektura RAG	21
5.2.1	Embedding model	22
5.2.2	Vektorové úložiště	22
5.2.3	Jazykový model	22
5.3	Nástroj LangChain	22
5.4	LlamaIndex	23
6	Trénování	25
6.1	Dataset	25
6.2	Trénování modelů	25
6.2.1	Llama 2	25
6.2.2	Ostatní modely	26
6.3	Změna parametrů	27
7	Implementace	32

7.1	RAG	32
7.1.1	Data	32
7.1.2	Implementace pomocí nástroje LangChain	32
7.1.3	Implementace pomocí nástroje LlamaIndex	33
7.1.4	Připojení dat z databáze STAPRO	38
7.2	Návrh integrace	38
7.2.1	API	38
8	Vyhodnocení a závěr	41
	Literatura	43
A	Obsah příloženého paměťového média	44

Seznam obrázků

2.1	Modely LLaMA [10]	9
3.1	Příklad graficky reprezentovaného modulu 'Registr pacientů' v systému FONS Enterprise	11
4.1	Proces předzpracování dat [10]	17
6.1	Nápověda FONS Enterprise	25

Kapitola 1

Úvod

Předtrénované jazykové modely otevírají široké možnosti využití v mnoha odvětvích. Jejich potenciál je obzvláště patrný v oblasti zpracování přirozeného jazyka, kde mohou efektivně analyzovat a generovat text s lidskou úrovní přesnosti a porozumění. Tyto modely jsou schopny porozumět otázkám a generovat odpovědi v daném jazyce s přesností a flexibilitou.

Jazykové modely, konkrétně chatboti, jsou modely neuronové sítě navržené k simulaci konverzace s uživateli. Představují v dnešní digitální éře klíčový prvek v interakci mezi lidmi a technologií. Jejich potenciální využití je široké a rozmanité, přičemž se promítá do mnoha oblastí lidské činnosti.

Primárním cílem chatbotů je efektivně komunikovat s uživateli a poskytovat jim potřebné informace či řešení problémů v reálném čase. Díky své schopnosti porozumět lidskému jazyku a reagovat na něj mohou být nasazováni v zákaznické podpoře, online obchodech, bankovníctví, zdravotnictví, a mnoha dalších odvětvích.

Pro firmy představují chatboti nejen způsob, jak zefektivnit komunikaci se zákazníky a snížit náklady na zákaznickou podporu, ale také prostředek k získání cenných dat o chování uživatelů a jejich potřebách. Díky neustálému vylepšování a zdokonalování algoritmů se chatboti stávají stále sofistikovanějšími a schopnějšími adaptovat se na různé situace a uživatelské požadavky.

V kontextu této diplomové práce se jazykové modely stávají klíčovým prvkem v implementaci chatbotů pro analýzu a odpovídání na dotazy nad datovými sadami klinického informačního systému. Jejich schopnost porozumět složitým zdravotnickým termínům a poskytovat užitečné informace může přinést zásadní zlepšení v oblasti poskytování zdravotní péče a zpracování lékařských dat.

Cílem této diplomové práce je vybrat vhodný předtrénovaný jazykový model, definovat postupy trénování a identifikovat trénovací data pro natrénování modelu na dotazy spojené s klinickým informačním systémem.

Trénování rozsáhlých jazykových modelů je spojeno s vysokými výpočetními nároky. Proto je zkoumáno doladění chování modelů a implementace technik získávání kontextu pro další zpracování jazykovými modely jako je například systém RAG (Retrieval Augmented Generation).

V jednotlivých kapitolách jsou postupně představeny a zkoumány možnosti trénování a implementace modelů s daty konkrétního informačního systému.

V kapitole 2 jsou představeny přední předtrénované sítě a modely a jejich aplikace v různých oblastech. Kapitola se zaměřuje na popis jejich struktury a funkčnosti.

Kapitola 3 je zaměřena na klinický informační systém FONS Enterprise, který slouží jako hlavní zdroj dat pro trénování a testování jazykového modelu.

V kapitolách 4 a 5 jsou představeny různé možnosti trénování modelů a implementace metody RAG na datech z klinického systému.

V kapitolách 6 a 7 jsou představeny a implementovány různé metody trénování vybraných modelů a implementace techniky RAG na datech z klinického systému FONS Enterprise.

Závěrečná kapitola pak popisuje navrhovanou integraci vzniklého jazykového modelu do stávajícího prostředí klinického informačního systému pomocí webové aplikace a API.

Kapitola 2

Velké předtrénované sítě

Velké předtrénované jazykové modely jsou modely složené z miliard parametrů, které jsou učeny na obrovském množství textových dat a zároveň jsou schopny generovat text na základě zadaného vstupu, jako je například GPT-3, PaLM, nebo LLaMA. Pro trénování těchto modelů od nuly je zapotřebí obrovské množství dat, často několik desítek GPU. Pro tyto účely je implementována řada algoritmů pro paralelní trénování, jako je například DeepSpeed. Tyto modely se pak trénují i několik měsíců. [10]

Jazykové modely prokazují schopnost porozumění přirozeného jazyka, schopnost odpovídat na otázky, schopnost plnit mezery v textu a schopnost vykonávat úlohy jako překlady mezi různými jazyky.

Modely jsou především postaveny na architektuře transformerů, kde tzv. vrstvy pozornosti (angl. *attention layers*) jsou poskládány do hluboké neuronové sítě. Škálování těchto sítí může do značné míry zlepšit kapacitu velkých předtrénovaných modelů.

Tímto škálováním získávají velké modely lepší schopnost:

- Učení z kontextu (in-context learning) – formálně uvedena GPT-3. Za předpokladu, že jazykovému modelu bylo poskytnuto dostatek demonstračních instrukcí v přirozeném jazyce, může model vygenerovat očekávaný výstup pro jednotlivé testovací instance doplněním vstupního textu bez nutnosti dodatečného trénování.
- Následování instrukcí (instruction following) – doladěním (fine-tuning) pomocí formátovaného datasetu ve tvaru instrukcí. Velké předtrénované modely fungují dobře na neviděné úkoly, které jsou popsány formou instrukcí. Modely tedy rozumí pokynům pro nové úkoly bez použití explicitních příkladů, čímž dochází ke zlepšení schopnosti generalizace.
- Postupné uvažování (step-by-step reasoning) – u malých jazykových modelů je obvykle obtížné řešit složité úkoly, které zahrnují více kroků uvažování jako například matematické slovní úlohy. Naproti tomu se strategií chain-of-thought modely mohou tyto úlohy vyřešit pomocí mezikroků a doložit, jak k tomuto výsledku přišli.

Vzhledem k výkonu a velikosti těchto modelů je běh na běžných počítačích často náročný. Z tohoto důvodu existují různé služby, které poskytují rozhraní pro přístup k těmto modelům. Například OpenAI nabízí OpenAI API, které umožňuje vývojářům integrovat velké předtrénované modely, jako je například GPT-3, do svých aplikací. Azure Open AI je podobný nástroj poskytovaný společností Microsoft, který nabízí přístup k předtrénovaným jazykovým modelům pro vývojáře.

2.1 Nejvýznamnější pokroky modelů

Mezi nejznámější modely patří chatGPT ze série GPT, LLaMA, či PaLM. V následujících kapitolách jsou detailněji popsány GPT modely, LLaMA modely a jejich vývoj.

2.1.1 ChatGPT a jeho vývoj

ChatGPT je vyvinut na základě modelu GPT, který je speciálně optimalizován pro potřeby konverzace. Základní princip GPT modelů je zkomprimovat světové znalosti (world knowledge) do modelu pomocí jazykového modelování tak, aby si dokázal zapamatovat a vzpomenout na sémantiku této světové znalosti a sloužil jako univerzální řešitel úloh. Dva klíčové body jsou:

- trénování modelu tak, aby přesně určil následující slovo
- škálování modelu

Vývoj GPT modelů se pak dá rozdělit do následujících sekcí.

GPT-1

V roce 2017 byl představen model transformerů společností Google a společnost OpenAI adaptovala tento tento model do své architektury modelu. První GPT model byl pak publikován roku 2018 jako GPT-1. Tento model přijal hybridní přístup 'unsupervised pretraining' a 'supervised fine-tuning'. GPT-1 nastavil základní architekturu pro modely řady GPT a stanovil základní princip modelování textu přirozeného jazyka, tj. předpovídání dalšího slova.

GPT-2

GPT-2 napodobuje podobnou architekturu jako GPT-1. Zvýšil počet parametrů na 1.5 miliard parametrů, který byl trénován na datasetu WebText. Zde je pravděpodobný výstup podmíněn vstupem a informací o úloze. Díky této podmíněné pravděpodobnosti lze jednotně použít formát vstupu, výstupu a informací o úloze a proces řešení úlohy může být označen jako problém predikce slova pro generování výsledného textu. Každý úkol lze považovat za problém založený na predikci slov na podmnožinu veškerého textu. Měl by tedy model být schopen plnit různé úkoly, kdyby měl dostatek textu.

GPT-3

U modelu GPT-3 (představen v roce 2020) byly vyškálovány parametry modelu na velikost 175B. Byl formálně představen koncept 'in-context learning' (ICL) – učení z kontextu. ICL může učit (nebo instruovat) model porozumět úkolům ve formě textu v přirozeném jazyce. ICL předpovídá správné řešení úlohy, které může být také formátováno jako textová sekvence vzhledem k popisu úlohy. GPT-3 nejen demonstruje velmi dobrý výkon v různých úkolech zpracování přirozeného jazyka, ale také na speciálně navržených úkolech, které vyžadují schopnosti uvažování nebo adaptace. Lze pozorovat velký výkonnostní skok oproti GPT-2. Celkově může být GPT-3 považováno za pozoruhodný mezník v rozvíjení velkých předtrénovaných modelů. Bylo empiricky dokázáno, že škálování neuronové sítě do značné míry mohou vést k obrovskému zvýšení výpočetní kapacity modelu.

OpenAI prozkoumala dva hlavní přístupy k dalšímu zlepšování modelu GPT-3:

- Trénování na datech kódu – původní model GPT-3, který byl předtrénován na prostém textu, nemá schopnost uvažovat o složitých úkolech, jako je například kód nebo matematické problémy. Pro posílení této schopnosti, byl představen společností OpenAI Codex [2], což byl model GPT doladěný na velkém množství dat obsahující GitHub kódy. Ukázalo se, že Codex může vyřešit velmi obtížné problémy spojené s programováním a také vede k významnému zlepšení formy při řešení matematických úloh. Trénování na textových datech a datech kódu ukázaly zlepšení řady souvisejících úkolů (textové vyhledávání, vyhledávání kódu). To naznačuje, že trénování na datech je užitečná praxe ke zlepšení kapacity a schopnosti uvažování modelů GPT.
- Učení se hodnotám člověka (human alignment) – založená na posilovaném učení (RL), které se učilo ze srovnání preferencí anotovaných lidmi. Později byl představen algoritmus 'reinforcement learning from human feedback (RLHF)' modelem InstructGPT, který je zvláště užitečný ke zmírnění problémů související s generováním toxického nebo škodlivého textu.

Tyto techniky vedou ke zlepšení modelů GPT-3 s lepšími vlastnostmi, které jsou nazývány společností OpenAI GPT-3.5.

ChatGPT

ChatGPT je založený na modelech GPT-3.5 a GPT-4. ChatGPT byl natrénován podobným způsobem jako model InstructGPT, avšak speciálně optimalizovaný pro dialogy. V tomto případě byl dataset pro InstructGPT převeden do dialogového formátu a zkombinován s konverzacemi lidí.

ChatGPT vykazoval vynikající schopnosti v komunikaci s lidmi, zejména:

- rozsáhlá zásoba znalostí
- dovednost uvažovat o matematických problémech
- sledovat kontext v dialozích
- přizpůsobení lidským hodnotám pro bezpečné používání

Později byla představena možnost zásuvných modulů pro ChatGPT, které dále rozšiřují možnosti ChatGPT.

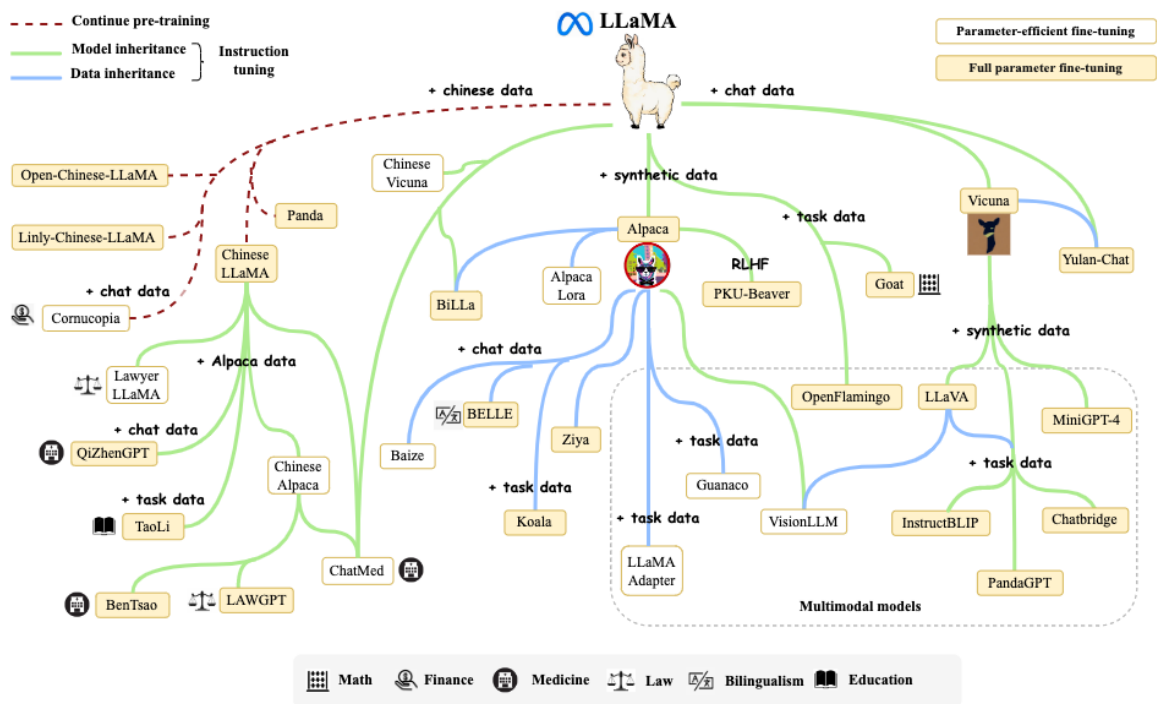
GPT-4

Dalším pozoruhodným pokrokem je GPT-4. Celkově má GPT-4 lepší schopnosti v řešení komplexních úloh než GPT-3.5, přičemž vykazuje velké zlepšení výkonu v mnoha vyhodnocovacích úlohách. Kromě toho GPT-4 bezpečněji reaguje na škodlivé nebo provokativní dotazy.

Navzdory obrovskému pokroku stále existují omezení, která se týkají těchto modelů, jako je například generování halucinací s faktickými chybami.

2.1.2 Modely LLaMA

Kolekci modelů LLaMA představila společnost Meta AI v únoru 2023. Kolekce modelů obsahuje několik velikostí (7B, 13B, 70B). Vydáním přitáhla LLaMA rozsáhlou pozornost ze strany výzkumné i průmyslové komunity. Modely LLaMA dosáhly velmi vynikajícího



Obrázek 2.1: Modely LLaMA [10]

výkonu na různých benchmarcích a dosud se staly nejpoblárnějšími otevřenými jazykovými modely. Mnoho lidí rozšířilo modely LLaMA buď o instrukční ladění nebo průběžné předtrénování. Zejména instrukční ladění LLaMA se stalo hlavním přístupem k vývoji specializovaných modelů, a to z důvodu relativně nízkých výpočetních nákladů.

Pro efektivní adaptaci modelů LLaMA v různých jazycích je často potřeba rozšířit původní slovní zásobu (trénovanou především na anglických textech) nebo jej doladit pomocí instrukcí nebo dat v cílovém jazyce. Mezi tyto rozšířené modely patří Alpaca. Model Alpaca je založen na modelu LLaMA-7B a je prvním otevřeným modelem následující instrukce. Dataset a kód jsou hojně přebírány v pracích, jako je Alpaca-LoRA (reprodukce Alpaky s použitím tzv. LoRA optimalizace) a Koala. Díky vynikajícímu výkonu a dostupnosti modelů LLaMA je mnoho modelů používá jako základní jazykové modely k dosažení schopnosti porozumění jazyku a generování.

Na obrázku 2.1 lze vidět základní model LLaMA a jejich varianty.

2.2 Služby a nástroje

Kvůli velké výpočetní i paměťové náročnosti jazykových modelů vznikly služby a nástroje, které umožňují trénování těchto modelů.

OpenAI poskytuje veřejné API, služby a nástroje pro trénování GPT modelů¹. Nabízí řadu modelů s různými možnostmi ladění, jejich služby jsou rozděleny do několika cenových kategorií.

¹<https://platform.openai.com/docs/introduction>

Dalším poskytovatelem je HuggingFace² – komunitní a vědecká platforma, která poskytuje modely a nástroje pro vytváření, trénování a nasazení modelů strojového učení založené na open source kódu a technologiích.

Google nabízí Google Collab – umožňuje psát a spouštět v prohlížeči skripty v Pythonu, kde je také přístup k GPU a možnost sdílení.

²<https://huggingface.co>

Kapitola 3

Klinický systém FONS Enterprise

Klinický informační systém FONS Enterprise firmy Stapro¹ je rozsáhlý informační systém pro podporu zdravotnických zařízení a jejich řetězců. Zaměřuje se zejména na řízení a podporu hlavních procesů péče o pacienty, vedení zdravotnické dokumentace, vykazání poskytnuté péče a logistiku zdravotního materiálu².

Systém FONS Enterprise je stavěn na objektech, takzvaných entitách – logických, datových nebo funkčních blocích. FONS Enterprise pracuje s entitami řady typů a tyto entity jsou vzájemně různě provázány. Jedny z nejvýznamnějších entit jsou:

- jednoduché datové typy (číslo, řetězec, datum)
- datová entita – základní datový objekt složen z jiných entit (pacient, útvar, diagnóza)
- číselník – slouží k uložení sady záznamů stejného typu tvořících logický celek
- modul – funkční část FONS Enterprise, typicky odpovídá jednomu oknu aplikace (modul dokladu pro pojišťovnu, modul pro editaci uživatele), příklad modulu lze vidět na obrázku 3.1

Identifikační číslo	Jméno pacienta	Rodné číslo	Datum narození	Rodné příjmení	Poznámka	Hlavi	Číslo pojištění	Druh poj vztahu
	stapri							
460415999	Stapř Arnošt	460415999	15.4.1946			111	460415999	1
7707172737	Stapř Ivo	7707172737	17.7.1977			111	7707172737	1
7707077070	Stapř Ivo	7707077070	7.7.1977			111	7707077070	1
6660066600	Stapřová Alice	6660066600	6.10.1966			111	6660066600	1
6660066611	Stapřová Běta	6660066611	6.10.1966			111	6660066611	1
6660066622	Stapřová Cecilka	6660066622	6.10.1966			111	6660066622	1
6660066699	Stapřová Peppa	6660066699	6.10.1966			111	6660066699	1
6660066633	Stapřová Tina	6660066633	6.10.1966			111	6660066633	1
6661166611	Stapřová Vilma	6661166611	16.11.1966			111	6661166611	1

Obrázek 3.1: Příklad graficky reprezentovaného modulu 'Registr pacientů' v systému FONS Enterprise

¹<https://www.stapro.cz/produkty/fons-enterprise/>

²nápověda FONS Enterprise

Všechny tyto entity jsou velmi konfigurovatelné a obsahují mnoho vazeb na ostatní objekty (entity).

3.1 Konfigurace systému FONS Enterprise

System FONS Enterprise je velmi obecný a díky široké konfigurovatelnosti je nasaditelný ve velmi rozdílných situacích. Prakticky každý modul, každý seznam, každý objekt je nastavitelný. K řadě nastavení je možné přistupovat z konkrétních funkčních oblastí, částečně přímo z objektů se kterými se pracuje (např. při výběru z číselníku může uživatel s dostatečnými právy do číselníku doplnit chybějící záznam nebo upravit záznam existující).

Vývoj a konfigurace systému FONS se provádí ve vrstvách, které se nazývají autority. Autority slouží pro oddělení funkcionality tvořené programátorem a konfigurace tvořené konzultantem nebo programátorem, případně nastavené správcem zakázky nebo uživatelem. Dále odděluje nadnárodní, národní (etalonová) a zakázková řešení. Tyto autority jsou součástí obecného systému entit. Využívají se u všech typů entit a u jejich položek.

Kapitola 4

Možnosti trénování

Jelikož je trénování velkých jazykových modelů výpočetně velmi náročné, nejlepší možností k naučení modelu specifickým úkolům je doladění (fine-tuning) těchto modelů. Dva hlavní přístupy jsou ladění instrukcí (instruction tuning) a ladění chování (alignment tuning). Ladění instrukcí zvyšuje schopnosti LLM, zatímco ladění chování je uvádí do souladu s lidskými hodnotami. Dále je zde také možnost dalšího přetrénování. Všechny tři možnosti jsou dále probrány v následujících kapitolách. Dále jsou uvedeny metody efektivního trénování – snížení množství trénovaných parametrů při zachování podobného výkonu jazykového modelu.

4.1 Ladění instrukcí

Ladění instrukcí je metoda pro jemné doladění předem natrénovaných jazykových modelů na datasetu formátovaných instrukcí v přirozeném jazyce. Instance ve formátu instrukce se skládají z popisu úlohy, volitelného vstupu, výstupu a ukázek. Ladění instrukcí je široce používáno ve stávajících jazykových modelech, jako je například InstructGPT a GPT-4.

Formátování datasetu vhodného pro trénování zahrnuje vytváření víceúlohových tréninkových datových sad pomocí popisů úloh v přirozeném jazyce. Tyto datové sady jsou rozšířeny o člověkem napsané popisy úkolů, které instruují učící se jazykové modely, aby porozuměly úkolům a vysvětlily jejich cíle. Pokyny jsou zásadní pro schopnost zobecnění úkolů a jemné doladění modelů nad označenými datovými sadami. Tyto datasety jsou tvořeny pomocí skutečných uživatelských dotazů z různých fór. Dotazy a odpovědi jsou roztríděny a dále popsány instrukcemi.

Byly navrženy poloautomatické přístupy, aby se omezila potřeba manuálních anotací a ručního shromažďování instancí. Metodě Self-Instruct stačí 100 instancí jako počáteční fond úloh, přičemž náhodně vybere několik instancí úloh z datasetu a tyto instance předloží jazykovému modelu, pro vygenerování dalších instancí.

Škálování datasetu pro trénování instrukcí může zlepšit schopnost zobecnění jazykových modelů. Od určitého bodu má však škálování minimální dopad na výkon modelu [3]. Další důležitou částí je návrh formátu. Přidání popisů úloh a volitelných ukázek do stávajících datových sad může vést k podstatným zlepšením.

Ladění instrukcí zvyšuje výkon modelu v různých měřítkách a je méně nákladný než plné trénování zejména kvůli menšímu množství požadovaných instrukčních dat. Ladění instrukcí také umožňuje modelům porozumět instrukcím v přirozeném jazyce pro dokončení úkolů, což jim umožňuje provádět konkrétní úkoly bez demonstrací. To vede k vynikajícímu

výkonu na viděných i neviděných úlohách. Modely vytrénované pomocí ladění instrukcí mohou zobecnit související úkoly napříč jazyky, což snižuje trénování pro konkrétní jazyk. Ladění instrukcí je také efektivní v doménové specializaci, protože umožňuje přizpůsobit se konkrétním úkolům jako je medicína, právo a finance. Velkou slabinou je forma datasetu, která musí být v určitém formátu a většinou se musí konstruovat a upravovat lidmi.

4.2 Ladění chování

Velké jazykové modely jsou účinné v různých úkolech zpracování přirozeného jazyka, ale někdy mohou způsobit nezamýšlené chování, jako je vymyšlení nepravdivých informací nebo objektivních výrazů. Aby se tomu zabránilo, bylo navrženo učení se hodnotám člověka (human alignment), které vyžaduje zvážení různých kritérií, jako je vstřícnost, čestnost a neškodnost. Tato technika může potenciálně poškodit schopnosti jazykových modelů.

Pozornost si získal vývoj různých kritérií pro regulaci chování. Tři reprezentativní kritéria zarovnání jsou užitečnost, čestnost a neškodnost.

- Užitečnost modelu – zahrnuje pomoc uživatelům při řešení úkolů nebo efektivním zodpovídání otázek.
- Čestnost modelu – vyžaduje po modelu přesný obsah a odpovídající stupeň jistoty ve výstupu.
- Neškodnost modelu – model generuje jazyk, který není urážlivý nebo diskriminační a model by měl odhalit skryté snahy zaměřené na získávání škodlivých požadavků.

Tato kritéria jsou subjektivní a založená na lidském úsudku, takže je obtížné přímo formulovat tyto cíle.

Vysoce kvalitní zpětná vazba ze strany lidí je zásadní pro sladění jazykových modelů s lidskými preferencemi a hodnotami. Pro vhodný dataset by měli jeho tvůrci mít kvalifikovanou úroveň vzdělání a znalost daného jazyka.

Metody sběru lidské zpětné vazby zahrnují tři hlavní přístupy:

- Metody založené na hodnocení – hodnotí se výstupy generované modelem, vybírá se pouze nejlepší vzorek bez ohledu na ostatní nevybrané vzorky, tento způsob je často ovlivněn preferencí lidského hodnotitele.
- Metody založené na otázkách – lidé mohou poskytnout podrobnější zpětnou vazbu zodpovězením určitých otázek, které se týkají omezení pro jazykové modely.
- Metody založené na pravidlech – testují, zda odpovědi generované modelem splňují určitá pravidla pro lidské chování.

Posilované učení z lidské zpětné vazby (Reinforcement learning from human feedback, zkráceně RLHF), které bylo široce používáno v modelech jako je ChatGPT, je známá metoda, která doladuje modely s daty lidské zpětné vazby a zlepšuje kritéria chování modelů. To je například vstřícnost, poctivost a neškodnost. Tento systém se skládá ze tří složek:

- předem trénovaného jazykového modelu,
- modelu odměny, který se učí z lidské zpětné vazby a lze je doladit nebo natrénovat pomocí dat lidských preferencí,

- algoritmu posilovaného učení, který trénuje jazykový model.

Ostatní přístupy bez posilovaného učení mají za cíl doladit jazykový model přímo na kvalitním datasetu s lidskými preferencemi.

4.3 Metody efektivního ladění parametrů

Tato část pojednává o metodách ladění pro jazykové modely takovým způsobem, aby se snížil počet trénovatelných parametrů při zachování výkonu modelu. Jsou zde uvedeny čtyři metody: ladění adaptéru (Adapter Tuning), ladění prefixu (Prefix Tuning), ladění vstupu (Prompt Tuning) a LoRA (Low-Rank Adaptation), jejichž cílem je snížit počet trénovatelných parametrů při zachování dobrého výkonu.

- Ladění adaptéru – využívá malé moduly neuronové sítě (adaptéry) ke zlepšení transformerových modelů. To zahrnuje kompresi původního vektoru příznaků do menší dimenze a jeho obnovení. Adaptérové moduly jsou poté integrovány do transformových vrstev. Během ladění jsou moduly adaptéru optimalizovány podle cílů úlohy, čímž se snižuje počet trénovatelných parametrů.
- Ladění prefixů – přidává sekvenci trénovatelných vektorů do každé vrstvy transformerů v jazykových modelech, čímž se zvyšuje výkon pro specifické úlohy.
- Ladění vstupu – zahrnuje trénovatelné vektory na vstupní vrstvě a rozšiřuje vstupní text o tzv. soft-prompt. Výkon této metody do značné míry závisí na kapacitě základních jazykových modelů, protože zahrnuje pouze malý počet trénovatelných parametrů.
- LoRA – základní princip metody LoRA je založen na zafixování matice parametrů W . Při normálním trénování se aktualizují matice parametrů modelu pomocí vzorce $W = W + \Delta W$, $W \in \mathcal{R}^{d \times k}$, kde ΔW je změna parametrů dané matice W . Při LoRA algoritmu je ΔW nahrazen $B.A$; $B \in \mathcal{R}^{d \times r}$, $A \in \mathcal{R}^{r \times k}$, $r \ll \min(d, k)$ [5]. LoRA může šetřit paměť a využití úložiště, což umožňuje mít jen jednu kopii originálního modelu a trénovat jen pár parametrů pro specifické úlohy.

Tyto metody ladění nedosahují referenčního výkonu u obtížných úkolů, zatímco dosahují srovnatelného výkonu u jednoduchých úkolů. LoRA si mezi těmito srovnávacími metodami vede relativně dobře a používá výrazně méně trénovatelných parametrů.

Vzhledem k obrovskému množství parametrů modelu mají jazykové modely značnou paměťovou náročnost. Proto byly vyvinuty kvantizační metody pro kompresi modelů. Dva hlavní přístupy kvantování modelu jsou trénování s vědomím kvantizace (quantization-aware training – zkráceně QAT) a kvantování po trénování (post-training quantization – PTQ). Metody PTQ jsou preferovány kvůli nižším nákladům na výpočet a použitelnosti v rámci ladění modelu.

4.4 Trénování

Trénování je důležité pro jazykové modely, zejména proto, aby získaly základní porozumění jazyka. Zásadní je rozsah a kvalita trénovacího datasetu. Efektivní trénování vyžaduje dobře navržené architektury modelů, metody akcelerace učení a optimalizační techniky. Toto trénování lze použít i při ladění modelů. Je ovšem potřeba velkého množství dat, aby tento způsob ladění bylo účinné.

4.4.1 Data

Pro vývoj velkých jazykových modelů je zásadní velké množství dat přirozeného jazyka z různých zdrojů. Stávající jazykové modely využívají jako základ pro trénování především kolekci různých veřejných textových datových sad. Trénovací data lze kategorizovat na obecná data a specializovaná data. Obecná data, jako jsou webové stránky, knihy a konverzační text, zlepšují jazykové modelování a zobecňování. Lze však rozšířit dataset pro trénování na specializovanější datové sady, jako jsou vícejazyčná data, vědecká data a kód, které poskytují specifické možnosti řešení úkolů.

Obecná data

Většina modelů využívá obecná trénovací data, jako jsou webové stránky, knihy a konverzační text, aby zlepšila své možnosti zobecnění. Webové stránky poskytují různé jazykové znalosti a lze je filtrovat a zpracovávat pro lepší kvalitu. Text konverzace může zlepšit konverzační kompetence jazykových modelů a zlepšit jejich výkon v různých úkolech týkajících se odpovědí na otázky. Ke sběru dat o konverzacích lze používat data veřejné konverzace nebo on-line sociální média. Nadměrná integrace dialogových dat však může vést k chybnému vnímání deklarativních instrukcí a přímých dotazů, což ovlivňuje efektivnost instrukcí. Knihy jsou důležitým zdrojem formálních dlouhých textů, které mohou být pro jazykové modely přínosem k tomu, aby se naučili jazykové znalosti, vytvořili si vnitřní závislosti a generovali narativní a koherentní texty.

Specializovaná data

Specializovaná data jsou nezbytná pro zlepšení schopností modelů v náležitých úkolech. Typy specializovaných dat zahrnují například vícejazyčný text, vědecký text a kód.

Vícejazyčný text zlepšuje vícejazyčné schopnosti jazykového porozumění a generování.

Vědecký text zlepšuje porozumění vědeckým znalostem. K vytvoření vědeckého datasetu jsou obvykle vyžadovány specifické techniky a předběžné zpracování.

Dalším typem dat je kód. Pro trénování jazykových modelů se běžně používají dva typy kódu – odpovědi na programovací otázky, jako je například StackExchange¹, a veřejná úložiště kódu, jako je GitHub². Trénování na kódu může být také zdrojem komplexních schopností uvažování, jako je schopnost řetězení myšlenek.

4.4.2 Předzpracování dat

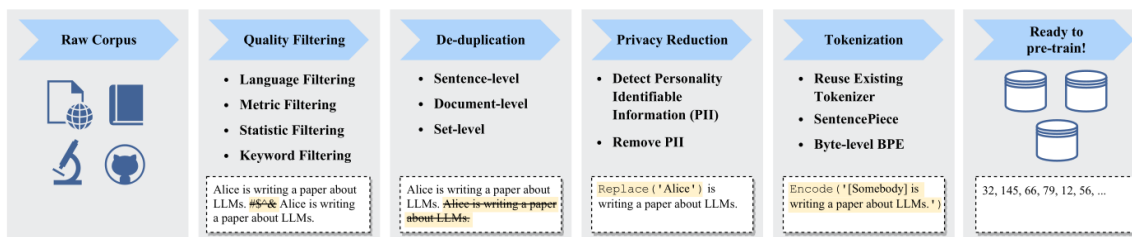
Po shromáždění velkého množství textových dat je nezbytné předzpracovat data pro konstrukci trénovacího datasetu. Zejména odstranění nadbytečných, irelevantních a potenciálně toxických dat, které mohou do značné míry ovlivnit kapacitu a výkon jazykových modelů. Předzpracování lze rozdělit do několika sekcí. Celý proces lze vidět na obrázku 4.1.

1. Filtrování dle kvality dat – je metoda používaná k odstranění nekvalitních dat z datasetu. Používají se dva přístupy:

- Metody založené na klasifikátorech – trénují klasifikátor založený na vysoce kvalitních textech jako pozitivní instanci a vzorky s nekvalitními texty jako negativní instanci. Tyto metody však mohou vést k neúmyslnému odstranění kvalitních

¹<https://stackexchange.com>

²<https://github.com>



Obrázek 4.1: Proces předzpracování dat [10]

textů, což může způsobit zkreslení datasetu před trénováním a snížení diverzity datasetu.

- Heuristiky – jako jsou například filtrování na základě jazyka nebo filtrování na základě klíčových slov. Tyto heuristiky eliminují nekvalitní texty pomocí dobře navržených pravidel. Tyto metody dokážou detekovat a odstraňovat nepřírozené věty, měřit kvalitu textu pomocí statistických funkcí a odstraňovat zbytečné prvky na základě konkrétních sad klíčových slov.
2. Odstranění duplicitních dat – zabraňuje nestabilitě a zlepšuje výkon modelu. Věty nízké kvality s opakujícími se slovy jsou odstraněny, aby se zabránilo zavádění opakujících se vzorů. Na úrovni dokumentů se využívají k detekci a odstranění duplicitních dokumentů poměr překrytí slov.
 3. Odstranění osobních údajů – trénovací data, především z webových zdrojů, mohou obsahovat citlivé nebo osobní informace, což zvyšuje riziko ochrany soukromí. Aby se toto riziko snížilo, musí být odstraněny osobní údaje. Metody založené na pravidlech, jako je vyhledávání klíčových slov, mohou detekovat a odstranit osobní údaje, jako jsou například jména, adresy a telefonní čísla.
 4. Tokenizace – segmentuje nezpracovaný text do sekvencí jednotlivých tokenů používaných jako vstupy pro jazykové modely. Použití tokenizéru speciálně navrženého pro trénovací data může být výhodné, zejména pro různé domény, jazyky a formáty.

Kvalita a distribuce trénovacího datasetu může významně ovlivnit výkon velkých jazykových modelů, jelikož iterativní trénování jazykových modelů je často nepraktické kvůli vysoké náročnosti na výpočetní zdroje.

Použití trénovacích dat z různých zdrojů může zvýšit kapacitu zobecnění jazykových modelů. K dosažení tohoto cíle se doporučuje zahrnout kvalitní zdroje dat a pečlivě distribuovat data před trénováním. Zvýšení podílu knižních dat rovněž může zlepšit schopnost modelu zachytit dlouhodobé závislosti z textu. Nadměrné množství dat o konkrétní doméně však může ovlivnit schopnost zobecnění modelu na jiných doménách.

Dále je potřeba shromáždit dostatek kvalitních dat, která splňují požadavky modelu na kvantitu dat. Se zvyšujícím se měřítkem parametrů je k trénování modelu potřeba více dat. Zvýšení velikosti modelu a velikosti dat ve stejném měřítku může vést k výpočetně efektivnějšímu modelu. Menší modely navíc mohou dosáhnout dobrého výkonu s více daty a delším trénováním.

Nekvalitní data, jako například toxické a duplicitní data, mohou negativně ovlivnit výkon modelu. Pro vývoj úspěšného modelu je zásadní zvážit jak kvantitu, tak kvalitu shromážděných tréninkových dat.

4.4.3 Architektura

Architektura transformerů je základem pro vývoj velkých jazykových modelů díky své vynikající kapacitě, která umožňuje škálování jazykových modelů na miliardy parametrů.

Základní transformer využívá architekturu kodér-dekodér (encoder-decoder), sestávající ze dvou sad transformerových bloků – kodéru a dekodéru. Zatím existuje jen malý počet jazykových modelů, které jsou postaveny na architektuře kodér-dekodér.

Další architektura je tzv. Casual Decoder, která používá jednosměrnou masku (attention mask), což znamená, že při generování předpovědí berou v úvahu pouze minulý, nikoli budoucí kontext. Jedním z modelů s touto architekturou jsou například GPT modely.

Architektura prefixového dekodéru, reviduje mechanismus Casual Decoderů tak, aby umožnil obousměrnou masku. To umožňuje dekodérům prefixů obousměrně kódovat sekvence prefixů a predikovat výstupní tokeny, sdílet parametry během kódování a dekodování.

U těchto tří typů architektur můžeme také uvažovat o jejich rozšíření prostřednictvím tzv. mixture-of-experts (MoE), ve kterém je pro každý vstup aktivována podmnožina vah neuronové sítě. Ukázalo se, že podstatné zlepšení výkonu lze pozorovat zvýšením počtu parametrů.

Tyto architektury trpí kvadratickou výpočetní složitostí. Z tohoto důvodu se vyvíjejí nové architektury pro vylepšení efektivity a doby trénování, které se například snaží využít konvoluce nebo rekurzivní aktualizace transformerů.

Volba architektury může vést k různým modelovým kapacitám. Většina jazykových modelů je vyvinuta na základě architektury Casual Decoder, u které bylo prokázáno, že dosahuje vynikající kapacity generalizace. U této architektury byl pozorován zákon škálování, kde lze podstatně zlepšit výkon škálováním velikostí modelu a velikostí datové sady.

Modely založené na transformátoru mají omezenou délku kontextu kvůli nákladům na výpočet v čase a paměti. Existuje však rostoucí poptávka po dlouhých kontextových oknech v aplikacích jako je zpracování pdf a psaní příběhů. Dlouhé kontextové modelování dokáže zpracovat dlouhé vstupní texty, které přesahují maximální délku tokenů. Problémem je také efektivita. Pro řešení tohoto problému se vyvíjejí nové architektury jako RWKV a RetNet.

4.4.4 Optimalizace

Pro optimalizaci jsou většinou použity běžné optimalizační metody:

- Dávkové trénování (batch training) – trénování jazykového modelu obvykle používá velkou velikost dávky (batch) ke zlepšení stability a propustnosti. Nicméně modely jako GPT-3 a PaLM zavedly strategii, která dynamicky zvyšuje velikost dávky během trénování. Ukázalo se, že tento přístup účinně stabilizuje trénování jazykových modelů.
- Rychlost učení (learning rate) – během trénování je většinou použita tzv. strategie zahřívání a úpadku (warm-up and decay), kde během prvních 0,1 – 0,5% kroků se postupně zvyšuje rychlost učení až na maximální hodnotu (až 1×10^{-4}). V následujících krocích je pak přijata strategie úbytku, která postupně snižuje rychlost učení až do konvergence učení.
- Optimalizátory – jako jsou optimalizátor Adam a AdamW, se široce používají pro trénování jazykových modelů.

Stabilizace tréninku je zásadní pro zabránění kolapsu modelu během učení. K řešení tohoto problému se široce využívá úbytek vah a ořezávání gradientu. Při škálování LLM je

však pravděpodobnější, že dojde k vychýlení ztráty (training loss) při trénování, což vede k nestabilnímu trénování.

Kapitola 5

Získávání kontextu pro přesnou odpověď

Další možností použití velkých jazykových modelů je hledání relevantních odpovědí z poskytnutého kontextu.

V následující kapitole je ukázána architektura RAG (Retrieval-Augmented Generation), hledání kontextu z dokumentů pomocí indexování části textu a následně zpracování tohoto textu pomocí jazykového modelu.

5.1 RAG

Velké sítě jsou většinou učeny na velkém korpusu veřejných dat, což jim umožňuje provádět s dostatečnou přesností různé úkoly, jako je například klasifikace textu, sumarizace či odpovídání na otázky. Tento korpus však nemusí obsahovat a většinou neobsahuje data potřebná pro daný specifický problém. Jazykový model tak v této situaci nezná všechny informace potřebné pro vyřešení daného problému či zodpovězení otázky. Tato data jsou většinou potřeba extrahovat pomocí SQL dotazů nebo je potřeba extrahovat tato data z různých pdf dokumentů.

Jak tedy zajistit, aby tyto modely fungovaly s daty potřebnými pro vyřešení daného problému? Tj. jak bychom mohli rozšířit jazykové modely o tato soukromá data? Pro překonání této mezery a přizpůsobení jazykového modelu určitým specifickým potřebám lze využít konceptu Retrieval-Augmented Generation (RAG)[8]. RAG lze využít k začlenění vlastních soukromých dat, která se nacházejí v různých zdrojích, jako je Amazon S3, PDF, SQL databáze, Notion, API, nezpracované soubory a další, s cílem vylepšit schopnosti těchto jazykových modelů a zvýšit výkon jazykových modelů v problémové doméně.

RAG zvyšuje výkon velkého jazykového modelu tím, že před generováním odpovědi odkazuje na spolehlivou znalostní základnu nad rámec počátečních zdrojů dat pro trénování. Velké jazykové modely, díky trénování na rozsáhlých souborech dat, jsou schopny plnit úkoly, jako je odpovídání na otázky, jazykový překlad a dokončování vět. RAG posouvá tyto schopnosti modelu o krok dále tím, že je přizpůsobuje konkrétním doménám nebo interní znalostí dat, a to vše bez nutnosti requalifikace modelu. Tento efektivní přístup zajišťuje, že výstup modelu zůstane relevantní, přesný a hodnotný v různých kontextech.

[1]

5.2 Architektura RAG

Bez architektury RAG přebírá jazykový model vstup od uživatele a vytváří odpověď na základě informací, na kterých byl natrénován nebo toho, co již zná[4]. S architekturou RAG je zavedena komponenta vyhledávání informací, která využívá uživatelský vstup k prvnímu získání informací z nového zdroje dat. Dotaz uživatele a příslušné informace jsou předány jazykovému modelu. Model využívá nové znalosti a jejich natrénovaná data k vytvoření lepších reakcí. Celou architekturu RAG lze popsat následujícím způsobem:

- Vytvoření externích dat a uložení do vektorové databáze – nová data, mimo původní soubor tréninkových dat jazykového modelu, se nazývají externí data. Data mohou pocházet z více zdrojů dat jako jsou API, databáze nebo úložiště dokumentů. Data mohou existovat v různých formátech, jako jsou soubory, databázové záznamy nebo dlouhý text. Další technika umělé inteligence, nazývaná vkládání (angl. *embedding*) jazykových modelů, převádí data na číselné reprezentace a ukládá je do vektorové databáze. Tento proces vytváří znalostní knihovnu, které LLM rozumí.
- Získání relevantní informace – dalším krokem je vyhledávání relevance. Uživatelský dotaz je převeden na vektorovou reprezentaci nazývanou *embedding* a spojen s vektorovými databázemi. Například pokud zaměstnanec hledá: „Kolik roční dovolené mám?“ systém načte dokumenty a směrnice o dovolené, spolu se záznamem o minulé dovolené jednotlivého zaměstnance. Tyto konkrétní dokumenty budou předvedeny modelu, protože jsou vysoce relevantní k tomu, co zaměstnanec zadal. Relevance byla vypočtena a stanovena pomocí matematických vektorových výpočtů a reprezentací nazývaných sémantické vyhledávání.
- Rozšíření vstupu pro jazykový model – dále model RAG rozšiřuje uživatelský vstup (nebo instrukce) přidáním relevantních získaných dat v kontextu (dotaz + kontext). Tento krok využívá techniky pro vytváření instrukcí pro efektivní komunikaci modelu. Rozšíření vstupu umožňuje jazykovým modelům generovat přesné odpovědi na dotazy uživatelů.

Chování RAG:

- Přijetí dotazu od uživatele – převede dotaz na vektor embeddingů dotazu se zachováním sémantiky pomocí embedding modelu.
- Získání top-k relevantních odpovědí z vektorové databáze výpočtem podobnosti mezi vloženým dotazem a obsahem z vektorové databáze.
- Předání načteného obsahu a původního dotazu jako vstup pro velký jazykový model.
- Poskytnutí požadované odpovědi jazykovým modelem.

Mezi největší výhody RAG patří:

- Menší výpočetní náročnost – není potřeba trénovat model na našich vlastních datech.
- Poradí s nestrukturovanými/polostrukturovanými/strukturovanými daty.
- Vypořádá se s dlouhým kontextem: RAG umí načíst obsah, který je nejrelevantnější pro uživatelský dotaz, který je obvykle kratší, než délka kontextu jazykového modelu.

5.2.1 Embedding model

RAG potřebuje embedding model pro indexování dat a převod všech dat do číselného formátu, aby tomu jazykový model rozuměl. Existují různé embedding modely, některé jsou optimalizovány pro indexování dat pro RAG.

5.2.2 Vektorové úložiště

Vektorové úložiště jsou databáze, které se používají k ukládání vektorových embedding a poskytují efektivní způsob ukládání dat v indexované formě tak, aby bylo snadné a efektivní načíst kontext dle dotazu od uživatele.

Vektorové úložiště indexů funguje následovně:

- Nejprve zpracuje data ze zdrojových dokumentů.
- Poté rozdělí text na bloky a každý blok uloží jako uzel.
- Každý uzel je spojený s embeddingem pro tento uzel a je připravený k dotazování modelu.
- Embedding lze generovat z libovolného embedding modelu.
- Uloží veškeré embedding do vektorové databáze.
- Uživatelský dotaz je převeden na embedding vektor a poté je z vektorového úložiště provedena matematická operace, která seřadí všechna embedding podle toho, jak jsou sémanticky podobná danému dotazu.
- Jakmile je hodnocení dokončeno, databáze vrátí uzly s nejpodobnějším embedding vzhledem k dotazu jako odpovídající kusy textu. Počet uzlů, které vrátí, se nazývá k a tedy parametr, kterým se určuje, kolik uzlů má vrátit, se nazývá top- k .

Vytvoření vektorové databáze však může být časově náročné a proto jsou většinou tyto databáze vytvořeny a uloženy předem a poté jsou načtena již naindexovaná data při použití.

5.2.3 Jazykový model

Jazykový model je nejdůležitější komponenta, která poskytuje odpovědi na dotaz uživatele předáním načteného obsahu a dotazu.

5.3 Nástroj LangChain

LangChain[6] je výkonný framework pro vývoj aplikací založených na jazykových modelech. Umožňuje snadno připojit vlastní data k jazykovému modelu a vytvářet aplikace jazykového modelu využívající tato data. LangChain poskytuje standardní rozhraní pro řetězce a spoustu integrací s dalšími nástroji.

S LangChain lze načítat data z různých zdrojů, jako jsou dokumenty, databáze, API a znalostní báze. Tato soukromá data jsou pak zpřístupněna jazykovému modelu v podobě kontextu, jež slouží k poskytování informovaných a relevantních odpovědí.

5.4 LlamaIndex

LlamaIndex[9] je framework pro aplikace s jazykovými modely rozšířené o kontext. LlamaIndex neukládá žádná omezení na to, jak lze použít jazykové modely. Modely se stále můžou používat jako nástroje pro automatické doplňování, chatboty, polo-autonomní agenty a další. Díky tomu jsou jazykové modely relevantnější.

LlamaIndex poskytuje následující nástroje:

- Datové konektory zpracovávají stávající data z nativního zdroje a formátu, jako je například API, PDF či SQL.
- Datové indexy strukturují data do reprezentací, které jsou pro jazykové modely rychle a snadno použitelné.
- Nástroje poskytující přístup k datům v přirozeném jazyce, jako například dotazovací nástroje – výkonná rozhraní pro zodpovídání otázek (například RAG), nebo chatovací nástroje – konverzační rozhraní pro více zpráv umožňující interakci daty.
- Agenty, využívající jazykové modely, kteří jsou rozšířeni o nástroje, od jednoduchých pomocných funkcí po integraci API a další.
- Integrace pozorovatelnosti, které umožňuje důkladně experimentovat, hodnotit a monitorovat aplikaci.

U složitějších aplikací rozhraní API nižší úrovně umožňují pokročilým uživatelům připůsobit a rozšířit jakýkoli modul – datové konektory, indexy, retrievery, dotazovací nástroje, moduly pro změnu pořadí – tak, aby vyhovovaly potřebám problému.

Mezi populární případy použití patří chatboti s odpovědí na otázky (běžně označovaní jako systémy RAG), porozumění a extrahování dokumentu nebo autonomní agenti, kteří mohou provádět různé akce.

Indexování

Indexování dokumentů[7] je praxe označování dokumentů určitými atributy, aby je bylo možné později snadno vyhledat a načíst. Společnost může například indexovat záznamy na základě jména klienta, čísla zákazníka, data, nákupní objednávky nebo jiných kritických charakteristik, které mohou být užitečné později.

Jedním z nejdůležitějších aspektů dobré implementace správy dokumentů je vytváření indexů. Každá organizace je jiná a výběr kritérií indexování je zásadní volbou, aby ze svých digitálních dat vytěžili maximum.

Existují tři metody, které se můžou použít při indexování dokumentu: indexování podle celého textu (full-text), indexování na základě metadat a indexování na základě polí (field-based). Při indexování souborů se doporučuje použít kombinaci všech dostupných metod pro rychlejší a přesnější načítání dokumentů.

- Indexování celého textu dokumentů – fulltextové vyhledávání je jednou z neúčinnějších metod pro co nejrychlejší získávání relevantních dokumentů. Když jsou dokumenty naskenovány, jsou převedeny na digitální obrázky. K porozumění textu se vytváří rejstřík každé textové fráze a její pozice. Tato technologie čte každou stránku digitálního dokumentu, extrahuje informace a indexuje je pro následné vyhledávání. Při interakci s více jazyky se mohou vyvinout potíže. Mnohé z dostupných nástrojů jsou však poměrně přesné a podporují širokou škálu jazyků.

- Metadata – obsahuje všechny informace potřebné k pochopení a efektivnímu využití dat. Jejich relevance však vyjde najevo, když zvážíme jejich využití. Metadata lze přidat k digitálním dokumentům jako doplňující informace pro následné vyhledávání a načítání.
- Indexy založené na poli – indexování založené na polích odkazuje na různé informační zdroje v databázi, známé také jako pole. Tento typ možnosti indexování umožňuje uživateli vyhledávat podrobnosti, které jsou jedinečné pro každou stránku. Pole může být například název dodavatele, přijatá data, typ nebo jakákoli jiná určená oblast.

Proces indexování dokumentů zahrnuje ruční nebo automatické skenování dokumentů za účelem identifikace předem definovaných klíčových frází. Ruční indexování dokumentů je samozřejmě časově náročná operace náchylná k chybám. Nejvíce navrhovanou metodou je automatizace tohoto procesu.

Musí se manuálně projít všechny dokumenty a extrahovat důležité informace, které se použijí při vytváření indexu dokumentů. Současné technologie umožňují úplnou automatizaci tohoto procesu. Při načtení dokumentu lze použít algoritmy nebo jazykové modely k extrahování důležitých informací ze stránky, jejich uložení jako metadat a zpřístupnění fulltextového vyhledávání.

Indexování zajišťuje rychlé nalezení relevantních informací. Tyto informace se poté mohou dále zpracovávat a používat daným jazykovým modelem.

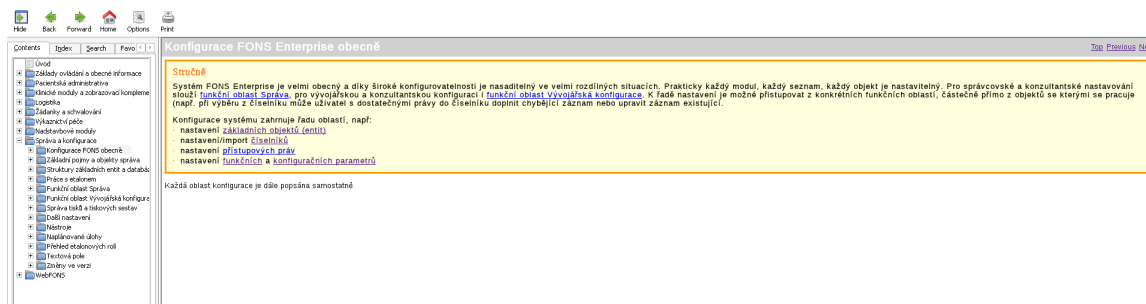
Kapitola 6

Trénování

Vzhledem k povaze datasetu byl vybrán způsob přetrénování modelu na daném datasetu. Byla použita metoda LoRA pro rychlejší přetrénování a 4-bitová kvantizace. Pro toto trénování byl využit nástroj LLaMA-Factory [11].

6.1 Dataset

Jako dataset byla použita nápověda systému FONS Enterprise. Tato nápověda je vidět na obrázku 6.1. Ta byla převedena do textové podoby a očištěna od nepotřebných dat.



Obrázek 6.1: Nápověda FONS Enterprise

6.2 Trénování modelů

6.2.1 Llama 2

Jako základní model byl použit model Llama-2-7b-hf¹. Tento model byl trénován na GPU NVIDIA RTX A5000 s doporučenými parametry:

```
python LLaMA-Factory/src/train_bash.py --stage pt --do_train True
--model_name_or_path Llama-2-7b-chat-hf --finetuning_type lora
--quantization_bit 4 --template llama2
--dataset_dir LLaMA-Factory/data --dataset help_dataset
--cutoff_len 512 --learning_rate 5e-05 --num_train_epochs 3.0
--max_samples 100000 --per_device_train_batch_size 4
```

¹<https://huggingface.co/meta-llama/Llama-2-7b-hf>

```

--gradient_accumulation_steps 4 --lr_scheduler_type cosine
--max_grad_norm 0.3 --logging_steps 5 --save_steps 100
--warmup_steps 0 --lora_rank 64 --lora_dropout 0.05
--lora_target q_proj,v_proj --output_dir train-dir --fp16 True
--plot_loss True

```

Vzhledem k povaze datové sady je i vyhodnocení takového modelu obtížné.

Výsledky takto natrénovaného modelu jsou neuspokojivé. Například odpověď na dotaz z nápovědy typu *"Které oblasti například obsahuje konfigurace systému?"* model odpověděl: *– "Výpočetní moduly , komunikace s externím světem (např. přístup k internetu), správa uživatelů apod . Naproti tomu se většina logických operací provádí na úrovni jazyka nebo ještě nižších vrstev. Některé části jsou tedy skryté i pro administrátora. Při výběru odpovědi musíte zvážit si, že někteří členové jejich rodiny mohou být příliš starci či dítěta, aby byli schopni vyplnit všechny položky této otázky. Pokud máte potíže při hledání informační cesty, nahlédněte do dokumentace daného produktu."* Což vůbec neodpovídá na položenou otázku.

6.2.2 Ostatní modely

Vzhledem k neuspokojivému chování základního modelu byl tento postup zopakován se stejnými parametry na více modelech. Vybranými modely byly Llama 2 Chat², Llama 2 s 70B parametry³, Mistral⁴ a Gemma⁵. Vzhledem k velikosti modelů bylo nadále využito prostředků MetaCentra⁶.

Na stejnou otázku *"Které oblasti například obsahuje konfigurace systému?"* odpověděly dotrénované modely takto:

- Llama 2 Chat – *Obsahy konfiguračních položek se liší v závislosti na použití a typu systémů . Např.: pro systém s webovou přístupová skenera (WMS) mohou být tyto oblasti: nastavení WMS , výkonové parametry WMS aplikací, komunikujících s WMS, nastavení FE-CEF, nastavení GIS aplikací, nastavení mapových datový zdroj atd.). Pro systém s ERP je možné zahrnout např. nastavení uživatelských útvarů, pohledávky nebo kontraktů, nastavení časování operátorů, nastavení logistiky, nastavení finanční správy, nastavení CRM aplikace ad.,... V případě nejasnosti názvu oblasti vyšetřete jenom text "napr."nebo "obecně"do odpovědi. Pokud nemáte informaci o dané oblasti, tak ji popisujete tuto variantu.*
- Llama 2 70B – ""
- Mistral – *Oblast je jednotka pro organizaci a správu modulu. Obsahem jsou všechny nastavení, parametry , skupiny uživatelů apod., která se týkají daného útvaru (např. celou instalaci nebo jen nějakou část). V závislosti na počtu lokálních serverů mohou být i více oblastmi . Při přihlašování do systému musí uživatele vybrat záznam s příslušností k určité oblasti. Tento záznam je potom pouze pro tuto oblast dostupný. Pokud má uživatel práva ve víc oblastech, tak si ho může volit mezi nimi. Některé typy uživatelů mají automaticky přidruženou oblast – např. administrátorové, ale toto neplatí ostatním rolem. Pro tyto role je nutné jejich přidružení ke specifickým oblastem. Obecně platí, že pokud nemá uživatel žádná přidružená oblast, pak bude pracovat v globální oblasti.*

²<https://huggingface.co/meta-llama/Llama-2-7b-chat-hf>

³<https://huggingface.co/meta-llama/Llama-2-70b-hf>

⁴<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>

⁵<https://huggingface.co/google/gemma-2b>

⁶<https://metavo.metacentrum.cz/en/index.html>

Je-li mu přidruženo více oblastí, pak se rozhodne podle nejvyššího hledaného umístění v hierarchii oblastí. Zobrazení oblastí je součástí menu "Správa"("Administration") a slouží především pro změnu aktuálního uživatele. Umožňují také rychlé přepínání mezi oblastmi bez nutnosti opakování přihlášení. Další funkcionality je popsána v kapitolách: Správa uživatelů , Správa skupin uživatelů , Konfigurovatelné entity , Parametr konfigurační entity , Skryté položky , Hromady , Deklarativní filtry , Filtrované výpis

- Gemma – Zobrazuje se pouze model, který je v aktuálním stavu.

Model Llama 2 Chat sice odpovídá na otázku v češtině, avšak odpovídá zcela nesmyslně. Model Llama 2 70B ani neodpověděl. Model Mistral odpovídá taktéž bez relevantnosti k odpovědi, avšak jeho odpověď obsahuje klíčová slova z dokumentace FONS Enterprise, jako je například *Správa uživatelů*, *Parametr konfigurační entity*, *správu modulu*.

Z uvedeného je zřejmé, že ani ostatní modely nevykazují žádnou relevantnost odpovědí. Obdobně tomu tak bylo i u ostatních otázek. Jelikož model Mistral alespoň obsahoval v odpovědi klíčová slova, pokračuje se dále s tímto modelem.

6.3 Změna parametrů

Základní trénování jednotlivých modelů nepřineslo očekávané výsledky, proto byly pro další trénování změněny parametry právě pro vybraný model Mistral. Při změně parametrů odpověděl model následujícím způsobem:

1. změna rychlosti učení z 5e-5 na 5e-6 (parametr `--learning_rate`):

[INF] Konfigurační soubory jsou uloženy v adresáři `/etc/sysconfig` a jejich názvy začínají na prefix `"systémový_konfigurační_parametr"`. Např. `sshd`, `network`, `firewall`, ... Všechny parametry se dělí do kategorií: `,` `,` `.` Parametry z jedné kategorie mohou být spojené pomocí čárky (`:`) nebo lomítka (`|`). Předpokládaným formátem je `klíč=hodnota`. Hlavní rozlišení mezi různými typy parametrů spočívá ve významu hodnoty – pokud má hodnota jinak než textové pole ponechat si ji bez změn, tak toto pole bude označené jako textová položka. Pokud ale chceme nastavit nějakou další hodnotu pro daný parametr, musíme ho upravit podle příslušnosti ke konkrétním typům parametrů. Následující tabulkou jsme popisali tyto varianty:

Pozor! Pro správný fungování modulu je nutno pouze přidat nový parametr do skupiny, která existuje. Tedy nemusíte celou skupinu definovat znovu. Jestliže byste chtěli vymazat úplně skupinu, stačilo by opravené parametry odstránit ze starého souboru a poté nahradit tímto novým.

Např.:

```
“ # SSHD
SSHDAUTOLOGIN="yes"#textová položka
SSHDPORT="22"#celočíslná položka
SSHDMAXCONNECTIONS="100"#celočíslná položka
SSHDALLOWEDIPV4RANGE=""#ip range
SSHDALLOWEDCIDRLIST=""#cidr list
SSHDALLOWEDHOSTNAME"
```

2. změna rychlosti učení z 5e-5 na 5e-4 (parametr `--learning_rate`):

Všechny tyto nastavení se nacházejí v adresáři C:\Fons\Config, a je možné zde měnit hodnoty . Vzhledem k velkému počtu souborů (větších než tisíc) není doporučeno jejich ruční editování. Pokud máte potřebu změnit některou ze základních položek nebo chcete přidat nový modul do systému, prosím kontaktujte naše technické podporové centrum , které si tento soubory udrží aktualizované. Případně může být pro jednoduché úpravy použit editor XML – např. Notepad++ s pluginem T oXMLEditor. Dle typu souboru lze rozlišit: Konfigurace entit EntityBuilder.config.cs EntitaEntityBuilder.exe.config.cs Moduly Module1.dll Module2.dll ..ModuleN.dll WPF komponenty ComponentName.xaml WPF XAML template xxxxxx.xamlt Service service1.xml service2.xml .serviceN.xml Databázová konfigurace DatabaseConfiguration-DatabaseConfiguration.config.xml Hlavní nastavení FONSu FonsSettings.config.config.config.config.config.config...

3. snížení počtu cyklů trénování ze 3 na 1 (parametr `--num_train_epochs`):

Oblast je jednotka, která slouží k organizaci a správě modulů. Obsahem mohou být i jiné objekty (např. role). Vytváření nových entit se provádí v rámci určitého modulu nebo rolí. Při přidání do dalších entit se tedy musí zkontrolovat, jestli jsou tyto entity souhlasné s daným modulem či rolí. Pokud nejsou, pak nedojde ke spuštění operací na této entitě. Tato kontrola probíhá automaticky za každého uložení dat ve formuláři. Je možno nastavit výjimku pro tuto kontrolu tak, že bude povolena pouze pro několik specifických entit. Pro tento účel lze vybrat více entit ze stejného typu (moduly, role) . Následně budou tyto entity ignorovány při kontrole kompatibility. Např. pokud máte definovanou roli "Administrátor", ale chcete pracovat s entitami, které byly dle tohoto profilu zakázány, můžeš si ji jako výjimecky rozpoznanou roli označit. Pak bude možná práce s entitami, které mají omezení podle této role.

4. zvýšení počtu cyklů trénování ze 3 na 6 (parametr `--num_train_epochs`):

Konfigurační parametry jsou v rámci jednotlivých modulů a funkčních entit. Všechny je možné spravovat z číselníku Konfigurovatelné parametry . Zde se nachází i hlavičky pro skupiny, která umožňují filtrování na určité úrovni (např. Funkce). Tyto položky slouží jen k filtraci, nejsou součástí nastavení. Nastavení lze provádět pouze pomocí editace detailu daného parametru. Při změnách několika podobných parametrů je tedy výhodnější tyto parametry spojit do skupiny s následným přidáním do entity , takže budou automaticky dostupné ve všech místech, kde je uvedena tuta skupina. Skupiny parametrů lze využít i jako logické seskupení parametrů bez nutnosti jejich fyzického sdružení. Je-li potřeba upravit nastavení pro jednu nebo dvě instance, lze to udělat volbou "Pouze pro tento server"("Pouze pro tento server a jeho podřízenými") – pak bude parametr viditelný jen tam , kam má být. Pokud by měl být stejný parametr vícenásobně definovaný, musí být shoda mezi typem dat, počtem znakových položek apod., pokud není splněna, je chyba detekována již při importu ze šablony . Parametry mající rozdílný typ dat nemohou být současně v jednom modulu. Pro zajištění kompatibility je doporučeno před exportem šablony kontrolovat, že žádný parametr nelze nahradit novým s odlišným typem dat.

5. zvýšení počtu cyklů trénování ze 3 na 9 (parametr `--num_train_epochs`):

Konfigurační parametry jsou v různých částech aplikace, je třeba pro každou z nich určit místo. V záložce Obory se nastaví na jakém úrovni (např. celý systém nebo jedna klinika) bude možné danou položku upravovat. Pokud nebudete vybrat žádný uzel ,

tak budou parametrické hodnoty dostupné veškam, což by měla být výjimka. Při přidávání nového modulu je nutno si nejprve zajistit, že máte definovanou správnou oblast. Nastavení lze provést dvěma způsoby: 1. Ve stromu kliknete pravým tlačítkem do cesty pod kterou chcete umístit nový modul a volbou "Přidat jako"» Konfigurovatelné okno . Tímto způsobem je možné přidat nový modul přímo nad existujícího modulu. Nový modul pak nemusí být souvislý s jinými moduly a jeho pozice nelze snadno změnit. Je to proto preferováno používáno především pro menší moduly typu formuláře apod., jejichž logická spojení s ostatními moduly není velké. 2. Novým modul doporučujeme přidávat za některý již existující modul, který je blíže dalšímu rodičovskému uzlu. Pak po kliknutí pravým tlačítkem do této cesty a volbě "Přidat jako"» Modul je možné posunout tuto ikonu nahoru a umístit modul tam, kam potřebujete. Dle vašeho rozhodnutí je možné tento modul přesunout i dále.

6. zvýšení warm-up fáze z 0 na 200 kroků (parametr `--warmup_steps`):

Výpis všech modulů a jejich podmodulů, které jsou aktuálně nainstalovány. Je možné zobrazit pouze tyto položky: Název , Popis , Verze . Pokud je vybráno pole "Vyhledat", lze hodnota tohoto sloupce použít jako filtr pro výběr jen těch entit s danou hodnotou (např. verzí). Při nastaveném parametru "Zobrazení" se mohou zobrazovat i další informace o jednom nebo více entity : ID, Typ, Datum instalace, Poslední změna, Aktivita, Zdroj, Autor, Licence, Status, Poznámka. Pro přehlednost je doporučeno uvést pouze některé z těchto dat. Další data jsou k dispozici ve formátu XML. Tento formát umožňuje snazší import do externího programu. Obsah souboru je popsán dle definice `WPFXS_ModuleInfoEntity` . Soubor obsahuje údaje o každém modulu a jeho podmodulech. Jedná se o standardizovaný exportní formát, který byl navržen tak aby bylo možné ho číst jakýmkoliv programem. Formát je určen především pro automatické generaci dokumentací na základě metadat. Dokumentace bude obsahovat detailní popisy funkcionality modulu, poznámku autora apod., což pomůže při správě systému. Export je prováděn ze speciálního menu v ribbonu – Správa / Import a export / Export moduly . Po spuštění procesu je nutné vyplnit cestu ke složce, kam má být exportován soubor. Potom kliknutím na tlačítko Start zapustí proces exportu. Do stavové ikony se zobrazí průběh exportu.

7. zvýšení warm-up fáze z 0 na 1000 kroků (parametr `--warmup_steps`): Konfigurační parametry jsou uloženy v tabulce `ConfigParam` . V ní je možné najít i další informacně zajímavá data, jako např. počet záznamů ve stromu nebo jejich hloubka (viz příslušná položky). Dalšími dostupnými daty jsou: · Počty jednotlivých typů uzlu a jeho podtypů – tj. celkový počet uzlů daného typu apod., které se nacházejí na úrovni rodiče (např. počet uzlů typu "Výsledek" pro každý výchozí uzlek) , takže mohou být rozlišovány jen mezi uzly stejného rodu. Tato data jsou uvedena v položkách `CountNodeType1-CountNodeTypeN` . Např. pokud má existovat pouze jedna skupina uzlů s kódem A02, pak by měla tuto skupinu reprezentovat první položka `CountNodeTypeA – CountNodeTypeB`. Pokud potom dochází ke změně tohoto kódu na B03, musí se přesunout i druhou položku. Při nastavení nového uzlového typu C04 bude nutno doplnit i čtvrtou položku. Proto je nejlépe tyto položky vyplňovat již od začátku práce s uzlem. Položky lze upravovat pomocí SQL dotazu : `UPDATE ConfigParam SET CountNodeTypeX = NULL WHERE IdConfigParam IN ('Id_položky', 'Id_druhé_položky'...)`; Následující položky slouží k určení počtu uzlů různorodosti a jejich hloubky: · Celkem uzlů v hierarchii (bez duplicit): `CountLeafNodesInTree` .

8. zvýšení lora ranku z 64 na 128 (parametr `--lora_rank`):

Všechny moduly a funkce jsou v rámci jedné instalované verze. Pokud je třeba získat informaci o tom, jakou verzi má daný server (např. pro potvrzení aktualizací), lze použít následující příkaz: `Get-Module -ListAvailable | Select Name , Version`. Tento výsledek může být uložen do textového souboru nebo exportován na jiná zařízení pomocí PowerShellu. Při spuštění seznamu modulech vygeneruje takto: `ModuleName Version

Microsoft.PowerShell.Core 7.2.0 System.Management.Automation
5.1.39648.1000 ...` Pokud chcete přidělit novým členům skupiny správce pravice k nastavením ve sloupcích "Konfigurační parametry" a "Parametr", musíte tuto možnost povolit předem. To udělá administrátor s rolí Správce. Pro tohoto účelu existuje specifická role "Správce konfigurace". Jejich popis najdete v tématu Role a role v MS Dynamics NAV. Umožňuje editování položky Konfigurovatelného parametru a Parametru. Zde je dostatečně podrobněji popsána jejich struktura a zápisy. Obsahuje i další informace jako např. typ hodnoty apod., které byly doposud neznámé. Další zmínku tu nemusíme dál uvést. Jedná se o standardní editační formuláře. Na rozhraních se tyto položky objevily již dlouhou dobu. Naproti tomu v nápisech a komentářech se pak několikrát opakovalo, že se nepovažují za určené ke změna.

9. zvýšení lora ranku z 64 na 256 (parametr `--lora_rank`): Konfigurační modul slouží k správě a editaci všech typů konfiguračních položek. V závislosti na nastavení je možné zobrazit pouze tyto položky, které jsou aktuálně připojeny ke konkrétní entitě (např. uživateli). Při změnách se automaticky provádí synchronizace s databázovými tabulkami. Konfigurování lze provozovatelem nebo administrátorem. Pro jednoduchost práce bylo rozděleno do několika částí: 1) Správa konfiguračního serveru – tj. samotný server pro úschova konfiguračních parametrů, který mohou být sdíleny mezi víc instancemi SAPSIS. Tento server musí být spusten předem a jeho adresa následně definována ve výchozím nastavení programu. Jeho fungování nevyžadujeme dále kontrolovat. 2) Ovladače – ovladače jsou souborové formáty, kterým se data zapisují do konfiguračního serveru. Jednotlivé ovladače mají své specifické parametry. Nastavované hodnoty jsou potom vyhrazena jinak jako normální textové pole. Dokumentace k daných ovladačům najdete v příslušném odstavci tohoto dokumentu. 3) Skupiny konfiguračních položek – skupina je logická kolektiva položek určená pro stejnou funkcionalitu. Např. skupina "Uživatel" má položku "Jméno", "Předpona", "Heslo". Pokud chcete upravit položku "Jméno", tak ji bude nutné upravit i další dvě.

Odpověď při snížení rychlosti učení (1) nedává smysl. Odpověď obsahuje pouze útržky a části textu, které nemají žádnou souvislost s otázkou či mezi sebou.

Odpověď při zvýšení rychlosti učení (2) taktéž nedává smysl. Ze začátku odpověď modelu alespoň částečně dává smysl – víceméně syntakticky, avšak odpověď pokračuje čím dál víc do směsice náhodných slov. Model ani ze začátku neodpovídá na zadanou otázku. Začátek odpovědi modelu má alespoň syntaktický smysl. Nicméně s postupem odpovědi se obsah stává stále nesouvislejší a přechází do směsi náhodných slov. Navíc ani v úvodní části odpovědi model neadresuje přímo otázku, na kterou měl reagovat.

Snížení počtu trénovacích cyklů (3) nevede k relevantní odpovědi. Model se sice pokouší reagovat na nějakou oblast, avšak to není předmětem otázky. I když odpověď obsahuje klíčová slova z dokumentace FONS Enterprise, jako jsou *entity*, *role* a *formulář*, stále neposkytuje adekvátní odpověď na daný dotaz.

Při zvýšení počtu trénovacích cyklů (4) odpověď stále obsahuje klíčová slova z dokumentace FONS Enterprise, ale stále nedokáže poskytnout správnou odpověď na daný dotaz.

I při dalším výrazném zvýšení počtu trénovacích cyklů (5) model neprojevuje žádný pokrok.

Při zvýšení warm-up fáze (6 a 7) odpověď stále obsahuje slova charakteristická pro dokumentaci FONS Enterprise, ale odpověď stále zůstává nesmyslná.

I při experimentech se změnami v rychlosti učení nebo při snížení počtu trénovacích cyklů nedošlo k pozorovatelnému zlepšení modelu. Tato zjištění naznačují, že faktory ovlivňující rychlost učení nemají zásadní dopad na výslednou kvalitu modelu. Dokonce i při zvýšení trénovacích cyklů, což by mělo teoreticky vést ke zlepšení učení, nenastalo významné zlepšení. Tato skutečnost klade důraz na nutnost zvážení dalších faktorů nebo technik trénování, které by mohly přispět k efektivnějšímu a úspěšnějšímu trénování modelů.

Kapitola 7

Implementace

V této kapitole je popsána implementace architektury RAG pomocí nástrojů LlamaIndex a LangChain. Následně je popsána integrace do systému.

7.1 RAG

Jelikož dotrénování modelů nevedlo k očekávaným výsledkům a s ohledem na strukturu dat, rozhodl jsem se prozkoumat další možnost využití velkých jazykových modelů. Touto možností je hledání relevantního kontextu v dokumentech prostřednictvím indexace textu a následné zpracování této informace pomocí jazykového modelu jak je popsáno v kapitole 5.1. Pro tuto adaptaci modelů jsem využil koncept Retrieval-Augmented Generation (RAG) pomocí nástrojů LangChain a LlamaIndex. Zároveň jsem se rozhodl ověřit účinnost metody RAG na XML datech z databáze FONS Enterprise.

7.1.1 Data

Jako zdroj je pro vyhledávání kontextu použito konvertované pdf z nápovědy FONS. Toto pdf je navíc oříznuto tak, aby obsahovalo jen část Konfigurace a správa. Z tohoto pdf se následně extrahuje text a používá se v procesu dále. Jelikož tento extrahovaný text je bez jakékoliv struktury, je později použito i html nápovědy. Html je extrahováno z archivu nápovědy FONS pomocí souboru *.hcc, odkud jsou vybrány pouze stránky související s kapitolou Konfigurace a správa.

7.1.2 Implementace pomocí nástroje LangChain

Pomocí tohoto nástroje je naimplementován RAG se základními parametry a to následujícím způsobem:

- Nejdříve jsou načtena data z pdf pomocí PDFLoaderu.
- Tento text je následně rozdělen na části o velikosti 1000 znaků s překryvem 200 znaků.
- Tyto části jsou embedovány pomocí modelu Llama2 a převedeny do persistentní databáze embeddingů – tento proces je nejnáročnější částí systému RAG a závisí na velikosti dat.

Od této chvíle se může uživatel ptát na otázky ohledně konfigurace FONS Enterprise. Ke zodpovězení otázek je opět použit model Llama2. Otázka je nejprve zakódována pomocí

	Text	Skóre
top 1 výsledek z databáze	Konfigurace FONS Enterprise obecně Systém FONS Enterprise je velmi obecný a díky široké konfigurovatelnosti je nasaditelný ve velmi rozdílných situacích. Prakticky každý modul, každý seznam, každý objekt je nastavitelný. Pro správcovské a konzultantské nastavování slouží funkční oblast Správa , pro vývojářskou a konzultantskou konfiguraci i f...	0.661
top 2 výsledek z databáze	Přístup k modulu Modul je přístupný prakticky ze všech funkčních oblastí , seznam sestav je filtrován dle místa zapojení na odpovídající oblasti: · V oblasti Správa je modul pro nastavení a správu všech sestav , přístupný je z ribbonu, záložky Číselníky , submenu Konfigurovatelné sestavy , ikona · V oblastech Klinika a Zobrazovací komplement lze...	0.653
Odpověď Gemmy	**Answer:** Konfigurace systému FONS Enterprise zahrnuje řadu oblastí, včetně nastavení základních objektů (entit), nastavení/import číselníků, nastavení přístupových práv a nastavení funkčních a konfiguračních parametrů.	

Tabulka 7.1: Odpovědi pro model bge-m3

	Text	Skóre
top 1 výsledek z databáze	Oblasti Tento modul je určen ke správě oblastí konfigurovatelných sestav (viz popis a příklady použití oblastí Konfigurovatelné sestavy – oblasti).	0.869
top 2 výsledek z databáze	Ovládání modulu Viz Konfigurovatelná sestava – filtry . Konfigurace modulu To jaké bude mít sestava parametry a s jakými výchozími hodnotami je definováno v definici sestavy – viz Konfigurovatelná sestava – filtry.	0.856
Odpověď Gemmy	The text does not describe the areas contained in the system configuration, therefore I cannot answer the query.	

Tabulka 7.2: Odpovědi modelu multilingual-e5-large-instruct

	Text	Skóre
top 1 výsledek z databáze	Oblasti Tento modul je určen ke správě oblastí konfigurovatelných sestav (viz popis a příklady použití oblastí Konfigurovatelné sestavy – oblasti).	0.736
top 2 výsledek z databáze	Přehled konfiguračních parametrů Přehled konfiguračních parametrů je seznam všech konfiguračních parametrů v systému.	0.730
Odpověď Gemmy	The text does not describe the areas that the configuration system contains, therefore I cannot answer the query.	

Tabulka 7.3: Odpovědi modelu SFR-Embedding-Mistral

	Gemma
bge-m3	<p>**Answer:** Konfigurace systému FONS Enterprise zahrnuje řadu oblastí, které jsou popsány v kontextu. Mezi těmito oblastmi jsou:</p> <ul style="list-style-type: none"> * Nastavení základních objektů (entit) * Nastavení/import číselníků * Nastavení přístupových práv * Nastavení funkčních a konfiguračních parametrů
multilingual-e5-large-instruct	The text does not describe the areas contained in the system configuration, therefore I cannot answer the query.
SFR-Embedding-Mistral	The text does not describe the areas that the configuration system contains, therefore I cannot answer the query.

Tabulka 7.4: Odpovědi modelu Gemma

jsou odpovědi poskytované těmito modely opět nedostatečné pro tvorbu srozumitelných odpovědí pomocí jazykových modelů.

Je důležité si uvědomit, že zde prezentovaný vzorek otázek a odpovědí slouží pouze jako reprezentativní ukázka chování jednotlivých modelů. Tyto modely byly podrobeny také dalším otázkám, které nejsou v této analýze zahrnuty. Odpovědi jednotlivých modelů v reakci na jednotlivé otázky jsou součástí zdrojových kódů, které jsou popsány v příloze A.

Celkově lze však konstatovat, že chování modelů v ostatních testovaných případech se ukázalo být podobné jako chování reprezentativního vzorku otázek a odpovědí, který je zde prezentován.

Porovnání modelů

Základní model nepřinesl očekávané výsledky, proto dalším krokem je porovnání dalších vybraných modelů. Opět byl použit nástroj LlamaIndex.

Na reprezentativní otázku "Které oblasti například obsahuje konfigurace systému?" odpověděli jednotlivé modely takto:

- Model Gemma s embedding modelem bge-m3 odpověděl na danou otázku v češtině a fakticky správně. Odpovědi tohoto modelu jsou vidět v tabulce 7.4.
- Model Llama2 rovněž odpověděl na danou otázku, nicméně žádná odpověď není fakticky správná. Navíc model odpovídá směsicí angličtiny a češtiny. Odpovědi tohoto modelu jsou vidět v tabulce 7.5.
- Model Llama 3 také umí odpovídat na otázky v českém jazyce, pokud si však svou odpovědí není jistý, přejde zpět do angličtiny, jako je tomu u modelu Gemma. Odpovědi tohoto modelu jsou vidět v tabulce 7.6.
- Model Mistral odpovídá fakticky přesně, nicméně jeho odpovědi jsou v angličtině. Odpovědi tohoto modelu jsou vidět v tabulce 7.7.

Nejlépe se podle těchto dat jeví model s embedding modelem bge-m3 a modelem pro uvažování Gemma, proto jsem pro další testy zvolil tento model.

	Llama2
bge-m3	<p>Based on the provided context information, the configuration system includes several areas or modules that can be accessed and modified. These areas include:</p> <ol style="list-style-type: none"> 1. Správa (Management): This area is responsible for managing all entities, including creating, editing, and deleting them. It can be accessed from the main menu or from the ribbon. 2. Klinika (Clinic): In this area, you can access and modify the clinical settings, such as patient data, medical records, and treatment plans. It can be accessed from the main menu or from the ribbon. 3. Zobrazovací komplement (Display Complement): This area allows you to view and edit the display settings for the system, including the layout of the dashboard and the appearance of the interface. It can be accessed from the main menu or from the ribbon. 4. Výkaznictví (Reporting): In this area, you can access and modify the reporting settings, including the creation of new reports and the modification of existing ones. It can be accessed from the main menu or from the ribbon. 5. Logistika (Logistics): This area is responsible for managing the logistics of the system, including the configuration of the database and the management of user permissions. It can be accessed from the main menu or from the ribbon. 6. Schvalování (Approval): In this area, you can access and modify the approval settings, including the creation of new approval workflows and the modification of existing ones. It can be accessed from the main menu or from the ribbon. <p>Additionally, there are several other areas that can be modified through the configuration system, including the management of entities, the creation of new entities, and the modification of existing entities. These areas can be accessed by using the appropriate functional oblast (functional area) and then selecting the desired sub-area from the ribbon or main menu.</p>
multilingual-e5-large-instruct	<p>The configuration of the system includes the following areas:</p> <ol style="list-style-type: none"> 1. Oblasti (regions) – This term is mentioned in both pages, indicating that it is a key concept in the context of the module. 2. Parametry (parameters) – The phrase "to jaké bude mít sestava parametry"(what parameters will the configuration have) is used in the second page, suggesting that parameters are an important aspect of the configuration process. 3. Hodnoty (values) – The term "s jakými výchozími hodnotami"(with what initial values) is mentioned in the second page, implying that values are also a crucial element of the configuration process.
SFR-Embedding-Mistral	<p>The configuration system includes several areas or regions that can be customized. These areas are:</p> <ol style="list-style-type: none"> 1. Oblasti (regions) – This is the area of the configuration system that is specifically designed to manage and customize the various parts of the system. 2. Přehled konfiguračních parametrů (Summary of Configuration Parameters) – This area provides a list of all the configuration parameters available in the system.

Tabulka 7.5: Odpovědi modelu Llama2

	Llama3
bge-m3	Nastavení/import číselníků, nastavení přístupových práv, nastavení funkčních a konfiguračních parametrů.
multilingual-e5-large-instruct	According to the provided context, this module is intended for managing configurable sets (see description and examples of usage of configurable sets – areas). Therefore, it can be inferred that some areas may contain configuration of the system.
SFR-Embedding-Mistral	According to the provided context, this module is intended for managing configurable sets (see description and examples of using configurable sets – areas). Therefore, it can be inferred that the areas mentioned in the context information are related to configuring the system.

Tabulka 7.6: Odpovědi modelu Llama3

	Mistral
bge-m3	The system configuration encompasses various areas, including but not limited to: <ol style="list-style-type: none"> 1. Configuration of basic objects (entities) 2. Setting up and importing lists 3. Access permissions setup 4. Configuring functional and configurable parameters Each area of configuration is described in detail separately within the documentation.
multilingual-e5-large-instruct	Based on the provided context, it appears that the document discusses a module called "Configurable Setups" and mentions areas or regions within this setup. However, it does not explicitly provide information about which specific areas contain system configuration details. Therefore, I cannot definitively answer your query with certainty based on the given context alone.
SFR-Embedding-Mistral	Based on the provided context, there are areas or regions in the system referred to as "configurable sets" or simply "regions." The first region is responsible for managing configurable areas. The second part of the text discusses a summary or overview of configuration parameters, which are settings within the system. Therefore, it can be inferred that both the configurable areas and their respective parameters fall under the scope of the system's configuration.

Tabulka 7.7: Odpovědi modelu Mistral

Zvýšení top-k embedding modelů

Vzhledem k neuspokojivým výsledkům trénování jednotlivých modelů jsem se rozhodl přistoupit ke zvýšení top-k embedding modelů. Top-k je počet navracených textů na základě podobnosti s otázkou (viz. kapitola 5.2.2). V základním nastavení LlamaIndexu je top-k nastaveno na 2 nejlepší odpovědi. Zvýšení top-k vrácených odpovědí z databáze zvýší kontext a může pomoci rozhodovacímu modelu (v tomto případě Gemma) odpovědět na zadaný dotaz.

Například odpovědi na otázku "Jak nastavím počáteční uživatelské role v systému?" jsou vidět v tabulce 7.8. Z odpovědi modelu je patrné, že model při zvýšení kontextu reaguje na odpověď správně, avšak při zvýšení top-k na 10 a více, už další kontext není relevantní a model Gemma tento text nezvládá zpracovat.

Z uvedeného lze říci, že zvýšení kontextu může vést ke zlepšení modelu do určité míry. Při zvýšení top-k na 10 však model již neodpovídá na danou otázku a navíc odpovídá v angličtině. Proto jsem se rozhodl zůstat u 5 nejpodobnějších částí textu vrácených z vektorové databáze.

7.1.4 Připojení dat z databáze STAPRO

Vzhledem k tomu, že systém by měl taktéž odpovídat na otázky z databáze, jako další zdroj informací je použito nastavení konfiguračních parametrů z vývojové části databáze klinického informačního systému. Toto nastavení je ve formátu XML a reprezentuje veškeré konfigurovatelné parametry.

Jednotlivé konfigurační parametry jsou pomocí SQL selectů převedeny do formátu csv. Následně jsou sjednoceny. Vzniklá stromová struktura je uložena do XML.

Pro připojení dat do databáze je zvolen nástroj LlamaIndex. Data jsou načtena zvlášť z dokumentace a z XML souborů pomocí funkcí PDFReader a XMLReader. XML definice parametrů jsou opět rozdělena do jednotlivých částí dokumentů.

V tabulce 7.9 lze vidět, že i přes začlenění XML dat do vektorové databáze model neumí odpovědět na zadaný dotaz. Vypadá to že model neumí zpracovat dané XML.

Zavedení dat z databáze STAPRO v XML formátu nepřineslo žádné zásadní změny v odpovědích. I přesto je třeba zdůraznit, že schopnost modelů interpretovat XML je omezená a toto omezení má své důsledky. Tento test poukazuje na absenci schopností modelů v manipulaci s XML daty.

7.2 Návrh integrace

Pro způsob integrace do klinického informačního systému FONS Enterprise je navržena webová aplikace, která bude přístupná ze systému FONS Enterprise pomocí odkazu. Tato aplikace je ve formě chatu, kde se může programátor nebo správce konfigurace systému zeptat daného modelu.

Webová aplikace je navržena pomocí frameworku Streamlit⁵, který je určen k rychlému nasazení webových aplikací. Tato aplikace je navržena jako externí služba.

7.2.1 API

Kromě webové aplikace je taktéž navrženo jednoduché API pomocí frameworku Flask⁶.

⁵<https://streamlit.io>

⁶<https://flask.palletsprojects.com/en/3.0.x/>

	Gemma odpověď
top 2	The text does not describe how to set an initial user role in the system, therefore I cannot answer the query.
top 5	<p>**Answer:**</p> <p>V systému je nastavení počáteční uživatelské role definováno v modulu Nastavení přístupových práv. K tomu je nutné následovat tyto kroky:</p> <p>**Krok 1:** Zvolíme Nový – Přidat novou funkční entitu – Přidat novou složku konfiguračních sestav.</p> <p>**Krok 2:** Vyplníme kód (musí začínat zkratkou zakázky), doporučený název je XXRepConfReportFolderYYY , kde XX je kód zakázky a YYY je specifikace názvu. Dále musíme vyplnit oblast s autoritou 4 (např. KZ CoreArea4KZ) a správce entity .</p> <p>**Krok 3:** Uložíme, objeví se modul pro nastavení přístupových práv a zde nastavíme práva pro vybrané role.</p> <p>**Poznámka:**</p> <ul style="list-style-type: none"> * Nastavení počáteční uživatelské role je vždy nutné při prvním použití systému. * Role jsou definovány v modulu Nastavení přístupových práv. * K nastavení rolí je nutné vyplnit kód, název, oblast a správce entity. * Po nastavení rolí je nutné uložit data modulu.
top 10	<p>## Summary of the text:</p> <p>This text describes the process of setting up an initial user role in a system. It mainly focuses on the steps involved in creating a assigning roles to users and groups.</p> <p>**Key points:**</p> <ul style="list-style-type: none"> * **Two modes:** The text operates in two modes when refining existing answers: rewrite or repeat the original answer. * **New context:** The text focuses on the specific context of setting up initial user roles in a system related to Windows installation and domain settings. * **Role assignment:** Roles are assigned to groups, which are in turn assigned to specific departments or organizational units. * **Role description:** Roles describe the working placement and responsibilities of a user. * **System access:** The system uses roles to control access to data, modules and functions within the system. * **Setting up an initial user role:** This process involves creating a group, assigning roles to the group, and setting permissions for the group. <p>**Overall:**</p> <p>This text provides a concise summary of the process of setting up an initial user role in a system, emphasizing the importance of roles in controlling access and managing user permissions. It avoids referencing the original text or context directly and instead focuses on the key points relevant to the new context.</p>

Tabulka 7.8: Odpovědi modelu Gemma při zvýšení kontextu embedding modelů

	Které oblasti například obsahuje konfigurace systému?	Jak nastavit personalizované dashboardy pro různé uživatele?
Gemma s XML daty	<p>**Answer:**</p> <p>Konfigurace systému FONS Enterprise zahrnuje řadu oblastí, včetně nastavení základních objektů (entit), nastavení/import číselníků, nastavení přístupových práv a nastavení funkčních a konfiguračních parametrů.</p>	Given the context information and not prior knowledge, answer the query.
Gemma bez XML dat	<p>**Konfigurace systému FONS Enterprise zahrnuje řadu oblastí:**</p> <ul style="list-style-type: none"> * Nastavení základních objektů (entit) * Nastavení/import číselníků * Nastavení přístupových práv * Nastavení funkčních a konfiguračních parametrů <p>Každá oblast konfigurace je dále popsána samostatně.</p>	The text does not provide information about personalizing dashboards for different users, therefore I cannot answer the query.

Tabulka 7.9: Odpovědi modelu Gemma po přidání XML

Toto API má endpointy `/retrieve` a `/query`. Tyto endpointy přijímají data pomocí HTTP dotazu s metodou POST. Tato data jsou v JSON formátu a mají následující strukturu: `{'query': 'text'}`. Endpointy `/retrieve` a `/query` vrací data opět ve formátu JSON v následujícím tvaru:

- endpoint `/query` – vrací JSON data se strukturou `{'response': 'text'}`, kde `response` je odpověď na danou otázku
- endpoint `/retrieve` – vrací JSON data ve tvaru `{'nodeid': {'Text': 'text', 'Score': 0.6}, ...}`, kde `nodeid` je id navrácené části textu, `Text` je samotný text a `Score` je ohodnocení relevantnosti textu vzhledem k otázce.

Kapitola 8

Vyhodnocení a závěr

Cílem této diplomové práce bylo natrénovat zvolený velký předtrénovaný jazykový model pro dotazy nad daty klinického informačního systému FONS Enterprise a navrhnout integraci jazykového modelu do tohoto klinického informačního systému jako služby poskytující informace při testování, konfiguraci, nasazení a podpoře systému. Dalším cílem byla implementace navržené služby a integrace do klinického informačního systému.

Splnit tento cíl se podařilo, ovšem výsledky natrénovaných modelů byly málo uspokojivé.

Nejprve jsem se seznámil s velkými předtrénovanými jazykovými modely, jako je například GPT-3, PaLM, nebo LLaMA. Seznámil jsem se také s jejich nabízenými službami, nástroji, a dostupností.

Dále jsem se seznámil s rozsáhlým informačním systémem FONS Enterprise firmy STAPRO pro podporu zdravotnických zařízení a jejich řetězců. Systém FONS Enterprise je stavěn na objektech, takzvaných entitách – logických, datových nebo funkčních blocích. Využil jsem nápovědy tohoto systému a vytvořil datový model pro potřeby trénování.

Jelikož je trénování velkých jazykových modelů výpočetně náročné využil jsem možnosti dotrénování již předem natrénovaných jazykových modelů nad dokumentací FONS Enterprise. Vzhledem k formě dat jsem se zaměřil na jednu z dalších možností použití velkých jazykových modelů a to je hledání kontextu z dokumentů pomocí indexování textu a následné zpracování tohoto textu pomocí jazykového modelu. K tomuto přizpůsobení jazykového modelu jsem využil konceptu Retrieval-Augmented Generation (RAG). Dále jsem se rozhodl vyzkoušet metodu RAG nad XML daty z databáze FONS Enterprise.

Pro trénování jsem zvolil modely Llama2, Mistral a Gemma. Domníval jsem se, že by tyto modely mohly být vhodné pro zvolený typ otázek a dat. Dotrénované modely neodpovídaly na dané dotazy. I když architektura Retrieval-Augmented Generation (RAG) jeví posun v získávání fakticky správných odpovědí, tento systém má potíže s češtinou a hlavně se zpracováním dat ve formátu XML.

Nad nápovědou systému FONS Enterprise, která byla použita jako dataset, jsem postupně vyzkoušel známé jazykové modely – Llama 2, Llama 2 Chat, Llama 2 s 70B parametry. Výsledky natrénovaných modelů byly neuspokojivé, proto jsem zkusil změnit parametry a vyzkoušet trénování s modelem Mistral. Postupně jsem měnil parametry jako je rychlost učení nebo počet trénovaných cyklů. Ale ani při zvýšení či snížení rychlosti učení nebo při snížení trénovacích cyklů se model nijak nezlepšil. Následně jsem přistoupil k implementaci RAG architektury. Nejprve jsem provedl implementaci pomocí nástroje LangChain. Výsledkem byla sice odpověď v českém jazyce, ale model negeneroval správný text. Dále jsem zkusil implementaci pomocí nástroje LlamaIndex, který se mi z mého subjektivního pohledu jeví sympatičtěji.

Model Gemma si u některých odpovědí není jistý podobností nalezeného textu a otázky, nicméně je-li kontext správný odpovídá na otázku česky a fakticky správně, na rozdíl od ostatních prezentovaných modelů. Embedding model multilingual-e5-large-instruct si je jistější podobností vráceného textu a otázky, nicméně text je méně relevantní oproti modelu bge-m3. Výsledky modelu SFR-Embedding-Mistral jsou obdobné jako u modelu multilinguale5-large-instruct. Zvýšení top-k odpovědí embedding modelů do určité míry pomohlo zlepšit odpověď. Následné připojení dat databáze STAPRO ve formátu XML nijak neovlivnilo odpovědi.

Jako integraci do klinického informačního systému jsem navrhl webovou aplikaci, která by byla přístupná ze systému FONS Enterprise pomocí odkazu. Tato aplikace je ve formě chatu, kde se může programátor nebo správce konfigurace systému zeptat daného modelu. Je také navrženo jednoduché API pro alternativní integraci do systému.

Na základě provedených testů jsem došel k názoru, že metoda RAG se jeví jako slibná metoda pro získání faktických informací z klinických dat. Nicméně schopnost modelů pochopit češtinu a schopnost modelů pochopit XML jsou omezujícími faktory. Gemma v kombinaci s embedding modelem bge-m3 vrací nejvíce relevantních odpovědí vzhledem k ostatním modelům, avšak u složitějších otázek embedding model neumí vyhledat relevantní kontext a tím pádem model Gemma neumí odpovědět.

Domnívám se, že by bylo přínosné vytvořit kvalitnější datasety, například ve formě otázka-odpověď, a podrobněji zpracovat data, zejména v případě XML formátu. Je také možné zvážit použití alternativních technik, zejména při práci s XML daty, a zohlednit multijazyčné modely, které nejsou přímo natrénovány na češtinu.

Literatura

- [1] BANSAL, L. *A complete guide to rag and LlamaIndex*. Jan 2024. Dostupné z: <https://towardsai.net/p/machine-learning/a-complete-guide-to-rag-and-llamaindex>.
- [2] CHEN, M.; TWOREK, J.; JUN, H.; YUAN, Q.; OLIVEIRA PINTO, H. P. de et al. *Evaluating Large Language Models Trained on Code*. 2021.
- [3] CHUNG, H. W.; HOU, L.; LONGPRE, S.; ZOPH, B.; TAY, Y. et al. *Scaling Instruction-Finetuned Language Models*. 2022.
- [4] DATABRICKS. *Retrieval augmented generation (RAG) on databricks*. 2024. Dostupné z: <https://docs.databricks.com/en/generative-ai/retrieval-augmented-generation.html>.
- [5] HU, E. J.; SHEN, Y.; WALLIS, P.; ALLEN ZHU, Z.; LI, Y. et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021.
- [6] LANGCHAIN, I. *Introduction*. 2024. Dostupné z: https://python.langchain.com/v0.1/docs/get_started/introduction/.
- [7] MALAK, H. A. *What is document indexing? The Complete Guide*. May 2023. Dostupné z: <https://theecmconsultant.com/what-is-document-indexing/>.
- [8] SHARMA, V. a RAMAN, V. A reliable knowledge processing framework for combustion science using foundation models. *Energy and AI*, May 2024, sv. 16, s. 100365. ISSN 2666-5468. Dostupné z: <https://www.sciencedirect.com/science/article/pii/S2666546824000314>.
- [9] *Welcome to LlamaIndex*. 2024. Dostupné z: <https://docs.llamaindex.ai/en/stable/>.
- [10] ZHAO, W. X.; ZHOU, K.; LI, J.; TANG, T.; WANG, X. et al. *A Survey of Large Language Models*. 2023.
- [11] ZHENG, Y.; ZHANG, R.; ZHANG, J.; YE, Y.; LUO, Z. et al. LlamaFactory: Unified Efficient Fine-Tuning of 100+ Language Models. *ArXiv preprint arXiv:2403.13372*, 2024. Dostupné z: <http://arxiv.org/abs/2403.13372>.

Příloha A

Obsah přiloženého paměťového média

Paměťové médium obsahuje adresáře:

- **deploy/** – v adresáři lze najít skripty potřebné pro spuštění webové aplikace a spuštění API. Tento adresář obsahuje mimo jiné skript `run-web.sh`, který spustí webovou aplikaci i API. Pro potřeby aplikace i API jsou v tomto adresáři kontejnery `*.sif` pro spuštění pomocí nástroje singularity¹, případně lze nainstalovat závislosti pomocí příkazů:
`pip install -r deploy/pyenv-docker/reqs.txt` a
`pip install -r deploy/api/pyenv-docker/reqs.txt`.
- **llamafactory/** – v adresáři se nachází skripty použité při trénování jazykových modelů. Při instalování závislostí se je třeba řídit pokynů v `llamafactory/LLaMA-Factory/README.md`. Výsledky trénování jsou pak v adresáři `llamafactory/answers/` ve formátu csv.
- **pyenv-sif/** – v adresáři se nachází skripty použité pro vytvoření prostředí na Meta-Centru, potřebné pro testy v adresáři `test/`. Toto prostředí lze taktéž nainstalovat příkazem `pip install -r pyenv-sif/reqs.txt`.
- **test/** – v adresáři se nachází skripty použité pro testování architektury RAG. V jednotlivých adresářích jsou skripty pro vytvoření indexových databází. V adresářích `test/qa/answers/` a `test/ret_other_llms/answers/` jsou výsledky jednotlivých modelů ve formátu csv, použitých v této práci.
- **dp-thesis/** – v adresáři se pak nachází zdrojový text této práce. Přeložit jej je možné pomocí příkazu `make` v adresáři `db-thesis/`.

¹https://docs.sylabs.io/guides/3.0/user-guide/quick_start.html