FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Dotazování nad relačními databázemi s pomocí velkých jazykových modelů (LLM)
Student:	Tomáš Machorek
Vedoucí:	Ing. Ivo Lašek, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Umělá inteligence 2021
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2025/2026

Pokyny pro vypracování

Výzkum a aplikace různých modelů strojového učení, zejména těch založených na principu velkých jazykových modelů (LLM), nabízí široké spektrum možností pro zlepšení a automatizaci různých procesů v informatice. Jedním z nadějných přístupů se ukázal i překlad přirozeného jazyka do příkazů databázových dotazovacích jazyků (SQL). Cílem této práce je prozkoumat potenciál a vhodnost vybraných LLM modelů pro překlad přirozeného jazyka do příkazů SQL a jejich následné užití pro extrakci dat z daných databázových systémů.

Dále se práce zaměří na tvorbu interaktivního uživatelského rozhraní koncipovaného jako webová stránka ve formě obdobné chatbotu jako např. ChatGPT jež umožní snadné přepínání mezi jednotlivými LLM modely. Toto rozhraní by mělo být intuitivní a snadno použitelné pro široké spektrum uživatelů, a to i pro ty, kteří nemají hluboké technické znalosti. Cílem je poskytnout nástroj, který uživatelům umožní efektivně komunikovat s databázemi prostřednictvím přirozeného jazyka, a tím jim zjednodušit práci s daty.

V rámci práce s daty se ukazuje, že vizualizace v reálném čase je mimořádně užitečná pro identifikaci trendů a vzorců, což představuje třetí a závěrečný úkol této práce. Cílem je umožnit uživatelům snadnou tvorbu vizualizací a grafů na základě jejich vstupů v přirozeném jazyku na základě čehož vybrané LLM modely rozhodnout využít předpřipravené funkce napsané v jazyce Python. Tento přístup má za úkol zpřístupnit komplexní analýzu dat bez nutnosti hlubokých technických znalostí v oblasti datové

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 1. března 2024 v Praze.

analýzy nebo programování.

Pokyny k vypracování práce:
1. Proveďte rešerši dostupných řešení a potenciálně využitelných LLM modelů. Na základě rešerše vyberte 2-3 nejvhodnější modely, které následně v aplikaci využijete (uživatel bude mít možnost mezi jednotlivými modely přepínat prostřednictvím uživatelského rozhraní).
2. Vyberte několik vhodných reálných datových sad, na kterých celou aplikaci otestujete (např. NorthWind a Pubs, AdventureWorks, Strukturovaný výstup ze StackOverflow od Brenta Ozara)
3. Navrhněte, implementujte a otestujte aplikaci, která nabídne základní uživatelské rozhraní, které umožní vybrat jednu z datových sad a jeden z LLM modelů a položit nad datovou sadou dotazy v přirozeném jazyce.
4. LLM model následně přeloží uživatelský dotaz na SQL dotaz, provede ho nad vybraným datasetem a vrácené výsledky odprezentuje uživateli a to buď jako sumarizaci dat opět pomocí vybraného LLM modelu, nebo jako vizualizaci.
5. Pro potřeby vizualizace stačí základní grafické reprezentace tabulkových dat pomocí některé standardní vizualizační knihovny (např. D3).
6. Celou aplikaci následně využijte pro prozkoumání a otestování možností vybraných LLM modelů. Jejich porovnání shrňte v samotné závěrečné práci.

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 1. března 2024 v Praze.

Bakalářská práce

DOTAZOVÁNÍ NAD RELAČNÍMI DATABÁZEMI S POMOCÍ VELKÝCH JAZYKOVÝCH MODELŮ (LLM)

Tomáš Machorek

Fakulta informačních technologií
Katedra aplikované matematiky
Vedoucí: Ing. Ivo Lašek, Ph.D.
16. května 2024

České vysoké učení technické v Praze
Fakulta informačních technologií

Odkaz na tuto práci: Machorek Tomáš. *Dotazování nad relačními databázemi s pomocí velkých jazykových modelů (LLM)*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2024.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především svému vedoucímu práce, panu Ing. Ivo Laškovi, Ph.D., za jeho neocenitelnou pomoc a podporu během celého psaní této práce. Velký dík patří také mé rodině, která mi byla stálou oporou ve všech fázích mého studia. Dále bych chtěl poděkovat přátelům ze skupiny Tangus, kteří mi vždy ochotně poskytli pomocnou ruku, kdykoli jsem to potřeboval. Na závěr bych chtěl poděkovat restauračnímu zařízení Baru 10 a oběma jeho majitelům za podporu, již mi poskytoval také v průběhu celého mého studia.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 16. května 2024

Abstrakt

V posledních letech nabyly velké jazykové modely (LLM) značné popularitě a také rapidnímu rozvoji jak ve kvalitě svých výstupů, tak i v možnostech jejich užití. Tato práce se zaobírá možnostmi využití vybraných LLM k překladu dotazů z přirozeného jazyka do strukturovaného dotazovacího jazyka SQL a jejich následného vyvolání pro zisk strukturovaných dat z relačních databází a prezentací získaných dat pro účely prvotní datové analýzy formou tabulovaných dat.

Klíčová slova přirozený jazyk, NL2SQL, NLP, datová analýza, databáze, LLM, doladťování, Streamlit, Unsloth

Abstract

In recent years, large language models (LLMs) have gained considerable popularity and have also developed rapidly both in the quality of their outputs and in the possibilities of their use. This paper discusses the possibilities of using selected LLMs to translate queries from natural language to structured query language SQL, and then invoke them to extract structured data from relational databases and present the extracted data for initial data analysis in the form of tabulated data.

Keywords natural language, NL2SQL, NLP, data analysis, database, LLM, fine-tuning, Streamlit, Unsloth

Seznam zkratek

DDL	Data Definition Language
DQL	Data Query Language
FNN	Feed Forward Network
GQA	Grouped-Query attention
LoRA	Low-Rank adaptation
QLora	Quantized Low-Rank Adaptation
LLM	Large Language Model
NLP	Natural Language Processing
NL2SQL	Natural Language to SQL
RNN	Recurent Neural Network
SWA	Sliding Window Attention

Kapitola 1

Úvod

1.1 Motivace

Množství generovaných dat se v posledních letech řádově zvětšilo a dle predikcí tento růst nehodlá upadnout ani v blízké budoucnosti [1]. Se samotným růstem objemu dat vzniká i přímočará potřeba schopnosti tyto data interpretovat, prezentovat a následně zredukovat na potřebné podmnožiny pro užití na vybrané problémy. S nutností těchto schopností ovšem vzniká další problém, a to, že tyto schopnosti se kvalifikovaní datoví analytici (či programátoři) učí několik let. Jedním z nadějných východisek se v posledních letech ukázalo využití velkých jazykových modelů (*angl. Large Language Model, LLM*) jež dokáže využít i méně zkušený či dokonce úplně nezkušený uživatel, který nemusí mít žádné znalosti programovacích jazyků či metodik datového analýzy. Tento postup poté umožňuje ulehčení práce datovým analytikům díky možnosti předelegování břemena samotné interpretace a prezentace dat na daného uživatele (jemuž stačí pouze zformalizovat jednoznačný dotaz v přirozeném jazyce), díky čemuž se poté expert může přímo zaměřit na jejich využití na vybranou problematiku. Tato situace nepředstavuje jen velkou příležitost pro zvýšení efektivity práce, ale také pro rozšíření obzorů mnoha profesionálů a podniků.

1.2 Cíle práce

Tato práce si ukládá za svůj primární cíl vytvoření webové aplikace sloužící k datové analýze dat. Tato stránka bude koncipovaná jakožto „chatovací“ prostředí. Do tohoto prostředí bude uživatel moct zadat dotazy v přirozeném jazyce. Tyto dotazy se následně pomocí vybraných LLM modelů přeloží do příkazů SQL. Uživatel získá data formou tabulovaných dat. Na základě těchto dat bude i nezkušený uživatel schopen provést úvodní průzkum získaných dat.

1.3 Organizační struktura práce

Práce se, spolu s úvodní kapitolou, skládá z šesti kapitol. V prvotní kapitole bude utvořen teoretický základ pro lepší uchopení struktury a architektury vybraných modelů. Druhá kapitola se skládá z dvou hlavních sekcí. V počáteční sekci této kapitoly jsou představeny vybrané velké jazykové modely s jejich nejdůležitějšími parametry, jenž budou užity v zbytku této práce, ve druhé sekci jsou následně představeny vybrané techniky zlepšování efektivity velkých jazykových modelů pro překlad přirozeného jazyka do strukturovaného jazyka SQL. V třetí kapitole je představen metodologický postup který byl užit pro přípravu modelu na využití v rámci webového rozhraní. Struktura a užité technologie webového rozhraní pro jeho vývoj budou diskutovány v páté kapitole. V závěrečné kapitole budou poté diskutovaný možnosti zlepšení výsledku daného rozhraní, přípravy modelu a technologické limitace stávajícího řešení.

Kapitola 2

Teoretický základ

I přestože se velké jazykové modely dostaly do lidského podvědomí až v posledních pár letech, mají kořeny již několik desetiletí zpátky. Historie těchto modelů začíná již v 50. letech 20. století, kdy se začaly objevovat první pokusy o modelování jazyka na základě statistických metod. Postupem času se ovšem ukázalo, že tyto relativně jednoduché postupy pro reprezentaci jazyka nejsou adekvátní. Disciplíně rozpoznávání přirozeného jazyka tehdy přišly na pomoc hluboké neuronové sítě, jmenovitě poté úspěchy nových algoritmů jako jsou například rekurentní neuronové sítě a algoritmus zpětné propagace. Dalším velice důležitým milníkem poté bylo v nedávných letech zavedení architektury *Transformer*, která se pro zpracování přirozeného jazyka používají dodnes. Tato kapitola si klade za cíl vybudovat teoretický základ, který umožní čtenářům hlubší porozumění metodám zpracování přirozeného jazyka, jež budou využívaný modely (do jisté kapacity) sloužícími k překladu jazyka přirozeného do strukturovaného dotazovacího jazyka SQL.

2.1 Neuronové sítě

Neuronové sítě jsou inspirovány strukturou a funkcí lidského mozku. Představují výpočetní modely, které se učí z dat a napodobují způsob, jakým se biologické neurony učí a adaptují. Základní myšlenkou je propojení jednoduchých jednotek, nazývaných neurony nebo perceptrony, do sítě, která je schopná řešit komplexní úlohy. Každý neuron přijímá několik vstupů, provádí na nich určité výpočty a vytváří výstup, který je předán dalším neuronům v síti.

Neuronové sítě se využívají v mnoha oblastech, jako je rozpoznávání obrazu, zpracování přirozeného jazyka, předpověď časových řad a mnoho dalších. V této sekci se zaměříme na základní typy neuronových sítí, jejich architekturu a principy fungování. Prozkoumáme perceptron, dopředné neuronové sítě a rekurentní neuronové sítě, které tvoří základ moderních přístupů v oblasti strojového učení a umělé inteligence.

2.1.1 Perceptron

Perceptron představuje nejzákladnější stávební kamen neuronových sítí a zároveň je samotným nejzákladnější formou neuronové sítě. Koncept perceptronu byl představen již v roce 1958 v článku *„The perceptron: A probabilistic model for information storage and organization in the brain.“* [2] kde byl koncipován jakožto matematický model pro lidský neuron. Obdobně jako lidský neuron má několik vstupů a jeden výstup, základní tvar matematické funkce, jenž jej popisuje poté vypadá následovně:

$$f(x) = \theta(\mathbf{w} \cdot \mathbf{x} + \mathbf{b}) \tag{2.1}$$

kde pro $n \in \mathbb{N}$ je $\mathbf{x} \in \mathbb{R}^n$ je vektor vstupních hodnot, $\mathbf{w} \in \mathbb{R}^n$ je vektor váh, $\mathbf{b}$ je tzv. *bias*, jenž slouží k posunu rozhodovací hranice od počátku nadroviny, operace $\cdot$ je standardní skalární součin jenž je definován následovně:

▶ **Definice 2.1** (Standardní skalární součin)**.** *Nechť $x, y \in \mathbb{T}^n, n \in \mathbb{N}$. Poté definujeme skalární součin předpisem:*

$$x \cdot y := \sum_{i=1}^{n} \bar{x_i} y_i \tag{2.2}$$

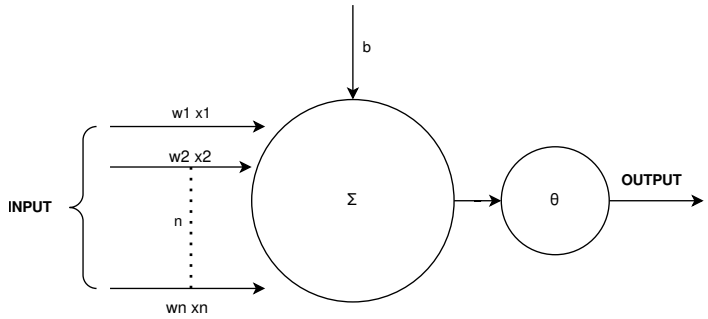
kde $\bar{x_i}$ značí reálnou část čísla x.

a konečně symbol θ značí tzv. Heavisideovu funkci (která v tomto modelu reprezentuje tzv. *aktivační funkci*) jenž je definována následujícím předpisem:

▶ **Definice 2.2** (Heavisideova funkce (s parametrem ρ))**.** *Heavisideovu funkci θ definujeme následujícím předpisem:*

$$\theta(x) = \begin{cases} 0 & x > 0 \\ \rho & x = 0 \\ 1 & x < 0 \end{cases} \tag{2.3}$$

kde obvykle $\rho \in \{0, \frac{1}{2}, 1\}$



Obrázek 2.1 Perceptron

Postupem času se ovšem ukázalo, že perceptron dokáže modelovat pouze lineární závislosti, jak bylo ukázáno na dnes již slavném příkladu XOR problému [3]. Toto zjištění způsobilo velkou ztrátu zájmu u výzkum v poli umělé inteligence. Postupem času se ovšem ukázalo, že v případě nahrazením Heavisideova funkce za vhodnější aktivační funkci nelineárního

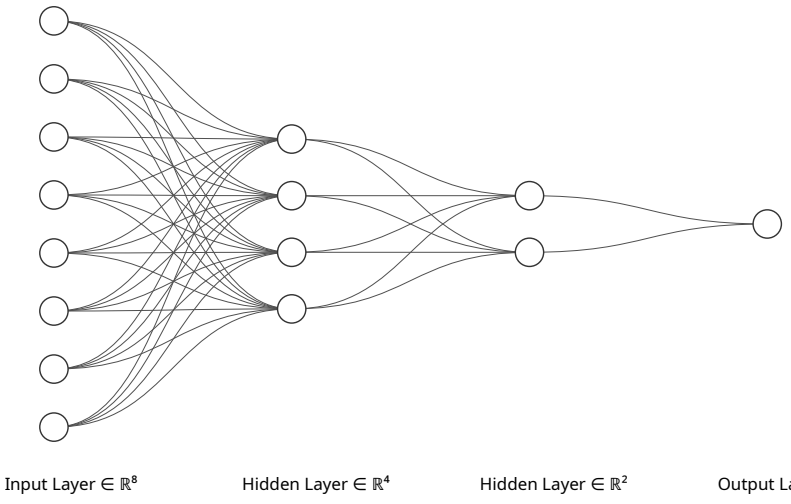
charakteru spolu s utvořením více vrstvého modelu jde podchytit i závislosti nelineární.

2.1.2 Dopředné neuronové sítě

Dopředné neuronové sítě (angl. *feedforward neural networks, FNN*) jsou jedním ze základních typů neuronových sítí. Tyto sítě se skládají z několika vrstev perceptronů, kde je výstup každého perceptronu napojen na vstup všech perceptronu ve vrstvě následující. Celý účel tohoto propojení je ve své nejzákladnější podstatě aproximovat nějakou funkci f^* pro nějaký vstup $\mathbf{x}$, přesněji hledáme funkci $y = f(\mathbf{x}, \theta)$ která mapuje vstup x za pomocí sady parametrů θ. Dle vět o univerzální aproximaci bylo poté ukázáno, že síť s alespoň jednou skrytou vrstvou tvořenou perceptrony s nelineární aktivační funkcí s oborem hodnot $[0, 1]$, jejichž výstupy jsou finálně agregovány lineárními kombinacemi, lze na výstupu s libovolnou přesností aproximovat jakoukoliv spojitou funkci s omezeným nosičem v $\mathbb{R}^p$. [4]. Díky této vlastnosti tedy dokážeme modelovat libovolné reálné funkce a do této třídy samozřejmě patří i již zmiňovaný XOR problém a zároveň navrátila naději do výzkumu umělé inteligence.

Dopředné neuronové sítě jsou charakterizované třemi třídami vrstev, jmenovitě:

- Vrstva vstupní - Prvotní vrstva modelu, která přijímá surové data $\mathbf{x} \in \mathbb{R}^n$, na obrázku 2.2 je to první vrstva zleva a tento model dokáže přijímat vstup o rozměru 8.
- Vrstvy skryté - Vrstvy sloužící k výpočtu vážených sumací vstupů a provádění nelineárních aktivací a transformací vstupních data.
- Výstupní vrstva - Vždy poslední vrstva v modelu. Počet perceptronů v této vrstvě určuje typ úlohy, na obrázku 2.2 lze vidět pouze jeden výstupní perceptron a můžeme tedy předpokládat, že by to byl model na binární klasifikace.



Obrázek 2.2 Příklad neuronové sítě s dvěma skrytými vrstvami

Co je dále charakteristické k této architektuře je skutečnost, že se informace šíří pouze

jedním směrem a to právě od vrstvy vstupní, přes vrstvy skryté až finálně do vrstvy, matematicky tuto skutečnost poté můžeme zapsat následujícím vztahem:

$$f(x) = f^{(l-1)} \circ f^{(l-2)} \circ \cdots \circ f^{(2)} \circ f^{(x)} \tag{2.4}$$

kde l značí hloubku sítě [5]. Pro získání výstupu dopředné neuronové sítě jsou postupně tvořeny lineární kombinace vstupů, vah a biasu. Informace o váhách přechodů jsou poté obvykle uchovávaný v tzv. *matici vah*.

▶ **Definice 2.3** (Matice vah)**.** *Nechť poslední skrytá vrstva produkuje vektor* $n_{(l-1)}$ *skrytých příznaků* $f(x) = f^{(l-1)} \circ f^{(l-2)} \circ \cdots \circ f^{(2)} \circ f^{(x)}$ *a* $\mathbf{b} \in \mathbb{R}^{n_{(l)}}$ *je vektor biasů. Matice* $\mathbf{W}$ *se poté skládá ze sloupců kde i-tý sloupec a i-tá složka* $\mathbf{b}$ *jsou váhy i-tého perceptronu.*

S pomocí této definice lze uvést vztah, jenž slouží k tvorbě hodnot které následně budou užity k predikci:

$$\mathbf{z} = \mathbf{W}^T\mathbf{x} + \mathbf{b}, \tag{2.5}$$

kde $\mathbf{z}$ je výstup určený k predikci, $\mathbf{x}$ je vstupní vektor a $\mathbf{b}$ je vektor biasů. V případě, že bychom ovšem pouze tvořili tyto lineární kombinace, tak dostáváme opět pouze lineární aktivační funkce (jelikož by ve své podstatě šlo o identické funkce) a nenaplnil by se tak předpoklad Vět o univerzálních aproximacích. Kvůli této nutnosti je potřeba na tyto výstupy ještě aplikovat nějakou nelineární aktivační funkci. jako např. RELU, jenž má následující předpis:

$$RELU(x) = \max(0, x), \tag{2.6}$$

či Sigmoida s nálsedujícím předpisem:

$$\sigma(x) = \frac{e^x}{1 + e^x}, \tag{2.7}$$

nebo jedna ze spousty dalších (Leaky RELU či SELU, ...).

I přestože jsou dopředné neuronové sítě velice silným nástrojem a vrstvy tohoto charakteru se do dnes používají ve velké části většiny moderních modelů, mají velkou nevýhodu, a to, že nemají žádný zpětný mechanismus a jsou tedy ve své podstatě bezpaměťové. Intuitivním řešením tohoto problému se poté stává implementace zpětného předání informací. Jedním z řešení tohoto problému jsou právě rekurentní neuronové sítě.

2.2 Rekurentní neuronové sítě

Rekurentní neuronové sítě (RNN) jsou třídou neuronových sítí, které mají schopnost zpracovávat sekvence dat s proměnnou délkou. Tato vlastnost z těchto sítí dělá ideální nástroj a stavební blok pro zpracování přirozeného jazyka.

Základní stavební jednotkou rekurentních neuronových sítí je obdobně jako u FNN perceptron. Vitální rozdíly této sítě jsou ovšem tzv. *skryté stavy* a perceptrony zpětné vazby.

Díky těmto mechanismům je v každém časovém kroku t zpracováván jak vstup $\mathbf{x}_t$ tak i skrytý stav h_t. Pomocí akumulace informace držené ve skrytém stavu h_t můžeme poté držet kontext až teoreticky nekonečné délky. Samotný výstup modelu v závislosti na daném skrytém stavu v čase t poté definujeme následovně:

$$\hat{y}_t = f(x_t, h_{t-1}) \tag{2.8}$$

a informace kumulujeme ve skrytých stavech následující rekurentní rovnicí:

$$h_t = f(\mathbf{W}_{hh} h_{t-1} + \mathbf{W}_{xh} \mathbf{x}_t + \mathbf{b}_h) \tag{2.9}$$

kde

- $\mathbf{W}_{hh} \in \mathbb{R}^{d,d}$ je standardní matice vah dle definice 2.3 mezi skrytými stavy kde d je dimenze skrytého stavu.
- $\mathbf{W}_{xh} \in \mathbb{R}^{d,m}$ je standardní matice vah dle definice 2.3 mezi skrytým stavem a vstupním vektorem kde d je dimenze skrytého stavu a m je vstupního vektoru.
- $\mathbf{b}_h \in \mathbb{R}^d$ je bias skrytých stavů.

Bohužel ani tento postup nestačí na dnešní potřeby, ale rekurentní neuronové sítě jsou již jedním ze stavebních bloků architektury Transformer a je zároveň v dobu psání této práce *State-of-the-art* architekturou pro zpracovávání přirozeného jazyka.

2.3 Modely Transformer

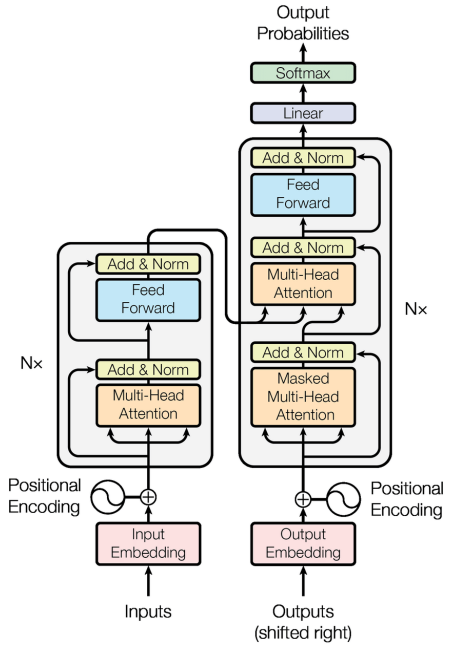
Modely architektury Transformer byla představeny v roce 2017 v průkopnickém článku *Attention Is All You Need* [6]. Tato architektura byla stvořena pro zpracovávání sekvenčních dat a jejich transformaci z dat vstupních do dat výstupních na základě paralelní *encoder-decoder* architektury. K dosažení tohoto cíle tato architektura využívá mechanismus tzv. *Attention (pozornost)*, jenž má emulovat lidský způsob extrakce sémantických spojitostí na základě kontextů dané věty. Krom samotného tvoření výběru nejdůležitějších slov ze vstupních dat je model také auto-regresivní a tedy v každém kroku predikce dalších tokenů konzumuje svůj předchozí výstup k zlepšení odhadu následujícího tokenu. Dodatečně má tato architektura na rozdíl od předchozích architektur schopnost modelovat i velice dlouhé závislosti v rámci svého tzv. *kontextového okna* díky právě schopnosti výběru pozornosti dle daného kontextu pro generaci dalších tokenů.

2.3.1 Architektura

Jak lze vidět z obrázku 2.3, data do modelu nepřicházejí v surové podobě, ale nejprve procházejí technikou zvanou *word embedding*, která reprezentuje jednotlivá slova ve vektorovém prostoru pomocí modelů pro word embedding. Tato technika prokázala zvýšení efektivity výstupního modelu pro různé účely v oblasti zpracování přirozeného

jazyka [7]. Dále je vstupní slovní vektor obohacen o svou pozici ve vstupní sekvenci (větě), což rovněž přispívá ke zvýšení koncové efektivity, a to společně s tzv. *position-wise FNN*, která je součástí obou hlavních částí této architektury. Hlavní dvě částí této architektury se poté jmenují Encoder a Decoder. Tyto části mají následující architekturu:

- **Encoder** – Skládá z šesti identických vrstev. Každá z těchto vrstev se poté skládá ze dvou podvrstev, jmenovitě z vrstvy vícenásobného mechanismu pozornosti a plně provázané *position-wise FNN*. Každá z těchto podvrstev navíc užívá metody zbytkového propojení [8] a vrstvové normalizace [9]. Finální výstup každé z podvrstev poté je dán vzorcem $y = LayerNorm(x + f(x))$ kde $f(x)$ je funkce dané pod vrstvy. Finální cíl Encoderu je zakódovat vstup do skrytých stavů, které následně budou dekódovány Decoderem do výstupního formátu.
- **Decoder** – Skládá se také z šesti identických vrstev, ovšem ke každé vrstvě přibude jedna pod vrstva, jenž se skládá opět z vícenásobného mechanismu pozornosti, který je ovšem zaměřen na výstup z Encoderu. Finálně poté Decoder dekóduje skryté stavy na výstup pomocí lineární a softmaxové vrstvy po provedení Decoder stacku.



Obrázek 2.3 Architektura Transformer, převzato s povolením z [6]

2.3.2 Attention (Pozornost)

Poslední a nejdůležitější pojem, který byl již v této kapitole několikrát zmíněn je pozornost, nebo také pozorností funkce. Pozornost v rámci kontextu vstupu odkazuje model na slova, o kterých si myslí, že mají největší vliv na generování dalšího slova. Mechanismus (určení) pozornosti se poté skládá z nasledujích kroků:

1. Nechť na vstupu dotazovací vektor $q \in \mathbb{R}^d$, množina vektorů klíčů $\mathbf{K} = \{\mathbf{k_1}, \mathbf{k_2}, \ldots, \mathbf{k_n}\}$ kde $\forall i \in \hat{n} : \mathbf{k_i} \in \mathbb{R}^d$, množina vektorů hodnot asociovaných s prvky množiny vektorů klíčů $\mathbf{V} = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_n}\}$ kde $\forall i \in \hat{n} : \mathbf{v_i} \in \mathbb{R}^m$ (tj. vektory hodnot nemusí být nutně stejné dimenze jako vstupní vektor).
2. Pro $\forall i \in \hat{n} : \mathbf{k_i} \in \mathbf{K}$ se vypočte skóre kompatibility s_i pomocí funkce kompatibility $C(\mathbf{q}, \mathbf{k_i})$
3. Všechny skóre kompatibility se normalizují a převedou na pozornosti váhy:

$$a_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}, \forall i \in \hat{n} \tag{2.10}$$

4. K danému klíči se vytvoří vážený součet jenž v tomto kroku je výstupem pozornostního mechanismu pro daný klíč:

$$o = \sum_{i=1}^{n} a_i v_i \tag{2.11}$$

▶ Poznámka 2.4. Funkce kompability: Existuje mnoho různých funkcí kompatibility a otázka, která z nich je nejlepší, je stále předmětem výzkumu. V původním článku byla použita tzv. *Scaled Dot-Product Attention*, která je definována následujícím vzorcem:

$$C(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.12}$$

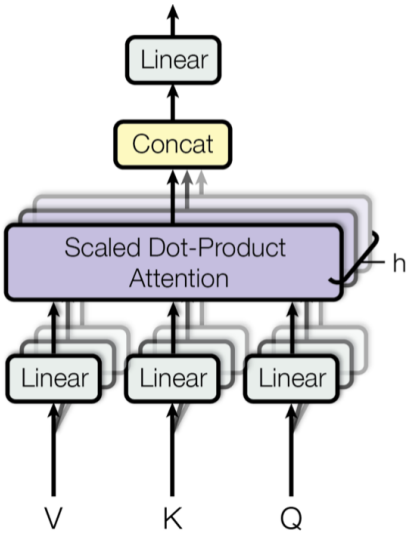
kde Q je pouze matice stvořena z více dotazovacích vektorů, K je matice vektorů klíčů a V je matice vektorů hodnot asociovaných s vektory matice vektorů klíčů.

Tento postup sám o sobě funguje, ovšemže pouze sekvenčním způsobem. Dalším logickým krokem bylo utvoření vícenásobného mechanismu pozornosti, jenž umožňuje díky lineárním projekcím vstupního vektoru na různé podprostory právě tížené paralelní zpracovávání sekvenčních vstupů. Samotný užitý předpis k dosáhnutí tohoto vztahu poté vypadá následovně:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, \ldots, head_h)W^O \\ \text{where } head_i &= C(QW_i^Q, KW_i^K, VW_i^V), \forall i \in \hat{h} \end{aligned} \tag{2.13}$$

kde $W_i^Q \in \mathbb{R}^{d_{model}, d_k}, W_i^K \in \mathbb{R}^{d_{model}\, d_k}, W_i^V \in \mathbb{R}^{d_{model}, d_v}$.

A díky tomuto postupu je následně dosaženo vytvoření více kontextových oken a paralelních zpracování v době psaní s nejvyšší efektivitou v disciplíně zpracovávání přirozeného jazyka, který je tématem této práce.



Obrázek 2.4 Vícenásobný mechanismus pozornosti, převzato s povolením z [6]

2.4 Velké jazykové modely

Velké jazykové modely (angl. *Large Language Models, LLMs*) jsou výkonným nástrojem pro zpracování přirozeného jazyka. Tyto modely jsou ve svém nitru založeny na neuronových sítích s možnostmi zpracovávání sekvenčních dat.

Klíčovým rysem LLM je jejich schopnost generovat smysluplné odpovědi a rozhodovat se i přes složité dotazy díky rozsáhlému trénování na korpusech textových dat. Tento rys poté umožňuje lehkou integraci s dalšími technologiemi, díky čemuž můžou LLM sloužit jako velice silné jádro pro spoustu jiných aplikací, než je pouze porozumění přirozeného jazyka.

Kapitola 3

Vybrané velké jazykové modely a techniky pro zlepšení jejich efektivity

V dnešní době se umělá inteligence stává součástí našich každodenních životů a její rozsah se neustále rozšiřuje. Jedním z klíčových aspektů této expanze jsou i velké jazykové modely. Tyto modely umožňují i nezkušeným uživatelům snadno komunikovat s technologií prostřednictvím přirozeného jazyka.

První sekce této kapitoly je zaměřena na vybrané velké jazykové modely, které budou užity v rámci této práce, spolu s jejich nejdůležitějšími aspekty, a v druhé sekci této kapitoly budou poté zmíněny techniky, jež jsou v dnešní době užívány pro zlepšení efektivity funkcionality těchto modelů jakožto rozhraní pro překlad přirozeného jazyka do strukturované dotazovacího jazyka SQL.

3.1 Vybrané jazykové modely

S každým dnem přibývá nových velkých jazykových modelů, ať už jde o zcela nové modely nebo o upravené verze zaměřené na specifické problémy. Díky tomu je stále jednodušší najít vhodný model podle kritérií pro konkrétní aplikace. V rámci této práce bylo hlavním kritériem schopnost modelu ve své základní, nedoladěné podobě excelovat ve zpracovávání přirozeného jazyka. Ideálními kandidáty se v tomto ohledu staly zejména nedávno vydaný model Llama 3 a poněkud starší, ale stále výkonný model Mistral, oba volně dostupné veřejnosti.

3.1.1 Llama 3

Tento model patří do rodiny open source modelů Llama a zároveň je v době psaní ve své nejsilnější variantě nejlepším modelem dostupným pro širokou veřejnost [10].

Model samotný byl publikován společností Meta AI ve ve dvou variantách: Llama-8B a Llama-70B. Bohužel v době psaní této práce společnost Meta AI nevydala přidružený výzkumný článek[1] specifikující přesné parametry jenž model má, mimo několika základních, či náturu datové sady, na které byl model trénován. Jediné informace, které jsou o o obou z variant modelů momentálně známe jsou, že modely byl trénovány na korpusu různorodých, veřejných dat s objemem více než 1.5×10^{13} tokenů a s užitím varianty pozornostního mechanismu *Grouped-Query Attention (GQA)*, který je založen na shlukování hlav pozornostního mechanismu do skupin uniformní velikosti a sdílení klíčů a hodnot asociovaných s klíči v rámci daných skupin [12] pro docílení lepší škálovatelnosti inferenčního mechanismu. Využití pozornostního mechanismu napovídá tomu, že oba modely opět spadají do kategorie transformerových modelů, ovšem toto tvrzení se nedá potvrdit bez samotného vydání asociovaného článku.

Název	Počet parametrů	Kontextové okno	Znalostní hranice
Llama-8B	8.0×10^9	8192 tokenů	Do Března, 2023
Llama-70B	7.0×10^{10}	8192 tokenů	Do Prosince, 2023

Tabulka 3.1 Známé parametry modelu Llama 3

I přes tyto neznalosti přesných specifikací modelu zůstává faktem, že tento model je momentálně nejmodernějším a napříč testovacími sadami nejlepší model z třídy volně dostupných modelů a z tohoto důvodu bude využit jako primární model v rámci této práce pro překlad přirozeného jazyka do strukturovaného jazyka SQL. Z důvodu nároků na výpočetní techniku byl v rámci této práce vybrán model Llama-8B, jenž sice není nejlepší nabízenou variantou ale i přes tento fakt si v rámci testování vede obstojně i vůči proprietárním modelům jako je např. ChatGPT 4.5 Turbo [13][10].

3.1.2 Mistral 7B

Jako sekundární model byl vybrán model Mistral 7B, který byl představen ve stejnojmenném článku společností Mistral AI [14]. Tento model, obdobně jako modernější model Llama3, využívá varianty pozornostního mechanismu GQA, ale navíc kromě využití této techniky aplikuje i další variantu pozornostního mechanismu jménem *Sliding window attention (SWA)*, sloužího k tvorbě izolovaného lokálního kontextu. Tato metoda prokazatelně zvyšuje efektivitu finálního modelu pro řešení různorodých problémů v rámci užití velkých jazykových modelů [15]. Tento přístup, podobný konvolučním metodám, postupně tvoří pozornostní okno s šířkou n tokenů. Toto okno je poté postupně „posouváno“ v rámci vstupního (tokenizovaného) dotazu pro tvoření lokalizovaného kontextu [16], jenž spolu s metodu GQA pomáhá extrahovat jak lokální sémantické významy, tak udržovat znalostní kontext pro celý vstupní dotaz.

[1] Nepočítaje oznamující článek vydání modelu [10] a modelovou kartu [11]

Název parametru	Hodnota parametru
dim	4096
n_layers	32
head_dim	128
hidden_dim	14336
n_heads	32
n_kv_heads	8
window_size	4096
context_len	8192
vocab_size	32000

Tabulka 3.2 Přehled parametrů modelu Mistral 7B [14]

Díky relativnímu stáří[2] tohoto modelu je i známý fakt, že tento model patří do kategorie transformer modelů spolu s parametry uvedenými v tabulce 3.2.

Model samotný byl vybrán jako evaluační a překládací model v rámci této práce z tří hlavních důvodů:

1. Díky své architektuře, jež umožňuje modelování jak lokálních tak globálních závislostí.
2. Kvůli „stáří“ pro kontrast nejmodernějšího modelu a modelu, jenž byl koncipován již téměř rok zpátky a také hlavně
3. Proto, že je open source , a tedy umožní jeho snadné doladění a následné využití.

3.2 Techniky pro zlepšení efektivity modelů

Technik pro zlepšení efektivity velkých jazykových modelů existuje široká řada, ve finále se ale dají všechny kategorizovat pomocí dvou možností a to:

- Zlepšení vstupních dat
- Systematické zlepšování modelů

Těchto technik je samozřejmé opět nespočet s různými mírami úspěšnosti a proto je v této práci opět vyjmenovaných pouze pár ze všech kategorií, jenž dosahují největší úspěšnosti právě u disciplíny překladu přirozeného jazyka do strukturovaného jazyka SQL.

3.2.1 Zlepšení vstupních dat

Kategorie zlepšování vstupních dat má ve své nejzákladnější podstatě jediný cíl: Předat vybranému modelu co nejkvalitnější vstup, aby mohl s co největší přesností přímočaře a efektivně rozhodnout o daném výstupu. V kontextu disciplíny zpracovávání přirozeného

[2] Ve srovnání s modelem Llama3.

jazyka, specificky pro překlad do strukturovaného jazyka SQL, je poté cílem předat modelu vstupní větu se všemi potřebnými informacemi, ze které bude model schopen extrahovat sémantické spojitosti pro tvorbu tíženého dotazu v jazyce SQL. Nejzákladnější technikou tohoto rázu je poté tzv. Vnořování slov (angl. *Word embedding*).

Vnořování slov

Vnořování slov je skupina technik, snažící se o reprezentaci slov jakožto hustých vektorů reálných čísel v omezeném vektorovém prostoru. Každé vstupní slovo je poté namapováno přesně na jeden vektor, jehož hodnoty jsou natrénovány často spolu se samotnými velkými jazykovými modely (a většinou tvoří první vstupní vrstvu dat do modelů).

Techniky vnořování jsou založeny na tzv. distribuční hypotéze, která tvrdí, že slova jenž se vyskytují a používají ve stejných kontextech často mívají podobný význam či jsou synonymní [17]. Díky této skutečnosti jsou modely schopné tvořit spojitostí i napříč přeformulovanými kontexty a větnými stavbami.

V rámci zvyšování efektivity využití velkých jazykových modelů k překladu do příkazů jazyka SQL je předávání kontextu modelu v položené otázce v přirozeném jazyce ovšem komplexní disciplína. Kontext pro nás z většiny tvoří informace o struktuře databáze, jejím schématu. Tato disciplína předávání se nazývá *Schema linking* nebo *Schema linking problém*. Jedním z řešení této problematiky je připojení vhodné reprezentace databáze k samotnému dotazu

Schema linking problém - Připojení k dotazu

Řešení Schema linking problému připojením samotné databáze je nejpřímočařejším řešením předání kontextu celé databáze modelu. Prvotní snahy pro tvorbu kvalitní serializované reprezentace databázových struktur probíhaly pomocí tzv. *n-gram modelů* [18]. Tyto modely jsou již v dnešní době nahrazeny RNN, které jsou zobecněním n-gramových modelů, ovšemže metodologie připojování samotného dotazu se prokázala velice slibnou a dokonce při vhodném užití šablony dotazů mohou modely dosahovat inferenční schopnost blízkou lidské schopnosti překládat dotazy [19].

Výpis kódu 3.1 Příklad serializované tabulky

```
CREATE TABLE price( `ID` integer PRIMARY KEY, `price` real );
Example:
+----+-------------+
| ID |    price    |
+----+-------------+
| 1  | 25561.59078 |
| 2  | 24221.42273 |
| 3  | 27240.84373 |
+----+-------------+
```

I přestože je tento postup, jak již bylo zmíněno, velice silný, tak není perfektní, jelikož v případě větších databází tento přístup vnáší do samotného požadavku na model spoustu šumu ve formě irelevantních tabulek a sloupců. Tento šum se, díky narušení pozornostního mechanismu, poté přímo projevuje v přesnosti výsledného modelu. Snaha o zbavení se tohoto šumu postupně vedla k rozvoji technik dekompozice počátečních dotazů.

Schema Linking Problem - Decomposition of Initial Queries

Techniky dekompozice počátečních dotazů spočívají na principech rozděl a panuj. Jednou z nejpopulárnějších a nejlepších technik je využití samotných velkých jazykových modelů jako modelů v multiagentních systémech. Průkopnickým multiagentním systémem se při svém vydání stal MAC-SQL [20], jenž se skládá ze tří agentů, jmenovitě *Selector*, *Decomposer* a *Refiner*.

Tito agenti fungují sekvenčně. Počáteční zpracování dat provádí agent Selector, který slouží jako mechanismus pro tvorbu kvalitního kontextu. Ze samotného dotazu, s pomocí expertních znalostí, extrahuje nejdůležitější části schématu pro překlad přirozeného dotazu do strukturovaného jazyka SQL.

Druhým agentem v pořadí je Decomposer, který se pomocí výstupního kontextu relevantních sloupců a tabulek od agenta Selector snaží o dekompozici úvodního dotazu do menších poddotazů pro zlepšení efektivity výstupu [21]. Pro každý z těchto poddotazů je poté vytvořena vhodná odpověď ve strukturovaném jazyce SQL. Tyto páry se postupně kumulují v seznamu párů dotaz-odpověď od nejvíce atomického až po celkový vstupní dotaz spolu s jeho odpovědí.

Posledním článkem tohoto systému je agent Refiner, který má k dispozici kromě vybraného velkého jazykového modelu také externí nástroje umožňující exekuci finálního dotazu nad přidruženou databází. Agent sám poté na základě definovaných kritérií vyhodnotí, zda je dotaz adekvátní odpovědí na prvotní dotaz.

Tato metoda a obecně metody dekompozice počátečních dotazů dosahují v jejich finálních vyhodnoceních lepších výsledků. Problematický je ovšem předpoklad externích expertních znalostí, který není v rámci reálných využití možné naplnit stabilně. Tento předpoklad lze do jisté míry překonat pomocí systematického zlepšování modelů pro specifickou aplikaci

3.2.2 Systematické zlepšování modelů

V průběhu systematického zlepšování modelů se snažíme o zlepšení výstupu na základě specializace samotných modelů na danou problematiku. V kontextu disciplíny překladu přirozeného jazyka do strukturovaného jazyka SQL pomocí velkých jazykových modelů to znamená zaměření jejich obecných schopností pro generaci výstupních sekvencí na pouhý překlad.

Jelikož jsou velké jazykové modely trénovány na obrovských korpusech textových dat, jsou schopny odpovídat na širokou škálu dotazů napříč různými disciplínami [22]. Tato schopnost je ovšem u transformerových modelů, které jsou momentálně průmyslovým standardem pro velké jazykové modely, nežádoucí kvůli inherentní roztroušenosti pozornostního mechanismu na irelevantní sémantické spojitosti. Jedním z nejslibnějších východisek, jak tuto nežádoucí vlastnost potlačit, je proces zvaný dolaďování (angl. *Finetuning*).

Supervizované dolaďování

Díky mohutnosti velkých jazykových modelů je jejich přetrénování na specifické úkoly časově nevýhodné. Východiskem je právě technika dolaďování modelů, která je de facto průmyslovým standardem pro modifikaci již přetrénovaných modelů na specifické úkoly požadované uživatelem. Technika spočívá v tvorbě malé datové sady (většinou zlomek velikosti oproti sadě, na které byl model původně trénován) k přetrénování již natrénovaných parametrů právě na nějaký specifický úkol. Tato datová sada je poté využita spolu s vhodnou šablonou pro nasměrování modelu správným směrem. Součástí této šablony jsou zpravidla následující části:

- **Instrukce** - Slouží k určení úkolu modelu při inferenci a případně předání dodatečných znalostí.
- **Vstup** - Vstupní větná sekvence.
- **Výstup** - Výstupní sekvence spojená se vstupem na základě instrukce.

Výpis kódu 3.2 Šablona pro doladění LLM dle šablony Standford Alpaca [23]

```
### Instruction:
Find out which city is the capital city of given country.
If it is not a country reply with 'Not␣a␣country' otherwise
reply with only the name the capital city and nothing more.

### Input:
Czech Republic

### Output:
Prague
```

Po samotném procesu dolaďování je model promptován obdobnou vstupní sekvencí jako ve šabloně, vyjma vynechání samotného výstupu. Model, díky dolaďování na základě instrukcí a vstupních dat (tj. v případě disciplíny NL2SQL žádostí o překlad přirozeného jazyka na strukturovaný jazyk SQL), výstup doplní. Metoda prostého dolaďování je ovšem stále velmi časově a výpočetně náročným úkolem. Tuto problematiku se snaží vyřešit metoda LoRA (Low-Rank Adaptation).

LoRA (Low-Rank Adaptation)

Metoda LoRA [24] spočívá v rozkladu hustých vrstev modelu do série řídkých matic. Přesněji pro množinu dvojic skládajících se z vstupních a výstupních sekvencí tokenů $\mathcal{Z} = (x_i, y_i), i \in \hat{N}$ kóduje informaci o váhách pomocí množiny parametrů Θ, pro kterou platí $|\Theta| \ll |\Phi_0|$, kde Φ_0 jsou prvotní natrénované váhy. Proces ladění a nalezení ideálních vah $\Delta\Phi$ se poté stane optimalizační úlohou hledání optimální Θ pomocí následujícího vztahu:

$$\max_{\Theta} \sum_{(x,y)\in\mathcal{Z}} \sum_{t=1}^{|y|} \log(P_{\Phi_0+\Delta\Phi(\Theta)}(y_t|x, y_{<t})) \tag{3.1}$$

kde $P_\Phi(y|x)$ je parametrizovaný model parametrem Φ. Tento přístup se ukázal jako velmi efektivní díky dolaďování pouze množiny parametrů $|\Theta|$, kde v jistých případech platí až vztah

$$\frac{|\Theta|}{|\Phi_0|} \approx 0.001 \tag{3.2}$$

Díky této skutečnosti je možné dolaďovat modely s parametry čítajícími miliardy, i na slabších systémech, což by jinak nebylo možné.

Kapitola 4

Příprava modelů

Tato kapitola pojednává o přípravě modelů, které budou užity ve finální implementaci webového rozhraní.

4.1 Výběr datové sady

Výběr datové sady je vitálním krokem v rámci jakéhokoliv kroku užití libovolného modelu (jsou-li vstupní data potřebná) – ať se jedná o jeho trénování, dolaďování či konečnou inferenci, klasifikaci, regresi či jakýkoliv jiný finální krok daného modelu. Proto je před samotným výběrem datové sady potřebné si prvotně ustanovit kritéria, která datová sada splňuje, aby bylo docíleno co největší efektivity.

4.1.1 Kritéria výběru

V rámci výběru datové sady bylo zkoumáno mnoho možností na základě různorodých kritérií, ale jako nejdůležitější kritéria byly poté vybrány následující:

- **Rozmanitost dotazů** – Na datovou sadu byl kladen důraz, aby obsahovala široké spektrum dotazů, od těch nejjednodušších až po složité dotazy, které interně na straně strukturovaných dotazů vyžadují pokročilejší funkce relačních databází.
- **Zastoupenost dat z reálného světa v datové sadě** – Další důležitý aspekt pro datovou sadu bylo to, zda reflektuje situace, které mohou nastat v reálném životě. Data na produkčních databázových systémech často nemají unifikovaný charakter, a proto trénování a evaluace modelu na datech, která nereflektují tuto skutečnost, by bylo v rozporu s cílem této práce.
- **Kvalita anotace** – Kvalitní anotace je další charakteristikou zvolené datové sady. Jelikož v dolaďování modelu je užito paradigma učení s učitelem (viz další sekce), nekvalitní anotace může drasticky ovlivnit kvalitu výstupního modelu.

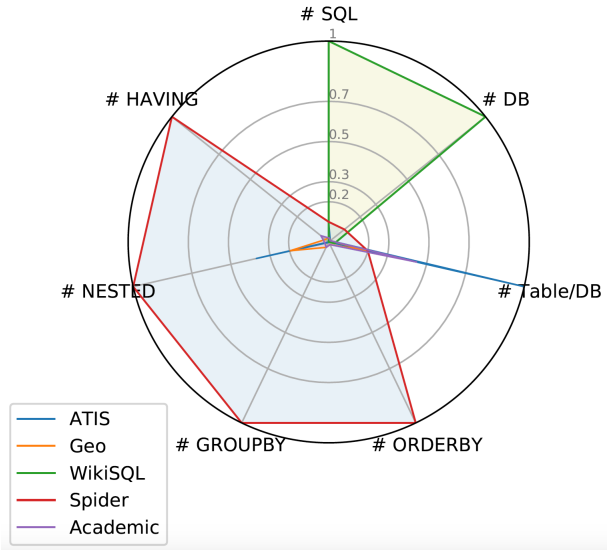
- **Vhodnost datové sady na N2SQL** – Finální vlastností datové sady je ta, že je tvořena na účel překladu přirozeného jazyka do jazyka SQL, na což se váže i aspekt kvalitní anotace.

4.1.2 WikiSQL

Z klasických datových sad se poté nabízela datová sada *WikiSQL*[25]. Tato datová sada obsahuje přes 25 000 tabulek spolu s více než 80 000 anotovanými dotazy. Problematická je ovšem samotná struktura daných dotazů spolu s chybovostí anotace. Struktura dotazů je po překladu z položeného dotazu do strukturovaného jazyka SQL velice elementární a obsahuje pouze klauzule $SELECT$, $WHERE$ a $FROM$[26], které nedokážou zachytit komplexnější vztahy reálných dat. Tato datová sada byla proto také zamítnuta.

4.1.3 Spider

Jako velice slibné východisko se tvářila datová sada Spider [27], která byla sestavena přesně dle kritérií, jež byly na začátku výběru datové sady v rámci metodologických postupů uloženy.



Obrázek 4.1 Struktura dotazů v datové sadě Spider oproti obdobným datovým sadám na NL2SQL problém, převzato z [28]

V této datové sadě, jak lze vidět z obrázku 4.1, již byl vyřešen problém pouze elementárních dotazů. Jediný problém, který tato datová sada obnáší, je její časté užití přímo žádaných hodnot v jednotlivém dotazu, jak lze vidět na příkladu 4.1. Tento problém do jisté míry modelu ulehčuje hledání sémantických spojitostí v otázce přirozeného jazyka. Z tohoto důvodu tato datová sada nebyla vybrána pro dolaďovací účely, avšak bude také využita pro evaluaci finálního modelu. Idea tohoto testování je předpoklad

na uživatelské straně. I přestože se neočekává, že vyvinuté prostředí bude užíváno odborníkem, tak se předpokládá, že uživatel bude alespoň do jisté míry mít povědomí o datech, na která se dotazuje, díky čemuž může nastat situace nezáměrně simulovaná charakterem dotazů v datové sadě Spider, která nám umožní do jisté míry odhadnout efektivitu modelu v případě základních expertních znalostí využívaného systému ovládaného uživatelem prostředí. Samotná evaluace probíhá na základě přidružené sady testovacích nástrojů.

Výpis kódu 4.1 Příklad dotazů datové sady Spider

```
--- Query
List the name of clubs that do not have players.
--- Gold SQL
SELECT Name FROM club
    WHERE Club_ID NOT IN (SELECT Club_ID FROM player);
```

4.1.4 BIRD

Poslední datová sada, která byla zvažována[1], byla BIRD (celým názvem ***BI****g bench for la****R****ge-scale* ***D****atabase grounded in text-to-SQL tasks*) [29]. Tato datová sada byla záměrně vytvořena z reálných dat, s komplexními pojmenováními sloupců spolu s jejich neočekávanými datovými typy. Tento přístup docílí nutnosti uchopení jemných sémantických nuancí k tvoření kvalitních strukturovaných dotazů. Bohužel, tento postup tvoření párů dotazů a strukturovaných odpovědí v jazyce SQL má také jeden velký problém – data jsou v momentální podobě až příliš náročná. Samotní autoři datové sady tvrdí, že velké jazykové modely v jejich momentální kapacitě nemají dostatečné prostředky pro překlad dotazů v přirozeném jazyce do jazyka SQL, a dokonce toto tvrzení bylo postaveno na faktu, že ke každému z dotazů v přirozeném jazyce je přiložena část expertních znalostí. Tento předpoklad je v reálném světě nerealistický a nemůže být na něm stavěno.

Výpis kódu 4.2 Příklad komplexních názvosloví sloupců v datové sadě BIRD

```
CREATE TABLE frpm (
CDSCode TEXT NOT NULL PRIMARY KEY,
Academic Year TEXT,
County Code TEXT,
--- Omitted for readability
Free Meal Count (Ages 5-17) REAL,
Percent (%) Eligible Free (Ages 5-17) REAL,
FRPM Count (Ages 5-17) REAL,
Percent (%) Eligible FRPM (Ages 5-17) REAL,
2013-14 CALPADS Fall 1 Certification Status INTEGER
);
```

[1] Vyjma ostatních historicky nezmíněných datových sad, které naráží na stejné problémy jako datová sada WikiSQL

Datová sada BIRD má ovšem i jednu velikou výhodu, a to přidruženou sadu testovacích nástrojů, které poskytují dvě metriky: exaktní totožnost vrácených dat a míru shody vrácených dat. Díky těmto metrikám spolu s náročností dat byla proto tato datová sada vybrána a bude na ní jak dolaďována, tak testována. U samotné evaluace není očekávána velká úspěšnost v jakékoliv z metrik, jelikož nejvyšší úspěšnost se bez expertních znalostí pohybuje okolo 20 %, avšak je předpokládáno, že díky ladění na této datové sadě budou modely schopny vnímat i složité závislosti.

4.2 Užité techniky pro zlepšení efektivity modelu

Jelikož modely ve své základní formě nejsou stavěny na překlad přirozeného jazyka do strukturovaného jazyka SQL, bylo prvotně potřeba modely na tento úkol připravit. Samotné modely proto byly dolaďovány pomocí knihovny Unsloth, specificky její Open verze[2].

Jako základní modely byly zvoleny již diskutované modely Llama-3-8B a Mistral-7B, ovšem v jejich kvantizované verzi pomocí knihovny Bits and Bytes[3]. Tato verze modelu má sice nižší finální přesnost, ale za to je dosáhnuta drastická redukce potřebných výpočetních prostředků. Ke kvantizovaným modelům byly přidány LoRA adaptéry, specificky jejich nadřazená verze QLoRA (Quantized Low-Rank Adaptation), jelikož nativní LoRA adaptéry nepodporují kvantizované modely. Modely byly načteny jakožto PEFT (*Parameter-Efficient Fine-Tuning*) modely s následující konfigurací:

Parametr	**Hodnota**
r	64
taget_modules	q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj
lora_alpha	32
lora_dropout	0
bias	None
use_gradient_checkpointing	unsloth
random_state	42

Tabulka 4.1 Parametry pro načtení PEFT modelu

Kde jednotlivé hodnoty jsou vybrány jako ideální pro generalizaci modelu v rámci specifického úkolu, v případě této práce překladu přirozeného jazyka do strukturovaného jazyka SQL [24].

4.3 Předpracovávání dat

V rámci předpracovávání dat byly prvotně serializovány všechny databáze z testovací sady datové sady BIRD [29] do příkazu DDL, který by bylo možné vyvolat pro vytvoření všech tabulek v dané databázi, spolu s datovým vzorkem.

[2] https://github.com/unslothai/unsloth
[3] https://github.com/TimDettmers/bitsandbytes

Samotný datový vzorek byl tvořen zpravidla prvními třemi záznamy v dané tabulce pro danou databázi a skládal se z hodnot všech sloupců vyjma sloupců s datovým typem Blob (Binary Large Object), který je určen pouze pro binární reprezentaci komplexnějších datových objektů (jako např. fotky). Tato reprezentace je zpravidla svým počtem tokenů outlier a bez použití technik rozšiřování kontextu (jako například RoPE škálování [30]) není možné datový vzorek obsahující i tuto informaci předat modelu, nehledě na inherentní snížení přesnosti modelu v důsledku použití těchto metod spolu s irelevantností těchto dat pro extrapolaci sémantických významů z původního dotazu díky binární povaze těchto dat.

Mimo serializaci databází a tvoření datových vzorků nebyla použita žádná jiná technika předpracovávání dat díky kvalitní anotaci trénovací datové sady.

Výpis kódu 4.3 Serializovaná databáze spolu s datovým vzorkem

```
CREATE TABLE Highschooler(
`ID` number PRIMARY KEY,
`name` text,
`grade` number );
\FIRST THREE ROWS DATA SAMPLE:
(1510, 'Jordan', 9)
(1689, 'Gabriel', 9)
(1381, 'Tiffany', 9)
... Omitted
);
```

4.4 Dolaďování

Dolaďování modelů proběhlo na základě předzpracovaných dat z trénovací sady datové sady BIRD [29], ze které se udělal vzorek 300 datových bodů. Samotný proces dolaďování proběhl v rámci placeného prostředí Google Colab za pomocí technologie CUDA na grafické kartě NVIDIA A100-SXM4-40GB, která byla v průběhu celého dolaďování téměř stoprocentně vytížená.

Před samotným dolaďováním byly prvotně vyřazeny datové body, jejichž počet tokenů po naplnění dolaďovací šablony 4.4 příslušnými daty po tokenizaci přesahoval rozsah kontextového okna obou modelů, tj. byl větší než 8192.

Výpis kódu 4.4 Šablona pro doladění inspirována šablonou Standford Alpaca [23]

```
### Instruction:
Translate the input question into SQLite SQL.
Always abide by these rules:

1. Analyse `TABLE CREATION SCRIPTS AND DATA SAMPLES` for context.
2. ALWAYS enclose columns in backticks (`).
The generated output is otherwise USELESS.
3. ALWAYS end your generated SQL with semicolon (;).

### TABLE CREATION SCRIPTS AND DATA SAMPLES
{}

### Input:
{}

### Output:
{}
```

V rámci dolaďování byl využit *Supervised Fine-Tuning Trainer*[4] z knihovny *trl* společnosti Hugging Face s následujícími parametry:

Parametr	Hodnota
per_device_train_batch_size	4
gradient_accumulation_steps	2
warmup_steps	100
learning_rate	2e-5
num_train_epochs	10
optimizer	adamw_8bit
lr_scheduler_type	linear

Tabulka 4.2 Parametry SFTTrainer

4.5 Evaluace doladěných modelů

Po procesu doladění byla finálně vyhodnocena přesnost obou dvou modelů. V rámci evaluace byly zvoleny již zmiňované datové sady Spider a BIRD. Evaluace proběhla s pomocí šablony 4.4 s otevřeným koncem v rámci sekce *Output*. Ke všem dotazům, které byly předány modelu k inferenci, byla přidána informace o samotné databázi spolu s datovým vzorkem dle příkladu 4.3, ke které se dotaz vztahoval.

4.5.1 Datová sada Spider

K evaluaci přeložených vstupních sekvencí (dotazů) bylo použito prostředí *test-suite-sql-eval* [31].

[4] https://huggingface.co/docs/trl/en/sft_trainer

Doladěný model Llama 3-8B-4bnb dosáhl následujících přesností překladu přirozeného jazyka do jazyka SQL:

Metric	Easy	Medium	Hard	Extra	All
Count	248	446	174	166	1034
Execution Accuracy					
Execution	0.504	0.374	0.293	0.259	0.373
Exact Matching Accuracy					
Exact Match	0.323	0.135	0.040	0.000	0.142

Tabulka 4.3 Výsledky modelu Llama 3-8B-4bnb na datové sadě Spider

Doladěný model Mistral 7B-4bnb dosáhl následujících přesností překladu přirozeného jazyka do jazyka SQL:

Metric	Easy	Medium	Hard	Extra	All
Count	248	446	174	166	1034
Execution Accuracy					
Execution	0.391	0.300	0.276	0.265	0.312
Exact Matching Accuracy					
Exact Match	0.153	0.056	0.006	0.000	0.062

Tabulka 4.4 Výsledky modelu Mistral 7B-4bnb na datové sadě Spider

Bohužel, tyto výsledky nejsou ideální. Jedním z hlavních možných bude samozřejmě nedostatek expertních znalostí daného systému, vůči kterému byly tyto dotazy položeny. Přikládání expertních znalostí k daným dotazům není pro případ použití této práce užitečné díky nerealistickému předpokladu nabytí těchto informací v případě práce s reálnými daty, jak již bylo diskutováno sekci 3.2.1.

Sekundárním možným problémem je kvantizace samotných modelů, která, i přes převážné zachování kvality výstupní inference, do jisté míry stále snižuje přesnost. Bez užití kvantizačních technik není možné modely realisticky využít bez obrovských dedikovaných prostředí s výpočetní kapacitou přesahující 200 GB VRAM paměti, což je opět v rozporu s cílem této práce vytvořit jednoduché nenáročné prostředí pro prvotní analýzu dat.

Třetím potenciálním možným důvodem této nízké přesnosti je samotný proces doladťování pomocí nevhodné šablony či hyperparametrů. Jediným možným východiskem by bylo lépe zkoumat vhodnost daných parametrů pro tento úkol, ale i zde narážíme na problém nároků na výpočetní techniku.

Z těchto důvodů bylo rozhodnuto, že dosažené stupně přesnosti budou v rámci této práce považovány za dostatečné.

4.5.2 Datová sada BIRD

V rámci evaluace nad datovou sadou BIRD byly použity přidružené skripty v samotném GitHubovském repozitáři dané datové sady.

Doladěný model Llama 3-8B-4bnb dosáhl následujících přesností překladu přirozeného jazyka do jazyka SQL:

Metric	Simple	Moderate	Challenging	All
Count	925	465	144	1534
Execution Accuracy				
Execution	0.0876	0.0473	0.0208	0.0691
Valid Efficiency Score (VES)				
VES	0.0647	0.0419	0.0155	0.0532

Tabulka 4.5 Výsledky modelu Llama 3-8B-4bnb na datové sadě BIRD

Doladěný model Mistral 7B-4bnb dosáhl následujících přesností překladu přirozeného jazyka do jazyka SQL:

Metric	Simple	Moderate	Challenging	All
Count	925	465	144	1534
Execution Accuracy				
Execution	0.0973	0.0452	0.0278	0.0750
Valid Efficiency Score (VES)				
VES	0.0402	0.0246	0.0616	0.0375

Tabulka 4.6 Výsledky modelu Mistral 7B-4bnb na datové sadě BIRD

Výsledky obou modelů na datové sadě BIRD jsou o dost horší. Tyto výsledky jsou, jak již bylo avizováno, nejspíše způsobeny potřebou expertních znalostí pro jakékoliv lepší než velice špatné výsledky. Jak lze vidět na příkladu 4.5 níže, který byl klasifikován tvůrci datové sady BIRD jako „Simple“, tak dotazy jsou dosti komplikovaného charakteru i se samotnými expertními znalostmi.

Co je ovšem překvapivé, je že model Mistral 7B, i přes jeho menší počet parametrů, dosahuje překvapivě lepších výsledků u této datové sady. Tento fakt je možná způsoben jiným korpusem, na kterém byl model trénován, popřípadě také možností, že model byl trénován právě na šabloně Stanford Alpaca[23], což spolu s doladťovacím procesem pomohlo modelu lépe se aklimatizovat na styl dotazů nad rámec datové sady BIRD k extrakci komplikovaných sémantických spojitostí i navzdory nepřipojeným expertním znalostem. Bohužel, tento odhad je pouze spekulativní povahy a nelze přesně určit, z jakého důvodu je přesnost tohoto modelu vyšší.

Výpis kódu 4.5 Příklad Gold SQL vs Odhadnutého dotazu + Expertí znalost

```
### Oracle knowledge:
### Eligible free rate for K-12 =
###    `Free Meal Count (K-12)` /
###    `Enrollment (K-12)`

#Question:
What is the highest eligible free rate for K-12
students in the schools in Alameda County?

### Predicted query
select max(t1.percent_())
    from schools as t1
    inner join
    frpm as t2 on t1.cds = t2.cdscode
        where t1.county = 'alameda'

### Gold SQL
SELECT `Free Meal Count (K-12)` / `Enrollment (K-12)`
    FROM frpm WHERE `County Name` = 'Alameda'
        ORDER BY (
            CAST(
                `Free Meal Count (K-12)` AS REAL) /
                `Enrollment (K-12)`)
        DESC LIMIT 1
```

Na závěr této kapitoly lze říci, že ačkoli dosažené výsledky modelů Llama-3-8B-4bnb a Mistral 7B-4bnb nejsou ideální, ukazují určitý potenciál. Přesnost modelů při překladu přirozeného jazyka do SQL může být ovlivněna několika již zmíněnými faktory, jako je nedostatek expertních znalostí, kvantizace modelů a nevhodné dolaďovací šablony či hyperparametry. Přesto však výsledky dosažené na datových sadách Spider a BIRD poskytují cenné poznatky pro další optimalizace a vývoj systémů na základě těchto modelů..

Kapitola 5

Návrh a implementace webové stránky

Tato kapitola se zabývá návrhem a implementací webového rozhraní pro využití modelů, jejichž příprava byla popsána v předchozí kapitole.

5.1 Návrh architektury systému

Architektura systému se skládá ze dvou primárních komponent:

- Front-end
- Back-end

Tyto dvě komponenty architektury, díky zvoleným technologiím, mohou mezi sebou plynule komunikovat, čímž je zajištěn plynulý chod celé webové aplikace. V následujících sekcích bude tento návrh představen podrobněji.

5.1.1 Front-end

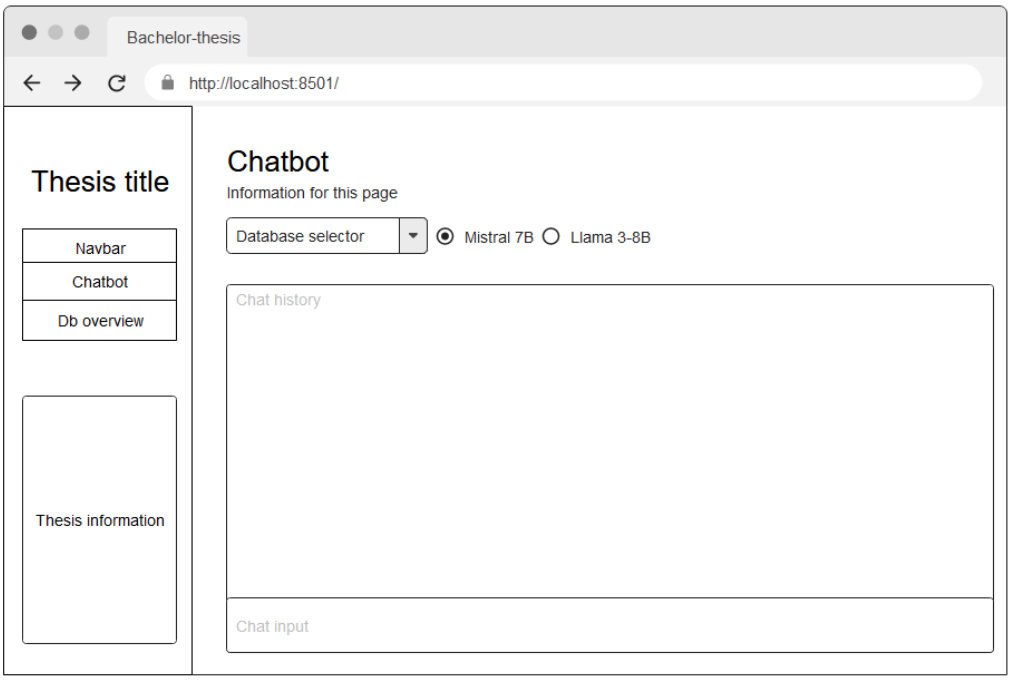
Front-end představuje uživatelské rozhraní, skrze které budou uživatelé komunikovat se systémem pro překlad přirozeného jazyka do příkazů SQL. Pro vývoj této komponenty byl zvolen framework Streamlit[1], jenž je postaven na základě jazyka Python.

5.1.1.1 Design a UX

V rámci designu uživatelského rozhraní byl kladen primární důraz na přehlednost, která uživateli umožní rychlé dotazování pomocí přirozeného jazyka nad požadovanou da-

[1] https://streamlit.io/

tabází. Webová stránka samotná se skládá ze dvou stránek: „Chatbot“ a „Přehled databází“.

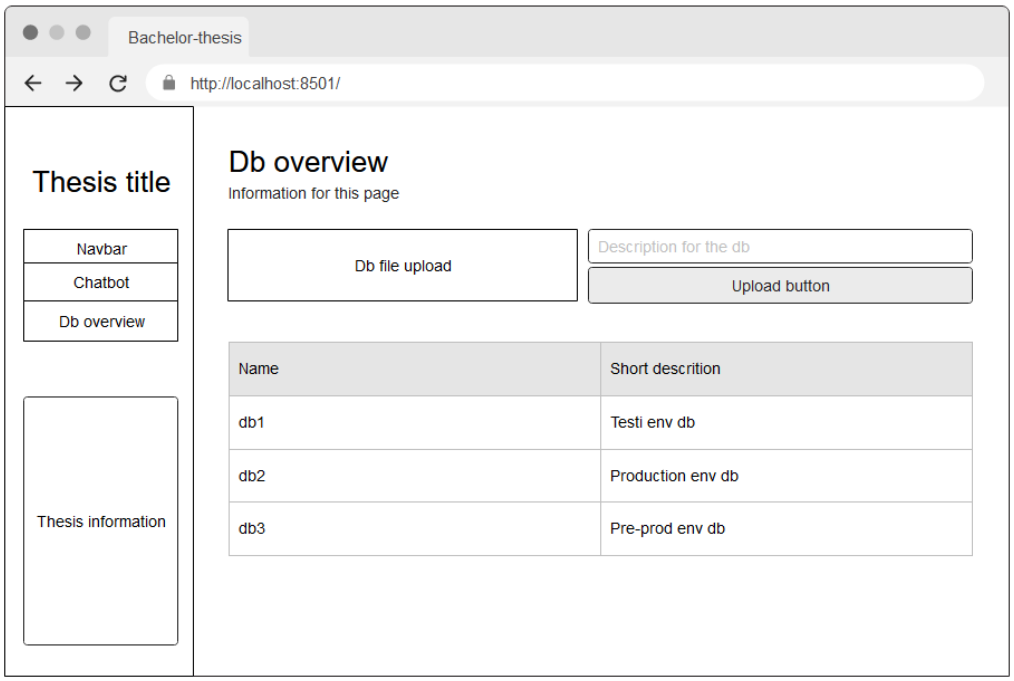


Obrázek 5.1 Mock-up stránky Chatbot

Pro hlavní stránku pro interakci s vybraným z modelů „Chatbot“ byl v prvotní fázi návrhu vytvořen návrhový mock-up 5.1. V rámci tohoto mock-upu se tato stránka skládala z možnosti výběru některé z nahraných databází (v rámci stránky „Přehled databází“), výběru jednoho z vybraných velkých jazykových modelů a konečně interaktivního chatu pro pokládání dotazů v přirozeném jazyce.

V rámci tvoření mock-upů pro design výsledné stránky byl vytvořen i mock-up pro podpůrnou stránku „Přehled databází“ 5.2. Tato stránka se skládá primárně z mechanismu nahrávání zvolených SQLite databází s volitelnou možností přidání popisu databáze. Sekundární využití této stránky je poté zobrazení jednotlivých již nahraných databází spolu s jejich popisy v případě jejich existence.

Tento přehled databází nabízí uživatelům možnost snadného přístupu a správy jejich nahraných databází. Po nahrání se databáze okamžitě objeví v seznamu dostupných databází na stránce. U každé databáze v tomto seznamu je uveden název databáze a volitelně připojený popis.



Obrázek 5.2 Mock-up stránky Přehled databází

5.1.2 Back-end

Back-end této webové aplikace byl navržen s hlavní myšlenkou jednoduchosti a modularity. K dosažení těchto principů je proto využit pouze *Session state*[2] z již zmiňovaného frameworku Streamlit. Na základě tohoto *Session state* jsou poté plynule předávány informace z komponenty „Front-end“ do komponenty „Back-end“.

5.2 Implementace webové aplikace

Implementace webové stránky proběhla velice přímočaře na bázi souběžného vývoje komponent „Front-end“ a „Back-end“ díky povaze frameworku Streamlit, jenž umožňuje jednoduché prototypování a vývoj na „obou frontách“. Specificky využité komponenty a interakce s modely budou přiblíženy v následujících podsekcích.

5.2.1 Front-end

Front-endová část celé aplikace byla postavena z velké většiny z nativních komponent frameworku Streamlit dle původních mock-up návrhů. V rámci samotného vývoje byly ovšem využity i komponenty z volně dostupného *component marketplace* frameworku Streamlit, jmenovitě to byly tyto komponenty:

[2] https://docs.streamlit.io/develop/api-reference/caching-and-state/st.session_state

1. **AgGrid**[3] – Nadstavba nad stejnojmennou populární JavaScriptovou komponentou. Tato komponenta slouží jako součást odpovědi velkého jazykového modelu pro efektivní prezentaci tabulovaných dat s nativní podporou filtrace dle široké škály filtrů.
2. **OptionMenu**[4] – Jednoduchá komponenta sloužící k výběru jednotlivých stránek v rámci boční navigační lišty.
3. **Streamlit Pandas Profiling**[5] – Nadstavba nad Pythonovskou knihovnou *ydata_profiling* sloužící k tvorbě komplexních reportů, které jsou součástí odpovědí v rámci stránky „Chatbot“.

V rámci vývoje byla také použita technika injektování HTML kódu díky limitacím frameworku Streamlit pro využití externích JavaScriptových skriptů či stylizování pomocí CSS souborů.

Výpis kódu 5.1 Injekce HTML kódu v frameworku Streamlit

```
st.markdown(
"""
<style>
    [data-testid="baseButton-header"] {
        display: none
    }
</style>
""",
    unsafe_allow_html=True)
```

Finálně byly všechny komponenty připojeny na mechanismus *Session state*, jenž volně předal informace do komponenty architektury „Back-end“.

5.2.2 Back-end

Komponenta architektury „Back-end“ byla navržena tak, aby reagovala na informace předané z komponenty architektury „Front-end“ pomocí mechanismu *Session state*. Pro samotné využití primární stránky „Chatbot“ je uživatel nejprve vyzván k nahrání databáze pomocí podpůrné stránky „Přehled databází“. Na této stránce je po nahrání souboru nejprve zkontrolováno, zda nahraný soubor je skutečně databázový soubor SQLite pomocí jednoduchých regulárních výrazů. Po jeho samotném nahrání je tento soubor uložen lokálně.

Jakmile je tento soubor uložen, je zpřístupněn hlavní chatovací mechanismus stránky „Chatbot“. Tento mechanismus ovšem není funkční do doby zvolení jednoho z dostupných velkých jazykových modelů. Po jeho zvolení je uživateli umožněno pokládat dotazy nad vybranou databází.

Po samotném položení dotazu je nejprve uživatelův dotaz přeformulován dle šablony 4.4 a následně poslán jako *payload* na dedikovaný *endpoint* vybraného velkého jazy-

[3] https://pypi.org/project/streamlit-aggrid/
[4] https://pypi.org/project/streamlit-option-menu/
[5] https://pypi.org/project/streamlit-pandas-profiling/

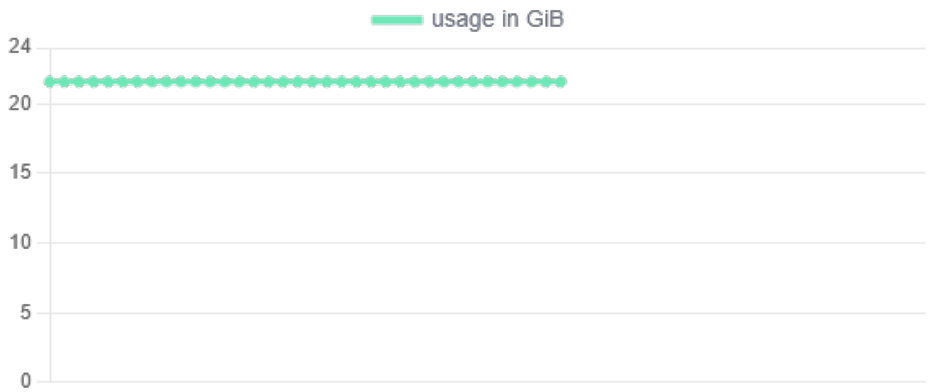
kového modelu, který dotaz přeloží do jazyka SQL, jenž je následně užit pro získání dat z vybrané databáze. Tato data jsou poté prezentována uživateli pomocí již zmíněných komponent „AgGrid“ a „Streamlit Pandas Profiling“ v rámci odpovědi v oblasti chatovacího okna spolu s přeloženým dotazem do jazyka SQL.

Dedikované endpointy byly v rámce vývoje hostovány pomocí ekosystému *Hugging Face*, konkrétně služby *Inference Endpoints*. Adresy těchto endpointů jsou editovatelné pomocí JSON souboru `config.json`, jenž se nachází v kořenové složce webové aplikace (viz Obsah příloh).

Na těchto *endpointech* v rámci vývoje poté byly nasazeny samotné doladěné modely, jenž jsou dostupné v rámci veřejným repozitáři s následujícími adresami:

- **Llama 3-8B** – https://huggingface.co/Fallperino/llama3-FT-8b-Bird-4bnb-ctxs
- **Mistral 7B** – https://huggingface.co/Fallperino/mistral-FT-7b-Bird-4bnb-ctxs

Oba dva *endpointy* měli dedikovanou grafickou kartu Nvidia A10G s 24 GB VRAM, která je hraničně dostačující, jak lze vidět z obrázku 5.3.



Obrázek 5.3 Využití VRAM v momentech inference v rámci dedikovaných endpointů

5.2.3 Testování

Testování této aplikace proběhlo lokálně na databázích z datové sady Spider se zařízením s následujícími parametry:

- **GPU** – Nvidia GeForce GTX 1650, 4 GB VRAM
- **RAM** – 8 GB DDR4
- **CPU** – Intel® Core i5-9300H

Navzdory poněkud slabšímu systému proběhlo testování úspěšně díky offloadování nejnáročnějších výpočetních operací na GPU jenž jsou hostované v rámcí zmiňovaných Hugging Face endpointů.

Testování probíhalo manuálně, vždy nahráním vybraného databázového souboru a dotazováním pomocí přirozeného jazyka. V rámci jednoduchých dotazů byl systém téměř stoprocentně úspěšný, ale u složitějších dotazů bohužel úspěšnost klesala kvůli limitacím překladového jádra skládajícího se z velkých jazykových modelů.

5.3 Návod na spuštění webové aplikace

K spuštění webové aplikace jsou zapotřebí následující prerekvizity:

- Nainstalovaná poslední verze programovacího jazyka Python.
- Pár adres dedikovaných endpointů na kterých jsou hostované repozitáře dolaďěného modelu Llama 3-8B 5.2.2 a Mistral 7B 5.2.2 ke kterým má uživatel přístup.

Po naplnění těchto prerekvizit stačí naplnit `config.json` s jednotlivými adresami a následné zavolání těchto příkazů pomocí příkazové řádky v kořenové složky webové aplikace:

Výpis kódu 5.2 Instalace balíčku

```
$ pip install -r requirements.txt
$ pip install ydata_profiling
```

Instalace balíčků musí probíhat takto odděleně kvůli problematické provázanosti balíčků v rámci knihovny `streamlit_pandas_profiling`, která závisí na starší verzi této knihovny pojmenované `pandas_profiling`.

Po nainstalování všech potřebných balíčků poté stačí použít příkaz

```
streamlit run app.py [ACCESS TOKEN]
```

a aplikace bude spuštěna v rámci defaultního prohlížeče na adrese `localhost:8501`.

Kapitola 6

Závěr

Tato práce se zaměřila na vývoj webové aplikace, která využívá velké jazykové modely (LLM) pro překlad dotazů v přirozeném jazyce do strukturovaného dotazovacího jazyka SQL. Výsledná aplikace je navržena tak, aby poskytovala uživatelům snadný a efektivní způsob interakce s databázemi bez potřeby hlubokých znalostí programování nebo SQL.

6.1 Shrnutí výsledků a cílů práce

V úvodní kapitole byla objasněna motivace pro vytvoření aplikace a stanoveny cíle práce. Bylo zdůrazněno, že růst objemu dat vyžaduje nové metody pro jejich interpretaci a prezentaci, které jsou snadno přístupné i pro uživatele bez hlubokých znalostí programování.

Kapitola o teoretickém základu poskytla přehled historie a vývoje neuronových sítí, rekurentních neuronových sítí (RNN) a modelů typu Transformer. Popsala klíčové mechanismy, jako je pozornost (Attention) a vícenásobná pozornost (Multi-Head Attention), které umožňují efektivní zpracování přirozeného jazyka.

Ve třetí kapitole byly představeny modely Llama 3 a Mistral 7B, které byly v této práci použity. Byly také diskutovány techniky zlepšování efektivity těchto modelů, včetně zlepšování vstupních dat a systematického dolaďování modelů pomocí metod jako Supervizované dolaďování a LoRA (Low-Rank Adaptation).

Následující kapitola se věnovala samotnému před-připravení modelů na jejich finální užití v rámci webové aplikace. Prvotně byly vybrány vhodné datové sady na kterých byly modely následně dolaďovány. Po samotném procesu doladění proběhlo vyhodnocení přesností modelů na *dev* částech obou vybraných datových sad a byly diskutovány jednotlivé výsledky.

V předposlední páté kapitole byla navržena architektura webové stránky a následně byla popsána její implementace. V rámci popisu implementace byly diskutovány použité

technologie a finálně byl poskytnut návod, jak na lokálním zařízení zprovoznit běh této aplikace.

Celkově práce dosáhla svého primárního cíle vytvoření webové aplikace, která umožňuje uživatelům zadávat dotazy v přirozeném jazyce a získávat odpovědi ve formě SQL dotazů. Aplikace byla úspěšně implementována a testována, což potvrzuje její funkčnost a použitelnost v reálném prostředí.

6.2 Limity

Primárním identifikovaným limitem je přesnost modelů. Tento limit by bylo možné zmírnit delším procesem dolaďování nebo použitím lepších modelů pro danou problematiku. Tento proces však naráží na druhý problém, kterým je nedostatek výpočetních prostředků.

Modely dolaďované v rámci této práce, i když byly slabší variantou dostupných modelů, využívaly výpočetní prostředky prostředí Google Colab na maximum. Tento limit bohužel nelze překonat bez pronajímání velkých výpočetních středisek.

6.3 Budoucí směr

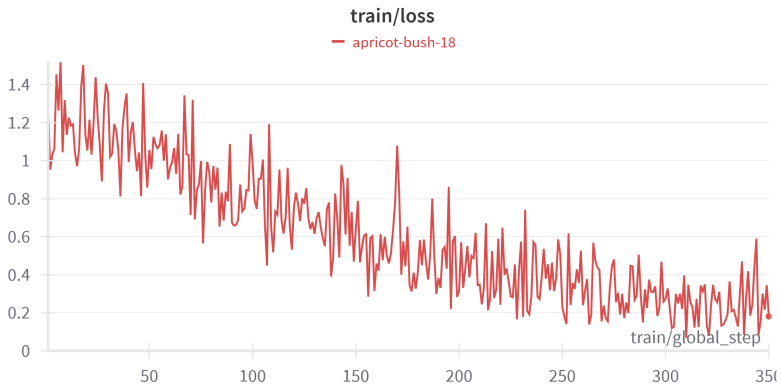
V rámci vylepšení této aplikace se nabízí několik možností. Zlepšení přesnosti modelů by mohlo být dosaženo prodloužením procesu dolaďování a použitím lepších modelů pro danou problematiku. Důležitým krokem je také optimalizace výpočetních zdrojů, například pronájmem výkonnějších výpočetních středisek nebo využitím cloudových řešení pro lepší výkon. Rozšíření funkcionality aplikace by mohlo zahrnovat implementaci podpory pro více databázových systémů a zlepšení uživatelského rozhraní. Dále by se mohly využít pokročilé techniky schema linking pro efektivnější zpracování komplexních dotazů. Všechny tyto kroky by mohly zlepšit jak využitelnost této aplikace, tak její hodnotu jako takovou.

6.4 Závěrečné myšlenky

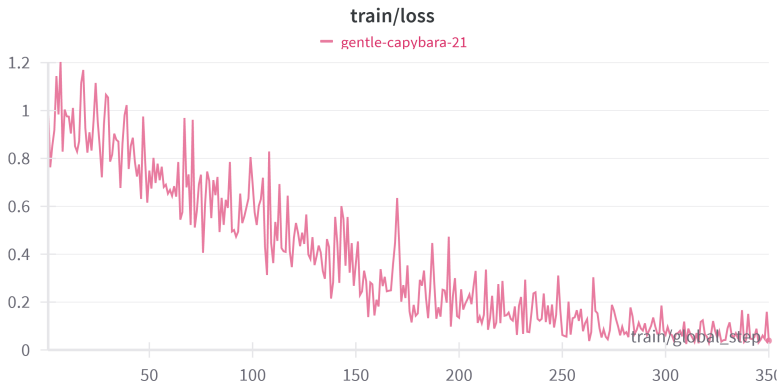
Tato práce demonstrovala potenciál využití velkých jazykových modelů pro překlad dotazů v přirozeném jazyce do SQL v kontextu webové aplikace. I když dosažené výsledky naznačují, že je stále prostor pro zlepšení, základní cíle byly úspěšně splněny. Vyvinutá aplikace poskytuje uživatelům jednoduchý způsob, jak komunikovat s databázemi, což může být užitečné v mnoha oblastech, od vzdělávání po profesionální datovou analýzu. Dosažené výsledky a identifikované limity poskytují pevný základ pro budoucí vývoj a optimalizaci tohoto přístupu.

Příloha A

Grafy z procesu dolaďování



(a) Llama



(b) Mistral

Obrázek A.1 Ztrátová funkce modelů v průběhu dolaďování

Bibliografie

1. HAMMAD, Khalid Adam Ismail; FAKHARALDIEN, Mohammed Adam Ibrahim; ZAIN, J; MAJID, M. Big data analysis and storage. In: *International conference on operations excellence and service engineering*. 2015, s. 10–11.
2. ROSENBLATT, Frank. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*. 1958, roč. 65, č. 6, s. 386–408. Dostupné z DOI: 10.1037/h0042519.
3. MINSKY, Marvin L; PAPERT, Seymour A. *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press, 1969. ISBN 0262130110.
4. HORNIK, Kurt; STINCHCOMBE, Maxwell; WHITE, Halbert. Multilayer feedforward networks are universal approximators. *Neural Networks*. 1989, roč. 2, č. 5, s. 359–366. ISSN 0893-6080. Dostupné z DOI: https://doi.org/10.1016/0893-6080(89)90020-8.
5. GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. MIT Press, 2016. Dostupné také z: http://www.deeplearningbook.org.
6. VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N.; KAISER, Lukasz; POLOSUKHIN, Illia. *Attention Is All You Need*. 2023. Dostupné z arXiv: 1706.03762 [cs.CL].
7. SOCHER, Richard; BAUER, John; MANNING, Christopher; Y., Ng. Parsing with Compositional Vector Grammars. In: 2013, sv. 1, s. 455–465.
8. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, s. 770–778.
9. BA, Jimmy Lei; KIROS, Jamie Ryan; HINTON, Geoffrey E. *Layer Normalization*. 2016. Dostupné z arXiv: 1607.06450 [stat.ML].
10. META AI. *Introducing Meta Llama 3: The most capable openly available LLM to date* [https://ai.meta.com/blog/meta-llama-3/]. 2024. Accessed: 2024-05-06.
11. AI@META. Llama 3 Model Card. 2024. Dostupné také z: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md. Accessed: 2024-05-06.
12. AINSLIE, Joshua; LEE-THORP, James; JONG, Michiel de; ZEMLYANSKIY, Yury; LEBRÓN, Federico; SANGHAI, Sumit. *GQA: Training Generalized Multi-*

Query Transformer Models from Multi-Head Checkpoints. 2023. Dostupné z arXiv: 2305.13245 [cs.CL].

13. PARK, Daniel. *Open-LLM-Leaderboard-Report*. 2023. Dostupné také z: https://github.com/dsdanielpark/Open-LLM-Leaderboard-Report.
14. JIANG, Albert Q.; SABLAYROLLES, Alexandre; MENSCH, Arthur; BAMFORD, Chris; CHAPLOT, Devendra Singh; CASAS, Diego de las; BRESSAND, Florian; LENGYEL, Gianna; LAMPLE, Guillaume; SAULNIER, Lucile; LAVAUD, Lélio Renard; LACHAUX, Marie-Anne; STOCK, Pierre; SCAO, Teven Le; LAVRIL, Thibaut; WANG, Thomas; LACROIX, Timothée; SAYED, William El. *Mistral 7B*. 2023. Dostupné z arXiv: 2310.06825 [cs.CL].
15. KOVALEVA, Olga; ROMANOV, Alexey; ROGERS, Anna; RUMSHISKY, Anna. Revealing the Dark Secrets of BERT. In: INUI, Kentaro; JIANG, Jing; NG, Vincent; WAN, Xiaojun (ed.). *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, 2019, s. 4365–4374. Dostupné z DOI: 10.18653/v1/D19-1445.
16. BELTAGY, Iz; PETERS, Matthew E.; COHAN, Arman. *Longformer: The Long-Document Transformer*. 2020. Dostupné z arXiv: 2004.05150 [cs.CL].
17. HARRIS, Zellig S. Distributional structure. *Word*. 1954, roč. 10, č. 2-3, s. 146–162.
18. GUO, Jiaqi; ZHAN, Zecheng; GAO, Yan; XIAO, Yan; LOU, Jian-Guang; LIU, Ting; ZHANG, Dongmei. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In: KORHONEN, Anna; TRAUM, David; MÀRQUEZ, Lluís (ed.). *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, s. 4524–4535. Dostupné z DOI: 10.18653/v1/P19-1444.
19. RAJKUMAR, Nitarshan; LI, Raymond; BAHDANAU, Dzmitry. *Evaluating the Text-to-SQL Capabilities of Large Language Models*. 2022. Dostupné z arXiv: 2204.00498 [cs.CL].
20. WANG, Bing; REN, Changyu; YANG, Jian; LIANG, Xinnian; BAI, Jiaqi; CHAI, Linzheng; YAN, Zhao; ZHANG, Qian-Wen; YIN, Di; SUN, Xing; LI, Zhoujun. *MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL*. 2024. Dostupné z arXiv: 2312.11242 [cs.CL].
21. ZHANG, Zhuosheng; ZHANG, Aston; LI, Mu; SMOLA, Alex. *Automatic Chain of Thought Prompting in Large Language Models*. 2022. Dostupné z arXiv: 2210.03493 [cs.CL].
22. LING, Chen; ZHAO, Xujiang; LU, Jiaying; DENG, Chengyuan; ZHENG, Can; WANG, Junxiang; CHOWDHURY, Tanmoy; LI, Yun; CUI, Hejie; ZHANG, Xuchao; ZHAO, Tianjiao; PANALKAR, Amit; MEHTA, Dhagash; PASQUALI, Stefano; CHENG, Wei; WANG, Haoyu; LIU, Yanchi; CHEN, Zhengzhang; CHEN, Haifeng; WHITE, Chris; GU, Quanquan; PEI, Jian; YANG, Carl; ZHAO, Liang. *Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey*. 2024. Dostupné z arXiv: 2305.18703 [cs.CL].
23. TAORI, Rohan; GULRAJANI, Ishaan; ZHANG, Tianyi; DUBOIS, Yann; LI, Xuechen; GUESTRIN, Carlos; LIANG, Percy; HASHIMOTO, Tatsunori B. *Stanford*

Alpaca: An Instruction-following LLaMA model [https://github.com/tatsu-lab/stanford_alpaca]. GitHub, 2023.

24. DETTMERS, Tim; PAGNONI, Artidoro; HOLTZMAN, Ari; ZETTLEMOYER, Luke. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. Dostupné z arXiv: 2305.14314 [cs.LG].
25. ZHONG, Victor; XIONG, Caiming; SOCHER, Richard. Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning. *CoRR*. 2017, roč. abs/1709.00103.
26. KATSOGIANNIS-MEIMARAKIS, George; KOUTRIKA, Georgia. Deep Learning Approaches for Text-to-SQL Systems. In: *EDBT*. 2021, s. 710–713.
27. YU, Tao; ZHANG, Rui; YANG, Kai; YASUNAGA, Michihiro; WANG, Dongxu; LI, Zifan; MA, James; LI, Irene; YAO, Qingning; ROMAN, Shanelle; ZHANG, Zilin; RADEV, Dragomir. *Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task*. 2019. Dostupné z arXiv: 1809.08887 [cs.CL].
28. YU, Tao. *Spider: One More Step Towards Natural Language Interfaces to Databases* [https://medium.com/@tao.yu/spider-one-more-step-towards-natural-language-interfaces-to-databases-62298dc6df3c]. 2024. Accessed: 2024-05-05.
29. LI, Jinyang; HUI, Binyuan; QU, Ge; YANG, Jiaxi; LI, Binhua; LI, Bowen; WANG, Bailin; QIN, Bowen; GENG, Ruiying; HUO, Nan et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*. 2024, roč. 36.
30. SU, Jianlin; LU, Yu; PAN, Shengfeng; MURTADHA, Ahmed; WEN, Bo; LIU, Yunfeng. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. Dostupné z arXiv: 2104.09864 [cs.CL].
31. ZHONG, Ruiqi; YU, Tao; KLEIN, Dan. *Semantic Evaluation for Text-to-SQL with Distilled Test Suites*. 2020. Dostupné z arXiv: 2010.02840 [cs.CL].

Obsah příloh

- `readme.txt` ..... stručný popis obsahu média
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `web` ..... kořenová složka webové aplikace
      - `db` ..... složka určená pro ukládání nahraných databází Sqlite
      - `apps` ..... jednotlivé stránky webové aplikace
      - `helpers` ..... pomocné funkce webové aplikace
    - `ft_jupyter` ..... soubory použité k dolaďování modelů
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF