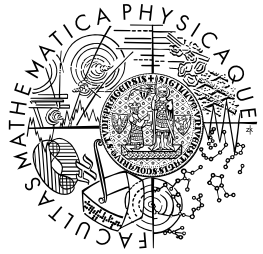


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Michaela Hubená

Automatické generování Einsteinových hádanek v přirozeném jazyce

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: RNDr. David Mareček, Ph.D.
Studijní program: Informatika
Studijní obor: Programování a vývoj software

Praha 2023

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Poděkování Chtěla bych poděkovat svému vedoucímu RNDr. Davidu Marečkovi, Ph.D. za pravidelné konzultace a zpětnou vazbu k vývoji aplikace i k textu bakalářské práce.

Název práce: Automatické generování Einsteinových hádanek v přirozeném jazyce

Autor: Michaela Hubená

Ústav: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: RNDr. David Mareček, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: V rámci bakalářské práce byla vytvořena aplikace na příkazové řádce pro generování Einsteinových hádanek v přirozeném jazyce pomocí jazykového modelu GPT-3 (generativní předtrénovaný Transformer třetí generace). Pro generování Einsteinových hádanek byla použita metoda few-shot, kdy je jazykovému modelu kromě zadání požadované úlohy předáno i několik vyřešených příkladů této úlohy, pomocí kterých se má jazykový model úlohu naučit přímo při generování. Vytvořená aplikace umožňuje generovat Einsteinovy hádanky různých velikostí a obtížností na libovolné téma v českém či anglickém jazyce. Při generování je kladen důraz na kreativitu a originalitu Einsteinových hádanek.

Klíčová slova: Einsteinova hádanka zebra generování textu přirozený jazyk

Title: Automatic generation of Einstein's puzzles in natural language

Author: Michaela Hubená

Institute: Institute of Formal and Applied Linguistics

Supervisor: RNDr. David Mareček, Ph.D., Institute of Formal and Applied Linguistics

Abstract: In this bachelor thesis was created command line application for generating Einstein's riddles in natural language using language model GPT-3 (third generation Generated Pre-trained Transformer). The few-shot method was used to generate Einstein's riddles, where, in addition to entering the required task, the language model is also given several solved examples of this task, with which the language model is supposed to learn the task directly during generation. The created application allows user to generate Einstein's riddles of various sizes and difficulties on any topic in Czech or English language. During generation the emphasis is placed on the creativity and originality of Einstein's riddles.

Keywords: Einstein's riddle zebra puzzle text generating natural language

Obsah

Úvod

Jazykové modely se v posledních pár letech rychle vyvíjejí a zlepšují. Nabízí se proto otázka, k jakým úlohám lze tyto modely využít a jak dobrých výsledků lze pomocí nich dosáhnout. V této práci se pokusíme pomocí jazykového modelu GPT-3 (generativní předtrénovaný Transformer třetí generace, více viz kapitola 2) generovat zadání Einsteinovy hádanky tak, aby byly jazykově rozmanité, ale zároveň srozumitelné a řešitelné.

Einsteinova hádanka je logická hádanka, jejíž zadání může vypadat takto:

> *Je 5 domů namalovaných odlišnými barvami. V každém domě žije jeden člověk s jinou národností. Každý majitel domu pije určitý typ nápoje, hraje určitý sport a chová určitého mazlíčka. Žádní majitelé nemají stejného mazlíčka, nehrají stejný sport ani nepijí stejný nápoj. Kdo chová rybu?* [1]

K tomuto popisu hádanky, ještě patří následující nápovědy:

1. Brit žije v červeném domě.
2. Švéd chová psi jako mazlíčky.
3. Dán pije čaj.
4. Zelený dům je nalevo od bílého domu.
5. Majitel zeleného domu pije kafe.
6. Člověk, který hraje fotbal, chová ptáky.
7. Majitel žlutého domu hraje baseball.
8. Muž bydlící v prostředním domě pije mléko.
9. Nor žije v prvním domě.
10. Muž, který hraje volejbal, bydlí vedle toho, kdo chová kočky.

11. Muž, který chová koně, bydlí vedle toho, kdo hraje baseball.
12. Majitel, který hraje tenis, pije pivo.
13. Němec hraje hokej.
14. Nor bydlí vedle modrého domu.
15. Muž, který hraje volejbal, má souseda, který pije vodu. [1]

S těmito informacemi už můžeme hádanku vyřešit, tak že postupně vyplňujeme tabulku, dokud nezjistíme, kdo chová ryby. Můžeme začít tím, že použijeme 9. pravidlo, které přímo říká, že v prvním domě žije člověk norské národnosti. Na to lze navázat použitím 14. pravidla, jelikož už víme, kde bydlí Nor, odvodíme, že druhý dům je modrý. Tímto způsobem pokračujeme a nakonec získáme zcela vyplněnou tabulku, z které zjistíme, že ryby chová Němec (kompletní řešení viz tabulka 1.1).

Věty, které jsou nápovědami hádanky, mohou být velmi různorodé, kreativní a originální. Avšak existující aplikace značně omezují formát nápověd, protože dávají důraz na korektnost a gramatickou správnost před originalitou nápověd a vytvářejí proto nápovědy doplňováním do pevně daných šablon (viz sekce 1.5). Motivací této práce je generovat zajímavější hádanky, jejichž nápovědy nebudou mít jednotný formát a budou proto různorodější. Podobně jako je tomu ve výše uvedeném příkladu, kde v jedné nápovědě je zmíněno, že někdo *bydlí vedle*, a v další, že někdo *má souseda*. Preferujeme tak originalitu nápověd před korektností a gramatickou správností nápověd, ale samozřejmě nemůžeme generovat zcela nesrozumitelné věty, jelikož pak by nebylo možné hádanku vyřešit.

Cílem práce je tak vytvořit aplikaci, která vygeneruje s pomocí jazykového modelu GPT-3 korektní zadání Einsteinovy hádanky v přirozeném jazyce s jediným řešením. Nápovědy této hádanky by měly být dostatečně srozumitelné a pochopitelné pro vyřešení hádanky, ale zároveň originální, kreativní a různorodé. Aplikace umožní uživateli zvolit jednotlivé parametry hádanky jako je velikost, obtížnost, jazyk či téma hádanky (podrobněji viz sekce 4.1.2). Nebudou tedy různorodé pouze nápovědy v rámci jedné hádanky, nýbrž i jednotlivé vygenerované hádanky vůči sobě navzájem. V kapitole 5 zhodnotíme kvalitu generovaných zadání hádanek a porovnáme hádanky generované v češtině a v angličině.

Kapitola 1

Einsteinova hádanka

Einsteinova hádanka, taktéž nazývána zebra, je hádanka, kterou podle legendy vymyslel Albert Einstein ve svém dětství. V úvodu jsme si ukázali, jak taková Einsteinova hádanka vypadá na konkrétním příkladu. V této kapitole si zadefinujeme formát hádanky, který budeme nadále používat v celé práci, a předvedeme ho na úvodním příkladu. Nakonec se podíváme na existující programy, které se zabývají Einsteinovými hádankami.

1.1 Řešení úvodního příkladu

V úvodu jsme si ukázali příklad Einsteinovy hádanky. Zde pro úplnost uvádíme jeho kompletní řešení v tabulce 1.1.

1.2 Formát Einsteinovy hádanky

Einsteinovy hádanky můžeme najít v různých více či méně odlišných formátech. Abychom se vyhnuli případným nedorozuměním, definujeme si konkrétní formát Einsteinovy hádanky, který budeme nadále používat v této práci.

	Dům 1	Dům 2	Dům 3	Dům 4	Dům 5
Národnost	Nor	Dán	Brit	Němec	Švéd
Barva	žlutá	modrá	červená	zelená	bílá
Nápoj	voda	čaj	mléko	kafe	pivo
Sport	baseball	volejbal	fotbal	hokej	tenis
Mazlíček	kočky	koně	ptáci	**ryby**	psi

Tabulka 1.1 Řešení ukázkové Einsteinovy hádanky

Zadefinujme si nejprve pojmy, které budeme používat k popisu hádanek. *Kategorie hádanky* je skupina vlastností či oblíbených věcí obyvatel jednotlivých domů. *Instance hádanky* je konkrétní vlastnost či oblíbená věc z jedné kategorie hádanky, která patří obyvateli jednoho z domů. Každá kategorie má několik konkrétních instancí, jejichž počet odpovídá počtu domů. V našem formátu bude hádanka vždy čtvercová, a proto počet kategorií bude vždy stejný jako počet domů a počet instancí v jedné kategorii. *Velikost hádanky* je číslo odpovídající počtu kategorií, počtu instancí v jedné kategorii i počtu domů. *Nápověda hádanky* je věta obsahující alespoň jednu instanci, která poskytuje nějakou informaci k vyřešení hádanky. Počet nápověd není pevně určen a není jakkoli spjat s velikostí hádanky. Typy jednotlivých nápověd si zadefinujeme v části 1.3.

Jednotlivá zadání hádanek se budou lišit právě v kategoriích, instancích, velikosti hádanky a nápovědách. Stejný bude pro každou hádanku cíl, který sestává z vyplnění příslušné tabulky, jejíž sloupce patří jednotlivým domům a řádky jednotlivým kategoriím. Řešení hádanky musí být právě jedno.

1.2.1 Ukázka formátu na příkladu

Ukažme si na příkladu uvedeném v úvodu práce, čemu odpovídají nově zadefinované pojmy. Tato hádanka téměř odpovídá našemu formátu, jediným rozdílem je, že pokládá otázku, která je cílem řešení. Nicméně jelikož jde otázku, kdo chová ryby, zodpovědět až po úplném vyplnění tabulky, není tento rozdíl příliš podstatný (viz tabulka 1.1).

V ostatních parametrech hádanka přesně odpovídá našemu formátu. Jedná se o hádanku čtvercovou, jejíž velikost je 5. Máme 5 kategorií, konkrétně národnosti, barvy, nápoje, sporty a mazlíčky. Pro každou kategorii máme 5 instancí, například národnosti jsou Nor, Dán, Brit, Němec, a Švéd. A máme také 5 domů. Hádanka má právě jedno řešení.

1.3 Typy nápověd

V Einsteinových hádankách se používají různé typy nápověd. V této části popíšeme 6 typů nápověd, které doplňují formát hádanky uvedený v předchozí části a které používá aplikace při generování hádanky. Jednotlivé typy nápověd demonstrujeme na nápovědách z ukázkového příkladu uvedeném v úvodu práce.

1.3.1 Informující nápověda

První a nejjednodušší nápovědou je *informující nápověda*. Tato nápověda řešitele hádanky pouze informuje, že existuje instance, která není uvedena v

žádné jiné nápovědě. Tato instance bude ve výsledném řešení patřit na místo, které zbyde, jelikož se na něm podle ostatních nápověd nemůže nacházet žádná jiná instance. Ukázkový příklad tento typ nápovědy neobsahuje, nicméně mohli bychom ho tam dodat, jelikož v žádné z nápověd není zmíněna instance *ryby*. Přidaná informující nápověda by mohla znít například takto: *Někdo chová ryby.*

1.3.2 Poziční nápověda

Druhou nápovědou je *poziční nápověda*, která udává na jaké pozici se nachází instance, kterou obsahuje. Pozice je uvedena jako číslo domu a jednoznačně určuje místo instance v řešení hádanky. V ukázkovém příkladu je tento typ použit dvakrát. V osmé nápovědě *Muž žijící v prostředním domě pije mléko.*, kde instance je *mléko* a pozice zde sice není uvedena číselně, ale lze snadno odvodit, že je 3. A v deváté nápovědě *Nor žije v prvním domě.*, kde instance je *Nor* a pozice je 1.

1.3.3 Přiřazující nápověda

Třetím typem nápovědy je *přiřazující nápověda*, která přiřazuje k sobě dvě instance z odlišných kategorií, které se nacházejí na stejné pozici v řešení. Konkrétní pozice však není uvedena a řešitel ji musí určit pomocí ostatních nápověd. V ukázkovém příkladu je tento typ použit mnohokrát, například hned v prvním pravidle *Brit žije v červeném domě.*, kde instance jsou *Brit* a *červená*. Další přiřazující nápovědy jsou druhá, třetí, pátá, šestá, sedmá, dvanáctá a třináctá nápověda z ukázkového řešení.

1.3.4 Množinová nápověda

Dalším typem nápovědy je *množinová nápověda*, která je velmi podobné přiřazující nápovědě. Rozdílem je, že jedna z instancí je nahrazena množinou instancí o velikosti 3. Množinová nápověda přiřazuje k sobě uvedenou samostatnou instanci a jednu instanci z uvedené množiny, které se nacházejí na stejné pozici. Při použití množinové nápovědy je nutné určit, které instance z množiny se pravidlo týká a na jaké pozici se dvě přiřazené instance nacházejí. Ukázkový příklad tuto nápovědu neobsahuje, jelikož bylo vymyšleno za účelem zpestření hádanky, aby obsahovala více odlišných typů nápověd a byla tak různorodější. Konkrétně množinová nápověda byla přidána, protože se jedná o zajímavý typ nápovědy z hlediska generování nápověd v přirozeném jazyce, kde je nutné pro 3 instance reprezentující množinu najít nějaké zastřešující slovo, které spojuje tyto 3 instance ale vyčleňuje ostatní instance ze stejné kategorie. Dvanáctou nápovědu z ukázkové hádanky lze upravit na množinovou nápovědu *Majitel, který hraje*

tenis, pije studený nápoj., kde první instance je *tenis* a trojice instancí popsána pojmem *studený nápoj* je *voda*, *mléko* a *pivo*.

1.3.5 Sousedící nápověda

Zcela odlišným typem je *sousedící nápověda*, jelikož nepřiřazuje dvě instance na jednu pozici, nýbrž na dvě odlišné ale sousedící pozice. Pozice jsou sousedící, pokud jsou přesně o jedna vedle. Jelikož jsou pozice instancí odlišné, nemusí být odlišné kategorie instancí a může se stát, že sousedící nápověda obsahuje dvě instance jedné kategorie. Pokud známe pozici první instance, můžeme zjistit či alespoň upřesnit pozici druhé instance. Ukázkový příklad obsahuje čtyři tyto nápovědy, konkrétně desátou, jedenáctou, čtrnáctou a patnáctou. Například desátá nápověda *Muž, který hraje volejbal, bydlí vedle toho, kdo chová kočky.* upřesňuje pozici instancí *volejbal* a *kočky*.

1.3.6 Řadící nápověda

Posledním typem nápovědy je *řadící nápověda*, která je podobná sousedící nápovědě. Instance uvedené v této nápovědě sice mohou ale nemusí sousedit, jelikož nápověda udává jejich vzájemnou polohu. První instance se nachází na pozici vlevo od pozice druhé instance. Pozice instancí jsou odlišné a je určené pořadí zleva, v jakém se nacházejí, avšak není určena vzdálenost těchto pozic. Ukázkový příklad obsahuje jednu řadící nápovědu, konkrétně čtvrtou nápovědu *Zelený dům je nalevo od bílého domu.*, kde první instance je *zelená* a druhá instance je *bílá*.

1.4 Symbolická reprezentace

Symbolická reprezentace Einsteinovy hádanky je reprezentací, která namísto slov a vět používá symboly. Tuto reprezentaci používá aplikace jako mezistav při generování hádanky (viz kapitola 3). Stejně jako Einsteinova hádanka v přirozeném jazyce musí i Einsteinova hádanka v symbolické reprezentaci splňovat formát uvedený v sekci 1.2. Instance hádanky jsou reprezentovány jednotlivými symboly. Kategorie hádanky nemají vlastní symboly, nýbrž jsou charakterizovány pouze jako množiny symbolů reprezentujících instance. Nápovědy hádanky jsou reprezentována symboly instancí, ke kterým se nápověda vztahuje, a dalším symbolem či několika symboly, které reprezentují vztah mezi jednotlivými instancemi a typ nápovědy (viz sekce 1.3).

Nyní si ukážeme, jak mohou vypadat konkrétní nápovědy v symbolické reprezentaci na několika nápovědách z ukázkového příkladu uvedeném v úvodu práce.

Například první nápověda *Brit žije v červeném domě.* lze v symbolické reprezentaci zapsat jako *B = Č*, kde Brit je B, červená je Č a rovnítko mezi dvěma instancemi značí, že jde o přiřazující nápovědu. Podobně čtvrtou nápovědu *Zelený dům je nalevo od bílého domu.* lze zapsat jako $G \rightarrow W$, kde zelená je G, bílá je W a šipka značí vzájemné umístění instancí u řadící nápovědy. Zajímavá je devátá nápověda *Nor žije v prvním domě*, kterou zapíšeme jako *N = 0*, kde Nor je N, první dům je reprezentován jako 0, jelikož v symbolické reprezentaci číslujeme od nuly, a rovnítko v tomto případě reprezentuje poziční nápovědu, jelikož není mezi dvěma instancemi, nýbrž instancí a pozicí. Nakonec čtrnáctou nápovědu *Nor bydlí vedle modrého domu.* reprezentujeme jako *N - M*, kde Nor je opět N, modrá je M a pomlčka reprezentuje sousedění instancí neboli sousedící nápovědu.

1.5 Analýza existujících aplikací

Existuje mnoho různých aplikací více či méně spjatých s Einsteinovými hádankami, zejména her pro mobilní telefony. V této části si představíme pár z nich a rozebereme v čem jsou stejné a v čem jsou odlišné od aplikace, která je předmětem této práce.

1.5.1 einstein-puzzle

einstein-puzzle[1] je open source aplikace, která je remakem původní staré DOS hry Sherlock, která byla inspirována právě Einsteinovou hádankou. Cílem hry je správně určit rozložení symbolů na hrací desce velikosti 6x6. Hráč dostane k dispozici nápovědy, která říkají, že dva symboly jsou ve stejném sloupci, že symboly sousedí nebo že jeden symbol je více vlevo než druhý.

Einsteinova hádanka je v této aplikaci převzata bez zásadních úprav, vzhledem k principu hádanky a její obtížnosti. Používá některé z typů nápověd, které jsme si zavedli. Taktéž používá vlastní avšak podobnou symbolickou reprezentaci, která není mezistavem nýbrž cílem vygenerování hádanky. Jedním z rozdílů je, že aplikace poskytuje přehledné grafické uživatelské rozhraní pro řešení hádanky a neskončí pouze vygenerováním hádanky. Nicméně zásadním rozdílem je to, že tato aplikace vůbec nepoužívá přirozený jazyk.

1.5.2 Einstein's Riddle Logic Puzzle

Einstein's Riddle Logic Puzzle[2] je hra pro mobilní telefony. Cílem každé úrovně této hry je vyřešit Einsteinovu hádanku tak, že pro každé pole na hrací desce

[1] `https://github.com/lksj/einstein-puzzle`

[2] `https://play.google.com/store/apps/details?id=com.rottzgames.logic`

zvolíme správný obrázek předmětu, který odvodíme pomocí dostupných nápověd.

Formát Einsteinovy hádanky v této hře je velmi podobný tomu, který jsme si zavedli v sekci 1.2, rozebereme si proto pouze, v čem se tato hra liší. Prvním z rozdílů je opět, že se jedná o hru, která poskytuje GUI pro řešení hádanky. Druhým rozdílem je, že přestože se dá hra kromě angličtiny přepnout do několika dalších jazyků, čímž se přepne i jazyk nápověd hádanky, čeština není jedním z těchto jazyků. Dalším rozdílem je, že přestože nápovědy hádanky jsou zadány v přirozeném jazyce, věty reprezentující nápovědy jsou velmi jednoduché a mají stále téměř stejný formát. Nápovědy mají pouze dva typy: přiřazující nápovědy a znegované přiřazující nápovědy, neboli zda dvě instance patří či nepatří do stejného sloupce hrací desky. Formát se sice mírně liší pro různé kategorie, například je použité odlišné sloveso, ale celkový počet kategorií je 16, tedy relativně nízký. Patrně se jedná pouze o vzory vět pro jednotlivé kategorie a jazyky, do kterých jsou doplňovány konkrétní názvy instancí a případná negace.

Kapitola 2

GPT

V této kapitole si probereme teoretický základ této práce. Vysvětlíme, co je *jazykový model*, popíšeme model *Transformer* a krátce vysvětlíme, jak funguje model GPT a jaké má verze.

2.1 Jazykový model

Jazykový model je, zjednodušeně řečeno, model strojového učení, který je schopný podívat se na část věty a predikovat další slovo. Nejznámějšími jazykovými modely jsou klávesnice chytrých telefonů, které napovídají další slovo na základě toho, co bylo právě napsáno. [2] Avšak v praxi není výstupem jazykového modelu pouze jedno predikované slovo. Ve skutečnosti je výstupem pravděpodobnostní skóre všech slov, které daný jazykový model zná, přičemž velikost této "slovní zásoby" jazykového modelu se může pohybovat od pár tisíc do více než milionu slov. Aplikace postavená nad jazykovým modelem, například zmíněná aplikace klávesnice, pak musí najít slova s nejvyšší pravděpodobností a prezentovat je uživateli. [3]

2.2 Transformer

Jelikož je jazykový model GPT založen na modelu Transformer [4] popíšeme si nejprve tento model. Transformer je model původně navržený ke strojovému překladu a skládá se z enkodéru, dekodéru a propojení těchto komponent. . [4]

V enkodéru se střídají dva typy vrstev: *self-attention vrstva* a *dopředná neuronová síť*. Self-attention je metoda, kterou používá Transformer, aby zahrnul "porozumění" ostatních relevantních slov do slova, které právě zpracovává. V self-attention vrstvě jsou proto závislosti mezi jednotlivými cestami, kterými

putují zpracovávané tokeny. Dopředná neuronová síť, neboli *feed forward neuronová síť*, je neuronová síť, která neobsahuje cykly, nemůže se vracet a postupuje pouze dopředu. V dopředné neuronové síti, narozdíl od self-attention vrstvy, se jednotlivé tokeny zpracovávají nezávisle, a tak může být tato část snadno zparalelizována. [4]

Dekodér je podobný enkodéru, ale má jeden typ vrstvy navíc. Celkem má tedy dekódér tři střídající se typy vrstev: *self-attention vrstvu*, *Encoder-Decoder attention vrstvu* a *dopřednou neuronovou síť*. Self-attention vrstva je zde mírně odlišná, jelikož nyní smí pohlížet jenom na dřívější pozice, než kterou právě zpracovává. Budoucí pozice totiž při generování ještě neexistují, a tak je nelze použít. Tato úprava se provádí pomocí maskování budoucích pozic hodnotami, které neovlivní výsledek. Encoder-Decoder attention vrstva se dívá do enkodéru, a pomáhá tak dekódéru, aby se zaměřil na vhodné specifické segmenty vstupní sekvence. Dopředná neuronová síť je stejná jako v enkodéru. [4]

2.3 Modely GPT

V této sekci popíšeme modely GPT nejprve souhrnně a poté rozebereme vývoj jednotlivých modelů a základní rozdíly mezi nimi. Na závěr této sekce rozebereme, proč jsme zvolili právě model GPT-3 pro generování Einsteinových hádanek.

2.3.1 Popis modelů GPT

Zkratka GPT znamená *Generative Pre-trained Transformer* (česky generativní předtrénovaný Transformer) a z názvu je proto jasně patrné, že model GPT je založen na modelu Transformer [5], který jsme popsali v sekci 2.2. Model GPT je složen pouze z dekodéru, jehož vrstvy jsou mírně odlišné od vrstvev dekodéru v původním modelu Transformer, jelikož neobsahují Encoder-Decoder attention vrstvu. Dekóder v GPT se tedy skládá pouze ze dvou střídajích se typů vrstev: *self-attention vrstvy* a *dopředné neuronové sítě* [6]. Dvojici vrstev dekodéru, kde první vrstva je self-attention a druhá vrstva je dopředná neuronová síť, budeme označovat jako jeden blok dekodéru.

Podívejme se podrobněji, jak model GPT funguje. Začneme vstupem modelu, který se zpracovává po jednotlivých tokenech. Jeden token je slovo či část slova, v případě GPT-3 jeden token odpovídá v průměru 4 znakům textu pro běžný anglický text. [1] Pro každý token ze vstupu nalezne model odpovídající vektorovou reprezentaci tokenu v embedding matici, která je součástí natrénovaného modelu. Další krok spočívá v zakódování pozice tokenu ve vstupní sekvenci, přičemž

[1] `https://platform.openai.com/tokenizer`

příslušný vektor reprezentující pozici nalezneme v další matici, která je součástí modelu. Takto zpracovaný vstup je předán prvnímu bloku dekodéru. [6]

Token nejprve zpracuje self-attention vrstva, která provádí maskovanou self-attention tak, jak jsme popsali u dekodéru modelu Transformer. Při zpracování tokenu self-attention vrstvou se tedy berou v potaz pouze předchozí tokeny. Následně předá token vrstvě s plně propojenou dopřednou neuronovou sítí, která je složená ze dvou podvrstev, které postupně zpracují daný token. Poté co první blok dekodéru zpracuje daný token, pošle jej vzhůru zásobníkem dalšímu bloku dekóderu. Proces zpracování je stejný v každém bloku dekodéru, ale každý blok dekodéru má odlišné váhy v obou svých vrstvách. [6]

Poté co poslední blok dekodéru na vrchu zásobníku zpracuje všechny tokeny, získáme výsledky pomocí embedding matice, kterou jsme na začátku použili k nalezení reprezentace tokenů. Výsledky interpretujeme jako pravděpodobnosti pro každý token ze slovní zásoby modelu. Pomocí těchto pravděpodobností zvolíme výsledný token. Například můžeme vybrat token s nejvyšší pravděpodobností nebo vybrat ze všech tokenů náhodně s použitím získaných pravděpodobností tokenů. Střední cestou je, že nejprve předvybereme nějaký počet tokenů s nejvyššími pravděpodobnostmi a poté z nich vybereme náhodně. [6]

Tím model dokončil iteraci, jejímž výsledkem je výstup jediného tokenu. Následně je tento vygenerovaný token přidán do vstupní sekvence a tato nová sekvence je vstupem další iterace modelu. Tento postup se jmenuje *auto-regrese*. Model pokračuje v iteraci, dokud není vygenerován celý kontext nebo dokud není vytvořen token konce sekvence. [6] Velikost kontextu se liší v jednotlivých verzích (viz sekce 2.3.3).

2.3.2 Metody používání modelů

Fine-tuning *Fine-tuning* je proces adaptování předtrénovaného modelu neuronové sítě, jako je GPT, na novou úlohu nebo dataset pomocí dalšího trénování modelu na dané úloze či datasetu. Fine-tuning GPT zahrnuje úpravu parametrů předtrénovaného modelu tak, aby byly optimalizovány výsledky na následné konkrétní úloze, například klasifikaci textu nebo generování textu. [5] Hlavní nevýhodou fine-tuningu je potřeba nového velkého datasetu pro každou úlohu a potenciál pro špatnou generalizaci mimo distribuci. [7]

Few-Shot *Few-shot* je jeden ze způsobů použití jazykového modelu bez úprav vah modelu. Modelu pouze poskytneme několik demonstrací úlohy v čase odvozování jakožto podmiňování. Hlavní výhodou je výrazné snížení potřeby dat specifických pro danou úlohu a snížený potenciál naučit se příliš omezenou distribuci z velkého ale úzkého fine-tuning datasetu. Hlavní nevýhodou je, že

výsledky této metody byly, alespoň pro zatím, výrazně horší než výsledky fine-tuned modelů. Navíc stále potřebujeme malé množství dat specifických pro danou úlohu. [7]

Zero-Shot *Zero-shot* je velmi podobný few-shotu, taktéž se jedná o použití jazykového modelu bez úprav vah. Rozdíl spočívá v tom, že modelu neposkytujeme žádné demonstrace úlohy, ale pouze instrukce v přirozeném jazyce popisující danou úlohu. Tato metoda poskytuje maximální pohodlnost používání, potenciál pro robustnost a výhýbá se falešným korelacím, ale zároveň se jedná o nejnáročnější nastavení. Nicméně alespoň pro některá prostředí je zero-shot přístup nejblíže tomu, jak lidé plní úkoly. [7]

2.3.3 Vývoj modelů GPT

GPT-1 OpenAI[2] vydalo GPT-1 v roce 2018. [6] GPT-1 byl první model, který uměl přečíst text a odpovídat na dotazy. Jednalo se o důležitý krok kupředu ve vývoji umělé inteligence, protože GPT-1 umožnilo počítačům pochopit text přirozeněji než kdy dřív. Tento generativní jazykový model byl schopný se naučit širokou škálu spojení a získat obrovské znalosti o souvislých textech a dlouhých úsecích textu. GPT-1 používá 12-vrstvou architekturu dekódovačů se systémem self-attention pro trénování. GPT-1 má 117 milionů parametrů. [5]

GPT-2 S cílem vylepšit první verzi GPT vytvořilo OpenAI v roce 2019 GPT-2 tak, že použili větší dataset a více parametrů. [8] S 1,5 miliardou parametrů je tak GPT-2 desetkrát větší než GPT-1. Díky použití pouze nezpracovaného textu jako vstupu a využití minima trénovacích příkladů je GPT-2 efektivní model pro úlohy příbuzné překladu, sumarizaci textu a exceluje v predikování vět. [5]

GPT-3 GPT-3 je jazykový predikční a produkční model vytvořený OpenAI, který umí produkovat dlouhé pasáže textu. [7] GPT-3 se ukázal jako přelomový jazykový software umělé inteligence. Jednoduše řečeno se jedná o kus software, který umí sám o sobě vytvořit věty, které jsou tak charakteristické, že téměř znějí, jako by je psal člověk. Klíčovou výhodou je kapacita GPT-3, která je zhruba 175 miliard parametrů, tedy stokrát větší než u GPT-2. K učení byl použit korpus o 500 miliardách slovech s názvem "Common Crawl", který byl sesbírán z rozsáhlého archivu obsahu a z internetu. Jeho další pozoruhodnou a nepředvídanou vlastností je schopnost provádět základní matematické operace, psát kousky kódu a provádět chytré úlohy. OpenAI později tajně vydalo GPT-3.5, vylepšenou verzi GPT-3 natrénovanou na kombinaci textu a kódu. Z rozsáhlého množství dat sesbíraných

[2] `https://openai.com/about`

z webů, které zahrnovali tisíce záznamů Wikipedie, příspěvky ze sociálních sítí či novinové články, se tato vylepšená verze naučila rozpoznávat vztahy mezi slovy, větami a jinými komponentami. [5]

GPT-4 Nejnovější verzí modelu GPT od OpenAI je GPT-4, což je multimodální velký jazykový model, který byl spuštěn 14. března 2023. [9] GPT-4 byl předtrénován na veřejných datech a "datech licencovaných od poskytovatelů třetích stran". Následně byl model upraven pomocí zpětnovazebního učení založeného na vstupu od lidí a na umělé inteligenci s cílem, aby model reprezentoval lidské hodnoty a dodržoval odpovídající zásady. Byly vytvořeny dvě varianty GPT-4 s kontextovými okny velikosti 8192 a 32768 tokenů, pro srovnání model GPT-3 měl kontextová okna velikosti pouze 4096 a 2049 tokenů. [5]

2.3.4 Zvolení modelu GPT-3

V aplikaci pro generování Einsteinových hádanek používáme model GPT-3, konkrétně vylepšenou verzi GPT-3.5. Verze GPT-1 a GPT-2 nebyly zvoleny, jelikož jsou již zastaralé a nepodporované OpenAI. Verze GPT-4 se teprve na začátku června 2023 začala postupně zpřístupňovat běžným uživatelům[3] a nebyla ještě volně dostupná v době vývoje aplikace. Verze GPT-3 je vhodná pro aplikaci přístupu *few-shot*, který chceme používat pro generování hádanek (více viz sekce 3.3). Verze GPT-3.5 je vylepšením verze GPT-3 a lze také dobře použít pro aplikaci přístupu *few-shot*. Verzi GPT-3.5 jsme tedy zvolili, protože se jedná o nejlepší verzi GPT, která byla v době vývoje aplikace dostupná a která splňuje naše požadavky.

[3] `https://openai.com/blog/gpt-4-api-general-availability`

Kapitola 3

Generování hádanky

V této kapitole si popíšeme postup, který používá aplikace pro vygenerování hádanky. Tento postup zahrnuje tři kroky: vygenerování řešení hádanky v symbolické reprezentaci, vygenerování zadání hádanky v symbolické reprezentaci a převedení zadání hádanky ze symbolické reprezentace do přirozeného jazyka.

3.1 Generování řešení hádanky

Abychom zajistili, že hádanka bude mít právě jedno řešení, budeme postupovat pozpátku a nejprve vygenerujeme řešení hádanky. Protože v této fázi generujeme hádanku v symbolické reprezentaci, která je pouze přechodná, nezáleží na tom jaké symboly přiřadíme jednotlivým instancím, jediným požadavkem je, že jsou všechny symboly unikátní. Vygenerujeme tedy příslušný počet unikátních symbolů, který odpovídá druhé mocnině velikosti hádanky, a rozdělíme symboly do kategorií o velikosti hádanky. Samotné generování řešení je přímočaré, jelikož pouze vytvoříme tabulku požadované velikosti a každou řádku vyplníme permutací instancí příslušné kategorie. Příklad řešení hádanky velikosti 4 je k vidění v tabulce 3.1.

Kategorie	Dům 1	Dům 2	Dům 3	Dům 4
{ A, B, C, D }	C	D	B	A
{ G, H, I, J }	G	H	I	J
{ M, N, O, P }	P	M	O	N
{ S, T, U, V }	T	S	U	V

Tabulka 3.1 Příklad vygenerovaného řešení v symbolické reprezentaci

3.2 Generování zadání hádanky v symbolické reprezentaci

Pomocí řešení poté vygenerujeme zadání hádanky sestávající z jednotlivých nápověd. Idea postup je následující: vygenerujeme všechny možné existující nápovědy pro již vygenerované řešení hádanky a náhodně budeme z této množiny vybírat jednotlivé nápovědy, dokud nezískáme dostatek nápověd pro vyřešení hádanky. Nyní si tento postup podrobněji vysvětlíme a ukážeme si, proč funguje.

3.2.1 Generování nápověd

Nápovědy generujeme zvlášť podle jejich jednotlivých typů.

Informující nápověda Informující nápověda je výjimkou, jelikož tento typ nápovědy v této fázi negenerujeme. Poté co vygenerujeme a vybereme ostatní nápovědy, zjistíme, které instance nebyly zmíněny v žádných nápovědách a pro ty rovnou vytvoříme informující nápovědy v přirozeném jazyce.

Poziční nápověda Pro konkrétní řešení hádanky existuje jedna poziční nápověda pro každé pole v tabulce s řešením. Projdeme tedy již vygenerované řešení hádanky a pro každé pole vygenerujeme odpovídající poziční nápovědu. Například pro první pozici v řešení uvedeném v tabulce 3.1 by byly vygenerovány následující nápovědy: C = 0, G = 0, P = 0 a T = 0, jelikož jak bylo uvedeno v sekci 1.4, v symbolické reprezentaci číslujeme pozice od nuly. Nápovědy pro další pozice vygenerujeme stejným způsobem.

Přiřazující nápověda Existuje právě jedna přiřazující nápověda pro každou dvojici instancí nacházející se na stejné pozici v daném řešení hádanky. Nápověda je symetrická, a proto pro každou dvojici instancí vygenerujeme pouze jednu přiřazující nápovědu. Pro instance na první pozici v řešení uvedeném v tabulce 3.1 by tedy bylo vygenerováno těchto šest nápověd: C = G, C = P, C = T, G = P, G = T a P = T.

Množinová nápověda Pro každou dvojici instancí nacházející se v řešení na stejné pozici můžeme vytvořit hned několik různých množinových nápověd. K jedné nebo druhé instanci přidáme dvě instance ze stejné kategorie, čímž vytvoříme množinu. Tímto postupem vznikne mnoho různých množinových nápověd pro jedno konkrétní řešení hádanky, výrazně více než pro jiné typy nápověd, což nás přivádí ke dvěma problémům. Zaprvé pokud budeme následně nápovědy vybírat uniformně náhodně, větší počet množinových nápověd způsobí,

že vygenerovaná hádanka bude obsahovat více množinových nápověd oproti ostatním typům nápověd. Zadruhé může výsledná hádanka obsahovat dvě velmi podobné množinové nápovědy, například *Martinova nejoblíbenější barva je buď modrá, zelená nebo červená.* a *Martinova nejoblíbenější barva je buď modrá, žlutá nebo zelená.* Přestože toto není problém pro vyřešení hádanky, dle mého názoru by jednotlivá nápovědy měly být pokud možno různorodá, aby byla hádanka zajímavější.

Nebudeme tedy generovat všechny možné množinové nápovědy, ale generování omezíme. Tím odstraníme uvedené problémy, aniž bychom museli při vybírání nápověd složitě kontrolovat, zda jsme nevybrali dvě příliš podobné nápovědy a zda nevybíráme hodně množinových nápověd oproti ostatním typům nápověd. Generování omezíme tak, že pro každou dvojici instancí na stejné pozici v řešení vygenerujeme pouze dvě množinové nápovědy. Jednu nápovědu, kde první instance bude součástí množiny, a druhou nápovědu, kde naopak druhá instance bude součástí množiny. Další instance, které se doplní do příslušné množiny, se v obou případech vyberou náhodně ze všech ostatních instancí odpovídající kategorie.

Pro instance na první pozici v řešení z tabulky 3.1 můžeme vytvořit například tyto nápovědy: C = [G, H, J], G = [A, C, D], C = [N, O, P], P = [B, C, D], C = [T, U, V], T = [A, B, C], G = [M, N, P], P = [G, H, I], G = [S, T, V], T = [G, H, I], P = [S, T, U], T = [N, O, P]. Jelikož generování množinových nápověd není deterministické, můžeme ze stejného řešení vygenerovat částečně či zcela odlišnou množinu nápověd než v uvedeném příkladu.

Sousedící nápověda Narozdíl od množinové nápovědy bude generování sousedí nápovědy opět přímočaré. Pro konkrétní řešení existuje jedna sousedící nápověda pro každou dvojici instancí na sousedních pozicích. Stejně jako přiřazující nápověda i sousedící nápověda je symetrická, a proto pro každou dvojici instancí vygenerujeme pouze jednu nápovědu. Pro dvojice instancí z první a druhé pozice v řešení uvedeném v tabulce 3.1 vzniknou následující sousedící nápovědy: C - D, C - H, C - M, C - S, G - D, G - H, G - M, G - S, P - D, P - H, P - M, P - S, T - D, T - H, T - M a T - S.

Řadící nápověda Existují dvě řadící nápovědy pro každou dvojici instancí na odlišných pozicích v konkrétním řešení hádanky. Buď můžeme mít nápovědu, že první instance je nalevo od druhé instance nebo že druhá instance je napravo od první instance. Protože tyto dvě varianty poskytují stejnou informaci, nechceme je generovat obě ale pouze jednu z nich. Budeme proto pro každou dvojici instancí generovat právě jednu řadící nápovědu takovou, že první instance je nalevo od druhé instance. Pro dvojice instancí na první a druhé pozici v řešení z tabulky

3.1 vzniknou následující nápovědy, které jsou stejné jako pro sousedící nápovědy pouze pomlčka je nahrazena šipkou znázorňující směr řadící nápovědy: C → D, C → H, C → M, C → S, G → D, G → H, G → M, G → S, P → D, P → H, P → M, P → S, T → D, T → H, T → M a T → S.

3.2.2 Výběr nápověd

Nyní máme vygenerované nápovědy všech typů a potřebujeme z nich vybrat takovou množinu nápověd, aby pomocí ní bylo možné vyřešit hádanku a dospět k jedinému správnému řešení, jehož vygenerování bylo popsáno v sekci 3.1. Nicméně tento požadavek není jediný, který na zadání hádanky máme, jelikož chceme generovat různě obtížná zadání: jednoduchou, normální a těžkou hádanku.

Základní myšlenka výběru spočívá v tom, že vezmeme všechny vygenerované nápovědy a vytvoříme z nich náhodnou permutaci, laicky řečeno je zamícháme. Poté budeme brát jednotlivé nápovědy popořadě z této permutace a u každé nápovědy rozhodneme zda ji přidáme do vybraných nápověd či nikoli. Toto rozhodnutí se bude zakládat na tom, zda právě zkoumaná nápověda přináší novou informaci, kterou ještě nemáme z nápověd, které už byly vybrány před touto nápovědou. Pokud přináší novou informaci, pak bude přidána do vybraných nápověd, jinak nikoli. Například můžeme snadno nahlédnout, že nápověda na první pozici permutace bude vždy vybrána. S výběrem nápověd skončíme ve chvíli, kdy přidání právě zkoumané nápovědy do výběru způsobí, že hádanku už lze pomocí aktuálního výběru nápověd vyřešit.

Avšak tuto myšlenku potřebujeme rozvinout a upravit, abychom mohli generovat různě obtížná zadání hádanky. Nebudeme proto vybírat ze všech typů nápověd dohromady, ale nejprve vybereme pouze omezený počet pozičních a přiřazujících nápověd a poté k těmto nápovědám doplníme výběr nápověd ostatních typů. Poziční a přiřazující nápovědy jsou zvoleny, protože se jedná o jednoduché a přímočaré typy nápověd. Počty vybraných pozičních a přiřazujících nápověd jsou pevně určeny podle obtížnosti a velikosti hádanky, přičemž přiřazujících nápověd je vždy dvojnásobek oproti pozičním nápovědám. Pro každou kombinaci obtížnosti a velikost hádanky je počet pozičních i přiřazujících nápověd větší než nula. Na závěr, poté co jsme doplnili výběr pozičních a přiřazujících nápověd o ostatní typy již uvedeným způsobem, přidáme pro lehkou a normální obtížnost hádanky ještě několik náhodných nápověd. Tyto přidané nápovědy již neposkytují nové informace, jelikož hádanka by šla vyřešit i bez nich, tak jako je tomu u těžké obtížnosti. Slouží pouze ke zjednodušení hádanky, jelikož díky duplikování některých informací umožňují více způsobů jak hádanku vyřešit.

Popsali jsme tedy celý postup výběru nápověd, avšak potřebujeme ještě dokázat, že pomocí tohoto postupu opravdu vygenerujeme zadání hádanky pro každé řešení. Teoreticky se totiž může stát, že projdeme všechna nápovědy a

přesto bude vytvořené zadání nekompletní a nepůjde pomocí něj hádanka vyřešit. Předpokládejme pro spor, že tato situace nastala. Máme tedy množinu vybraných nápověd, pomocí které nelze vyřešit hádanku, a zbylé nevybrané nápovědy, která nepřinášejí žádné nové informace pro vyřešení hádanky.

Uvažujme stav, kdy jsme použili všechny vybrané nápovědy a máme částečně vyřešenou hádanku. Protože hádanka není vyřešená zcela, musí existovat instance, jejíž pozici v řešení hádanky neznáme. Také víme, že existuje nějaká instance, jejíž pozici známe, protože zadání každé hádanky obsahuje alespoň jedno poziční pravidlo. Navíc musí existovat dvojice instancí se známou a neznámou pozicí, které spolu sousedí. Označme instance této dvojice X, pro instanci jejíž pozici neznáme, a Y pro instanci jejíž pozici známe.

Můžeme snadno nahlédnout, že pozici instance X bychom mohli určit pomocí sousedící a řadící nápovědy vzhledem k pozici instance Y, kterou známe. Jelikož sousedící i řadící nápovědy generujeme všechny, které existují pro dané řešení, musely být tyto nápovědy k dispozici při výběru nápověd. Buď byly obě nápovědy vybrány, pak bychom ale uměli určit pozici instance X, čímž docházíme ke sporu. Nebo alespoň jedna z nápověd nebyla vybrána, čímž opět docházíme ke sporu, jelikož tato nevybraná nápověda přináší novou informaci pro řešení hádanky. K vygenerování neúplného zadání proto nemůže dojít a uvedený postup vždy dospěje k vygenerování korektního zadání s jediným předem vygenerovaným řešením.

3.3 Konverze hádanky do přirozeného jazyka

Posledním krokem celého postupu generování hádanky je konverze hádanky ze symbolické reprezentace do přirozeného jazyka. K tomu použijeme jazykový model GPT-3, který jsme si představili v kapitole 2. Koverze hádanky spočívá ze 3 kroků, nejprve vygenerujeme kategorie hádanky, poté vygenerujeme jednotlivé instance a nakonec zkonvertujeme všechny nápovědy.

3.3.1 Generování kategorií

Prvním krokem konverze hádanky je vygenerování kategorií hádanky. V symbolické reprezentaci jsou jednotlivé kategorie pouze množiny instancí, které do dané kategorie patří, nicméně nyní budeme chtít reprezentovat kategorie slovy z přirozeného jazyka, což následně využijeme při generování instancí z těchto kategorií. Kategorie nebudeme volit náhodně, jelikož celá hádanka bude mít jedno téma, které si zvolí uživatel aplikace. Vygenerované kategorie by proto měli být spjaty s tímto tématem a mělo by se jednat o nadřazená slova, abychom pak z těchto kategorií mohli vygenerovat instance, které budou naopak slovy

podřazenými. Například slovo *zelenina* by mohlo být jednou z kategorií, jelikož jde o nadřazené slovo vůči jednotlivým druhům zeleniny jako *salát*, *paprika* nebo *rajče*, které by mohli být instancemi této kategorie.

Kategorie budeme generovat pomocí jazykového modelu GPT-3, kterému předáme vzory a téma, pro které chceme vygenerovat kategorie. Vzory jsou vyřešené příklady generování, pomocí kterých model GPT-3 vyřeší zadanou úlohu. Používáme tedy učení metodou *few-shot*, kterou jsme popsali v předchozí kapitole v sekci 2.3.2. Jeden vzor obsahuje nějaké téma a k němu vygenerované příslušné kategorie, kde počet vygenerovaných kategorií je o jedna menší než velikost hádanky. Vzhledem k tomu, že dané kategorie se vztahují k lidem žijícím v domech, je vždy první kategorie daná a tou je *křestní jméno*. Další kategorie se generují pomocí popsaných vzorů.

Představme si, že chceme vygenerovat hádanku velikosti 4 na téma *kinematografie*. Vzory pro generování kategorií v češtině pro hádanku velikosti 4 vypadají takto:

- malířství → malíři, obrazy, styly
- jídlo → zelenina, mléčné výrobky, maso
- hudba → skladby, žánry, zpěváci
- příroda → zvířata, rostliny, národní parky
- atletika → disciplíny, sportovci, náčiní

GPT-3 tedy předáme řetězec složený z těchto vzorů tak, že na každé řádce je jeden vzor a na poslední řádce je naše téma, tedy *kinematografie*, a za ním šipka. Celý řetězec vypadá následovně:

```
malířství => malíři, obrazy, styly
jídlo => zelenina, mléčné výrobky, maso
hudba => skladby, žánry, zpěváci
příroda => zvířata, rostliny, národní parky
atletika => disciplíny, sportovci, náčiní
kinematografie =>
```

Za tuto šipku má GPT-3 vygenerovat svou odpověď, která může být například: *filmy*, *režiséři*, *žánry*. Doplníme první kategorii, která je pevně daná, a získáme všechny čtyři kategorie: *křestní jména*, *filmy*, *režiséři* a *žánry*.

3.3.2 Generování instancí

Postup generování instancí je velmi podobný generování kategorií. Stejně jako jsme pro téma hádanky vygenerovali kategorie, nyní pro každou kategorii vygenerujeme jednotlivé instance. Opět využijeme vzory, avšak odlišné než pro generování kategorií a počet instancí v každém vzoru bude odpovídat velikosti hádanky, jelikož nebudeme mít žádnou pevně danou instanci. Důvodem proč používáme odlišné vzory je, že nyní chceme generovat konkrétní instance, které nemusí být slovy nadřazenými. Dalším důvodem je, že vzory budou obsahovat nejen kategorii a odpovídající instance, ale i téma celé hádanky. Po samotném vygenerování instancí ještě spárujeme nyní vygenerované instance v přirozeném jazyce s původními instancemi v symbolické reprezentaci, čímž vytvoříme slovník, který využijeme při konvertování nápověd.

Pro hádanku s tématem *kinematografie* máme vygenerované kategorie *křestní jména*, *filmy*, *režiséři* a *žánry*, pro které chceme vygenerovat jednotlivé instance. Vzory pro generování instancí v češtině pro hádanku velikosti 4 jsou následující:

- malířství : styly → realismus, impresionismus, moderna, renesance
- jídlo : pečivo → houska, chleba, donut, koláč
- hudba : žánry → pop, rock, klasika, jazz
- atletika : náčiní → oštěp, koule, tyčka, disk
- příroda : zvířata → medvěd, lev, kůň, liška

Tato verze vzorů není původní, jelikož vzory byly postupně upravovovány, což popíšeme v sekci 5.1. GPT-3 opět předáme řetězec složený z těchto vzorů tak, že jeden vzor je na jedné řádce a na poslední řádku přidáme naše téma a kategorii. Pro první kategorii bude tedy poslední řádka řetězce vypadat takto: *kinematografie : křestní jména* →. Odpověď vygenerovaná GPT-3 může být například: *Tomáš*, *Anna*, *Kačka*, *David*. Pro ostatní kategorie postupujeme stejně a použijeme stejný vzor pouze vyměníme příslušnou kategorii na poslední řádce zkonstruovaného řetězce.

Na závěr vytvoříme překladový slovník mezi instancemi v symbolické reprezentaci a v přirozeném jazyce. V jednotlivých kategoriích nezáleží na tom, které symboly přiřadíme ke kterým slovům reprezentujícím instance, proto je jednoduše spárujeme podle pořadí. Pro první kategorii dostaneme následující dvojice neboli překlady: A = *Tomáš*, B = *Anna*, C = *Kačka*, D = *David*, jelikož první kategorie v symbolické reprezentaci je množina { A, B, C, D } a první kategorie v přirozeném jazyce jsou křestní jména konkrétně právě *Tomáš*, *Anna*, *Kačka* a *David*. Ke každému překladu přidáme kategorii, do které patří příslušná instance,

jelikož se nám to bude hodit při konverzi nápověd. Například z dvojice A = *Tomáš* tak vznikne trojice A = (*křestní jména*, *Tomáš*).

3.3.3 Konverze nápověd

Posledním krokem vygenerování celé hádanky je konverze nápověd do přirozeného jazyka. Postup konverze a využití GPT je obdobné jako u generování kategorií a instancí, avšak pro nápovědy budeme mít výrazně více odlišných sad vzorů. Pro každou nápovědu tak nejprve musíme určit, která sada vzorů má být použita, poté doplníme řetězec sestavený z vybraných vzorů a předáme ho GPT-3.

Pro konkrétní nápovědu zvolíme správnou sadu vzorů pomocí dvou kroků. V prvním kroku zúžíme výběr na sady vzorů patřící odpovídajícímu typu nápovědy. V druhém kroku zohledníme zda daná nápověda obsahuje první kategorii, tedy *křestní jména*, a případně na jaké straně pravidla se nachází, zda vlevo či vpravo. Máme tedy pro každý typ nápověd tři sady vzorů. První pro nápovědy, která neobsahují první kategorii. Druhý pro nápovědy, která obsahují první kategorii na levé straně nápovědy. A třetí pro nápovědy, která obsahují první kategorii na pravé straně nápovědy a zároveň neobsahují první kategorii na levé straně nápovědy, jelikož u některých typů nápověd se může jedna kategorie vyskytovat na obou stranách nápověd. Sady vzorů máme rozdělené tímto způsobem z toho důvodu, že instance z kategorie *křestní jména* budou používány v konvertovaných nápovědách jako podmět, narozdíl od instancí z ostatních kategorií, které budou používány jako předmět.

Když máme vybranou správnou sadu vzorů, potřebujeme ji správně doplnit. V každém vzoru je zahrnuto jak téma hádanky, tak i kategorie instancí, které jsou obsažené v nápovědě, pro kterou byl vzor vytvořen. Jelikož vzory jsou již v přirozeném jazyce, využijeme nyní překladový slovník, který jsme zkonstruovali po vygenerování instancí. Vezmeme tedy všechny symboly instancí, které obsahuje dané nápověda, a přeložíme je pomocí překladového slovníku do přirozeného jazyka. Zároveň také získáme kategorie v přirozeném jazyce, do kterých jednotlivé instance patří. S těmito informacemi již můžeme správně doplnit příslušnou sadu vzorů.

Ukážeme si tento postup na příkladu hádanky s tématem *kinematografie*, který jsme již použili u generování kategorií a instancí. Mějme nápovědu D - H. Víme, že se jedná o sousedící nápovědu a že D je instance z první kategorie, čímž určíme konkrétní sadu vzorů:

- malířství : křestní jména - Martin = barvy - modrá → Martinova nejoblíbenější barva je modrá.
- jídlo : křestní jména - Jana = pečivo - chleba → Jana ráda jí chleba.

- hudba : křestní jména - Jakub = žánry - pop → Jakub rád poslouchá pop.

Použijeme překladový slovník a zjistíme, že symbol D odpovídá kategorii *křestní jména* a instanci *David* a symbol H odpovídá kategorii *filmy* a instanci *Tenet*. Z těchto informací vytvoříme poslední řádku řetězce: *kinematografie* : *křestní jména - David = filmy - Tenet* →. Nakonec předáme sestavený řetězec GPT a počkáme na odpověď, která by mohla být například: *Davidův nejoblíbenější film je Tenet.*

Jak můžeme vidět z uvedených vzorů, nápovědy jsou konvertovány do vět v přirozeném jazyce tak, že obsahují všechny instance, které patří do dané nápovědy. Toto platí i pro množinové nápovědy, jejichž konverze je proto značně zjednodušena, jelikož není nutné najít souhrnný pojem pro danou množinu, jak jsme si uvedli v sekci 1.3.4, ale stačí uvést výčet instancí. Pro nalezení zastřešujících pojmenování množin by bylo nutné u konverze hádanky do přirozeného jazyka přidat ještě jeden krok, ve kterém by se vygenerovala tato pojmenování množin. Teprve poté by se generovali instance takovým způsobem, aby správně patřily nebo naopak nepatřily do daných množin, což by bylo výrazně náročnější než generování instancí v současném stavu. Tento krok je nutné přidat, jelikož nalézt pojmenování množin zpětně po vygenerování instancí by bylo téměř nemožné. Navíc by bylo nutné ještě více omezit generování množinových nápověd. Nyní je totiž možné, že bude vygenerováno více překrývajících se množin v jedné kategorii a pro dvě takto podobné množiny bude obtížné ne-li přímo nemožné, vymyslet souhrnná pojmenování. Zcela by se musely zakázat množinové nápovědy, které obsahují *křestní jména* na pravé straně nápovědy, jelikož pro množiny obsahující křestní jména nelze vymyslet vhodné zastřešující pojmy. Množství nutných úprav je vysoké a časově náročné, a proto jsme se v současné chvíli uchýlili k této zjednodušené verzi konvertování množinových nápověd s tím, že uvedené úpravy mohou být implementovány dodatečně v dalších verzích aplikace.

Kromě konverze již vytvořených nápověd musíme navíc zkonstruovat informační nápovědy, které jako jediné nebyly vytvořeny v symbolické reprezentaci. Projdeme tedy všechny nápovědy vytvořené v symbolické reprezentaci a zjistíme, které symboly v nich nejsou použity. Poté již postupujeme stejně jako u ostatních typů nápověd. Použijeme tedy překladový slovník, který jsme zkonstruovali při generování instancí, a přeložíme nalezené symboly do přirozeného jazyka. Kategorie instancí tentokrát ignorujeme, protože je nebudeme potřebovat. Nyní použijeme vzory pro informační nápovědy, které jsou stejné pro všechny informační nápovědy a vypadají následovně:

- Jakub → V jednom z domů bydlí Jakub.
- Usain Bolt → Někdo fandí Usainu Boltovi.

- růže → Někdo má rád růže.

Opět vytvoříme řetězec, kde na každé řádce je jeden vzor a doplníme na poslední řádku instanci právě konvertované informační nápovědy následované šipkou. Odpověď GPT-3 je kompletní informační nápovědou v přirozeném jazyce, kterou přidáme do zadání.

Tímto způsobem postupně zkonvertujeme a přidáme do zadání všechny nápovědy včetně informačních, čímž je zadání kompletní a hádanka je vygenerována.

Zde jsme uvedli pouze několik příkladů vzorů pro generování, kompletní vzory pro generování kategorií, instancí a nápověd v českém i anglickém jazyce jsou k dispozice v příloze A.1.

Kapitola 4

Dokumentace

Tato kapitola je věnována dokumentaci aplikace, konkrétně uživatelské dokumentaci v sekci 4.1 a vývojářské dokumentaci v sekci 4.2.

4.1 Uživatelská dokumentace

V této sekci si proberem uživatelskou dokumentaci. Nejprve popíšeme, jak aplikaci nainstalovat, a poté, jak se aplikace používá.

4.1.1 Instalace aplikace

Aplikace se instaluje pomocí knihovny *setuptools* (viz sekce 4.2.2). Tuto knihovnu si lze stáhnout pomocí nástroje pip[1], který slouží k instalaci balíčků pro Python:

```
pip install setuptools
```

Samotná instalace aplikace se provádí pomocí skriptu *setup.py*:

```
python setup.py build
python setup.py install
```

4.1.2 Používání aplikace

Po instalaci můžeme program používat jako příkaz *einstein*. Bez dalších argumentů program vygeneruje hádanku normální obtížnosti o velikosti 5 a vypíše ji na standardní výstup. Vygenerování hádanky v symbolické reprezentaci je téměř

[1] `https://pip.pypa.io/`

okamžité. Oproti tomu vygenerování hádanky v přirozeném jazyce trvá kvůli komunikaci s jazykovým model GPT-3 několik desítek vteřin. Nyní uvedeme, jaké argumenty aplikace používá a co tyto argumenty dělají.

`-h, --help` Ignoruje všechny ostatní argumenty a pouze vypíše nápovědu k používání aplikace. Nápověda aplikace je stručným shrnutím této sekce.

`--symbolic` Vygeneruje hádanku pouze v symbolické reprezentaci namísto v přirozeném jazyce. Celý proces konverze hádanky do přirozeného jazyka se přeskočí.

`-S, --solution` Vypíše kromě zadání hádanky i řešení hádanky. Řešení je vygenerováno ve stejné reprezentaci jako zadání a je vypsáno na stejný výstup jako zadání.

`-D, --debug` Zapne ladící výpisy pro vývojáře, které jsou vypisovány na standardní výstup. Neslouží pro běžné uživatele aplikace.

`-v, --verbose` Zapne výpisy informující o průběhu postupu aplikace. Například může informovat, že jsou právě generovány kategorie hádanky a poté jaké kategorie byly konkrétně vygenerovány.

`-d, --difficulty` Nastavuje obtížnost hádanky. Možnosti jsou *jednoduchá*, *normální* a *těžká*, avšak zadávají se v angličtině. Za argument tedy můžeme doplnit buď hodnotu *easy*, *normal* nebo *hard*. Ve výchozím nastavení je generována hádanka normální obtížnosti.

`-s, --size` Nastavuje velikost hádanky. Možné velikosti jsou 4, 5 nebo 6, hodnotu příslušné velikosti doplníme za argument. Ve výchozím nastavení je generována hádanka velikosti 5.

`-L, --language` Nastavuje jazyk vygenerované hádanky. V současnosti je na výběr pouze čeština a angličtina, přičemž čeština je výchozím nastavením. Hodnoty, které se doplňují za argument, jsou zkratky příslušných jazyků: *cs* a *en*.

`-t, --theme` Nastavuje téma hádanky. Vybrané téma doplníme za argument, jazyk zadaného tématu musí odpovídat jazyku generované hádanky. Výchozím tématem hádanky je *televize* bez ohledu na jazyk vygenerované hádanky.

`-c, --categories` Nastavuje rovnou konkrétní kategorie hádanky, místo jednoho společného tématu celé hádanky. Jednotlivé kategorie oddělujeme mezerou. Počet zadaných kategorií musí být o jedna menší než je velikost hádanky, jelikož i v tomto případě je zachována povinná první kategorie, tedy *křestní jména*. Jazyk zadaných kategorií musí odpovídat jazyku generované hádanky. Nelze zároveň nastavit kategorie hádanky i téma hádanky.

`-o, --output` Nastavuje výstupní soubor, do kterého se generuje zadání a případně i řešení hádanky. Hodnota doplněná za argument je cestou k tomuto souboru. V případě, že soubor neexistuje, je vytvořen, avšak všechny adresáře v zadané cestě musí existovat. Pokud soubor existuje je přepsán. Ve výchozím nastavení je hádanka vypsána na standardní výstup.

`-a, --apikey` Nastavuje cestu k souboru, který obsahuje API klíč pro OpenAI. Ve výchozím nastavení se používá klíč obsažený v systémové proměnné s názvem `OPENAI_API_KEY`. Pokud chceme generovat hádanky v přirozeném jazyce, je nutné jedním z uvedených způsobů specifikovat, jaký klíč chceme použít. Jinak lze vygenerovat hádanku pouze v symbolické reprezentaci.

4.2 Vývojářská dokumentace

V této sekci popíšeme návrh architektury aplikace (sekce 4.2.1), rozebereme jednotlivé technologie použité pro vývoj aplikace (sekce 4.2.2) a zmínímě podrobnější vývojářskou dokumentaci popisující jednotlivé třídy a metody aplikace (sekce 4.2.3).

4.2.1 Návrh architektury

V této části si popíšeme návrh architektury aplikace, konkrétně členění aplikace na jednotlivé moduly. Aplikace je rozdělena do čtyř modulů: řídící modul, generátor, konvertor a konfigurace, které odpovídají jednotlivým zdrojovým souborům aplikace.

Řídící modul Řídící modul je hlavním modulem aplikace a jak vyplývá z jeho názvu, řídí celý chod aplikace a ostatní moduly. Nejprve má za úkol zpracovat jednotlivé argumenty předané aplikaci na příkazové řádce a některé z nich uloží do globálních proměnných z konfiguračního modulu. Dále má na starost vstup a výstup aplikace, musí tedy zpracovat jednotlivé argumenty předané aplikaci na příkazové řádce, adekvátně zpracovat zadaný požadavek a vypsat výstup.

Generátor Modul generátor slouží k vygenerování řešení a zádaní hádanky v symbolické reprezentaci, jak jsme popsali v sekcích 3.1 a 3.2.

Konvertor Modul konvertor slouží k převedení hádanky ze symbolické reprezentace do přirozeného jazyka způsobem, který jsme popsali v sekci 3.3. Tento modul obstarává veškerou komunikaci s jazykovým modelem GPT-3 pomocí API[2], které poskytuje OpenAI.

Konfigurace Modul konfigurace poskytuje několik globálních proměnných, které slouží ke konfiguraci aplikace a které nastavuje řídící modul.

4.2.2 Použité technologie

Celá aplikace je napsaná v programovacím jazyce Python. Dalšími technologiemi použitými pro vývoj aplikace jsou nástroje mypy, Pylint, setuptools a Sphinx.

Python Python[3] je vysokoúrovňový programovací jazyk s dynamickým typováním. Tento jazyk byl pro vývoj aplikace vybrán z několika důvodů. Zaprvé se jedná o snadno použitelný programovací jazyk, s kterým má autorka této práce předchozí zkušenosti. Zadruhé OpenAI poskytuje API[4] pro přístup ke svým modelům, včetně různých verzí jazykového modelu GPT-3, právě pro Python a také pro Node.js. Zatřetí se Python skvěle hodí pro tento typ aplikace, tedy malou a výpočetně nenáročnou aplikaci na příkazové řádce.

mypy Mypy[5] je nástroj pro statickou kontrolu typů pro programovací jazyk Python. K této kontrole používá typové nápovědy[6], které byly přidány do Pythonu ve verzi 3.5 a které jsou při standardním spuštění aplikace ignorovány. Cílem mypy je částečně vykrýt nevýhodu dynamického typování a odhalit chyby v typování bez spouštění programu.

Pylint Pylint[7] je statický analyzátor kódu, který odhaluje chyby, hledá code smells, vynucuje programovací konvence a navrhuje, jak by mohl být kód refaktorován.

[2] `https://platform.openai.com/docs/api-reference`
[3] `https://www.python.org`
[4] `https://platform.openai.com/docs/api-reference`
[5] `https://mypy.readthedocs.io`
[6] `https://peps.python.org/pep-0484/`
[7] `https://pylint.readthedocs.io`

Setuptools Setuptools[8] je knihovna navržená pro snadné balíčkování a sdílení projektů v Pythonu. Aplikace používá setuptools pro jednoduchou instalaci popsanou v sekci 4.1.1.

Sphinx Sphinx[9] je dokumentační generátor přímo pro programovací jazyk Python a je proto kompatibilní s konvencí formátování dokumentačních komentářů v Pythonu[10], která je odlišná od jiných programovacíh jazyků. Navíc při generování dokumentace ponechává v signaturách metod i typové nápovědy a není proto potřeba typy nesmyslně opisovat do dokumentačních komentářů, když leží v kódu hned pod signaturou metody. Sphinx tak splňuje unikátní požadavky Pythonu, narozdíl od jiných dokumentačních nástrojů jako je například *doxygen*.

4.2.3 Popis tříd a metod

Podrobnější vývojářskou dokumentaci s popisem jednotlivých tříd a metod lze nalézt v příloze A.3.2. Tato dokumentace je vygenerována z dokumentačních komentářů ve zdrojových souberech aplikace pomocí nástroje Sphinx[11] popsaného v sekci 4.2.2.

[8] `https://setuptools.pypa.io`
[9] `https://www.sphinx-doc.org`
[10] `https://peps.python.org/pep-0257/`
[11] `https://www.sphinx-doc.org`

Kapitola 5

Evaluace výsledků

V této kapitole zhodnotíme výsledky generování hádanek a rozebereme úpravy, kterými jsme se pokoušeli generované hádanky vylepšit. Dále porovnáme generování hádanek v češtině a generování hádanek v angličtině. Na závěr sekce ukážeme kreativitu vygenerovaných nápověd a navrhneme možnosti dalšího vývoje aplikace.

5.1 Vylepšování vzorů

První vzory, které byly vytvořeny pouze pro češtinu, měly jednodušší formát a každá sada vzorů obsahovala pouze tři vzory včetně vzorů pro generování kategorií a instancí. Vzory pro generování instancí neobsahovali téma a vzory pro generování nápověd neobsahovali ani téma ani příslušné kategorie instancí obsažených v nápovědě. Jak se ukázalo následným testováním, jednalo se o dobrý počáteční bod, jelikož již s těmito vzory bylo možné vygenerovat pěknou hádanku s nápovědami, které byly většinou srozumitelné, s minimem gramatických či stylistických nesrovnalostí a velmi různorodé.

Nicméně při testování jsme objevili několik problémů, jejichž výskyt v generovaných hádankách jsme se snažili omezit postupnými úpravami vzorů. První nalezený problém spočíval v tom, že v některých vygenerovaných nápovědách byla daná instance použita v jiném kontextu než z jaké kategorie byla vygenerována. Ukažme si tuto situaci na konkrétním příkladu hádanky, kterou vygenerovala aplikace s první verzí vzorů. Hádanka měla zadané téma *fotbal* a povinnou kategorii *křestní jména* doplnily vygenerované kategorie *fotbalisté*, *fotbalové týmy* a *fotbalové ligy*. Pro instance *La Liga* z kategorie *fotbalové ligy* a *Ostrava* z kategorie *fotbalové týmy* bylo vygenerováno následující přiřazující pravidlo: *Ten, kdo fandí La Lize, zná město Ostrava.* Přestože je tato nápověda gramaticky i stylisticky správně česky a lze pomocí ní úspěšně hádanku vyřešit, neodpovídá použití

instance *Ostrava* kategorii *fotbalové týmy*, jelikož v této nápovědě je *Ostrava* zmíněna jako město.

Tento problém jsme se pokusili vyřešit právě přidáním tématu do vzorů pro generování instancí a nápověd. Uvedený postup pomohl, avšak pouze mírně, proto jsme hledali lepší řešení. Vzhledem k tomu, že instance v dané nápovědě byla použita nejenom v jiném kontextu než zvolené téma hádanky, ale i než kategorie odpovídající této instanci, rozhodli jsme se přidat do vzorů pro generování nápověd i kategorie příslušných instancí. Uvedený nápad vznikl poté, co aplikace vygenerovala hádanku s kategorií *míče*, jejíž instance byly všechny přídavnými jmény, například *nafukovací*. Hned několik nápověd této hádanky mělo uvedený problém, kupříkladu: *Ten, kdo se rád prochází po Allianz Areně, má spoustu radosti z nafukovacích předmětů.* Bylo tedy jasně patrné, že při generování uvedené nápovědy by pomohlo, kdybychom GPT-3 předali informaci, že se jedná o *nafukovací míč*, a tak jsme se rozhodli tuto informaci do vzorů přidat. Nelze s jistotou říci, že již tento problém nemůže nastat, ale minimálně tato úprava značně snížila jeho výskyt.

Další problém se netýká generování nápověd, nýbrž již generování kategorií a instancí. Samotné generování nápověd tento problém tedy neovlivňuje, avšak zůstává při něm zachován. Občas se stane, že dvojice vygenerovaných kategorií si je natolik podobná, že instance vygenerované z těchto dvou kategorií nelze s jistotou rozlišit. V nejhorším případě se některé instance dokonce opakují, a výsledná hádanka je proto vyloženě matoucí. Například pro hádanku s tématem *sport*, byly vygenerovány kategorie *týmy* a *hry*, která ovšem obě obsahovali pouze sporty, navíc instance *basketbal* se vyskytovala v obou těchto kategoriích. Tento problém jsme se pokusili řešit úpravou a rozšířením vzorů pro generování kategorií i instancí. Přestože rozšíření vzorů generování hádanek obecně vzato pomohlo, naneštěstí výše uvedený problém přetrvává a nepodařilo se ho uspokojivě vyřešit. Se znalostí rozdělení instancí do kategorií, které aplikace vypisuje ve *verbose* módu, lze hádanku s tímto problémem vyřešit, pokud neobsahuje identické instance, což ovšem nastává zřídka. Do aplikace byla navíc přidána možnost specifikovat místo tématu rovnou kategorie hádanky, což poskytuje možnost se uvedenému problému vhodnou volbou kategorií vyhnout.

5.2 Srovnání výsledků v českém a anglickém jazyce

V této sekci porovnáme výsledky generování hádanek v českém a v anglickém jazyce. Srovnáváme pouze konečný výsledek aplikace, tedy kompletně vygenerované zadání hádanky. Pro každý z jazyků jsme vygenerovali 10 Einsteinových

hádanek. Pro vygenerování každé dvojice hádanek jsme použili přesně stejné parametry aplikace kromě zadaného jazyka a jazyka zadaného tématu, které bylo adekvátně přeloženo. Poté jsme prošli všechny nápovědy všech vygenerovaných hádanek a u každé jsme určili zda je srozumitelná či nikoli. Výsledky, reprezentující procenta srozumitelných nápověd, jsou zobrazeny v grafu 5.1 rozdělené podle jazyků hádanky a podle jednotlivých typů nápověd. Graf také obsahuje souhrnné výsledky všech typů nápověd pro oba jazyky.

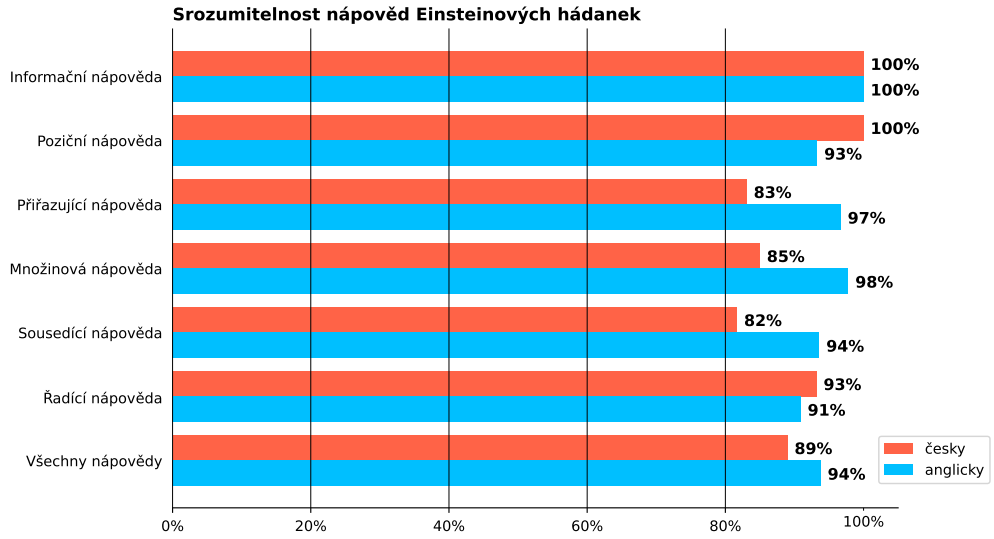
Z grafu můžeme pozorovat, že z celkového hlediska generuje aplikace srozumitelnější hádanky pro anglický jazyk a že výsledky pro jednotlivé typy nápověd jsou pro angličtinu srovnatelné. Generování hádanek v českém jazyce je mírně horší, kvůli značné nevyrovnanosti ve srozumitelnosti jednotlivých typů nápověd. Přiřazující, množinové a sousedící nápovědy ve vygenerovaných hádankách jsou častěji nesrozumitelné, oproti tomu poziční a řadící nápovědy mají lepší výsledky pro češtinu než pro angličtinu.

Tyto výsledky mají pravděpodobně dva hlavní důvody. Zaprvé je model GPT-3 rozsáhleji natrénován na anglických datech než na českých. Zadruhé čeština má složitější gramatiku se skloňováním a třemi odlišnými rody slov, což se projeví zejména při generování nápověd se složitějšími souvětími, kdy celková náročnost vygenerování srozumitelné hádanky je už zkrátka příliš vysoká.

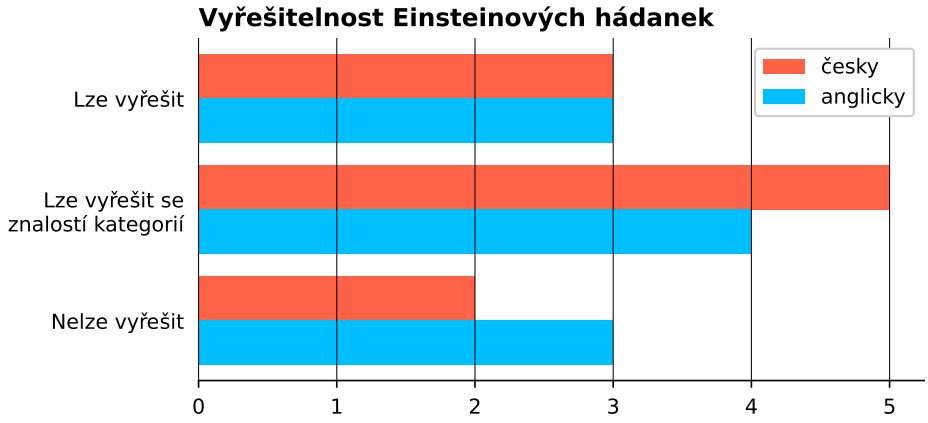
Nyní porovnáme vyřešitelnost hádanek v jednotlivých jazycích. Za vyřešitelnost hádanky považujeme, že jazyková stránka hádanky nebrání vyřešení logické hádanky, samotnou logickou část hádanky ovšem neřešíme. Budeme rozlišovat zda jde hádanka vyřešit bez informace o rozdělení instancí do jednotlivých kategorií, s touto informací či nejde vyřešit vůbec. Výsledky pro stejných 10 vygenerovaných Einsteinových hádanek, jako jsme použili pro porovnání srozumitelnosti nápověd, můžeme vidět v grafu 5.2. Jak můžeme snadno nahlédnout česká a anglická verze hádanky se ve vyřešitelnosti téměř neliší. Nutnost znát rozdělení instancí do kategorií je typicky způsobena příliš podobnými kategoriemi, jak jsme již zmínili v sekci 5.1. Nemožnost vyřešit hádanku nejčastěji způsobuje duplikace nějaké instance. Přesto je většina hádanek vyřešitelných, a výsledky generování jsou tak přijatelné.

5.3 Nelogičnost nápověd

Protože při generování spojujeme instance, které patří k jednomu člověku, náhodně, můžou vznikat zdánlivě nelogické kombinace instancí a z nich následně zdánlivě nelogické nápovědy. Například v hádance na téma *literatura* byly k jedné osobě přiřazeny instance *J. K. Rowling* z kategorie *autoři* a *faktografická literatura* z kategorie *žánry* a z těchto informací, pak byla vygenerována následující nápověda: *Ten, kdo má rád J.K. Rowlingovou, má rád i faktografickou literaturu.* Přestože



Obrázek 5.1 Graf zobrazující srozumitelnost jednotlivých typů nápověd a souhrnnou srozumitelnost nápověd v českém a v anglickém jazyce



Obrázek 5.2 Graf zobrazující vyřešitelnost Einsteinových hádanek v českém a v anglickém jazyce

každý může mít rád jakékoli i zdánlivě nesouvisející kombinace věcí, působí tato nápověda mírně nelogicky, když *J. K. Rowling* nepíše *faktografickou literaturu*. Avšak toto chování je požadované, jelikož kdybychom vždy párovali odpovídající instance, šla by hádanka částečně řešit pomocí znalostí o daném tématu hádanky. Nicméně protože chceme zachovat, že Einsteinova hádanka je logickou hádankou, nemůžeme v hádance řešiteli takto napovídat.

5.4 Zhodnocení kreativity nápověd

Kvalitu vygenerovaných hádanek nelze posoudit pouze podle jejich srozumitelnosti. V této sekci si proto ukážeme několik kreativních nápověd, které byly vygenerovány v rámci hádanek použitých pro porovnání jazykových verzí v předchozí sekci 5.2. Každá z uvedených nápověd obsahuje nějaké originální slovo (případně frázi), které není používané často a není obsaženo ve vzorech pro generování, a přesto je v následujících větách použito gramaticky a stylisticky zcela správně.

- Člověk, který obdivuje designéry jako Karl Lagerfeld, bydlí nalevo od člověka, který je *posedlý* grunge.
- Člověk, který se rád vypraví do Paříže, bydlí nalevo od člověka, který *se zamýšlí* nad návštěvou Itálie.
- John bydlí napravo od člověka, který si užil *nezapomenutelné* safari.
- Ten, kdo je *zapálený* pro poezii, bydlí v domě číslo 1.
- *Knihomol*, který miluje díla Honoré de Balzaca, sousedí se člověkem, který je vášnivým příznivcem Roméa a Julie.

5.5 Další vývoj aplikace

Aplikaci by šlo dále vylepšovat. Zaprvé by bylo vhodné opravit problém předestřený v sekci 5.1, buď úpravou vzorů nebo jiným nastavením jazykového modelu GPT-3. Dále je možné experimentovat se vzory, ať už z hlediska jejich formátu, příkladů v nich uvedených nebo jejich množství, a tím se pokusit ještě zlepšit srozumitelnost hádanek při zachování kreativity nápověd. Zajímavým rozšířením by bylo generování úvodního odstavce zadání, který seznamuje řešitele s kategoriemi dané hádanky. Ukázku takového odstavce můžeme vidět v příkladu Einsteinovy hádanky uvedeném v úvodu práce.

Závěr

V rámci této bakalářské práci jsme popsali způsob generování Einsteinových hádanek v přirozeném jazyce pomocí jazykového modelu GPT-3 a analyzovali jsme výsledky tohoto postupu.

Nejprve jsme definovali pojem *Einsteinova hádanka*. Specifikovali jsme, v jakém formátu budeme hádanky generovat a jaké typy nápověd budeme mít při generování k dispozici. Rozebrali jsme existující aplikace generující Einsteinovy hádanky a zjistili jsme, že žádné nesplňují naše požadavky na kreativitu a originalitu hádanek.

Dále jsme popsali, jak fungují jazykové modely GPT a v čem se liší jejich verze, na což jsme navázali vysvětlením celého postupu generování zadání hádanek, od vygenerování řešení v symbolické reprezentaci (viz sekce 3.1), přes vygenerování zadání v symbolické reprezentaci (viz sekce 3.2) až po konverzi hádanky do přirozeného jazyka (viz sekce 3.3).

V kapitole 5 jsme analyzovali Einsteinovy hádanky vygenerované aplikací. Zjistili jsme, že vygenerované hádanky jsou kreativní a různorodé, a také uspokojivě srozumitelné a vyřešitelné, přičemž hádanky generované v anglickém jazyce jsou mírně srozumitelnější než hádanky generované v českém jazyce. Zároveň jsme objevili několik nedostatků, z nichž některé se v současné chvíli nepodařilo odstranit. Aplikaci lze nadále vylepšovat, jak jsme uvedli v sekci 5.5). Cílem další práce by mohlo být zejména vylepšení srozumitelnosti generovaných hádanek a minimalizování problému s generováním podobných kategorií a duplikovaných instancí, aniž by byla obětována kreativita generovaných hádanek, které již bylo dosaženo (viz sekce 5.4).

Práci zakončíme dvěma Einsteinovými hádanámi, jedné v češtině a jedné v angličtině, které vygenerovala vytvořená aplikace. Obě hádanky mají velikost 4, lehkou obtížnost a téma *televize*. Hádanky ponecháme čtenáři jako cvičení, nicméně řešení jsou uvedena v příloze A.2.

Hádanka v češtině má tyto kategorie: *křestní jména*, *seriály*, *dabing*, *pořady* a byly pro ní vygenerovány následující nápovědy:

1. *Někdo chce vidět nové díly seriálu Friends.*
2. *Ten, kdo miluje seriál Stranger Things, bydlí v domě číslo 2.*

3. *Chris bydlí v domě číslo 1.*
4. *Ten, který preferuje španělské dabingované televizní pořady, bydlí v domě číslo 4.*
5. *David rád sleduje seriály Big Bang Theory.*
6. *Ian ovládá všechny díly seriálu Stranger Things.*
7. *Ten, kdo má rád italský dabing, rád se dívá na dokumentární pořady.*
8. *Ten, kdo má rád sitcomy, také se rád dívá na seriál Stranger Things.*
9. *Ten, kdo preferuje dabing v angličtině, rád se dívá na seriál Game of Thrones.*
10. *John rád sleduje pořady s anglickým dabingem.*
11. *Ten, kdo má rád italský dabing, bydlí nalevo od člověka, který se rád dívá na seriál Big Bang Theory.*
12. *Ten, kdo rád sleduje televizní programy v italštině, bydlí nalevo od člověka, který rád sleduje televizní programy v angličtině.*
13. *Ten, kdo má rád zábavné pořady, bydlí nalevo od člověka, který rád sleduje televizi s španělským dabingem.*
14. *Když se někdo rád dívá na dabing v angličtině, obvykle si vybírá mezi dokumentární pořad, reality show nebo zábavnou show.*
15. *David bydlí napravo od osoby, která ráda sleduje dokumentární pořady.*
16. *Ten, kdo má rád italštinu ve filmovém dabingu, bydlí nalevo od člověka, který rád mluví francouzsky.*
17. *Ten, kdo má rád Big Bang Theory, měl by si pustit seriál v kombinaci s francouzským, španělským nebo anglickým dabingem.*
18. *Člověk, který má rád zábavné show, vedle něho bude ten, kdo má rád Big Bang Theory.*
19. *Člověk, který si oblíbil španělský dabing, sousedí s někým, kdo rád sleduje seriál Hra o trůny.*
20. *John bydlí vedle člověka, který má rád reality show.*

Hádanka v angličtině má tyto kategorie: *first names*, *shows*, *channels*, *actors* a byly pro ní vygenerovány následující nápovědy:

1. *Monica lives in one of the houses.*
2. *Someone watches ABC.*
3. *The person who loves watching animated shows lives in house number 3.*
4. *Phoebe lives in house number 2.*
5. *The person who watches HBO lives in house number 3.*
6. *The person who likes Johnny Galecki also likes comedy shows.*
7. *The person who likes DISNEY channel enjoys watching cartoons.*
8. *The person who likes Kaley Cuoco also enjoys watching animated shows.*

9. *The person who enjoys Kaley Cuoco also likes to watch HBO channels.*
10. *The person who likes Jim Parsons would also enjoy watching cartoons.*
11. *Joey likes watching animated TV shows.*
12. *The person who loves Johnny Galecki lives to the left of a person who likes animation.*
13. *Rachel lives to the left of a person who enjoys watching Ellen DeGeneres' shows.*
14. *The person who likes Johnny Galecki lives to the left of a person whose favorite television show is a drama.*
15. *The person who watches comedies can be a neighbor of a person who likes watching cartoons.*
16. *Rachel lives next to a person who is a fan of Johnny Galecki.*
17. *The person who watches NBC lives to the left of a person who likes Ellen DeGeneres.*
18. *The person who enjoys watching Kaley Cuoco lives next to the person who likes Johnny Galecki.*
19. *Rachel lives to the left of a person whose favorite actor is Kaley Cuoco.*
20. *The person who is a fan of Johnny Galecki lives to the left of a person who is a fan of Kaley Cuoco.*
21. *The person who likes cartoons lives to the left of a person who is a fan of Kaley Cuoco.*

Seznam použité literatury

[1] Jeremy Stangroom. *Einstein's Riddle: 50 Riddles, Puzzles, and Conundrums to Stretch Your Mind*. A&C Black, 2009.

[2] Jay Alammar. "The illustrated gpt-2 (visualizing transformer language models)". In: *Jalammar. github. io. https://jalammar. github. io/illustrated-gpt2* (2019).

[3] Jay Alammar. "The illustrated word2vec (2019)". In: *URL: https://jalammar. github. io/illustrated-word2vec* (2021).

[4] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[5] Gokul Yenduri et al. "Generative Pre-trained Transformer: A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions". In: *arXiv e-prints* (2023), arXiv–2305.

[6] Alec Radford et al. "Improving language understanding by generative pre-training". In: *OpenAI blog, p. 9* (2018).

[7] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), s. 1877–1901.

[8] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: *OpenAI blog, p. 9* (2019).

[9] OpenAI. *GPT-4 Technical Report*. 2023. arXiv: 2303.08774 [cs.CL].

Příloha A

Přílohy

A.1 Vzory pro generování

Zde uvádíme kompletní vzory pro generování kategorií, instancí a nápověd v češtině i v angličtině. Vzory pro generování kategorií a instancí jsou uváděny pro velikost hádanky 6, tedy největší velikost. Pro menší velikosti odebereme od konce tolik kategorií či instancí o kolik je požadovaná hádanka menší a jinak vzory ponecháme stejné.

A.1.1 Vzory pro generování v češtině

Vzory pro generování kategorií

- malířství → malíři, obrazy, styly, barvy, galerie
- jídlo → zelenina, mléčné výrobky, maso, pečivo, ovoce
- hudba → skladby, žánry, zpěváci, nástroje, kapely
- příroda → zvířata, rostliny, národní parky, řeky, hory
- atletika → disciplíny, sportovci, náčiní, soutěže, kluby

Vzory pro generování instancí

- malířství : styly → realismus, impresionismus, moderna, renesance, expresionismus, kubismus
- jídlo : pečivo → houska, chleba, donut, koláč, bageta, brioška
- hudba : žánry → pop, rock, klasika, jazz, country, hip hop
- atletika : náčiní → oštěp, koule, tyčka, disk, kladivo, překážky
- příroda : zvířata → medvěd, lev, kůň, liška, klokan, bobr

Vzory pro generování informační nápovědy

- Jakub → V jednom z domů bydlí Jakub.
- Usain Bolt → Někdo fandí Usainu Boltovi.
- růže → Někdo má rád růže.

Vzory pro generování poziční nápovědy

- S křestním jménem vlevo:
  - malířství : křestní jména - Jakub = čísla - 0 → Jakub bydlí v domě číslo 1.
  - jídlo : křestní jména - Martin = čísla - 4 → Martin bydlí v domě číslo 5.
  - hudba : křestní jména - Jana = čísla - 2 → Jana bydlí v domě číslo 3.
- Bez křestního jména:
  - atletika : disciplíny - maraton = čísla - 0 → Ten kdo rád běhá maraton bydlí v domě číslo 1.
  - malířství : barvy - modrá = čísla - 4 → Ten jehož nejoblíbenější barva je modrá bydlí v domě číslo 5.
  - jídlo : pečivo - chleba = čísla - 2 → Ten kdo rád jí chleba bydlí v domě číslo 3.

Vzory pro generování přiřazující nápovědy

- S křestním jménem vlevo:
  - malířství : křestní jména - Martin = barvy - modrá → Martinova nejoblíbenější barva je modrá.
  - jídlo : křestní jména - Jana = pečivo - chleba → Jana ráda jí chleba.
  - hudba : křestní jména - Jakub = žánry - pop → Jakub rád poslouchá pop.
- Bez křestního jména:
  - hudba : zpěváci - Ed Sheeran = žánry - pop → Ten kdo má rád Eda Sheerana rád pouslouchá pop.
  - malířství : styly - impresionismus = barvy - modrá → Nejoblíbenější barva člověk který má rád impresionismus je modrá.
  - jídlo : zelenina - paprika = pečivo - chleba → Ten kdo rád jí papriku také rád jí chleba.
- S křestním jménem vpravo:
  - malířství : barvy - modrá = křestní jména - Martin → Martinova nejoblíbenější barva je modrá.
  - jídlo : pečivo - chleba = křestní jména - Jana → Jana ráda jí chleba.
  - hudba : žánry - pop = křestní jména - Jakub → Jakub rád poslouchá pop.

Vzory pro generování množinové nápovědy

- S křestním jménem vlevo:
  - malířství : křestní jména - Martin = barvy - [modrá, zelená, žlutá] → Martinova nejoblíbenější barva je buď modrá zelená nebo žlutá.
  - jídlo : křestní jména - Jana = pečivo - [chleba, donut, bageta] → Jana ráda jí buď chleba donut nebo bagetu.
  - hudba : křestní jména - Jakub = žánry - [pop, rock, jazz] → Jakub rád pouslouchá buď pop rock nebo jazz.
- Bez křestního jména:
  - hudba : zpěváci - Ed Sheeran = žánry - [pop, rock, jazz] → Ten kdo má rád Eda Sheerana rád pouslouchá buď pop rock nebo jazz.
  - malířství : styly - impresionismus = barvy - [modrá, zelená, žlutá] → Nejoblíbenější barva člověk který má rád impresionismus je buď modrá zelená nebo žlutá.
  - jídlo : zelenina - paprika = pečivo - [chleba, donut, bageta] → Ten kdo rád jí papriku také rád jí buď chleba donut nebo bagetu.
- S křestním jménem vpravo:
  - malířství : barvy - modrá = křestní jména - [Jana Vojta Aneta] → Buď Janina Vojtova nebo Anetina nejoblíbenější barva je modrá.
  - jídlo : pečivo - chleba = křestní jména - [Klára Simona Kačka] → Buď Klára Simona nebo Kačka ráda jí chleba.
  - hudba : žánry - pop = křestní jména - [Jakub Martin Michal] → Buď Jakub Martin nebo Michal rád poslouchá pop.

Vzory pro generování sousedící nápovědy

- S křestním jménem vlevo:
  - malířství : křestní jména - Martin = barvy - modrá → Martin bydlí vedle člověka jehož nejoblíbenější barva je modrá.
  - jídlo : křestní jména - Jana = pečivo - chleba → Janin soused rád jí chleba.
  - hudba : křestní jména - Jakub = žánry - pop → Jakub bydlí vedle osoby která ráda poslouchá pop.
- Bez křestního jména:
  - hudba : zpěváci - Ed Sheeran = žánry - pop → Ten kdo má rád Eda Sheerana a ten kdo rád pouslouchá pop jsou sousedé.
  - malířství : styly - impresionismus = barvy - modrá → Člověk který má rád impresionismus bydlí vedle člověka jehož nejoblíbenější barva je modrá.
  - jídlo : zelenina - paprika = pečivo - chleba → Ten kdo rád jí papriku sousedí s člověkem který rád jí chleba.
- S křestním jménem vpravo:

  - malířství : barvy - modrá = křestní jména - Martin → Martin bydlí vedle člověka jehož nejoblíbenější barva je modrá.
  - jídlo : pečivo - chleba = křestní jména - Jana → Janin soused rád jí chleba.
  - hudba : žánry - pop = křestní jména - Jakub → Jakub bydlí vedle osoby která ráda poslouchá pop.

Vzory pro generování řadící nápovědy

- S křestním jménem vlevo:
  - malířství : křestní jména - Martin = barvy - modrá → Martin bydlí nalevo od člověka jehož nejoblíbenější barva je modrá.
  - jídlo : křestní jména - Jana = pečivo - chleba → Jana bydlí nalevo od osoby která ráda jí chleba.
  - hudba : křestní jména - Jakub = žánry - pop → Jakub bydlí nalevo od osoby která ráda poslouchá pop.
- Bez křestního jména:
  - hudba : zpěváci - Ed Sheeran = žánry - pop → Ten kdo má rád Eda Sheerana bydlí nalevo od člověka který rád poslouchá pop.
  - malířství : styly - impresionismus = barvy - modrá → Člověk který má rád impresionismus bydlí nalevo od člověka jehož nejoblíbenější barva je modrá.
  - jídlo : zelenina - paprika = pečivo - chleba → Ten kdo rád jí papriku bydlí nalevo od člověka který rád jí chleba.
- S křestním jménem vpravo:
  - malířství : barvy - modrá = křestní jména - Martin → Martin bydlí napravo od člověka jehož nejoblíbenější barva je modrá.
  - jídlo : pečivo - chleba = křestní jména - Jana → Jana bydlí napravo od osoby která ráda jí chleba.
  - hudba : žánry - pop = křestní jména - Jakub → Jakub bydlí napravo od osoby která ráda poslouchá pop.

A.1.2 Vzory pro generování v angličtině

Vzory pro generování v angličtině jsou pouze přeložené vzory pro generování v češtině, avšak pro úplnost je zde taktéž uvádíme kompletní.

Vzory pro generování kategorií

- painting → painters, paintings, styles, colors, galleries
- food → vegetables, dairy products, meat, pastries, fruits
- music → songs, genres, singers, instruments, bands
- nature → animals, plants, national parks, rivers, mountains
- athletics → disciplines, athletes, equipment, competitions, clubs

Vzory pro generování instancí

- painting : styles → realism, impressionism, modernism, renaissance, expressionism, cubism
- food : pastries → bun, bread, donut, pie, baguette, brioche
- music : genres → pop, rock, classical, jazz, country, hip hop
- athletics : equipment → javelin, shot, pole, discus, hammer, hurdles
- nature : animals → bear, lion, horse, fox, kangaroo, beaver

Vzory pro generování informační nápovědy

- Jacob → Jacob lives in one of the houses.
- Usain Bolt → Someone roots for Usain Bolt.
- rose → Someone likes roses.

Vzory pro generování poziční nápovědy

- S křestním jménem vlevo:
  - painting : first names - Jacob = numbers - 0 → Jacob lives in house number 1.
  - food : first names - Martin = numbers - 4 → Martin lives in house number 5.
  - music : first names - Jane = numbers - 2 → Jane lives in house number 3.
- Bez křestního jména:
  - athletics : disciplines - marathon = numbers - 0 → The person who enjoys running marathons lives in house number 1.
  - painting : colors - blue = numbers - 4 → The person whose favorite color is blue lives in house number 5.
  - food : pastries - bread = numbers - 2 → The person who enjoys eating bread lives in house number 3.

Vzory pro generování přiřazující nápovědy

- S křestním jménem vlevo:
  - painting : first names - Martin = colors - blue → Martin's favorite color is blue.
  - food : first names - Jane = pastries - bread → Jane enjoys eating bread.
  - music : first names - Jacob = genres - pop → Jacob enjoys listening to pop music.
- Bez křestního jména:
  - music : singers - Ed Sheeran = genres - pop → The person who likes Ed Sheeran also enjoys listening to pop music.
  - painting : styles - impressionism = colors - blue → The favorite color of a person who enjoys impressionism is blue.
  - food : vegetables - pepper = pastries - bread → The person who likes peppers also enjoys eating bread.

- S křestním jménem vpravo:
  - painting : colors - blue = first names - Martin → Martin's favorite color is blue.
  - food : pastries - bread = first names - Jane → Jane enjoys eating bread.
  - music : genres - pop = first names - Jacob → Jacob enjoys listening to pop music.

Vzory pro generování množinové nápovědy

- S křestním jménem vlevo:
  - painting : first names - Martin = colors - [blue, green, yellow] → Martin's favorite color is either blue green or yellow.
  - food : first names - Jane = pastries - [bread, donut, baguette] → Jane enjoys eating either bread donuts or baguettes.
  - music : first names - Jacob = genres - [pop, rock, jazz] → Jacob enjoys listening to either pop rock or jazz music.
- Bez křestního jména:
  - music : singers - Ed Sheeran = genres - [pop, rock, jazz] → The person who likes Ed Sheeran also enjoys listening to either pop rock or jazz music.
  - painting : styles - impressionism = colors - [blue, green, yellow] → The favorite color of a person who likes impressionism is either blue green or yellow.
  - food : vegetables - pepper = pastries - [bread, donut, baguette] → The person who likes peppers also enjoys eating either bread donuts or baguettes.
- S křestním jménem vpravo:
  - painting : colors - blue = first names - [Jane, Tom, Emma] → Either Jane's Tom's or Emma's favorite color is blue.
  - food : pastries - bread = first names - [Clara, Simone, Kate] → Either Clara Simone or Kate enjoys eating bread.
  - music : genres - pop = first names - [Jacob, Martin, Michael] → Either Jacob Martin or Michael enjoys listening to pop music.

Vzory pro generování sousedící nápovědy

- S křestním jménem vlevo:
  - painting : first names - Martin = colors - blue → Martin lives next to a person whose favorite color is blue.
  - food : first names - Jane = pastries - bread → Jane's neighbor enjoys eating bread.
  - music : first names - Jacob = genres - pop → Jacob lives next to a person who enjoys listening to pop music.
- Bez křestního jména:
  - music : singers - Ed Sheeran = genres - pop → The person who likes Ed Sheeran and the person who enjoys pop music are neighbors.

  - painting : styles - impressionism = colors - blue → The person who likes impressionism lives next to a person whose favorite color is blue.
  - food : vegetables - pepper = pastries - bread → The person who enjoys eating peppers is neighbors with a person who likes bread.
- S křestním jménem vpravo:
  - painting : colors - blue = first names - Martin → Martin lives next to a person whose favorite color is blue.
  - food : pastries - bread = first names - Jane → Jane's neighbor enjoys eating bread.
  - music : genres - pop = first names - Jacob → Jacob lives next to a person who enjoys listening to pop music.

Vzory pro generování řadící nápovědy

- S křestním jménem vlevo:
  - painting : first names - Martin = colors - blue → Martin lives to the left of a person whose favorite color is blue.
  - food : first names - Jane = pastries - bread → Jane lives to the left of a person who enjoys eating bread.
  - music : first names - Jacob = genres - pop → Jacob lives to the left of a person who enjoys listening to pop music.
- Bez křestního jména:
  - music : singers - Ed Sheeran = genres - pop → The person who likes Ed Sheeran lives to the left of a person who enjoys pop music.
  - painting : styles - impressionism = colors - blue → The person who likes impressionism lives to the left of a person whose favorite color is blue.
  - food : vegetables - pepper = pastries - bread → The person who enjoys eating peppers lives to the left of a person who likes bread.
- S křestním jménem vpravo:
  - painting : colors - blue = first names - Martin → Martin lives to the right of a person whose favorite color is blue.
  - food : pastries - bread = first names - Jane → Jane lives to the right of a person who enjoys eating bread.
  - music : genres - pop = first names - Jacob → Jacob lives to the right of a person who enjoys listening to pop music.

A.2 Řešení Einsteinových hádanek

Zde uvádíme řešení dvou Einsteinových hádanek v tabulkách A.1 a A.2, jejichž zadání se nachází v závěru práce.

	Dům 1	Dům 2	Dům 3	Dům 4
Jména	Chris	Ian	John	David
Seriály	Friends	Stranger Things	Game of Thrones	Big Bang Theory
Dabing	italštině	francouzský	angličtině	španělský
Pořady	dokumentární pořad	sitcom	zábavná show	reality show

Tabulka A.1 Řešení Einsteinovy hádanky v českém jazyce

	Dům 1	Dům 2	Dům 3	Dům 4
First names	Rachel	Phoebe	Joey	Monica
Shows	cartoon	comedy	animation	drama
Channels	DISNEY	NBC	HBO	ABC
Actors	Jim Parsons	Johnny Galecki	Kaley Cuoco	Ellen DeGeneres

Tabulka A.2 Řešení Einsteinovy hádanky v anglickém jazyce

A.3 Přiložená složka `einstein`

Přílohou práce je zkomprimovaná složka `einstein` obsahující repozitář se zdrojovým kódem aplikace a vnořenou složku s vývojářskou dokumentací.

A.3.1 Repozitář `einstein_app`

Gitový repozitář uložený ve vnořené složce `einstein_app` obsahuje soubor `README.md`, veškerý zdrojový kód aplikace a konfigurační soubory používaných nástrojů (viz sekce 4.2.2).

A.3.2 Složka `einstein_docs`

Vnořená složka `einstein_docs` obsahuje kompletní vývojářskou dokumentaci s popisem modulů aplikace i podrobnou dokumentaci jednotlivých tříd a metod aplikace, vygenerovanou pomocí nástroje *Sphinx*[1]. Dokumentace je ve formě webové stránky, výchozí soubor je `index.html`.

[1] `https://www.sphinx-doc.org/en/master/`