

Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra kybernetiky

DIPLOMOVÁ PRÁCE

PLZEŇ, 2024

Bc. Tomáš LEBEDA

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení: **Bc. Tomáš LEBEDA**
Osobní číslo: **A22N0109P**
Studijní program: **N0714A150011 Kybernetika a řídicí technika**
Specializace: **Umělá inteligence a automatizace**
Téma práce: **Sémantická korespondence obrazu a řeči**
Zadávající katedra: **Katedra kybernetiky**

Zásady pro vypracování

1. Nastudujte problematiku reprezentace znalostí a faktů, ontologií, relační reprezentace dat.
2. Vhodnou reprezentaci použijte pro sémantický popis vybraného obrázku.
3. Navrhněte metodu, která vyhodnotí sémantickou shodu mezi řečovým popisem obrázku a jeho sémantickou reprezentací.
4. Metodu ověřte a dosažené výsledky vyhodnotěte, navrhněte případná vylepšení.

Rozsah diplomové práce:

40-50 stránek A4

Rozsah grafických prací:

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

Seznam doporučené literatury:

Dodá vedoucí práce.

Vedoucí diplomové práce:

Ing. Luboš Šmíd, Ph.D.

Katedra kybernetiky

Datum zadání diplomové práce:

2. října 2023

Termín odevzdání diplomové práce: **20. května 2024**

Doc. Ing. Miloš Železný, Ph.D.
děkan



Doc. Dr. Ing. Vlasta Radová
vedoucí katedry

Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne

.....

Chtěl bych poděkovat vedoucímu této práce Ing. Luboši Šmídlovi, Ph.D. za jeho lidský přístup a ochotu poradit v případě potřeby během vypracování této práce. Dále bych chtěl poděkovat své rodině a přátelům za jejich podporu.

Abstrakt

Zpracování přirozeného jazyka spolu se sémantickou analýzou a reprezentací znalostí jsou v dnešní době nedílnou součástí mnoha aplikací. Tato práce se zabývá návrhem a realizací systému pro určení sémantické korespondence obrázku a jeho popisu v přirozené řeči za účelem usnadnění diagnostických testů pro detekci kognitivních poruch, s důrazem na expertní přístup k problematice a z něj plynoucí detailní kontrolu nad jednotlivými fázemi procesu. V práci je popsán návrh systému a jeho částí, včetně zdůvodnění učiněných rozhodnutí. Poté je popsána implementace jednotlivých částí a řešení problémů, které se během práce vyskytly.

Klíčová slova

Zpracování přirozeného jazyka, sémantická analýza, reprezentace znalostí, sémantická korespondence obrazu a řeči, sémantický popis obrazu, bezkontextové gramatiky, porozumění řeči.

Abstract

Natural language processing, along with semantic analysis and knowledge representation, has become an integral part of many contemporary applications. This thesis focuses on the design and implementation of a system to determine the semantic correspondence between an image and its natural language description, aimed at facilitating diagnostic tests for the detection of cognitive disorders. Emphasis is placed on an expert approach to the subject, allowing for detailed control over the various stages of the process. The thesis outlines the design of the system and its components, including the rationale behind the decisions made. Subsequently, the implementation of the individual components and the solutions to the problems encountered during the work are described.

Keywords

Natural language processing, semantic analysis, knowledge representation, semantic correspondence of image and speech, semantic image description, context-free grammars, spoken language understanding.

Obsah

1	Úvod	10
2	Sémantika a zpracování přirozeného jazyka	11
2.1	Sémantika	11
2.2	Porozumění řeči	12
2.3	Sémantické gramatiky	14
3	Reprezentace znalostí	19
3.1	Dělení reprezentace znalostí	19
3.2	Sémantické sítě	21
3.3	Ontologie	22
4	Korespondence obrazu a jeho slovního popisu	24
4.1	Práce s obrazovými daty	24
4.2	Sémantický popis obrazu	26
5	Návrh a architektura systému	27
5.1	Obecná architektura systému	27
5.2	Reprezentace znalostí	28
5.2.1	Objekty ve scéně	30
5.2.2	Hierarchie objektů	31
5.2.3	Atributy objektů	32
5.2.4	Vazby mezi objekty	33
5.2.5	Další typy informací	34
5.3	Extrakce sémantické informace	35
5.3.1	Podoba sémantických entit	36
5.3.2	Extrakce sémantiky - objekty	36
5.3.3	Extrakce sémantiky - atributy	37
5.3.4	Extrakce sémantiky - vazby mezi objekty	38
5.4	Hodnotící algoritmus	38
5.4.1	Chybějící objekty	39
5.4.2	Chybějící atributy	42
5.4.3	Chybějící vazby mezi objekty	43

5.4.4	Atributy s chybnou hodnotou	44
5.4.5	Výstup hodnotícího algoritmu	46
6	Implementace a testování	47
6.1	Referenční popis obrázku	47
6.1.1	Validace referenčního popisu	51
6.1.2	Další příkazy pro práci s referenčním popisem	52
6.2	Sémantické parsování pomocí gramatik	54
6.2.1	Problémy s existující implementací a SRGS	55
6.2.2	Semantic Parsing Grammar Format (SPGF)	56
6.2.3	Vstupní a výstupní body SPGF gramatiky	59
6.2.4	Tagy v SPGF	61
6.2.5	Parsovací strategie a opakování v SPGF	62
6.3	Tvorba testovaného popisu	68
6.3.1	Algoritmus zpracování derivačních stromů	69
6.3.2	Číslování objektů	71
6.4	Hodnotící algoritmus a ztrátová tabulka	72
7	Vyhodnocení	76
8	Závěr	85

1 Úvod

V současném výzkumu aplikované lingvistiky a počítačového zpracování přirozeného jazyka zaujímá oblast zpracování textu, sémantická analýza a reprezentace znalostí význačnou pozici. Tyto problematiky jsou nedílnou součástí mnoha běžně používaných systémů, mezi které patří dnes již běžně dostupní hlasoví asistenti, automatické překlady a titulkování, či systémy pro extrakci informací v různých průmyslových nasazeních.

Cílem této práce je navrhnout a implementovat systém pro určení míry korespondence obrázku a jeho popisu v přirozené řeči, a to na sémantické úrovni. Motivací pro takovýto systém je možnost jeho využití pro vytvoření aplikace usnadňující vyšetření pro detekci kognitivních poruch s využitím metod strojového učení. Základní myšlenkou je využití předpokladu, že snížené kognitivní funkce se projeví mimo jiné i na schopnosti člověka detailně popsat komplexní scénu zobrazenou na předloženém obrázku.

Hlavní koncept prezentovaný v této práci spočívá v extrahování sémantické informace obsažené v přirozeném popisu, srovnání se vzorovým popisem a ohodnocení míry korespondence v podobě vektoru hodnot, který je možné následně použít jako vstup pro další zpracování, například jako vektor příznaků pro klasifikaci.

Během návrhu celkového konceptu systému, jeho jednotlivých částí a i následné implementace, byl kladen důraz na to, aby byly jednotlivé procesy „průhledné“, bylo možné detailně sledovat všechny kroky a v případě potřeby dělat detailní změny či vylepšení, pro zdokonalení výsledků v konkrétním nasazení. Dále bylo dbáno na modularitu systému, aby bylo možné v případě potřeby jednotlivé části nahradit za případné alternativy, nebo doplnit dodatečnými rozšířeními. Je vhodné poznamenat, že ačkoli bylo primární motivací již zmíněná podpora diagnostiky kognitivních poruch, byl systém navrhován obecně tak, aby jej bylo možné použít i pro případná další nasazení benefitující ze zde navržených metod.

V práci bude popsán návrh jednotlivých částí i celkové architektury, včetně zdůvodnění učiněných rozhodnutí. Následně bude popsána samotná realizace, společně s problémy a jejich řešením, které se v průběhu vypracování vyskytly.

2 Sémantika a zpracování přirozeného jazyka

Sémantika a zpracování přirozeného jazyka představují klíčové oblasti výzkumu v oboru počítačové lingvistiky a umělé inteligence. Kombinace těchto oblastí je základem pro mnoho metod zabývajících se analýzou vzájemné komunikace mezi lidmi, nebo způsobu předávání a uchovávání informací mezi člověkem a strojem.

2.1 Sémantika

Sémantika je vědní obor, který se zabývá zkoumáním významu. Je to obor související s lingvistikou a logikou, jelikož jeho předmětem je analýza významu frází, slov, vět a nebo i obecně jiných symbolů, pomocí kterých lze předávat informace. [1]

Sémantiku je možné dále rozdělit do subkategorií, podle konkrétního zaměření.

Lexikální sémantika se zaměřuje na význam slov, konkrétně se zabývá studiem buďto vnitřní významové struktury slova, nebo významové vazby, které se nacházejí v daném slovníku. [2]

Formální sémantika studuje význam výrazů v přirozeném jazyce pomocí matematických a logických nástrojů. Využívá exaktně definovaných pravidel a formalismů ke zkoumání přirozeného jazyka a souvislostí mezi významem a strukturou frází. Také se zabývá modelováním vazeb mezi jazykem, jeho strukturou, významem a informací, kterou popisuje. [3, 4]

Frázová sémantika (angl. phrasal semantics) se zabývá studiem významu vět a frází především z pohledu jejich skladby a kompozice. Hlavním předmětem je analýza toho, jak funguje skládání jednotlivých menších částí (slov či dílčích frází), jaké jsou významy jednotlivých elementů a jak jejich složení ovlivní celkový význam fráze [5].

Analýza jednotlivých částí se většinou zaměřuje na rozlišení subjektu, predikátu a argumentu. Subjekt většinou referuje na nějaký konkrétní objekt či entitu, predikát reprezentuje vlastnosti a události, které jsou se subjektem spojené a argument dodává doplňující informace, jako třeba konkrétní hodnotu dané vlastnosti. [6]

V kontextu výpočetní technologie a umělé inteligence existuje „výpočetní sémantika“ (angl. computational semantics). Výpočetní sémantika je disciplína, která kombinuje znalosti z formální sémantiky, výpočetní lingvistiky a automatického odvozování zna-

lostí. Jejím cílem je hledání a analýza metod, které by umožnily automatické zpracování a vytváření sémantických reprezentací pro výrazy z přirozeného jazyka a inferenční postupy. Dále se zabývá řešením neurčitostí, analýzou komplexních frází, vliv kontextu na význam a použití získaných znalostí pro automatické rozhodování. [7] [8]

2.2 Porozumění řeči

Porozumění řeči (angl. spoken language understanding, SLU) označuje strojové zpracování lidské přirozené řeči na sémantické úrovni. Jedná se tedy o extrakci významu z přirozené řeči pomocí automatických počítačových metod. Správné určení významu promluvy je často nezbytným krokem pro mnoho běžných úloh, například interpretace požadavků či příkazů, vyhledávání dokumentů nebo zodpovídání dotazů. [9]

Přirozený lidský jazyk je velmi variabilní a stejně sdělení je možné vyjádřit v mnoha podobách, které mohou být od sebe velmi rozdílné. Tato variabilita jazyka představuje hlavní problém, který metody porozumění řeči musí řešit. Existuje řada metod, jakými jsou z přirozené řeči extrahovány sémantické informace pro různé typy úloh. Následuje popis vybraných technik, které jsou široce používané v úlohách využívajících metody porozumění přirozené řeči, jako tokenizace, bag-of-words, latentní sémantická analýza, part-of-speech tagování, nebo sémantické sítě. [9]

Tato práce řeší danou problematiku expertním přístupem, konkrétně s využitím bezkontextových sémantických gramatik, jejichž principy budou rozebrány detailněji později v samostatné sekci 2.3.

1. Tokenizace

Tokenizace bývá jedním z prvních kroků při zpracování přirozeného jazyka, kdy je vstupní text rozdelen na jednotlivé diskrétní části, kterým jsou právě označovány jako *tokeny*. [9]

Častým přístupem k tokenizaci je rozdelení vstupního textu na jednotlivá slova, ale existují i jiné přístupy, které používají například ještě menší jednotky [10]. Zatímco pro anglický nebo český jazyk je rozdelení na slova relativně přímočarý proces, pro některé jazyky, které nepoužívají mezeru pro oddělení slov (např. Japonština), může být i pouhé rozdelení na slova obtížný úkol. [9]

Součástí tokenizace je často i proces *normalizace*, který upravuje vstupní data tak, aby odpovídaly nějakým požadavkům a bylo jednodušší s nimi pracovat dále. Typickým příkladem normalizace může být expanze zkratek. [9]

2. Bag-of-words

Přístup označený jako bag-of-words je jednoduchý způsob, jak analyzovat vstupní text na relativně povrchní úrovni. Tento způsob zpracování je také označovaný jako *vector space model*. [9]

Základní princip je vytvoření množiny slov ze vstupního textu a počítání jejich četnosti. Srovnání takto získaných vektorů hodnot je pak základem pro určení míry shody dvou nebo více zdrojů. Z tohoto přístupu plyne, že metoda zcela ignoruje jakékoli syntaktické informace a pořadí slov ve vstupním textu. [9]

Její součástí je také často vyřazení slov, která nepřispívají do sémantické informace (například předložky a spojky), tato vyřazená slova se označují jako *stop words*. Dalším častým rozšířením je *lemmatizace*, což je normalizace různých tvarů slov do jedné základní podoby. Výhodou přístupu bag-of-words je jeho jednoduchost a nezávislost na lingvistických znalostech. Naopak nevýhodou je nízká úroveň detailu a přesnosti. [9]

3. Latentní sémantická analýza

Metoda latentní sémantické analýzy je podobná předchozí metodě bag-of-words ve svém základním konceptu vytvoření množiny sledovaných prvků, spočítání jejich četnosti a následné práce se získaným vektorem hodnot.

Rozdíl oproti bag-of-words spočívá v tom, že namísto jednotlivých slov (nebo lemmat) jsou použité jejich významy a koncepty. Příkladem může být sledování vzorů, kde se často v textu vyskytují stejná slova blízko sebe. [9]

4. Part-of-speech (POS) tagování

Tento způsob zpracování přirozené řeči spočívá v přiřazení značek - *tagů* - k jednotlivým slovům. Tyto tagy přidávají dodatečnou informaci k daným slovům a tím tak obohacují vstupní text o sémantickou informaci. Často je POS tagování také využíváno jako doplňující proces k dalším, komplexnějším způsobům zpracování sémantického textu. [9]

Způsob realizace POS tagování může být založený na deterministických pravidlech a nebo může být proveden stochasticky. Pravidlový přístup je založen na použití často velmi rozsáhlé databáze pravidel, které jsou většinou tvořené ručně lidským expertem podle okolního kontextu. [9]

2.3 Sémantické gramatiky

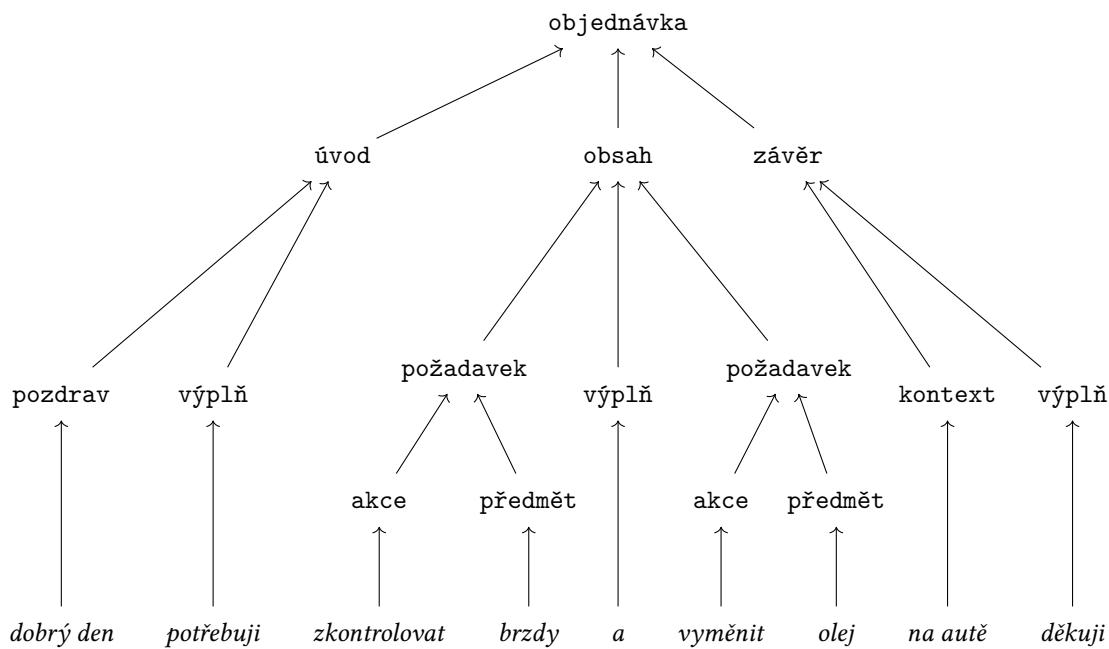
Další způsob, jakým lze zpracovat vstupní text, je založený na metodách formální syntaxe. Celkový vstupní text lze chápát jako jeden objekt, který lze rozdělit na menší části. Tyto menší části lze dále rekurzivně dělit až do okamžiku, než jsou jednotlivé části dále nedělitelné (atomické) - v ten moment se označují jako *primitiva*, která mají funkci *terminálních symbolů*. [11]

Tento proces rekurzivního dělení vytváří hierarchickou stromovou strukturu, kde kořenovým uzlem je původní vstupní text a koncové uzly (označované jako listy) jsou tvořené právě terminálními symboly. [11]

Na Obrázku 1 je zobrazený rozklad věty

„Dobrý den, potřebuji zkontolovat brzdy a vyměnit olej na autě, děkuji.“,

na kterém je možné vidět základní princip zpracování textu pomocí sémantických gramatik a výše zmíněný postupný rozklad na jednodušší jednotky.



Obrázek 1: Ukázka rozkladu věty

Stromová struktura umožňuje výhodně aplikovat metody pro analýzu syntaxe formálních jazyků. Množina těchto terminální symbolů se označuje jako *abeceda formálního jazyka*. Formálním jazykem je možné popsat množinu předmětů, které jsou tvořené složením terminálních symbolů do nějaké struktury (posloupnosti, řetězce). Tyto řetězce terminálních symbolů se označují jako *slova formálního jazyka* a pravidla, která popisují, jak je možné jednotlivé terminální symboly skládat do slov, jsou označována jako *substituční* nebo *produkční pravidla*. [11]

Pro lepší možnosti vyjádření jsou definované ještě takzvané *neterminální symboly*, které zastávají funkci pomocných částí popisu předmětů. Substituční pravidla mají tvar

$$\eta \rightarrow \omega \quad \text{nebo} \quad \eta \xrightarrow{P} \omega,$$

kde η a ω jsou řetězce terminálních a neterminálních symbolů a P označuje pravděpodobnost (pro stochastická pravidla). Substituční pravidla lze definovat i rekurzivně, což výrazně rozšiřuje vyjadřovací schopnosti a umožňuje popsat i nekonečné množiny předmětů pomocí konečného (a relativně malého) počtu pravidel. [11]

Tato pravidla, společně s množinou terminálních a neterminálních symbolů tvoří dohromady takzvanou *gramatiku*. Formálně lze gramatiku G popsat čtevějící

$$G = (V_N, V_T, \rho, S),$$

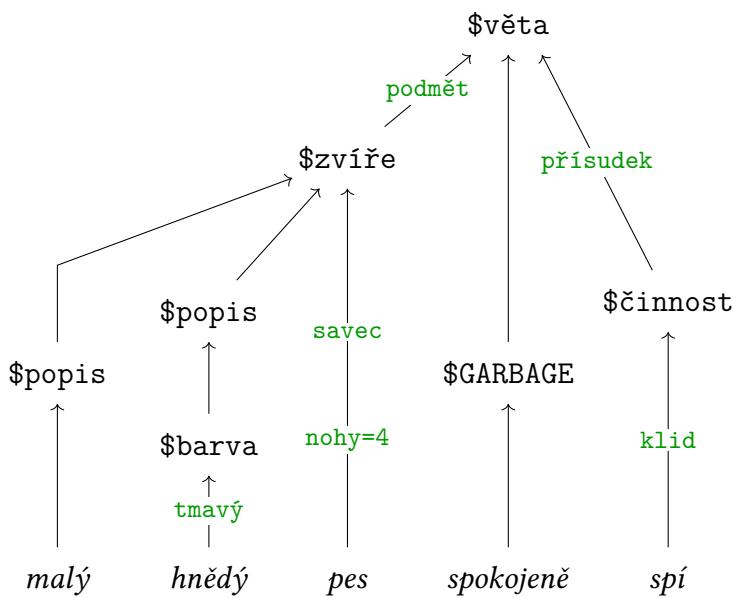
kde V_N je množina neterminálních symbolů, V_T je množina terminálních symbolů, ρ označuje množinu substitučních pravidel a $S \in V_N$ je počáteční symbol gramatiky. [11]

Konkrétní příklad ukázkové gramatiky je na Výpisu 1 a k ní příslušné dva derivační stromy, jeden pro vstupní text „malý hnědý pes spokojeně spí“ (Obrázek 2) a druhý pro vstupní text „bílý medvěd rychle a zdatně plave“ (Obrázek 3).

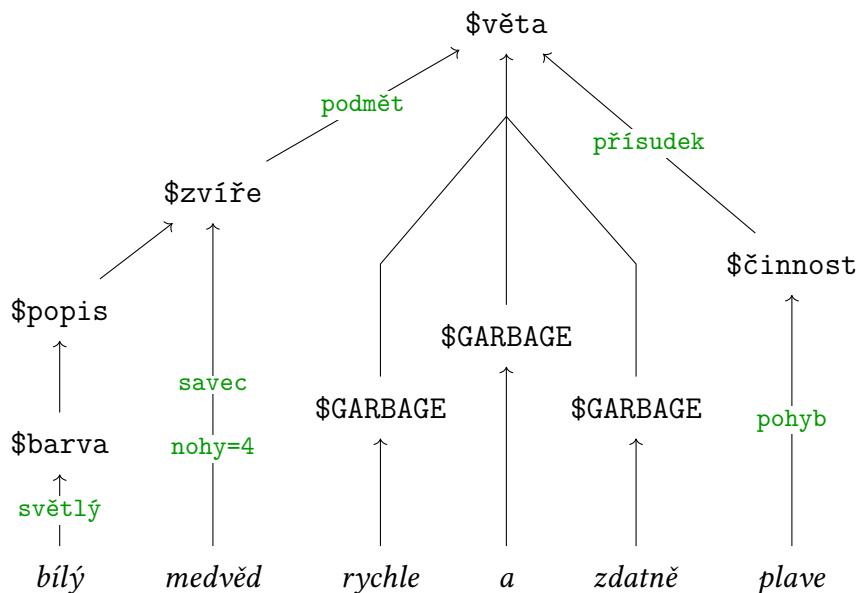
Gramatika na Výpisu 1 je ve formátu SPGF, který je představen v této práci. V gramatice a derivačních stromech na Obrázkách 2 a 3 je možné vidět použití některých dalších konceptů, jako jsou *tagy* (zeleně), které nemají vliv na proces vytváření derivačního stromu, ale přidávají do kontextu dodatečné sémantické informace.

```
1 public $věta = $zvíře {podmět} $GARBAGE<L:*> $činnost {příslušek};  
2  
3 $zvíře = $popis<0-3> ((pes | medvěd) {nohy=4} |  
4     delfín {nohy=0}) {savec} | kapr {nohy=0} {ryba};  
5  
6 $popis = $barva | velký | malý | chlupatý;  
7  
8 $barva = (hnědý | černý) {tmavý} | bílý {světlý};  
9  
10 $činnost = spí {klid} | žere | plave {pohyb};
```

Výpis 1: Ukázka gramatiky (ve formátu SPGF)



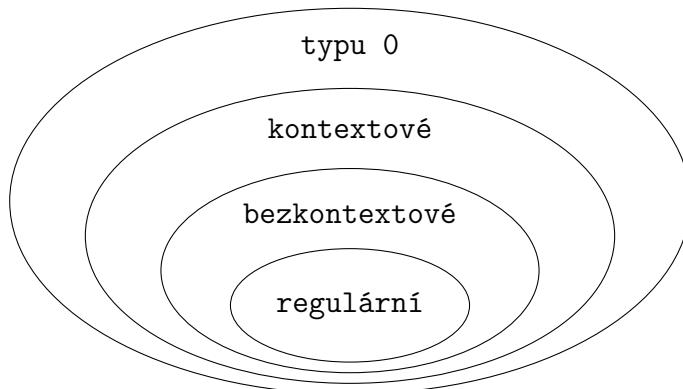
Obrázek 2: Ukázka derivačního stromu pro vstup „malý hnědý pes spokojeně spí“ při použití gramatiky z Výpisu 1



Obrázek 3: Ukázka derivačního stromu pro vstup „bílý medvěd rychle a zdatně plave“ při použití gramatiky z Výpisu 1

Gramatiky lze rozdělit na deterministické a stochastické, podle toho, zda jsou její pravidla definována deterministicky, nebo s využitím pravděpodobnosti [11]. Tato práce se zabývá pouze deterministickým přístupem.

Takto definované gramatiky lze rozdělit do čtyř základních kategorií, podle typu jejich pravidel. Nejobecnější a nejsložitější jsou *gramatiky typu 0*, poté jsou *kontextové gramatiky*, dále *bezkontextové gramatiky* a nejjednodušším typem jsou *regulární gramatiky*. Schématické znázornění této hierarchie je na Obrázku 4.



Obrázek 4: Schématické znázornění hierarchie formálních gramatik

Se složitostí gramatiky je také spojena její vyjadřovací schopnost, tedy pomocí složitějších gramatik je možné zachytit složitější struktury. Nejjednodušší regulární gramatiky nejsou například schopné zachytit vztah mezi částmi, které nejsou bezprostředně vedle sebe, což je pro text s přirozeným jazykem velmi silná limitace. Pro porozumění řeči se nejčastěji používají bezkontextové gramatiky. Kontextové gramatiky a gramatiky typu 0 by umožňovaly zachytit ještě další složitější vazby mezi částmi textu, ale jejich úroveň komplexnosti je natolik vysoká, že strojové zpracování je extrémně obtížné a výpočetně náročné. [11]

Výstupem zpracování pomocí bezkontextových gramatik je tedy strukturální informace, ze které lze zjistit, zda daný vstupní text odpovídá dané gramatice a pokud ano, jak vypadá jeho rozklad. Takto získaná stromová struktura jednotlivých rozpoznaných symbolů představuje sémantickou informaci, kterou je možné dále využít a zpracovat.

3 Reprezentace znalostí

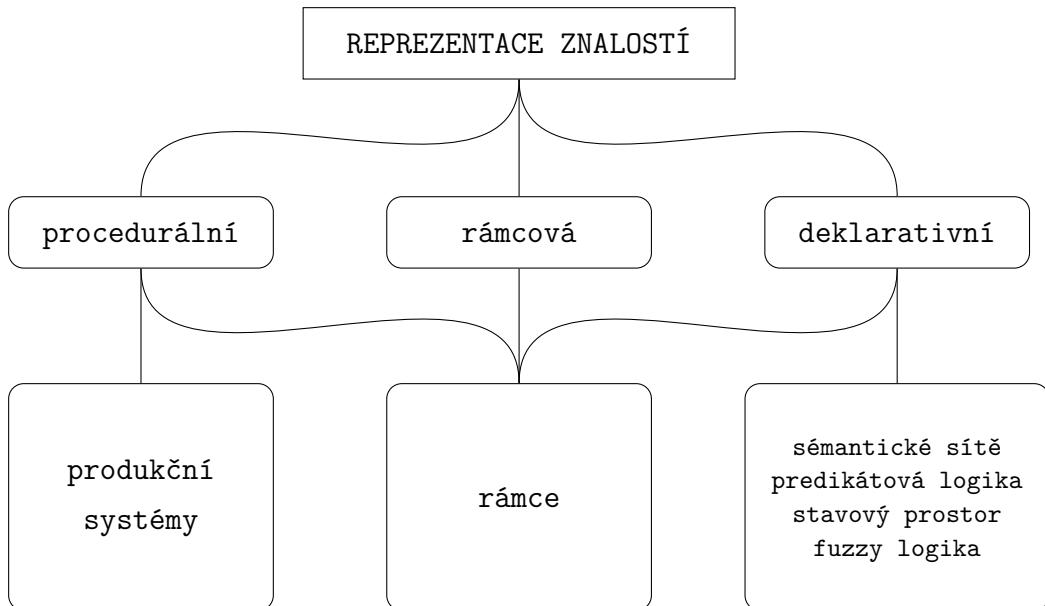
Pro řešení prakticky každého reálného problému je potřeba mít k dispozici nějaké znalosti minimálně z oblasti, ve které se problém nachází. Při řešení problémů pomocí automatický počítačových metod a metod umělé inteligence je pak nutné ještě navíc mít nějaký způsob, jak tyto znalosti zapsat a uchovat, jak s nimi manipulovat a provádět na nich operace a algoritmy.

Znalost je možné chápat jako nějakou informaci, která popisuje určitý fakt, nebo „pravdu o světě“. Jejich vyjádření v nějakém konkrétní podobě, neboli *formalismu*, je potom označováno jako *reprezentace znalostí*. Tato formální vyjádření by měla umožňovat provádět se znalostmi potřebné operace k vyřešení daného problému. [12]

Nové znalosti je možné získat v základu dvěma způsoby. Prvním z nich je pozorování objektů a jevů, což je označováno jako přímé získávání znalostí. Druhou metodou, která označována jako nepřímá, je odvození nových znalostí z těch, které jsou již známé. [12]

3.1 Dělení reprezentace znalostí

Podle učebního textu [12] je možné metody reprezentace znalostí rozdělit do několika kategorií. Schématické znázornění tohoto dělení je na Obrázku 5.



Obrázek 5: Schéma dělení metod reprezentace znalostí [12]

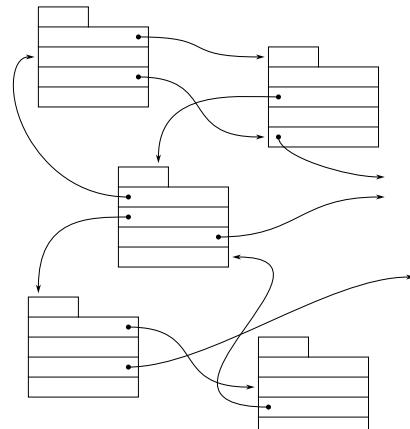
Deklarativní přístup je založený na vyjádření různých znalostí ve formě *poznatků*, množinou konkrétních a navzájem nezávislých faktů. S těmito fakty je pak možné manipulovat pomocí takzvaných *procedur*, někdy označovaných jako tvrzení. Mezi deklarativní metody se také řadí různé přístupy založené na matematice a logice, na algebře, relačních strukturách. Mezi deklarativní metody se řadí *sémantické sítě*, které znalosti reprezentují jako entity či objekty relací, které mezi těmito objekty popisují jejich vztahy. Dalším deklarativním typem reprezentace znalostí jsou *stavové prostory*, které popisují řešený problém jako množinu možných stavů, ve kterých se daný problém může nacházet. Mezi další formy reprezentace znalostí deklarativní povahy se řadí různé druhy logiky, jako jsou *predikátová logika* a *fuzzy logika*. [12]

Procedurální metody reprezentace znalostí zahrnují produkční systémy. To jsou systémy navržené tak, že pomocí předem dané množiny pravidel, které se označují jako produkční pravidla, se snaží dosáhnout nějakého stanoveného cíle a vyřešení určitého problému. [12]

Poslední skupina, označená jako rámcová metoda reprezentace znalostí, představuje kombinaci obou předchozích přístupů. Rámce jsou struktury, které popisují různé objekty podle předem dané formy a v průběhu řešení problému se doplňují konkrétní hodnoty k daným vlastnostem. [12]

Schématická ukázka sémantické sítě využívající rámce je na Obrázku 6.

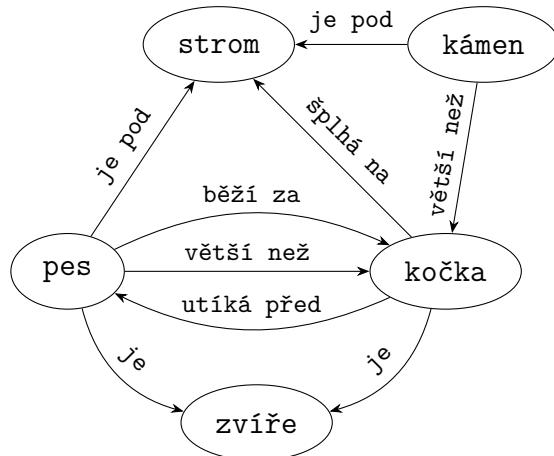
(FIDO)	
Slot	Value
INSTANCE-OF:	(DOG, PET)
Name:	„Fido“
Color:	(BROWN)
Father:	(BOWSER)
Mother:	(WEENIE)
Owner:	(MR.-SMITH)
Has-as-ears:	(LEFT-EAR, RIGHT-EAR)
Number-of-ears:	(2)



Obrázek 6: Ukázka rámců pro individuálního psa (vlevo) a schématické znázornění propojení více rámců (vpravo), převzato z [13]

3.2 Sémantické sítě

Sémantické sítě reprezentují znalosti pomocí grafových struktur složených z uzlů a hran. Uzly představují koncepty či objekty a hrany reprezentují relace mezi těmito jednotkami. Ukázka jednoduché sémantické sítě je na Obrázku 7.



Obrázek 7: Ukázka jednoduché sémantické sítě

Sémantické sítě umožňují použít různé metody z teorie grafů, logiky a matematiky pro práce s uloženými znalostmi, jejich analýzu a případně i odpovídání otázek či indukci nových znalostí. Sémantické sítě se používají v oblastech umělé inteligence pro porozumění jazyku, deduktivní databáze, vyhledávání dokumentů v databázi, expertní systémy a další. [13]

Sémantické sítě je možné realizovat také s využitím rámců (angl. frames) pro jednotlivé uzly. Rámec je chápán jako abstraktní popis nějakého objektu, která má pojmenované *sloty*, které popisují nějaké explicitní vlastnosti. Tyto rámce mohou být navzájem propojené a odkazovat se na sebe navzájem, čímž tvoří strukturu orientovaného grafu. Některé implementace také poskytují možnost dynamického zjišťování hodnot pro jednotlivé sloty pomocí procedurálních metod, což umožňuje zachytit komplexní chování a inferenci. [13]

Podle [13] je možné rozdělit vazby (hrany) v sémantických sítích do několika skupin. V relačních grafech, kde hrany popisují relace mezi objekty, tak se označují jako *deskriptivní* nebo *asertivní*. Oproti *strukturálním*, *definičním* nebo *inferenčním* vazbám, které

slouží k zachycení hierarchické struktury objektů, popisují vyšší abstrakce, kategorie a dědičnost. [13]

Aby bylo možné pomocí sémantických sítí zachytit i principy logiky, jako jsou:

- negace (např. „kámen NENÍ zvíře“),
- disjunkce (např. „osoba narozená v Británii NEBO potomek Brita“),
- existenční kvantifikátory (např. „VŠECHNY cesty vedou do Říma“),

a další komplexnější vztahy, je možné sémantické sítě rozšířit o další typy vazeb (např. IS-NOT-A) i celé strukturální koncepty, jako je rozdelení sítě do *prostorů*, které se mohou překrývat jako Vennovy diagramy, což umožňuje vyjádřit složitější logické operace. Dalším typem rozšíření, který se vyskytuje v *koncepcuálních grafech*, je použití *vnořených kontextů*, jejichž principem je využití metod logiky pro vyjádření složitějších operací skrze zapouzdření (angl. encapsulation) primitivních operací. [13]

3.3 Ontologie

Samotná forma reprezentace znalostí a způsob práce s nimi je pouze částí problematiky sémantických sítí, druhou otázkou, kterou je potřeba řešit, je samotný obsah neboli „co reprezentovat“. Tímto problémem se zabývá *ontologie*, která zahrnuje formální označení, strukturu a definice jednotlivých kategorií, vlastností, relací a samotných dat, které je potřeba reprezentovat.

Svátek [14] uvádí, že cílem ontologie je podpora porozumění mezi lidmi, zejména pak experty v určité oblasti, podpora komunikace mezi počítačovými systémy a usnadnění návrhu aplikací se znalostním zaměřením.

Rozdelení ontologií do typů či kategorií lze podle [14] dělat podle různých kritérií:

- *Dělení podle historického vývoje:*

V důsledku historického vývoje lze ontologie rozdělit na *terminologické* či *lexikální*, které se zaměřují na textové zdroje a lze je považovat za pokročilé slovníky, poté *informační ontologie*, které jsou spojené s databázovými systémy a schématy a *znalostní ontologie*, které jsou spojené s oblastí umělé inteligence a reprezentace znalostí.

- *Dělení podle míry formalizace:*

Existují typy ontologií, které nejsou celé striktně formalizované a například používají přirozený jazyk pro vysvětlení pojmu. Pak ale také existují striktně formalizované ontologie, které využívají formální jazyky. Podle [14] však většina formálních ontologií obsahuje do nějaké míry i neformální ontologie.

- *Dělení podle obsahu:*

Hlavní typy ontologií podle jejich obsahu jsou doménové, generické, úlohové a aplikační. Svátek [14] uvádí, že tento způsob dělení je „tradiční“, avšak s různými variantami, podle konkrétního autora.

Struktura ontologií obecně sestává ze *tříd*, *individuí*, *relací*, *pravidel* a *hodnot*. Třídy odpovídají množinám objektům, někdy jsou také označované jako *kategorie* nebo *koncepty*. Individua jsou v ontologiích označení pro konkrétní objekty, většinou reprezentující nějaké objekty v reálném světě. Mezi objekty je možné vytvořit *relace*, které popisují nějaké vazby mezi objekty, či mohou popisovat jejich vlastnosti. Pravidla, někdy označovaná jako *axiomy*, slouží v ontologických jazycích pro vyjádření logických formulí, jako je ekvivalence tříd, nebo jejich disjunktnost. [14]

Existuje řada různých přístupů a realizací [13], příkladem může být projekt *Cyc* [15], jehož cílem je poskytnout ontologii pro reálný svět. Jiným příkladem může být J. L. Jolleyův *Holotheme* [16], který se také snaží klasifikovat cokoli, ovšem v rámci jednoho schématu založeného na strukturální komplexitě. Dalším příkladem je *Wordtree* [17], který představuje pokus o definici významů slov pomocí kombinování primitivních konceptů a významů.

Jedním z ontologických jazyků je *RDF Schema* [18], které patří k *Resource Description Framework (RDF)* [19] standardu pod skupinou W3C¹. RDF je datový model, který byl navržený pro Web, se zaměřením na kompatibilitu dat a schopnost jejich sloučení. Je založený na syntaxi značkovacího jazyka XML a hojně využívá URL adresy.

¹<https://www.w3.org/>

4 Korespondence obrazu a jeho slovního popisu

Korespondence mezi obrazem a jeho popisem je problematika řešená v řadě reálných praktických aplikací. Tato kapitola popisuje základní principy práce s obrazem a spojitost s jeho popisem, a některé vybrané příbuzné problémy. Na závěr jsou v této kapitole zmíněny rozdíly v přístupu, který byl zvolen pro tuto práci.

4.1 Práce s obrazovými daty

Popis obrázku je v kontextu výpočetní technologie a umělé inteligence rozšířené téma, které nachází uplatnění v mnoha praktických aplikacích. Jedná se o jednu z mnoha problematik, které se obecně v úlohách počítačového zpracování obrazových dat řeší.

Úlohy, které se týkající zpracování obrazu, lze podle učebního textu [20] rozdělit do čtyř základních typů:

1. Klasifikace

Jedná se o úlohu přiřazování obrázku k nějaké třídě. Množina tříd, do kterých jsou obrázky přiřazované, jsou často známé předem a může jich být různá množství, od několika jednotek pro úzce specializované úlohy, například v medicíně, nebo pro obecnější úlohy u tisíce.

V úlohách klasifikace není brán ohled na prostorové informace, celý obrázek je chápán jako jeden celek, jemuž je přiřazena příslušnost do vybrané třídy. Typickým příkladem může být rozdelení obrázku do kategorií *kočka* nebo *pes* podle toho, jaké zvíře je na obrázku znázorněno.

2. Sémantická segmentace

Sémantická segmentace obrázků spočívá v přiřazení nějaké ze tříd každému pixelu. Tím dojde k rozdelení obrázku do disjunktních oblastí, které jsou (sémanticky) označené danou třídou.

V úlohách tohoto typu se již patrně zohledňuje prostorová informace jednotlivých částí, avšak jednotlivé instance různých objektů pod společnou třídou rozlišené nejsou. Příkladem typické segmentace může být určení toho, které pixely odpovídají obloze, které zemi, a které třeba člověku.

3. Detekce objektů

Principem detekce objektů je v obraze určit, zda se v něm nachází jeden nebo více hledaných objektů daného typu. Vstupem bývá obrázek a požadavek na detekci nějakého typu objektu a výstupem může být obrázek s označením objektů pomocí ohraničujících rámců (angl. bounding-box) s přiřazeným označením třídy.

Tento typ úlohy může být vhodný například pro sledování (tracking) lidí či zvířat, pro následnou klasifikaci nebo jinou analýzu.

4. Segmentace instancí

Tento typ úlohy představuje kombinaci předchozích typů úloh. Princip spočívá v tom, že v obrázku jsou označené pixely odpovídající nějakému hledanému objektu stejně jako při sémantické segmentaci, ovšem s tím rozdílem, že není nutné takto označit každý pixel v obrázku a naopak je dbáno na rozlišení jednotlivých instancí daných objektů.

Výsledkem je tedy informace o tom, které pixely odpovídají jakému konkrétnímu objektu. Například na obrázku, kde by bylo zobrazeno více psů a koček, by bylo cílem označit kde přesně se nachází který pes a která kočka.

Všechny výše zmíněné typy úloh, respektive jejich výstupy, lze považovat za popis obrázku v tom smyslu, že výstupem je jistá forma informace o tom, co daný obraz zachycuje. Nejedná se však přímo o hluboké či komplexní sémantické znalosti, spíše o „hrubé“ zařazení do nějaké třídy nebo forma výpisu zobrazených objektů.

V současnosti jsou obrazová data zpracovávána převážně pomocí velkých neuronových sítí. Díky pokrokům ve výpočetní síle a dostupnosti datových sad jsou tyto modely stále sofistikovanější a schopné řešit složitější úkoly. Součástí tohoto pokroku je také schopnost modelů analyzovat obrazová data a v kombinaci se zpracováním přirozené řeči i odpovídat na otázky týkající se komplexnější sémantiky. [přidat zdroj, třeba na ten nový GPT-4o, kde je ukázka? Tady nemám přímo nic faktického, tak by to mohlo stačit? Tenhle odstavec je hlavně pro nějaké „napojení“ popisu obrázků na sémantiku]

4.2 Sémantický popis obrazu

Jednou z oblastí, kde je využívána ontologie a sémantický popis obrázku, jsou systémy pro vyhledávání obrázků (angl. image retrieval).

Například Sarwar et al. [21] prezentuje metodu zabývající se zdokonalením přesnosti vyhledávání obrázků, na základě vstupního dotazu v přirozeném jazyce. Navrhuje metodu, která má tohoto cíle dosáhnout za využití doménové ontologie, lokální sémantické informace v podobě obrazových deskriptorů a jejich kvalitativní prostorové relace. Navržená architektura v první fázi předzpracuje poskytnuté obrázky tak, že je rozdělí na jednotlivé sektory, kterým jsou přiřazeny značky reprezentující obecné, předem určené, koncepty, jako jsou například „tráva“, „nebe“ nebo „písek“. Další část architektury, kterou Sarwar et al. [21] prezentuje, slouží k transformování uživatelského vstupu (dotazu) do několika předem definovaných forem využívajících RDF triplety. Poslední fázi zpracování je pak využití doménové ontologie pro určení sémantické podobnosti dotazu a dostupných anotovaných obrázků, ze kterého plynne výběr výstupního obrázku odpovídající vstupnímu dotazu.

Další příbuznou problematikou je automatická tvorba sémantických popisků k obrázkům, kterou ve své práci prezentuje Meiyu et al. [22]. Využívá k tomu SIFT [23] deskriptorů, kvůli jejich dobrému poměru mezi výpočetní náročností, informačním objemem a spolehlivostí. Následně je v práci [22] prezentováno, že díky internetu a z něj plynoucímu množství koexistujících obrázků a jejich textových popisů, byl sestaven model, který je schopen mapovat extrahované příznakové vektory (angl. feature vector) ze SIFT deskriptorů na textové anotace, ze kterých jsou získané sémantické popisy. Výsledky, které Meiyu et al. [22] prezentuje, pak zobrazují obrázek, k němuž jsou automaticky doplněné anotace objektů a konceptů, které se na něm vyskytují.

Doposud popisované problémy a jejich řešení využívají převážně neuronových sítí a statistických metod, jejichž základním předpokladem je dostatečný počet a kvalita trénovacích dat.

Vzhledem k motivaci této práce, kterou je hodnocení kvality popisu daného obrázku na sémantické úrovni pro usnadnění detekce kognitivních poruch, byl zvolen expertní přístup se zaměřením na kvalitativní analýzu výsledků, průhlednost celého procesu s možností detailních úprav a reprodukovatelnost.

5 Návrh a architektura systému

V této kapitole je popsán základní architektura systému prezentovaného v této práci. Budou popsány hlavní koncepty, algoritmy a přístupy použité při implementaci. Součástí popisu je i odůvodnění volby konkrétních postupů a účel jednotlivých částí systému.

5.1 Obecná architektura systému

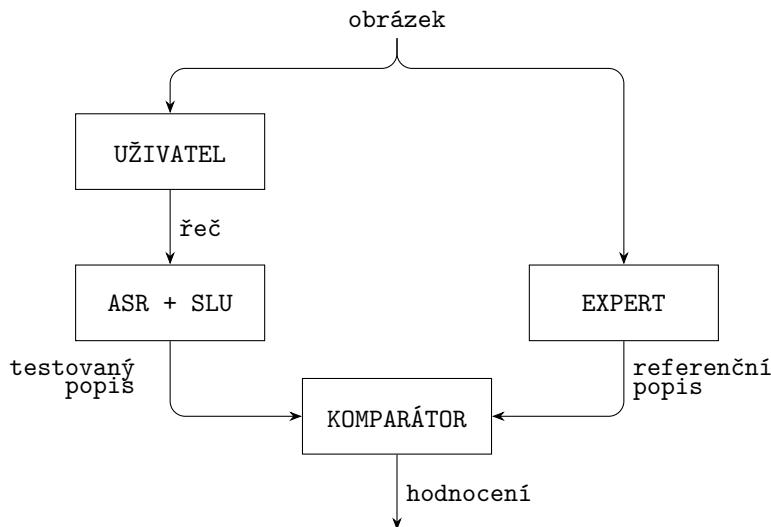
Obecná architektura celého systému vychází z jeho požadované funkčnosti, kterou je porovnání obrázku s jeho popisem v přirozené řeči, a to na sémantické úrovni. Z toho pak plyne, že celý systém se ve své podstatě skládá ze tří základních částí:

1. referenční (vzorový) popis daného obrázku
2. sub-systém pro zpracování přirozené řeči a tvorbu testovaného popisu
3. porovnání referenčního a testovaného popisu

Jednotlivé části spolu vzájemně fungují následujícím způsobem: Uživateli je prezentován obrázek a jeho úkolem je popsat, co na obrázku vidí. Získaný popis v přirozené řeči je převeden na text pomocí systému pro rozpoznání řeči (angl. automatic speech recognition, ASR). Z tohoto přepisu je extrahována sémantická informace, ze které je vytvořen testovaný popis. Testovaný popis je porovnaný s referenčním (vzorovým) popisem daného obrázku. Výsledek tohoto porovnání lze pak považovat za finální výstup, ale také je možné jej použít jako vstup pro další zpracování (např. vektor příznaků pro klasifikátor). Schématické znázornění je na Obrázku 8.

Tato práce řeší problematiku srovnání obrázku s jeho popisem v přirozeném jazyce na sémantické úrovni expertním přístupem. Od toho se poté odvíjí konkrétní algoritmy, formáty a postupy navržené a implementované v této práci, které jsou podrobněji popsány v pozdějších kapitolách. Je ale vhodné zmínit, že během návrhu bylo dbáno na to, aby bylo možné pro reálná nasazení některé implementace v případě potřeby zaměnit nebo upravit, aby lépe vyhovovaly specifickým požadavkům pro dané použití.

Jak je možné ze schématu na Obrázku 8 vidět, lidský expert je potřebný pro vytvoření referenčního popisu obrázku (a také pro další vstupní data pro komparátor, popsáno později). Všechna tato data, která jsou závislá na lidském expertovi, je ovšem možné



Obrázek 8: Schéma obecné architektury systému

vytvořit pouze jednou v rámci přípravy a poté je lze již bez nutnosti přítomnosti experta opakovaně používat.

5.2 Reprezentace znalostí

Pro porovnání obrázku s jeho popisem v přirozené řeči bylo nutné zvolit či navrhnout takovou formu reprezentace znalostí, která by umožnila zachytit sémantiku z obou zdrojů. Jak již bylo výše zmíněno, zvolen byl expertní přístup, a to v tomto případě znamená, že referenční popis obrázku je vytvořen lidským expertem, což klade další omezení na formát reprezentace znalostí.

Expertní přístup umožňuje velmi přesnou kontrolu nad obsahem a kvalitou vstupních dat, na rozdíl od automaticky získaných dat pomocí statistických metod. To je výhodné hlavně z toho důvodu, že kvalita referenčního popisu a dalších vstupních dat přímo ovlivňuje kvalitu výstupu. Další výhodou je průhlednost celého procesu, na rozdíl například od neuronových sítí, jejichž funkce je často označována jako black-box. To je také spojeno s možností dělat v případě potřeby drobné změny a opravy chyb v referenčním popise.

Při návrhu bylo tedy potřeba brát v úvahu následující požadavky a najít nějaký formát, který by představoval vhodný kompromis mezi nimi.

- **Čitelnost člověkem:**

Aby byl lidský expert schopen vytvořit, číst a případně upravit referenční popisy, je nutné, aby byl schopen porozumět formě a zápisu uložených dat. Toto omezení tedy upřednostňuje textové formáty a prakticky vyřazuje binární data.

Výjimku by mohl tvořit nějaký binární formát s přidruženým editorem, kde by člověk mohl v grafickém prostředí prohlížet a manipulovat data, ale takový případ je nad rámec této práce.

- **Kompaktnost a struktura dat:**

Dalším důležitým aspektem je struktura a kompaktnost dat. Pomocí počítače je poměrně snadné v krátkém čase zpracovat velké množství jednoduchých datových záznamů, nicméně člověk se bude lépe orientovat v nějakém kompaktnějším popisu, který ačkoli může být složitější ve své struktuře, tak bude pro člověka lépe názorný a uchopitelný.

- **Univerzálnost formátu:**

Podstatnou vlastností pro hledanou reprezentaci znalostí je její schopnost zachytit popis různých obrázků. Navržený formát tedy musí být dostatečně univerzální, aby pomocí něj šlo popsat co nejsírovší spektrum informací, od jednoduchých obrázků zobrazujících například jeden statický objekt, přes složitější obrázky zobrazující více objektů, až po dynamické komplexní scény zobrazující mnoho objektů, činnosti a vazby mezi nimi.

- **Počítačová zpracovatelnost:**

V neposlední řadě je také potřeba dbát na to, aby navržený formát bylo možné co nejsnadněji zpracovat programově, na počítači. Dynamické formáty s volnou strukturou bývají složitější na strojové zpracování, než fixní formáty s přesně definovanou podobou.

S ohledem na tyto body byla navržena reprezentace znalostí založená na sémantických sítích, která definuje 4 základní *aspekty popisu*: objekty, jejich hierarchii, statické atributy a dynamické vazby. Detailnější popis těchto jednotlivých aspektů je v následujících částech, konkrétní technická implementace je pak popsána v sekci 6.

5.2.1 Objekty ve scéně

Cokoli, co lze v obrázku ohraničit rámečkem (angl. bounding-box) a při separaci od zbytku scény neztratí nebo nezmění svůj význam, lze považovat za *objekt*.

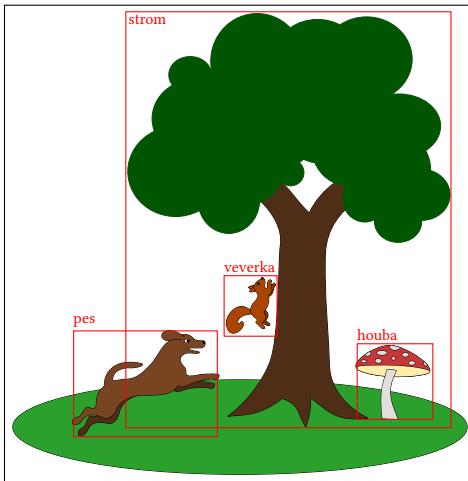
Jako objekt v obrázku lze tedy označit zobrazené fyzické předměty, postavy, zvířata, ale také nehmotné pojmy jako „nebe“, místa, lokace či místnosti (např. „kuchyň“ nebo „louka“) a části jiných objektů (např. „obličeji“ jsou součást hlavy nebo celého člověka).

Tato definice objektu byla záměrně navržena velmi obecně, aby byl definovaný formát univerzální a šel použít i pro popis velmi odlišných obrázků s různými účely. Potenciální nevýhodou, která plyne z univerzálnosti formátu, může být v některých situacích problém nejednoznačnosti.

V jednom obrázku lze definovat různé množiny objektů, podle toho, jak moc detailní popis expert vytvoří. Například pokud by byl na obrázku člověk, lze jej popsat jedním objektem jako „člověk“ nebo „osoba“, ale také bylo definovat ještě mnoho dalších objektů, například pro jednotlivé části těla nebo oblečení.

Kromě různých úrovní detailů lze také na problematiku nejednoznačnosti narazit v situaci, kdy je pro nějaký (dostatečně komplexní) obrázek možné sestavit různé množiny objektů podle toho, pro jaké potřeby je zrovna obrázek a referenční popis používán. Pro aplikaci, kde je podstatné zachycení živých objektů, může být množina objektů v referenčním popise tvořena lidmi či zvířaty. Pro jiné použití pak ale může být podstatné zachytit prostředí a neživé předměty, takže množina objektů by byla tvořena částmi prostředí (např. stromy, křoví, voda, skály), budovami nebo obecnými předměty.

Příkladem obrázku s několika vyznačenými objekty je ilustrace na Obrázku 9.

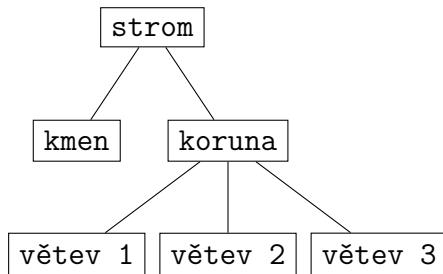


Obrázek 9: Ilustrace s vyznačenými objekty

5.2.2 Hierarchie objektů

Kromě množiny samotných objektů lze v obrázku také definovat jejich hierarchii. To přirozeně plyne z výše zmíněné definice *objektu*, která umožňuje specifikovat část existujícího objektu jako další samostatné objekty.

Příkladem takového popisu může být například situace, kdy je na obrázku strom. Strom je možné rozdělit na korunu a kmen, korunu pak je možné dále dělit na větve. Schématicky lze tento popis znázornit jako stromovou strukturu, viz Obrázek 10.



Obrázek 10: Schématické znázornění hierarchie objektů

Z pohledu daného objektu jsou „vyšší“ (obecnější) objekty označované jako *rodičovské objekty* (nebo jen *rodiče*) a „nižší“ (konkrétnější) objekty pak jako *potomci*.

Možnost definovat hierarchii objektů nabízí mimo jiné také způsob, jak vytvořit skupiny objektů, které k sobě nějakým způsobem patří. Například strom může být součástí lesa,

kráva může být součástí stáda nebo postava na hřišti může být součástí fotbalového týmu. Tato příslušnost objektu nějaké skupině je dalším typem sémantické informace, kterou umožňuje navržený formát zachytit bez nutnosti definovat další specializované struktury.

Mohou však nastat situace, kdy je potřeba jeden objekt zařadit do několika různých skupin. Z tohoto důvodu bylo rozhodnuto, že každý objekt může mít libovolné množství rodičů a libovolné množství potomků.

5.2.3 Atributy objektů

Vedle pouhého výčtu samotných objektů ve scéně je dále přirozeným požadavkem, aby byla reprezentace znalostí schopna zachytit i jejich vlastnosti. K tomu slouží třetí aspekt popisu - *atribut*.

Atributem je možné popsat jakoukoli informaci o objektu, která není závislá na jiném objektu. Jinými slovy, pokud bychom objekt izolovali od zbytku scény (obrázku), tak všechny vlastnosti, které se tím nezmění nebo nezaniknou, lze popsat pomocí atributů. Typickým příkladem je barva nebo tvar objektu či jeho části.

Atribut se skládá ze zvoleného *názvu* a přiřazené *hodnoty*, kdy konkrétní názvy a hodnoty jsou volbou experta, který tvoří referenční popis. Název atributu označuje jakou vlastnost daný atribut popisuje a přiřazená hodnota pak udává, jaké konkrétní hodnoty nabývá. Každý objekt může mít libovolné množství těchto atributů.

Názvy atributů pod jedním objektem nemusí být unikátní, lze specifikovat více stejnojmenných atributů s různými hodnotami. Typickým příkladem může být vícebarevný objekt, kde atribut s názvem „barva“ bude vícekrát, pro různé konkrétní barvy. Příklad jednoho takového popisu je na Obrázku 11, kde název objektu je „tričko“ a definované jsou tři atributy: dvě barvy a vzor.



tričko
barva: modrá
barva: zelená
vzor: pruhy

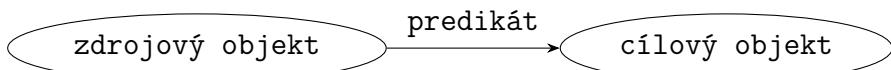
Obrázek 11: Příklad použití atributů pro popis objektu

Pro složitější objekty se opět naskytá problém toho, jaké vlastnosti vybrat a zapsat do referenčního popisu, a které je možné naopak zanedbat. Tato otázka je podobná problematice nejednoznačnosti, zmíněné v části 5.2.1 definující objekty. A i zde platí to, že popis byl záměrně vytvořen tak, aby byl univerzální a volba atributů do referenčního popisu závisí na expertovi a konkrétní aplikaci.

5.2.4 Vazby mezi objekty

Dalším typem informace, kterou by měl být referenční popis obrázku schopen zachytit, jsou vazby a vztahy mezi různými objekty. Může se jednat například o činnosti, které se týkají dvou objektů, nebo popis relativních vlastností, jako je velikost či pozice.

Každý takový záznam je označen jako *triplet* a skládá se ze *zdrojového objektu*, *cílového objektu* a *predikátu*, který popisuje danou vazbu nebo vztah. Tato struktura je inspirována běžným zápisem sémantických sítí a RDF standardem [19]. Obecné schéma jednoho tripletu je na Obrázku 12.

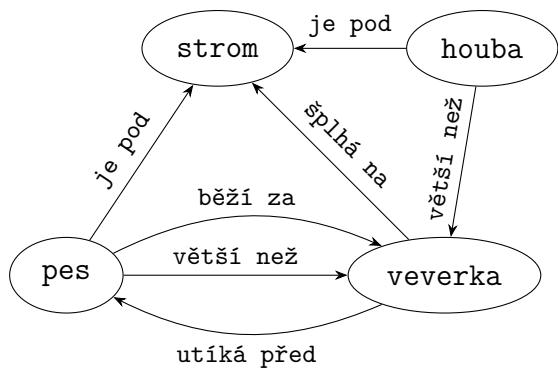
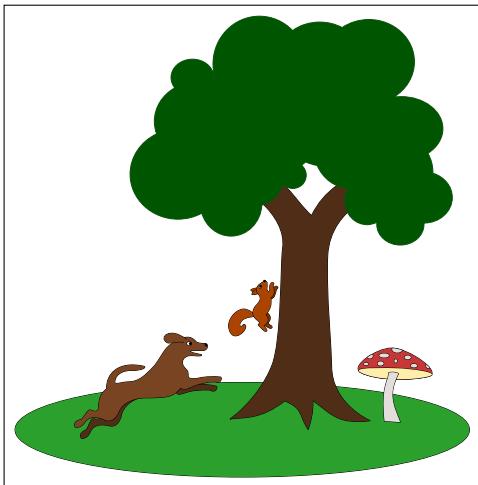


Obrázek 12: Obecné schéma tripletu

Každý objekt může být součástí libovolného počtu tripletů a to jak v pozici zdrojového, tak cílového objektu. Objekt také nemusí být součástí žádných vazeb a dokonce v celém popisu nemusí být žádné vazby definované. Například pokud bychom chtěli pouze testovat paměť uživatele, mohli bychom mu ukázat obrázek, pak jej skrýt a sledovat, jaké objekty si vybaví. V takové úloze nás vazby mezi objekty vůbec nemusejí zajímat a je tedy zbytečné, aby byly součástí referenčního popisu.

Při vytváření referenčního popisu obrázku se opět naskytá otázka, jaké vazby a vztahy mezi objekty do popisu zavést, a které lze ignorovat. Například relativní velikost nebo pozice objektů může být pro některé úlohy klíčová, ale pro jiné zcela nepodstatná. Tato problematika byla opět ponechána na expertovi, který vytváří referenční popis pro danou konkrétní aplikaci, aby rozhodl, které informace je potřeba v referenčním popise zachytit.

Konkrétní příklad několika tripletů je znázorněn na Obrázku 13.



Obrázek 13: Příklad použití tripletů pro popis vztahů mezi objekty

5.2.5 Další typy informací

Kromě výše zmíněných objektů, jejich vlastností a vazeb, je také možné pro některé aplikace požadovat, aby byly zachyceny i další jiné informace.

Jednou takovou dodatečnou informací může být například přesah zobrazené scény do reálného života, nebo extrapolace dějů a objektů. Tím jsou myšlené různé další objekty, děje, činnosti nebo vlastnosti, které nejsou přímo zobrazené na daném obrázku, ale z kontextu by bylo možné je nějakým způsobem předvídat. Příkladem takové situace může být ilustrace zobrazující například dítě, které si hraje blízko břehu řeky - v takovém případě je možné požadovat zachycení informace ve stylu „dítě může spadnout do vody“.

Druhým typem dodatečné informace může být meta-popis samotné scény. Pod tímto pojmem jsou myšlené obecné informace o obrázku, například celkový popis toho, co je na scéně vyobrazeno. Například pro Obrázek 19 by mohl být takovýmto meta-popisem například „scéna na břehu rybníka“.

Pro oba tyto typy informací nebyly navrženy žádné speciální datové struktury. Bylo rozhodnuto, že v případě potřeby je možné tyto informace zachytit dostatečně dobře pomocí již definovaných objektů, atributů a vazeb mezi objekty.

Případné rozšíření formy referenčního popisu o nějaké další specializované typy popisů bylo ponecháno jako předmět dalšího budoucího možného rozšíření.

5.3 Extrakce sémantické informace

Dalším klíčovým bodem bylo najít způsob, jak porovnat uživatelský popis obrázku s jeho referenčním popisem. To je úlohou subsystému pro získání sémantické informace z textu. Cílem je z přirozené řeči extrahovat informace v takové podobě, aby je bylo možné porovnat se strukturovaným referenčním popisem.

Extrakce sémantiky z přirozené řeči se běžně dělá z textového přepisu dané promluvy. I v této práci tedy extrakce sémantických informací probíhá z textu. To znamená, že pokud uživatel popíše obrázek mluvenou řečí, tak je potřeba promluvu převést do textu pomocí nějakého systému pro rozpoznání řeči. Problematika rozpoznání řeči a převodu audia do textu je nad rámec této práce a bude v praxi řešena nějakou již existující implementací. Ve zbytku práce bude tedy pro zjednodušení rovnou uvažovaným vstupem text.

Otázka extrakce sémantické informace z popisu přirozenou řečí se tedy zužuje na extrakci sémantiky z textu. Jako způsob řešení byl zvolen přístup založený na sémantickém parsování pomocí bezkontextových gramatik.

Tento přístup funguje tak, že na základě referenčního popisu obrázku bude expertem sestavena gramatika, podle které budou v textu detekované jednotlivé části sémantické informace. Gramatika je v tomto kontextu sada pravidel, která definují, jaké promluvy jsou v textu očekávané a jak mohou tyto promluvy vypadat. Program pak tato pravidla načte a prochází podle nich vstupní text a hledá, zda nějaké části textu „pasují“ na dané pravidlo. Samotný nalezený kus textu, který „pasuje“ na nějaké pravidlo, ale ještě nemá žádný užitečný význam. Proto pravidla obsahují zároveň i informaci o tom, jaká je struktura nalezené části textu a jaký význam mají jednotlivé části této struktury. V českém jazyce by se jednalo o obdobu větného rozboru, kdy se například určuje, která část věty je podmět a která přísudek.

Strukturovaná a označovaná část textu s doplněnou sémantickou informací je pak označována jako *sémantická entita*. Konkrétní syntaxe, použití a funkčnost těchto gramatik bude popsán později spolu s implementací v části 6.2. Množina všech těchto sémantických entit, které byly ze vstupního textu extrahované, jsou pak použité pro vytvoření testovaného popisu.

5.3.1 Podoba sémantických entit

První otázkou, kterou bylo potřeba vyřešit pro získání sémantické informace z přirozeného popisu, byla její podoba. Jinými slovy, jak by měla extrahovaná sémantika vypadat, aby ji bylo možné porovnat s referenčním popisem obrázku.

Vzhledem k tomu, že výše definovaný referenční popis (viz sekce 5.2) se skládá z objektů, jejich hierarchie, atributů a vazeb, tak se nabízí přímo použít tyto typy informací. Bylo tedy rozhodnuto, že z přepisu přirozené řeči budou extrahované objekty, jejich atributy a vazby mezi nimi.

5.3.2 Extrakce sémantiky - objekty

Základním typem informace, kterou musí být systém schopen z přepisu řeči získat, jsou samotné objekty, které expert definoval v referenčním popisu obrázku.

Například ze vstupní promluvy

„Na obrázku vidím psa s veverkou, strom a nějakou houbu.“

by měl systém vrátit množinu detekovaných objektů \mathcal{O} :

$$\mathcal{O} = \{ \text{pes, veverka, strom, houba} \}$$

Ve své nejjednodušší podobě by se mohlo jednat pouze o detekci nějakých klíčových slov, které odpovídají názvům objektů. Skutečná realizace je poněkud složitější, bere v potaz různá synonyma, tvary slov a alternativní vyjádření. Detailněji bude popsána později spolu s konkrétní implementací v kapitole 6.

Detekce samotných objektů by mohla být pro některé aplikace dokonce postačující sama o sobě. Mohlo by se jednat o úlohy, kde je hlavním předmětem pouze zjistit, kolik objektů na obrázku člověk popíše, případně které to jsou. Typickými příklady by mohli být nějaké testy paměti, pozornosti nebo jednoduché klasifikace.

5.3.3 Extrakce sémantiky - atributy

Druhým typem sémantické informace, kterou je potřeba, aby byl systém schopen najít a extrahovat z textu, jsou atributy popisující vlastnosti objektů.

Ve srovnání s detekcí samotných objektů se jedná o značně složitější problém, protože pro extrakci atributu je potřeba mít informace o objektu, na který se atribut váže, o názvu daného atributu a také o jeho přiřazené hodnotě.

Například ve větě

„Na obrázku vidím kluka v modrém tričku.“

by měl systém detekovat, že objekt „tričko“ má atribut „barva“ s hodnotou „modrá“.

Kromě toho, že se tato informace skládá z více nezávislých částí, tak je možné si všimnout, že se ve zdrojové větě nikde nevyskytuje slovo „*barva*“. Toto je tedy informace, kterou musí být systém schopen nějakým způsobem indukovat z okolních dat a referenčního popisu.

Dále by jiný uživatel mohl popsát stejný obrázek třeba větou

„Vidím nějakého kluka v tričku, které je modré.“

Tato promluva obsahuje stejný objekt, atribut i hodnotu, ale vyjádřenou ve zcela jiné podobě. Jak je tedy zřejmé, zde již není možné použít pouze nějakou formu detekce klíčových slov, ale bude potřeba komplikovanějšího přístupu.

Právě dříve zmíněná pravidla, která udávají různé formy hledané informace, umožňují zachytit i takovéto složitější struktury v různých formách. Konkrétní realizace bude opět popsána později, viz sekce 6.

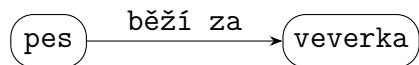
5.3.4 Extrakce sémantiky - vazby mezi objekty

Posledním typem informace, kterou je potřeba získat z textu, jsou vazby mezi objekty.

Například pro větu

„Na obrázku hnědého vidím psa, co běží za oranžovou veverkou.“

je třeba, aby systém dokázal detektovat vazbu mezi objektem psa a veverky:



Jak je zřejmé, pro sestavení vazby je - stejně jako pro atributy - potřeba tří částí informace: zdrojový objekt, cílový objekt a název vazby. Na rozdíl od detekce atributů ale představuje detekce vazeb unikátní problém, protože zdrojový a cílový objekt jsou stejného typu - oba jsou to objekty. To znamená, že v pří konstrukci tripletů musí být použit nějaký mechanismus, který rozpozná, který z objektů má být cílový a který zdrojový. Tento problém je řešen také v rámci definice pravidel v bezkontextové gramatice pomocí tagů, detailní popis v sekci 6.2.

Je vhodné zmínit, že ačkoli byly v kapitole 5.2 definované čtyři aspekty popisu, nyní jsou řešené pouze tři různé typy sémantických entit. To je proto, že hierarchie mezi objekty byla při návrhu považována za speciální případ vazby mezi objekty a nebyla pro ni vytvořena samostatná kategorie.

5.4 Hodnotící algoritmus

Za předpokladu, že je k dispozici referenční popis vytvořený expertem a testovaný popis získaný automaticky z přirozené řeči, je potřeba tyto dva popisy nějakým způsobem porovnat a určit, do jaké míry odpovídá testovaný popis referenčnímu.

Právě to je předmětem této kapitoly: navrhnout algoritmus, který by zajišťoval porovnání testovaného popisu s referenčním a jehož výstupem by bylo nějaké hodnocení.

Základ hodnotícího algoritmu vychází ze struktury referenčního a testovaného popisu. Oba typy se skládají z množiny objektů \mathcal{O} , množiny atributů \mathcal{A} a množiny vazeb \mathcal{V} . Dolním indexem je značeno, zda se jedná o množinu z referenčního popisu (R), nebo

testovaného popisu (T). Oba popisy obrázku lze tedy vyjádřit jako trojice:

$$\text{referenční popis obrázku} = (\mathcal{O}_R, \mathcal{A}_R, \mathcal{V}_R),$$

$$\text{testovaný popis obrázku} = (\mathcal{O}_T, \mathcal{A}_T, \mathcal{V}_T).$$

Porovnání a hodnocení podobnosti popisů by tak mohlo být převedeno na otázku porovnání podobnosti množin. Otázky teorie množin jsou ale nad rámec zadání a pro tuto práci byl definován hodnotící algoritmus, který je inspirován ztrátovými funkcemi.

Na obecné úrovni by se dalo říci, že hodnotící algoritmus počítá celkové ztráty pro chybějící objekty, chybějící atributy, atributy s chybnou hodnotou, chybějící vazby mezi objekty.

Vstupem hodnotícího algoritmu jsou tedy oba popisy (referenční a testovaný) a k tomu ještě ztrátová tabulka, která určuje, jaký typ chyby způsobí jak velkou ztrátu. Tuto ztrátovou tabulku také sestavuje expert, společně s referenčním popisem.

Výstupem hodnotícího algoritmu je pak množina označených číselných hodnot, které reprezentují celkové ztráty pro různé druhy chyb.

5.4.1 Chybějící objekty

Pokud byl v referenčním popisu expertem označený nějaký objekt, který chybí v testovaném popise, předpokládá se, že expert považoval daný objekt za důležitý a uživatel jej nezmínil.

Uživatel si třeba nemusel objektu všimnout, nebo jej nepovažoval za dostatečně důležitý. V obou případech se však jedná o nějaký rozpor se vzorovým popisem, který je třeba penalizovat.

Například pro situaci:

$$\mathcal{O}_R = \{\text{pes, veverka, houba, strom}\}$$

$$\mathcal{O}_T = \{\text{pes, veverka}\}$$

je zřejmé, že v testovaném popisu chybí dva objekty: houba a strom.

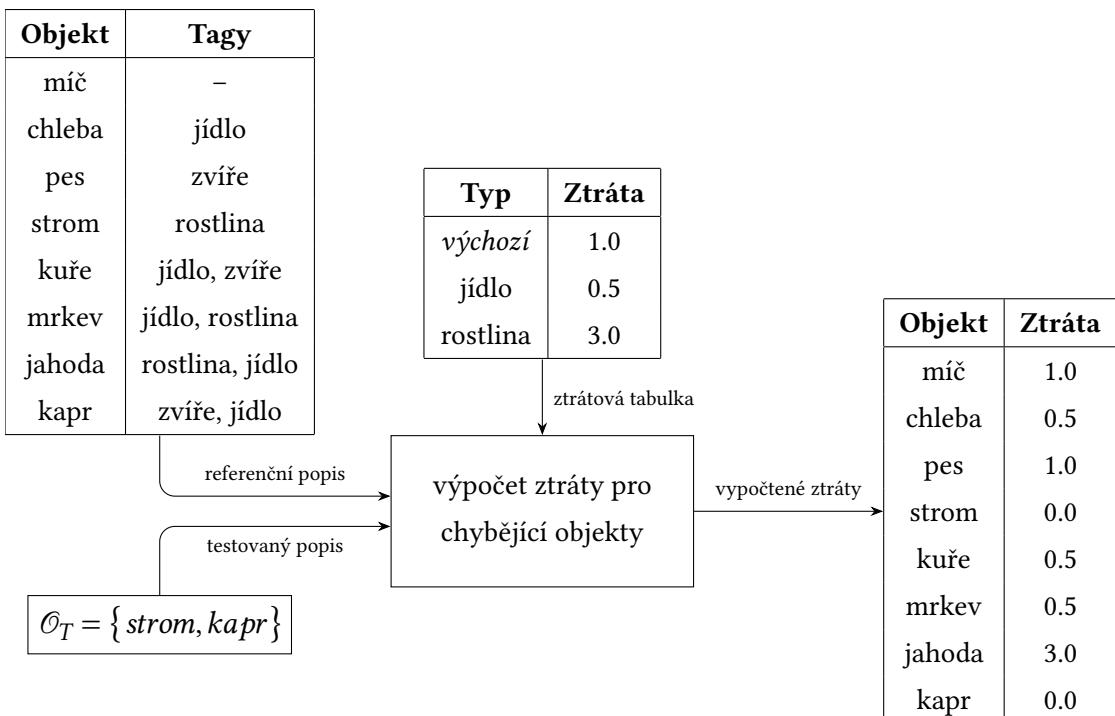
Lze očekávat, že různé objekty budou ve scéně různě důležité a bylo by vhodné, aby byl tento fakt zohledněn při výpočtu ztráty. Proto lze přiřadit objektům *tagy*. Jako *tag* je

v tomto kontextu chápána nějaká značka, která říká, že objekt patří do dané skupiny. Například objekt „tričko“ může mít přiřazen tag „*oblečení*“ a objekt „pes“ může mít přiřazen tag „*zvíře*“.

Pro větší variabilitu systému bylo dále rozhodnuto, že každý objekt může mít přiřazený libovolný počet tagů. Název tagů a jejich přiřazení objektům je volbou experta a je součástí tvorby referenčního popisu. Ve ztrátové tabulce pak může expert definovat ztrátové hodnoty pro jednotlivé tagy a tím tak přepsat hodnotu ztráty na daném objektu.

Pokud má objekt přiřazeno více tagů, na jejich pořadí záleží. Při určování ztrátové hodnoty pro daný objekt algoritmus postupně prochází jeden tag po druhém a kontroluje, zda je tento tag uvedený ve ztrátové tabulce. Pokud ano, je ztráta na daném jednom chybějícím objektu rovna této hodnotě a další tagy na tomto objektu již nejsou kontrolované. Pokud žádný z tagů není nalezen ve ztrátové tabulce, je použita výchozí hodnota ztráty pro jakýkoli chybějící objekt.

Pro názornost je schéma jednoho ukázkového případu na Obrázku 14.



Obrázek 14: Jednoduchá ukázka ztráty na chybějících objektech

Na příkladu zobrazeném na Obrázku 14 je možné vidět, že objekt *strom*, který byl nalezený v testovaném popisu, nezpůsobil žádnou ztrátu. Ostatní objekty, které v testovaném popise chybí, způsobily ztrátu odpovídající hodnotě ze ztrátové tabulky. Dále je možné si všimnout, že objekty *mrkev* a *jahoda* mají stejné tagy, ale v jiném pořadí, proto jsou jimi způsobené ztráty rozdílné.

Pro větší kompaktnost výstupu nejsou vypočtené ztráty prezentované pro každý objekt samostatně, ale jsou sečtené do finálního výsledku. Tento proces sčítání ztrátových hodnot je přes všechny objekty, ale také jsou počítané hodnoty přes jednotlivé tagy. Pokud bychom vzali tabulku vypočtených ztrát pro jednotlivé objekty z příkladu na Obrázku 14, šel by tento finální krok znázornit Obrázkem 15.

Objekt	Tagy	Ztráta		Chybějící objekty	Ztráta
míč	—	1.0		všechny	6.5
chleba	jídlo	0.5	→	objekty s „jídlo“	4.5
pes	zvíře	1.0		objekty s „zvíře“	1.5
strom	rostlina	0.0		objekty s „rostlina“	3.5
kuře	jídlo, zvíře	0.5			
mrkev	jídlo, rostlina	0.5			
jahoda	rostlina, jídlo	3.0			
kapr	zvíře, jídlo	0.0			

Obrázek 15: Finální krok při počítání ztrát na chybějících objektech

Zde je vhodné zmínit, že popsaný algoritmus výpočtu ztráty je pouze jednou z možností. Pro konkrétní nasazení by mohla být vhodnější jiná forma počítání ztrátových hodnot, například vypočítat průměr, sumu nebo maximum či minimum. Tento způsob počítání ztráty byl zvolen proto, že poskytuje expertovi další stupeň volnosti, jak vyjádřit důležitost objektů.

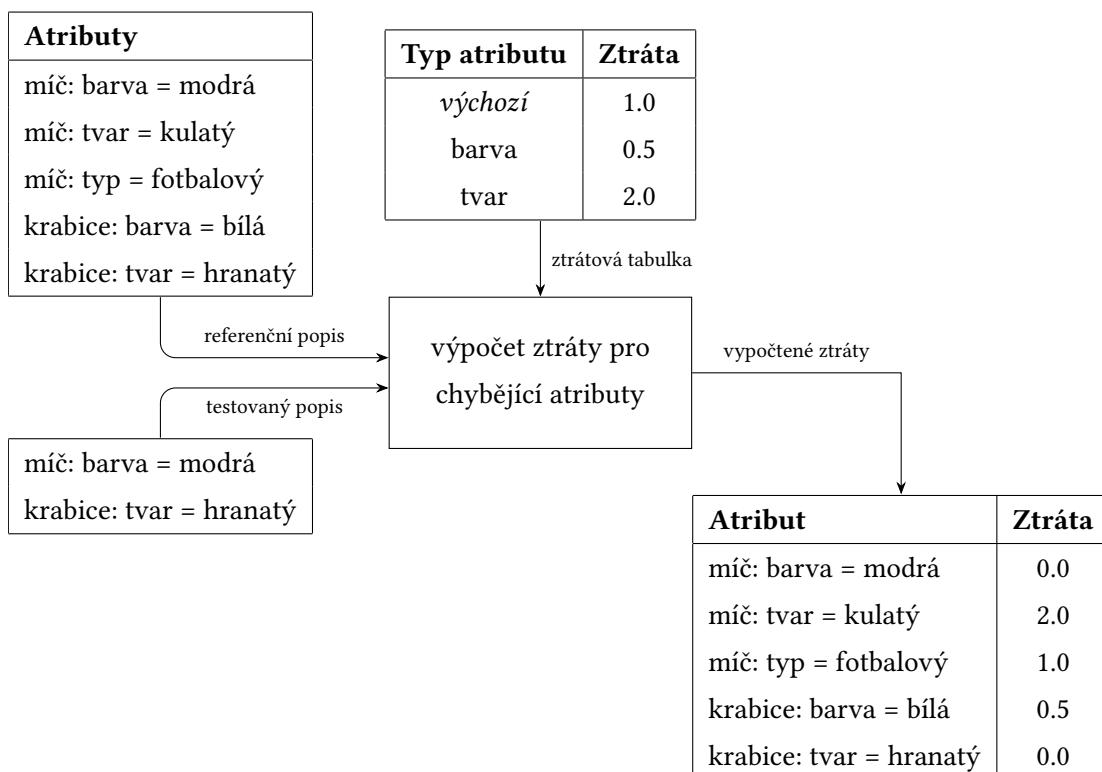
5.4.2 Chybějící atributy

Pokud expert popsal na daném objektu nějakou vlastnost a odpovídající atribut chybí v testovaném popise, je zde stejný předpoklad jako u chybějících objektů, a také se jedná o chybu, která způsobí navýšení ztrátové hodnoty.

Způsob hodnocení je obdobný jako pro chybějící objekty, pouze s tím rozdílem, že atributům nejsou přiřazovány žádné tagy. Tagy u objektů sloužily k detailnějšímu určení důležitosti objektu a později ke sčítání ztrátových hodnot, v případě atributů tuto funkci zastává samotný název atributu.

Například pokud by byl libovolný chybějící atributu penalizován ztrátou 1.0, pak by bylo možné specifikovat, že vynechání atributu „barva“ je méně závažné než všechny ostatní a penalizovat pouze ztrátou 0.5. Naopak vynechání atributu „výraz v obličeji“ (např. úsměv nebo zamračení) by mohl být považován za důležitý a expert by mohl ve ztrátové tabulce definovat, že jeho vynechání bude penalizováno ztrátou 2.0.

Jednoduchý příklad je znázorněn na Obrázku 16.



Obrázek 16: Ukázka výpočtu ztrát pro chybějící atributy

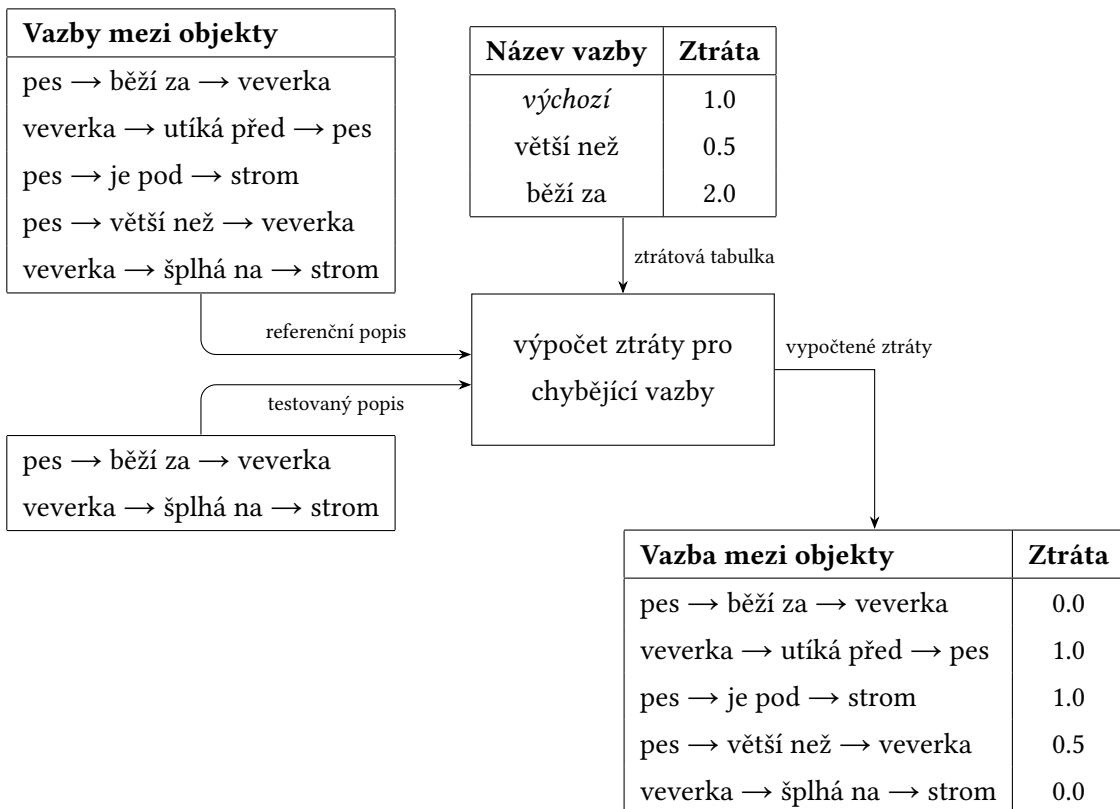
5.4.3 Chybějící vazby mezi objekty

Posledním typem entity, která může chybět při porovnávání referenčního a testovaného popisu, jsou vazby mezi objekty popsané *triplety*.

Zde je hodnotící algoritmus velmi podobný jako při počítání ztráty pro chybějící objekty. Rozdíl zde spočívá v tom, že místo tagů jsou zde použité samotné názvy vazeb, podobně jako u chybějících atributů jsou použité názvy atributů.

Expert ve ztrátové tabulce opět specifikuje, jaká je výchozí hodnota ztráty, pokud v testovaném popise bude chybět libovolná vazba mezi objekty. Následně může také specifikovat, že některé konkrétní typy vazeb (podle jejich názvu) mají větší či menší důležitost a tak jimi způsobená ztráta může být větší nebo menší. Algoritmus pak při výpočtu použije přednostně takovou hodnotu, která je více specifická.

Konkrétní příklad je znázorněn na Obrázku 17.



Obrázek 17: Ukázka výpočtu ztrát pro chybějící vazby mezi objekty

5.4.4 Atributy s chybnou hodnotou

Pokud byl v testovém popise nalezen atribut, který odpovídá objektem a názvem nějakému atributu z referenčního popisu, ale neodpovídá svou hodnotou žádnému referenčnímu atributu, pak se předpokládá, že uživatel udělal chybu při popisu a řekl špatnou hodnotu.

Příkladem takové neshody by mohlo být:

$$\mathcal{A}_R = \{ \text{tričko: barva} = \text{modrá} \},$$
$$\mathcal{A}_T = \{ \text{tričko: barva} = \text{zelená} \},$$

pak je možné předpokládat, že uživatel udělal při popisu chybu a řekl špatnou barvu.

Způsob počítání ztrátové hodnoty z těchto chybných hodnot je v principu stejný jako u chybějících objektů a atributů. Expert ve ztrátové tabulce definuje výchozí hodnotu a k tomu případně specifické případy, kdy je tato ztrátová hodnota jiná.

Rozdíl oproti předchozím výpočtům ovšem spočívá v tom, že různé hodnoty si mohou být různě blízko. Například pokud uživatel zamění hnědou a oranžovou barvu veverky, tak se pravděpodobně ve většině případů bude jednat o menší chybu, než kdyby oranžovou zaměnil třeba za zelenou.

Z toho důvodu je ztrátová tabulka navržena tak, aby expert mohl definovat výchozí ztrátu pro všechny chybné atributy, výchozí ztrátu pro konkrétní atribut bez ohledu na hodnoty, ztrátu pro konkrétní atribut a zároveň množinu konkrétních hodnot, jejichž záměna tuto ztrátu způsobí.

Pro lepší názornost následuje příklad, který předpokládá, že expert definoval ztrátovou tabulkou shodnou s Tabulkou 1. Tu je možné číst následujícím způsobem:

1. Libovolný chybný atribut způsobí ztrátu 1.0, pokud není ve zbytku tabulky definována více specifická alternativa.
2. Pokud je chybná hodnota v atributu „barva“, pak je způsobená ztráta rovna 0.5, pokud se nejedná o záměnu některých konkrétních hodnot:
 - Pokud je zaměněna barva „žlutá“ a „oranžová“, pak je ztráta 0.3.
 - Pokud je zaměněna barva „červená“, „modrá“ nebo „zelená“ (libovolná dvojice), pak je ztráta rovna 0.8.

3. Pokud je chybná hodnota v atributu „tvar“, pak je ztráta 1.5.

Ukázka některých konkrétních atributů s chybnými hodnotami a k nim vypočtených ztrát je zobrazena v Tabulce 18.

Atribut	Ztráta
výchozí	1.0
barva	•
tvar	•

Hodnoty	Ztráta
výchozí	0.5
žlutá, oranžová	0.3
červená, modrá, zelená	0.8

Hodnoty	Ztráta
výchozí	1.5

Obrázek 18: Část ukázkové ztrátové tabulky

Referenční atribut	Testovaný atribut	Ztráta
pes: barva = hnědá	pes: barva = hnědá	0.0
oheň: barva = žlutá	oheň: barva = oranžová	0.3
jablko: barva = červená	jablko: barva = modrá	0.8
tričko: barva = modrá	tričko: barva = zelená	0.8
čepice: barva = modrá	čepice: barva = žlutá	0.5
penál: tvar = válec	penál: tvar = kvádr	1.5
pes: činnost = leží	pes: činnost = leží	1.0

Tabulka 1: Výpočtu ukázkových ztrát chybných hodnot atributů podle Tabulky 18

Přirozeně se naskýtá otázka toho, jak počítat ztrátu při konfliktu objektů se stejnými atributy a nebo objekty s více atributy. Pro řešení těchto konfliktů bylo rozhodnuto, že hodnotící algoritmus bude optimistický a bude uvažovat nejmenší možnou chybu, pokud je podle ztrátové tabulky k dispozici více možných interpretací.

5.4.5 Výstup hodnotícího algoritmu

Výstup hodnotícího algoritmu, jak byl popsán v předchozích částech, obsahuje vždy celkovou ztrátu pro daný typ chyby a poté ještě akumulované ztrátové hodnoty stejného typu, ale rozdělené podle jednotlivých kategorií.

Konkrétně bude výstupní struktura obsahovat:

- celkovou ztrátu způsobenou všemi chybějícími objekty,
- celkovou ztrátu způsobenou všemi chybějícími atributy,
- celkovou ztrátu způsobenou všemi chybějícími vazbami mezi objekty,
- celkovou ztrátu způsobenou všemi atributy s chybnými hodnotami,
- ztráty způsobené chybějícími objekty s konkrétním tagem,
- ztráty způsobené chybějícími atributy s konkrétním názvem/typem atributu,
- ztráty způsobené chybějícími vazbami s konkrétním názvem/typem vazby,
- ztráty způsobené atributy s chybnými hodnotami, rozdělené podle názvu/typu atributu.

Je vhodné zmínit, že při návrhu hodnotícího algoritmu bylo předpokládáno, že pro různá konkrétní nasazení mohou být rozdílné požadavky na formu výstupu. V rámci implementace popsané později existuje proto „skrytý“ mezi-krok, ve kterém jsou k dispozici všechny jednotlivé ztráty pro konkrétní objekty, atributy a vazby, ze kterých je poté počítána právě popsaná výsledná struktura. Tento mezi-výsledek by bylo možné výhodně využít pro implementaci alternativních způsobů hodnocení, nebo případně rovnou použít jako výstup, pokud by daná aplikace benefitovala z podrobnější analýzy výsledků.

6 Implementace a testování

V této kapitole je rozebrána konkrétní technická implementace navržených a výše popsaných konceptů a algoritmů. Jako první bude popsána implementace referenčního popisu obrázku a odpovídající reprezentace znalostí, následuje popis sémantické analýzy pomocí bezkontextových gramatik a následná tvorba testovaného popisu. Poté je popsána implementace hodnotícího algoritmu včetně ukázek a na závěr jsou ukázány ilustrační příklady funkčnosti celého systému.

Systém byl vytvořen jako program s ovládáním z příkazové řádky (angl. command line interface, CLI). Sestává z jednotlivých příkazů, které jednotlivě vykonávají příslušné fáze celého systému, od načtení, validace a manipulace s referenčním popisem, přes sémantickou analýzu textu až po finální vyhodnocení.

Současná implementace slouží především k testování a vývoji navržených algoritmů, nicméně během implementace bylo dbáno na to, aby byly jednotlivé fáze a příkazy modulární a bylo možné je snadno použít pro zakomponování do konkrétních aplikací.

Celý systém byl programován v jazyce Rust, kompletní zdrojový kód, včetně použitých dat, je k dispozici online na adrese <https://github.com/TomLebeda/diplomka>.

6.1 Referenční popis obrázku

Jako první byl implementován systém pro práci s referenčním popisem obrázku. Aby bylo možné vytvořenou implementaci rovnou testovat, bylo nutné zvolit nějaký konkrétní obrázek, na kterém by bylo možné jednotlivé části zkoušet. Pro tento účel byla zvolena kresba na Obrázku 19, která pochází z projektu TAČR SIGMA_DC3 „Telemedicínské samovyšetření řeči a paměti pro rychlou detekci kognitivních poruch metodami strojového učení“.

Jak bylo definováno v kapitole 5.2, referenční popis obrázku se skládá z množiny objektů, jejich atributů a vazeb mezi nimi. Nyní bylo potřeba stanovit nějaký datový formát, který by umožnil tuto strukturu dobře zachytit.



Obrázek 19: Kresba použitá pro testování

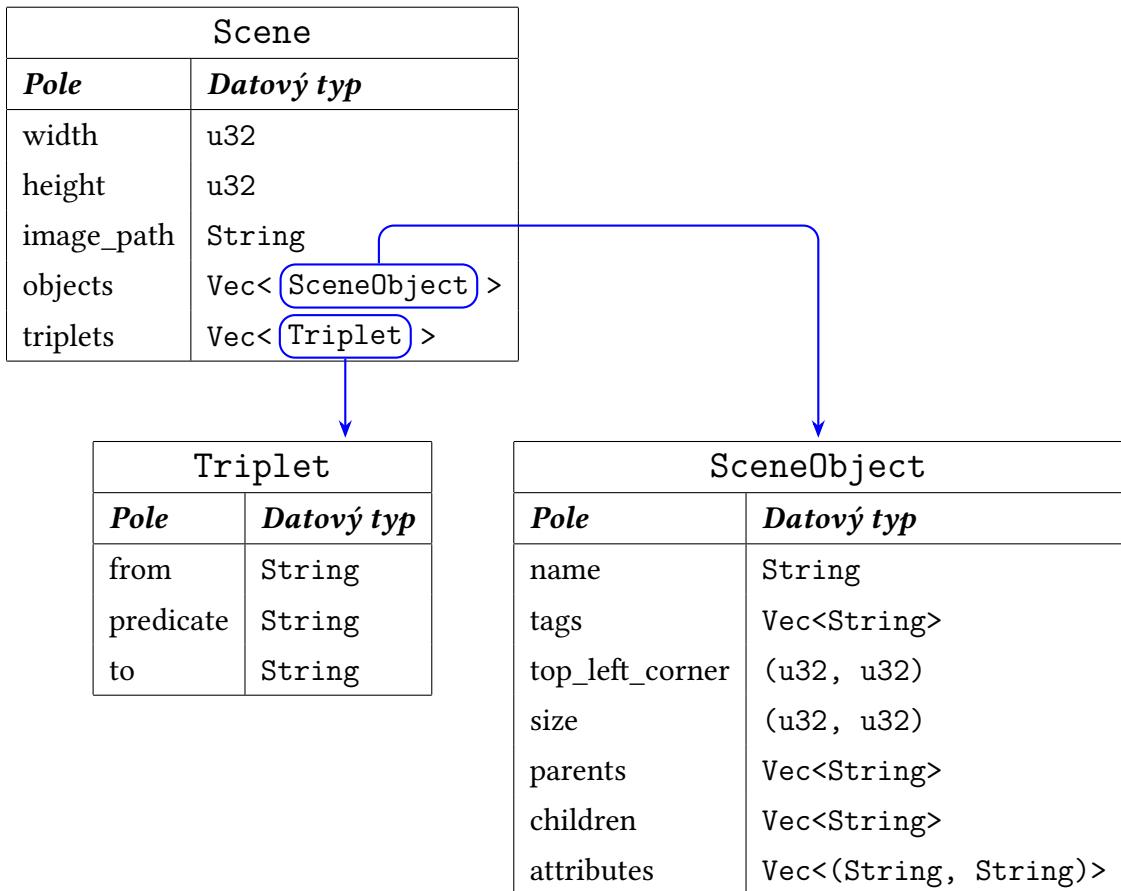
Základním prvkem referenčního popisu je struktura „Scene“, která obsahuje informace o obrázku, objekty s atributy a vazby mezi nimi.

Objekty byly definované strukturou „SceneObject“, která popisuje objekt ve scéně a jeho atributy. Každý SceneObject obsahuje informaci o svém názvu, umístění, velikosti a hierarchii vůči ostatním objektům. Dále si každý SceneObject udržuje list atributů a tagů, které mu byly expertem přiděleny.

Další strukturou, která byla definována jako součást referenčního popisu, je „Triplet“. Tato struktura slouží k popisu vazeb mezi objekty, obsahuje názvy dvou objektů a k nim název vazby.

Schéma implementovaných struktur a jejich provázání je na Obrázku 20. Zkratka pro datový typ u32 znamená „unsigned 32-bit int“ a je to nezáporné celé číslo na 32 bitech, druhý typ Vec<T> pak značí vektor (list) objektů typu T.

Ve struktuře Scene je význam jednotlivých polí poměrně jednoduchý. Pole width a height vyjadřují výšku a šířku obrázku v pixelech, image_path udává cestu k souboru, který obsahuje samotný obrázek. Pole objects a triplets pak obsahují list objektů a tripletů.



Obrázek 20: Schéma implementace referenčního popisu

Struktura `Triplet` je velmi jednoduchá. Její pole `from` a `to` jsou názvy zdrojového, respektive cílového, objektu. Pole `predicate` je název samotné vazby mezi těmito objekty.

Poslední struktura `SceneObject`, která popisuje samotné objekty, je nejsložitější. Pole `name` udává název daného objektu, který slouží zároveň jako identifikátor a musí být tudíž unikátní. Může nastat situace, že v obrázku bude více objektů, které by přirozeně byly označené stejně, například dva stromy. Pro tento případ byla implementována možnost objekty číslovat a tím tak jednoznačně rozlišit i objekty stejného typu. Číslování objektů je součástí jejich názvu a má formát „#n“ kde *n* je přirozené číslo označující index objektu. Například „strom #1“ a „strom #2“.

Dvojice polí `top_left_corner` a `size` udávají velikost a pozici objektu zdrojovém obrazku. Jedná se o souřadnice levého horního rohu ohraničujícího rámce (bounding-boxu) a velikost tohoto rámce v pixelech. Obě pole mají jako uvedený datový typ (u32, u32),

který reprezentuje souřadnice (i, j) kde $i, j \in \mathbb{N}^0$. V obrázku je jako počátek souřadnic uvažován levý horní roh.

Pole `tags` reprezentuje list tagů, které expert obrázku přiřadil. Jejich účel byl popsán v kapitole 5.4. Další dvojice polí `parents` a `children` pak reprezentuje list objektů, které jsou rodiče, respektive potomci, daného objektu.

Po formální definice formátů a datový typů bylo potřeba určit, jakým způsobem bude expert referenční popis tvořit. Pro tuto práci bylo rozhodnuto, že tvorba referenčního popisu bude spočívat v napsání strukturovaného textového souboru. Zvolen byl textový formát JSON, který představuje dobrý kompromis mezi čitelností člověkem a strojovou zpracovatelností.

Ukázka části referenčního popisu je na Výpisu 2.

```
1  {
2      "width": 2002,
3      "height": 1123,
4      "image_path": "./data/imgs/summer.png",
5      "objects": [
6          {
7              "name": "tree #1",
8              "tags": ["environment"],
9              "top_left_corner": [1194, 146],
10             "size": [264, 220],
11             "parents": [],
12             "children": ["branch"],
13             "attributes": [
14                 ["color", "green"],
15                 ["color", "brown"]
16             ]
17         }, {
18             "name": "bird",
19             "tags": ["animal"],
20             "top_left_corner": [1095, 193],
21             "size": [93, 47],
22             "parents": [],
23             "children": [],
24             "attributes": [[["color", "white"]]]
25         }, {
26             "name": "squirrel",
27             "tags": ["animal"],
28             "top_left_corner": [1655, 286],
29             "size": [116, 76],
30             "parents": [],
31             "children": [],
32             "attributes": [
33                 ["color", "orange"],
```

```

34     ["color", "brown"]
35   ]
36 }, {
37   "name": "bird",
38   "tags": ["animal"],
39   "top_left_corner": [1095, 193],
40   "size": [93, 47],
41   "parents": [],
42   "children": [],
43   "attributes": [[{"color": "white"}]]
44 }, {
45   "name": "branch",
46   "tags": ["environment", "background"],
47   "top_left_corner": [1130, 233],
48   "size": [138, 41],
49   "parents": ["tree #1"],
50   "children": [],
51   "attributes": []
52 }
53 ],
54 "triplets": [
55   {"from": "bird", "predicate": "sitting on", "to": "branch"},
56   {"from": "bird", "predicate": "sitting on", "to": "tree #1"},
57   {"from": "squirrel", "predicate": "climbing", "to": "tree #1"},
58   {"from": "squirrel", "predicate": "under", "to": "bird"}
59 ]
60 }

```

Výpis 2: Příklad jednoduchého referenčního popisu

6.1.1 Validace referenčního popisu

Jak bylo právě ukázáno, tvorba referenčního popisu je ve své podstatě psaní JSON souboru podle nějaké definované struktury a pravidel. Program při zpracovávání referenčního popisu předpokládá, že data budou validní a odmítne referenční popis, který by obsahoval chyby. Pro komplexnější scénu však může být náročné pro lidského experta neudělat v popise žádnou chybu a udržet celý referenční popis validní.

Z toho důvodu byl implementován v rámci celého programu příkaz `check`, který zajistí kontrolu všech dříve definovaných pravidel, které musí referenční popis splňovat:

1. Kontrola názvů objektů

Protože je název objektu použit jako identifikátor, tak musí být unikátní a žádné dva objekty ve scéně nesmí mít shodný název. V případě, že by referenční popis obsahoval duplicitní názvy objektů, program na tyto konflikty upozorní.

2. Kontrola přetékání objektů

Jelikož má program k dispozici informace o velikosti obrázku (v pixelech) a zároveň pro jednotlivé objekty také jejich pozici a velikost, tak lze předem ověřit, zda uvedené ohraničující rámce nezasahují mimo obrázek. Pokud se tak stane, program vypíše varovné hlášení a upozorní na jednotlivé problematické hodnoty.

3. Kontrola referencí na objekty

Objekty mohou v rámci hierarchie odkazovat na jiné objekty. Aby byly tyto reference validní, tak objekt nesmí odkazovat sám na sebe, referovaný objekt musí existovat a hierarchické vazby musí být oboustranné. Tím je myšleno to, že pokud objekt **A** specifikuje objekt **B** jako svého rodiče, pak musí i objekt **B** specifikovat **A** jako svého potomka a naopak.

V případě, že by byly detekované nějaké invalidní odkazy, bude opět vypsáno varovné hlášení o tom, co je v referenčním popise špatně.

4. Kontrola referencí v tripletech

Při specifikaci vazeb mezi objekty jsou také odkazované dva objekty. Aby byl triplet validní, musí být oba referované objekty definované ve scéně. V opačném případě bude vypsáno chybové hlášení s informací o chybějících objekty v tripletech.

6.1.2 Další příkazy pro práci s referenčním popisem

Kromě příkazu na kontrolu referenčního popisu byly implementované ještě následující pomocné příkazy. Tyto příkazy slouží obecně k prohlížení a porovnávání referenčních popisů a byly implementované zejména pro situace, kdy by bylo potřeba analyzovat komplexnější scény, porovnávat je mezi sebou a získat obecný přehled o scéně, aniž by potřeba zkoumat samotné JSON soubory.

1. Příkaz stats:

Tento příkaz vypíše základní vybrané statistiky daného referenčního popisu. Hlavním smyslem tohoto příkazu je poskytnout člověku představu o referenčním popisu k němu příslušejícímu obrázku.

Příkaz vypíše cestu k souboru s referenčním popisem, dále cestu k souboru, který obsahuje popisovaný obrázek a jeho velikost v pixelech. Dále vypíše počet objektů a počet vazeb (tripletů), které byly v tomto referenčním popise definované.

2. **Příkaz list:**

Příkaz `list` slouží k vypsání seznamu všech entit daného typu, podobně jako při použití databázových dotazů. Tento příkaz má několik pod-příkazů, které určují, jaký typ informace má být vypsán v podobě seznamu. Může se jednat o objekty, tagy, triplety, atributy, samotné názvy nebo hodnoty atributů, názvy vazeb mezi objekty a další. Seznam všech implementovaných pod-příkazů je možné si prohlédnout ve zdrojovém kódu.

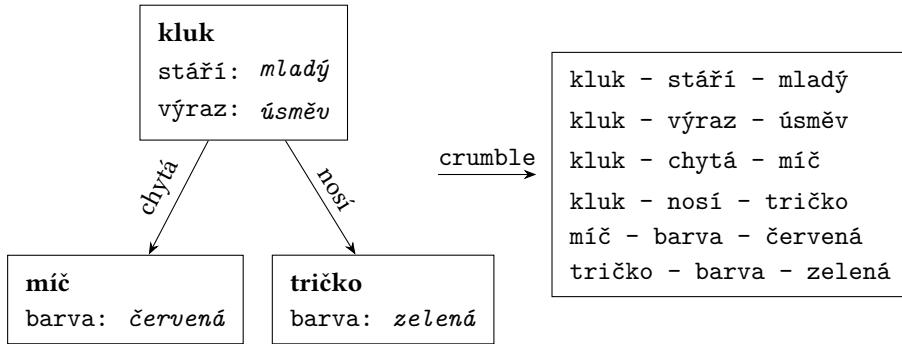
Cílem tohoto příkazu je poskytnout rychlý a pohodlný způsob, jak získat seznam všech prvků daného typu ve scéně, aniž by bylo potřeba přímo pracovat se složitější strukturou JSON souboru. Výstup je formátován tak, aby jej bylo možné snadno dále strojově zpracovat a integrovat do komplexnějších sekvencí příkazů.

3. **Příkaz crumble:**

Referenční popis obrazu byl v této práci definován jako množina objektů, jejich atributů a vazeb mezi objekty. Při pohledu na datový typ atributu a tripletu je ale možné si všimnout, že se v podstatě neliší - obojí se skládá ze tří částí, jejichž obsah může být zaměnitelný a rozdíl ve významu je dán pouze zavedenou interpretací. Bylo by tedy možné zobecnit všechny tyto entity a popsat je pouze jedním atomickým datovým typem, který obsahuje dva prvky a vazbu mezi nimi. V této podobě by bylo možné celý referenční popis zobrazit jako tradiční sémantickou síť, kde není rozlišeno, co je vazba mezi objekty a co je pouze popis vlastnosti.

Referenční popis nebyl navržen tímto obecnějším způsobem z toho důvodu, že pro komplexnější scény by počet těchto atomických trojic dosahoval tak vysokých hodnot, že by pro lidského experta bylo velmi obtížné se v datech orientovat. Pro strojové zpracování a případné vizualizace by ale mohlo být prospěšné, aby byly všechny typy informací prezentované v jednoduchém formátu. Právě k tomu slouží příkaz `crumble`, který, jak již název napovídá, daný referenční popis „rozdrobí“ a převede všechny informace do množiny prostých trojic.

Schématická ukázka procesu „rozdrobení“ je na Obrázku 21.



Obrázek 21: Ukázka procesu *crumble* („rozdrobení“)

4. Příkaz render:

Tento příkaz slouží k vytvoření vizuální reprezentace referenčního popisu.

Pro správnou funkci tohoto příkazu je nutné, aby byl na počítači dostupný program Graphviz [24], jehož DOT-engine je použitý pro vytvoření grafické reprezentace. Výhodou grafické prezentace je přehlednost pro člověka, nicméně pro velmi komplexní scény s mnoha vazbami ani grafická reprezentace nemusí být dostačená. Je vhodné zmínit, že jedním z problémů takto implementované vizualizace je to, že pro scény s mnoha objekty tvoří Graphviz velmi dlouhé a úzké obrázky, protože sázecí mechanismus skládá jednotlivé uzly pod sebe.

Grafická prezentace a metody vizualizace jsou nad rámec zadání této práce, proto bylo rozhodnuto, že pro ilustraci tato implementace dostačuje a případné komplexnější způsoby grafické prezentace jsou ponechány jako možná budoucí rozšíření.

6.2 Sémantické parsování pomocí gramatik

Extrakce sémantické informace byla jednou z hlavních řešených problematik. Jak již bylo řečeno dříve, byl zvolen přístup založený na parsování textu pomocí sémantických bezkontextových gramatik.

Při prvotních experimentech byla použita již existující implementace tohoto systému v platformě SpeechCloud, používající standard SRGS [25].

Během těchto prvních pokusů ale bylo zjištěno, že funkcionalita, kterou nabízí existující implementace, nebude pro potřeby této práce postačovat. Kvůli tomu byla navržena, implementována a otestována vlastní realizace stejného konceptu analýzy textu pomocí

bezkontextových gramatik, která lépe vyhovovala ostatním zde navrženým, realizovaným a popsaným postupům a metodám.

Detailedy této vlastní implementace a rozdíly oproti existujícím variantám jsou popsány v následujících kapitolách.

6.2.1 Problémy s existující implementací a SRGS

Při práci s existující implementací SRGS standardu na platformě SpeechCloud vyuštaly následující problémy:

1. Speciální pravidlo \$GARBAGE

První problém, který bylo potřeba vyřešit, byla absence pravidla \$GARBAGE. Jedná se o speciální pravidlo definované ve standardu SRGS, které má tu vlastnost, že dokáže reprezentovat libovolnou promluvu nebo libovolný token.

Lze tak velmi efektivně využít v situacích, kdy je potřeba specifikovat výplňová slova, která dopředu není možné odhadnout a nebo nás při analýze textu nezajímají.

Existující implementace však neměla toto speciální pravidlo implementováno. To představovalo problém, protože možnost specifikovat libovolný token je jeden ze základních konceptů, na kterém byl celý navržený koncept sémantické analýzy postaven.

2. Návratová datová struktura

Existující implementace nevracela ze svého API celý derivační strom, ale pouze list tagů, které se v derivačním stromu nacházely. To znamená, že byla ztracena hierarchická informace derivačního stromu spolu s informací o konkrétních pravidlech, které byly během zpracování textu použité.

Tyto informace nebyly zcela nezbytné pro další postup, avšak jejich absence by vyžadovala výrazně složitější definici parsovacích pravidel v bezkontextových gramatikách a s tím samozřejmě spojený složitější systém na zpracování obdržených výsledků.

Dostupnost celých derivačních stromů by tedy byla výhodná ve smyslu zjednodušení dalšího postupu.

3. Řešení nejednoznačných situací

Třetí problém vycházel z toho, že SRGS standard umožňuje existenci některých situací, kde není jednoznačně dán, jak má parsování probíhat, a teoreticky je možné získat z jednoho vstupu více různých derivačních stromů.

Existující implementace tyto situace dokázala zpracovat, avšak vrátila pouze jeden výsledek, který byl považovaný za nejlepší podle daného kritéria - v tomto případě podle délky zpracovaného textu.

Pro sémantickou analýzu textu potřebnou pro získání testovaného popisu by ale bylo výhodné, kdyby bylo možné toto kritérium změnit, a nebo ještě lépe, získat všechna možná řešení.

Po důkladném zvážení těchto problémů bylo rozhodnuto, že jako nejlepší způsob řešení bude implementace vlastního parseru.

Během reimplementace vlastního parseru byla jako základní reference využita specifikace SRGS [25], která přesně popisuje chování, funkčnost i syntax gramatik.

V průběhu reimplementace byly ovšem za účelem zdůraznění vynechány některé části SRGS standardu, které by pro sémantickou analýzu textu v této práci nebyly nijak užitečné a naopak byly přidány nějaké funkce navíc, které byly pak využity ve zbytku práce.

Výsledkem tak byl nový formát, označený „Semantic Parsing Grammar Format“ (SPGF). K němu byly samozřejmě vytvořené i základní softwarové nástroje, které nabízí:

- kontrolu syntaxe a případné hlášení o syntaktických chybách,
- TreeSitter [26] modul pro obarvení SPGF kódu v textových editorech,
- parser SPGF syntaxe s validací obsahu gramatiky i jednotlivých pravidel,
- parser přirozeného textu, který využívá právě SPGF gramatiky.

6.2.2 Semantic Parsing Grammar Format (SPGF)

Základním a nejvyšším prvkem SPGF je *gramatika*. Na rozdíl od SRGS definice, SPGF gramatika nevyžaduje žádnou hlavičku a nepodporuje dodatečné prvky, jako deklaraci jazyka nebo meta-dat. Gramatika má podobu textového souboru, ve kterém je definována množina parsovacích pravidel.

Každé pravidlo se skládá ze svého názvu, který je uvozený symbolem „\$“. Název pravidla slouží jako identifikátor, musí být tedy unikátní v dané gramatice. Druhou částí pravidla je expanze, která reprezentuje tělo pravidla. Mezi názvem a expanzí pravidla je znak rovnítka „=“. Pravidlo je vždy zakončeno středníkem. Například:

$$\$climbing = \underbrace{(\text{leze} \mid \text{šplhá})}_{\text{název pravidla}} \underbrace{[\text{na} \mid \text{po}]}_{\text{expanze (tělo pravidla)}};$$

Tělo pravidla se skládá z takzvaných *alternativ*, které jsou oddělené znakem „|“. Tato *alternativa* je pak dále definována jako posloupnost *elementů*. V zápisu má přednost mezera před „|“, takže několik tokenů po sobě je bráno jako jeden celek. Element má tři varianty, může jím být *token*, *reference* nebo *sekvence*. Například:

$$\$dog = \underbrace{\text{pejsek}}_{\text{alternativa 1}} \mid \underbrace{(\text{pes} \mid \text{psík})}_{\text{alternativa 2}} \mid \underbrace{\text{nejlepší} \underbrace{\text{přítel}}_{\text{alternativa 3}} \text{člověka}}_{\text{token token token}};$$

Nejjednodušší variantou *elementu* je *token*. Jedná se o terminální symbol (literál), který udává, jaký řetězec je při parsování právě přípustný. Token je možné chápat jako konkrétní slovo, které je v dané pozici v textu očekávané. Může se skládat z jednoho nebo více znaků, která mohou být písmena, číslice, tečka, pomlčka, podtržítko nebo dvojtečka. Součástí definice tokenu může také být (volitelně) definice opakování a případně libovolný počet tagů. Opakování elementů a tagy budou detailněji popsány později.

Druhou variantou elementu je *reference*. Tímto pojmem se rozumí odkaz na jiné pravidlo, a nebo odkaz na nějaké speciální pravidlo s vyhrazeným názvem. Reference začíná symbolem „\$“ po kterém následuje okamžitě (bez mezery) jméno referovaného pravidla definovaného ve stejné gramatice, nebo speciálního pravidla. SPGF umožňuje referovat pravidla rekurzivně, a to jak přímo (pravidlo obsahuje referenci na samo sebe), tak v cyklu.

Například jednoduchá gramatika pro (fiktivní) zápis vybraných matematických operací obsahuje rekurzivní pravidlo \$expr:

```
public $expr = $numer | $expr $operator $expr;
public $number = $digit<+>;
public $digit = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10;
public $operator = plus | minus | dot | frac ;
```

Formát SPGF definuje 5 speciálních pravidel s vyhraněnými jmény. Tři z nich jsou převzaté ze SRGS standardu, zbylé dvě byly přidány navíc, aby bylo možné vynutit jistá chování parsovacího algoritmu, která jsou v SRGS automaticky, ale v SPGF nikoli, kvůli odlišné strategii prohledávání (viz později).

1. Speciální pravidlo \$GARBAGE:

Akceptuje libovolný jeden token a posune parsování o akceptovaný token dále.

Speciální pravidlo \$GARBAGE je užitečné, když je potřeba vyjádřit, že na dané pozici v textu může být libovolné slovo, které buďto nedokážeme předvídat, nebo nás nezajímá.

2. Speciální pravidlo \$NULL:

Akceptuje prázdný řetězec. To znamená, že je vždy úspěšně aktivováno a nikdy neposune parsování o žádný znak dále.

Lze využít například pro situace, kde je žádoucí přidat do derivačního stromu (angl. parsing tree) nějaké tagy, aniž by byl učiněn postup v parsovaném textu.

3. Speciální pravidlo \$VOID:

Vždy selže, bez ohledu na parsovaný text. Nikdy tedy není úspěšné a sekvence obsahující \$VOID také nikdy nebude úspěšná.

Lze využít například pro dočasné blokování některých jiných pravidel, nebo pro zdůraznění, že daná sekvence tokenů se nesmí v textu vyskytovat.

4. Speciální pravidlo \$END:

Akceptuje konec textu. Bude úspěšné právě ve chvíli, kdy je parsování na konci textu a nezbývá již žádný znak ke zpracování.

Umístění speciálního pravidla \$END na konec nějaké expanze tak způsobí, že celá expanze bude úspěšná pouze ve chvíli, kdy dokáže pojmet text beze zbytku až do konce.

5. Speciální pravidlo \$BEGIN:

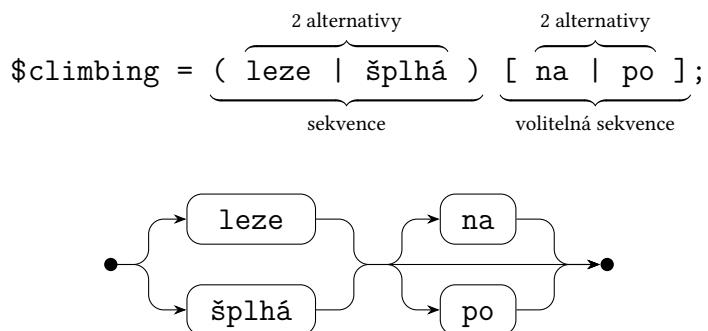
Akceptuje začátek textu. Bude úspěšné právě ve chvíli, kdy je parsování na začátku textu a žádný znak ještě nebyl zpracován.

Umístění speciálního pravidla \$BEGIN na začátek expanze tak způsobí, že celá expanze bude úspěšná pouze ve chvíli, kdy dokáže pojmet text od prvního znaku.

Poslední podobu, kterou *element* může mít, je *sekvence*. Sekvence může být „obyčejná“ nebo „volitelná“. V obou případech se jedná o *alternativy* oddělené znakem „|“, list alternativ je ohraničený kulatými, respektive hranatými, závorkami.

Důvodem pro existenci *sekvencí* je to, že umožňují seskupování elementů a přehlednější tvorbu složitějších konstrukcí, podobně jako jsou v matematice používané závorky v aritmetických či algebraických výrazech. Volitelná sekvence (s hranatými závorkami) je pouze syntaktická zkratka pro „obyčejnou“ sekvenci, která má minimální počet opakování roven nule (takže se nemusí vůbec v textu vyskytovat) a maximální počet opakování roven jedné. Jedná se o obdobu operátoru „?“ v regulárních výrazech.

Příklad zápisu pravidla s oběma typy sekvencí a jeho schématická reprezentace je pak na Obrázku 22.



Obrázek 22: Pravidlo \$climbing ve formátu SPGF a v podobě konečného automatu

6.2.3 Vstupní a výstupní body SPGF gramatiky

Základní rozdíl v parsovací strategii SPGF oproti SRGS spočívá v tom, co je považováno za úspěšný konec parsování. SRGS implementace jako úspěšně dokončené parsování považuje takové situace, kde poskytnutý text byl pokryt daným pravidlem, obojí (pravidlo i text) vyčerpané (využité) beze zbytku od začátku do konce. Naproti tomu SPGF parser nekontroluje využití celého textu, pouze celého pravidla. Pokud tedy pravidlo pokryje pouze prvních několik slov a zbytek textu nebude „pasovat“, SRGS bude tuto situaci brát jako za neúspěch, zatímco SPGF to bude považovat za úspěch.

Dalším rozdílem mezi SRGS a SPGF je ten, že SRGS gramatiky poskytují pouze jeden vstupní bod (pravidlo označené klíčovým slovem `root`). SPGF oproti tomu umožňuje specifikovat více pravidel jako „veřejná“ klíčovým slovem `public` a tím tak gramatika

bude mít více vstupních bodů. Parser po načtení SPGF gramatiky vezme všechna pravidla označená jako public a pokusí se pomocí nich zpracovat poskytnutý text. Výsledky úspěšných parsování jsou pak vrácené jako výstupy v podobě derivačních stromů, se kterými je možné dále pracovat.

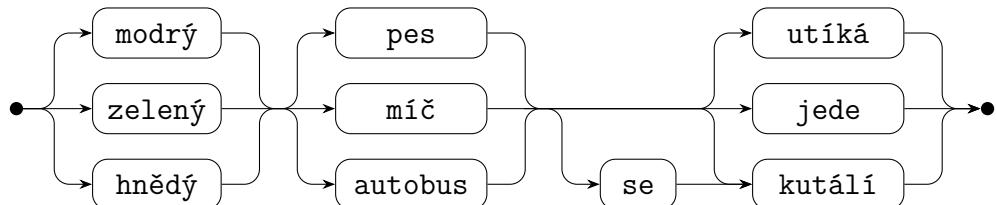
Pro příklad uvažujme vstupní text: „hnědý pes utíká pryč“ a gramatiku na Výpisu 3.

```

1  public $sentence = $color $object $action;
2  $color = modrý | zelený | hnědý;
3  $object = pes | míč | autobus;
4  $action = utíká | jede | [se] kutálí;
```

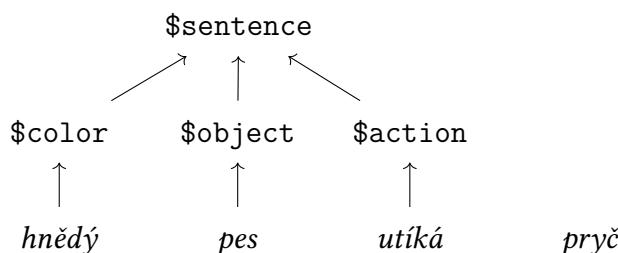
Výpis 3: Ukázková SPGF gramatika

Tuto gramatiku z Výpisu 3 lze zakreslit jako konečný automat na Obrázku 23.



Obrázek 23: Schéma gramatiky jako konečného automatu

Při zpracování textu by mohl vzniknout následující derivační strom, který je na Obrázku 24:



Obrázek 24: Ukázkový derivační strom

Jak je vidět, poslední slovo „pryč“ není součástí derivačního stromu, protože parsovací pravidlo \$sentence jej nedokáže pokrýt. V případě SRGS by tento derivační strom byl považovaný za neúspěch, zatímco SPGF vrátí zobrazený parsovací strom jako úspěšný výsledek.

6.2.4 Tagy v SPGF

K elementům je možné přiřadit tagy. Každý tag je libovolný text ohraničený složenými závorkami „{}“. Za každý element (s výjimkou speciálního pravidla \$VOID) je možné připsat libovolný počet těchto tagů, oddělených mezerami.

Tagy v SPGF nijak neovlivňují parsovací proces, slouží pouze jako dodatečná informace, kterou je možné k tokenům nebo jiným elementům přidružit.

V této práci jsou použité k tomu, aby obsahovaly dodatečnou sémantickou informaci. V definice tagů se SPGF liší od SRGS v tom, že tagy nejsou považované za samostatné elementy a není tak možné mít například pravidlo sestávající pouze z tagů nebo začínající tagem. Pro situace, kdy je žádoucí začít pravidlo nějakým tagem, je možné použít speciální pravidlo \$NULL a k němu přiřadit dané tagy.

Příklad gramatiky obsahující tagy je na Výpisu 4. K této ukázkové gramatice jsou pak na Obrázku 25 zobrazené tři derivační stromy pro promluvy „*pes spí*“, „*kapr plave*“ a „*delfín plave*“. Tagy jsou v derivačních stromech znázorněné zelenou barvou.

V tomto příkladu tagy přidávají informace o počtu nohou jednotlivých zvířat a jejich biologickou třídu. Dále je pomocí tagů označeno, která činnost je klidová a která v pohybu, a která část textu je podmět a která přísudek. Dále je možné pozorovat, že tagy jsou přiřazeny pouze k nejbližšímu předchozímu elementu a distribuuje se pouze přes závorky (sekvence).

```
1 public $věta = $zvíře {podmět} $činnost {přísudek};  
2 $zvíře = (pes {nohy=4} | delfín {nohy=0}) {savec} | kapr {nohy=0} {ryba};  
3 $činnost = spí {v klidu} | plave {pohyb};
```

Výpis 4: Ukázková SPGF gramatiky s tagy

Syntaxe SPGF je na Výpisu 5, detaily implementace je možné prozkoumat ve zdrojových kódech.

```
1 grammar ::= rule+  
2  
3 rule ::= ('public' ' ')? ruleName '=' ruleBody ';'  
4  
5 ruleName ::= '$' unicodeAlphanumeric+  
6  
7 ruleBody ::= ruleAlternative ('|' ruleAlternative)*
```



Obrázek 25: Derivační stromy s tagy

```

8 ruleAlternative ::= element (' ' element)*
9
10 element ::= token | ruleRef | sequence
11
12 sequence ::= bracedSequence | optionalSequence
13
14 bracedSequence ::= '(' bareSequence ')'
15
16 optionalSequence ::= '[' bareSequence ']'
17
18 bareSequence ::= ruleAlternative ('||' ruleAlternative)*
19
20 token ::= (unicodeAlphanumerical | '.' | '-' | '_' | ':' )+
21         repeatOperator? tag*
22
23 repeatOperator ::= '<' (( 'L' | 'G' | 'T' ) ':')?
24         (( digit+ '-' digit* ) | '*' | '+' | '?' ) '>'
25
26 ruleRef ::= '$' ruleName repeatOperator? tag*
27
28 tag ::= '{' [^}]+ '}'
29

```

Výpis 5: Syntaxe SPGF

6.2.5 Parsovací strategie a opakování v SPGF

Jak bylo zmíněno v sekci 6.2.2, v gramatice je možné ke každému elementu, s výjimkou speciálních pravidel \$END, \$BEGIN, \$VOID a \$NULL, přiřadit definici opakování.

Toto opakování udává, jaký je maximální a minimální počet bezprostředních opakování daného elementu během zpracování textu. Například pokud má daný token specifikováno, že jeho minimální počet opakování je 2 a maximální 3, tak v analyzovaném textu bude odpovídající slovo očekáváno dvakrát nebo třikrát po sobě.

Základní syntaxe je převzata ze standardu SRGS, tedy dvě přirozená čísla oddělená pomlčkou (bez mezer) a ohraničené špičatými závorkami: „`<min-max>`“. Pro pohodlnější zápis bylo definováno několik dalších alternativních způsobů zápisu, viz Tabulka 2. Pokud element žádné opakování nemá specifikované, implicitní hodnota je 1.

Syntax	Počet opakování		Poznámka
	min	max	
<code><m-n></code>	m	n	$m, n \in \mathbb{N}^0 \wedge m \leq n$
<code><m-></code>	m	∞	v implementaci max = $2^{32} - 1$
<code><m></code>	m	m	ekvivalentní s „ <code><m-m></code> “
<code><*></code>	0	∞	ekvivalentní s „ <code><0-></code> “
<code><+></code>	1	∞	ekvivalentní s „ <code><1-></code> “
<code><?></code>	0	1	ekvivalentní s „ <code><0-1></code> “

Tabulka 2: Možné způsoby zápisu opakování v SPGF

S opakováním použitím stejného elementu je úzce spojena i strategie parsování. Jedná se o způsob, jakým jsou řešené neurčité situace, kde je možné postupovat více způsoby. Tyto situace mohou vznikat právě v místech opakování elementů, nebo tam, kde je k dispozici více aplikovatelných alternativ.

Konkrétní příklad jedné z možných nejednoznačných situací (přímo převzatý ze SRGS specifikace [25]) je v Tabulace 3. Představuje situaci, kdy dvě alternativy mají různé tagy, ale stejně tokeny a tudíž není jasné, kterou z nich použít.

Expanze	<code>t1 {tag1} t1 {tag2} t2</code>
Vstupní řetězec	<code>t1</code>
Výstup (možnost 1)	<code>[t1, {tag1}]</code>
Výstup (možnost 2)	<code>[t1, {tag2}]</code>

Tabulka 3: Ukázka nejednoznačnosti SRGS

V existující implementaci SRGS, se kterou byly prováděné první experimenty, byl implementován pouze takzvaný „hladový algoritmus“, běžněji označovaný pod anglickým názvem *greedy matching*. Pro potřeby sémantické analýzy textu v této práci bylo ovšem zjištěno, že pouze *greedy* algoritmus nebude zcela postačovat.

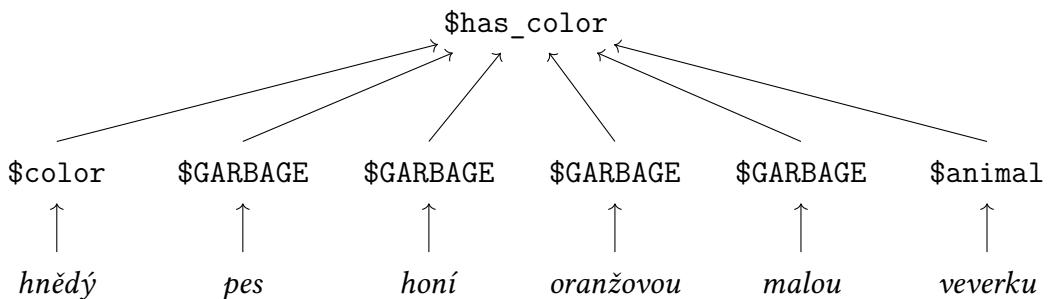
Příkladem takové situace by mohla být analýza textu

hnědý pes honí oranžovou malou veverku

s cílem zjistit vlastnosti objektů (v tomto případě zvířat), zde konkrétně jejich barvy. Pravidlo pro zachycení barvy zvířete by mohlo vypadat takto:

```
public $has_color = $color $GARBAGE<*> $animal;
```

kde \$color a \$animal jsou reference na pravidla akceptující konkrétní slova pro barvy a zvířata. Speciální pravidlo \$GARBAGE je zde definováno s operátorem opakování „*<*>*“, což lze chápat tak, že mezi samotnou barvou a zvířetem může být libovolný počet dalších slov, která nejsou podstatná. Při zpracování textu pak ale dochází k problému, že *greedy* algoritmus vrátí derivační strom zakreslený na Obrázku 26:



Obrázek 26: Derivační strom znázorňující problém s *greedy* algoritmem

Jak je na derivačním stromu na Obrázku 26 vidět, následující sémantická analýza by došla pravděpodobně k závěru, že v textu byla informace o hnědé veverce - což je chyba. Zde speciální pravidlo \$GARBAGE s neomezeným počtem opakování v kombinaci s *greedy* algoritmem způsobilo, že toto speciální pravidlo bylo „příliš agresivní“ a aktivovalo se i v případech, kdy by již bylo možné použít následující element.

Z tohoto důvody byla do systému implementována druhá strategie, která se běžně označuje jako *lazy matching*. Zatímco *greedy matching* se v každém okamžiku snaží posunout parsování co nejdále, tato *lazy* strategie se naopak snaží použít element co nejméněkrát.

Také by se dalo říci, že pokud parsovací algoritmus narazí na element e_1 následovaný elementem e_2 , pak:

- *greedy* strategie se vždy nejdříve pokusí e_1 použít (opakováně po sobě) a postoupí na e_2 až v moment, kdy již není možné dále opakovat e_1 ,
- *lazy* strategie se vždy pokusí nejdříve postoupit na element e_2 a vrátí se k e_1 pro opakování použití až v moment, kdy e_2 není možné použít.

Implementace *lazy* strategie umožnila zachytit sémantiku, kterou by bylo obtížné získat pomocí *greedy* algoritmu. Po několika dalších experimentech ovšem bylo zjištěno, že ani tato strategie nebude sama o sobě postačovat pro všechny potřeby sémantické analýzy. Hlavním problémem s *lazy* strategií bylo to, že v textech, kde bylo více možných vyjádření stejného typu, byla detekována vždy jen ta nejkratší.

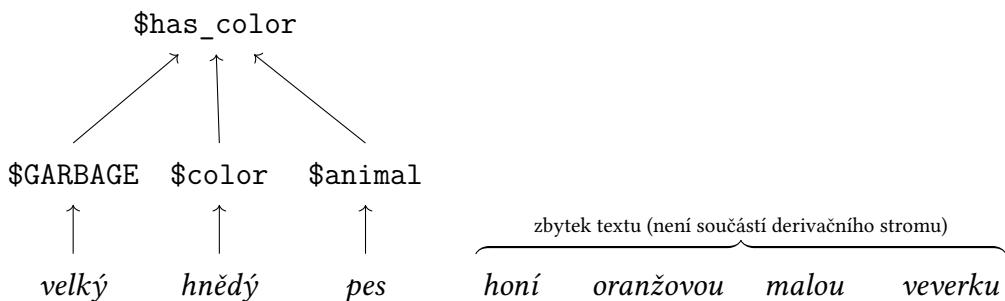
Například předchozí úloha určování barev zvířat se vstupním textem

velký hnědý pes honí oranžovou malou veverku

by mohla být zpracována SPGF pravidlem:

```
public $has_color = $GARBAGE<*> $color $GARBAGE<*> $animal;
```

V tomto případě bylo přidáno na začátek pravidla další $\$GARBAGE<*>$, které při použití *lazy* algoritmu způsobí, že text nemusí začínat přímo barvou, ale je možné hledat barvy a zvířata až později v textu. Derivační strom tohoto příkladu je zakreslen na Obrázku 27.



Obrázek 27: Derivační strom znázorňující problém s *lazy* algoritmem

Z derivačního stromu na Obrázku 27 je možné usoudit, že následující sémantická analýza by pravděpodobně detekovala, že v textu se nacházela informace o hnědém psovi - to je

správně. Nicméně je dále možné pozorovat, že v textu se vyskytovala také informace o oranžové veverce, kterou by systém nedokázal takto zachytit.

Z toho důvodu byl implementována ještě třetí strategie, která byla označena jako „*thorough matching*“. Myšlenka byla taková, že by bylo vhodné, aby algoritmus v místě nejednoznačnosti nemusel rozhodovat o tom, který postup je nejlepší, ale aby místo toho uvažoval všechny možnosti. Tento přístup se tedy od předchozích dvou liší v tom, že může vrátit pro jeden vstup více výstupů.

Základní koncept *thorough* strategie spočívá v tom, že v místě, kde by bylo možné po stupovat více způsoby, je řešení rozděleno do více paralelních větví (jedna pro každou možnost) a každá větev je dále zpracovávána nezávisle. Tímto způsobem se rekursivně vytváří strom možných řešení, jehož listy odpovídají finálním derivačním stromům. Po ukončení všech těchto paralelních parsování jsou úspěšné výsledky vráceny jako seznam derivačních stromů.

Na výstupu tedy bude množina všech možných způsobů, jakým bylo možné derivační strom pro dané pravidlo a vstupní text sestavit. To následně zaručí, že během zpracování textu nebyla opomenuta žádná sémantika (za předpokladu, že jsou expertem správně sestavena parsovací pravidla).

Například pro předchozí úlohu určování barev zvířat se vstupním textem

velký hnědý pes honí oranžovou malou veverku

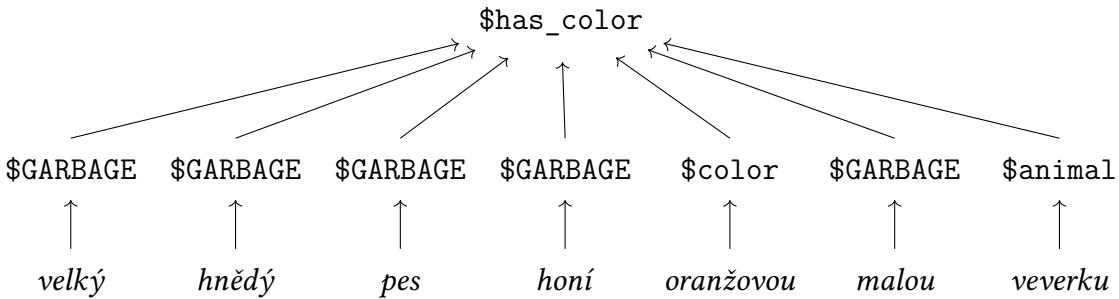
a SPGF pravidlem:

```
public $has_color = $GARBAGE<*> $color $GARBAGE<*> $animal;
```

by výstupem byly 3 derivační stromy. Dva z nich by byly shodné s výstupem *greedy* a *lazy* strategie (viz Obrázky 26 a 27), třetí (nový) derivační strom je na Obrázku 28.

Celkově by tedy z tohoto výstupu bylo možné získat informace o tom, že v textu byla hnědá veverka, hnědý pes a oranžová veverka. Dvě z těchto informací jsou správně, pouze informace o hnědé veverce je chybná, ta se v textu nevyskytla.

Jedná se o jednu z nevýhod *thorough* přístupu, že bude v textu hledat i takové informace, které z něj neplynou. Tento problém byl vyřešen následnou filtrace získaných entit oproti referenčnímu popisu. Detailní popis tohoto filtrování bude popsán v kapitole 6.3.1.



Obrázek 28: Derivační stromy z *thorough* algoritmu

Tímto způsobem je pak v testovaném popisu pouze podmnožina referenčního popisu. Jedná se o jednoduchý a efektivní způsob řešení falešně detekovaných sémantických informací. Také ale bohužel představuje omezení v tom smyslu, že neumožňuje detekovat sémantiku, která se v textu skutečně nacházela, ale nebyla zanesena do referenčního popisu - to klade očekávání na kvalitu referenčního popisu od experta. Sestavení nějakého složitějšího algoritmu pro detekci falešně extrahované sémantiky představuje jedno z možných rozšíření do budoucna.

Další nevýhodou *thorough* strategie je její výpočetní náročnost. Vzhledem k rekurzivnímu charakteru algoritmu a způsobu vytváření nových paralelních větví se naskytá riziko kombinatorické exploze počtu paralelních větví. Během testování se však ukázalo, že pro moderní výpočetní techniku nepředstavují vstupy o velikosti desítek až stovek tokenů žádný problém. Detailnější analýza výpočetní a paměťové náročnosti pro větší vstupy je nad rámec zadání a byla ponechána pro budoucí práce.

Posledním problémem, který bylo potřeba vyřešit v rámci parsovacích strategií, byl způsob, jakým strategie volit a přepínat. V první verzi parseru se jednalo o globální přepínač, kdy bylo možné před samotným parsováním zvolit, jaká strategie má být použita. Bylo ale zjištěno, že pro některé situace by bylo výhodné, aby bylo možné změnit výchozí strategii pro konkrétní pravidla nebo jejich části.

Například při zpracovávání číselných údajů je vhodná *greedy* strategie. Je totiž potřeba, aby posloupnost sobě jdoucí slov, které reprezentující číselný údaj, byla brána jako jeden celek.

Jiný příklad pak může být již zmíněný začátek pravidla `$GARBAGE<*>`, který libovolný počet prvních slov považovat za výplň a hledat tak význam až později v textu.

Pro tuto situaci je naopak *greedy* algoritmus zcela nevhodný, protože by celý text byl zpracován hned tímto počátkem. Zde je vhodná *lazy* strategie, která ve své podstatě bude dávat přednost následujícím elementům v pravidle a tento počátek bude používat až když zbytek pravidla selže.

Z těchto důvodů byla rozšířena syntaxe pro operátor definující opakování. Před samotné hodnoty opakování (tedy hned za otevírací špičatou závorku „<“) je možné napsat jedno z písmen L, G, T následované dvojtečkou a poté zbytek definice opakování. Tato písmena odpovídají jednotlivým strategiím (*lazy*, *greedy*, *thorough*) a umožňují tak vynutit danou strategii pro dané pravidlo nebo jeho část. Ve výsledku tak operátor opakování může vypadat například „*<G:1-3>*“, „*<L:**“, nebo „*<T:2->*“.

6.3 Tvorba testovaného popisu

Po implementaci systému pro zpracování přirozeného textu pomocí bezkontextových gramatik bylo potřeba získané derivační stromy nějakým způsobem zpracovat a vytvořit z nich testovaný popis, který by bylo možné srovnat s referenčním popisem od experta.

Zpracování derivačních stromů obecně představuje konflikt mezi obtížností automatického zpracování a omezením na strukturu. Striktně daná forma bude lépe zpracovatelná programově, ale může být obtížné zachytit všechny potřebné informace v dané struktuře. Naopak více volná forma bude umožňovat snadnější zachycení různých typů informace, ale bude obtížnější sestavit algoritmus, který by dokázal spolehlivě zpracovat předem neznámou strukturu.

Vzhledem k podobě testovaného a referenčního popisu, které byly popsány v kapitole 5, byl vytvořen algoritmus, který prochází získané derivační stromy a ukládá navštívené uzly s tagy do seznamu. Z tohoto seznamu tagů pak určuje, jaká sémantická informace je v derivačním stromu obsažena.

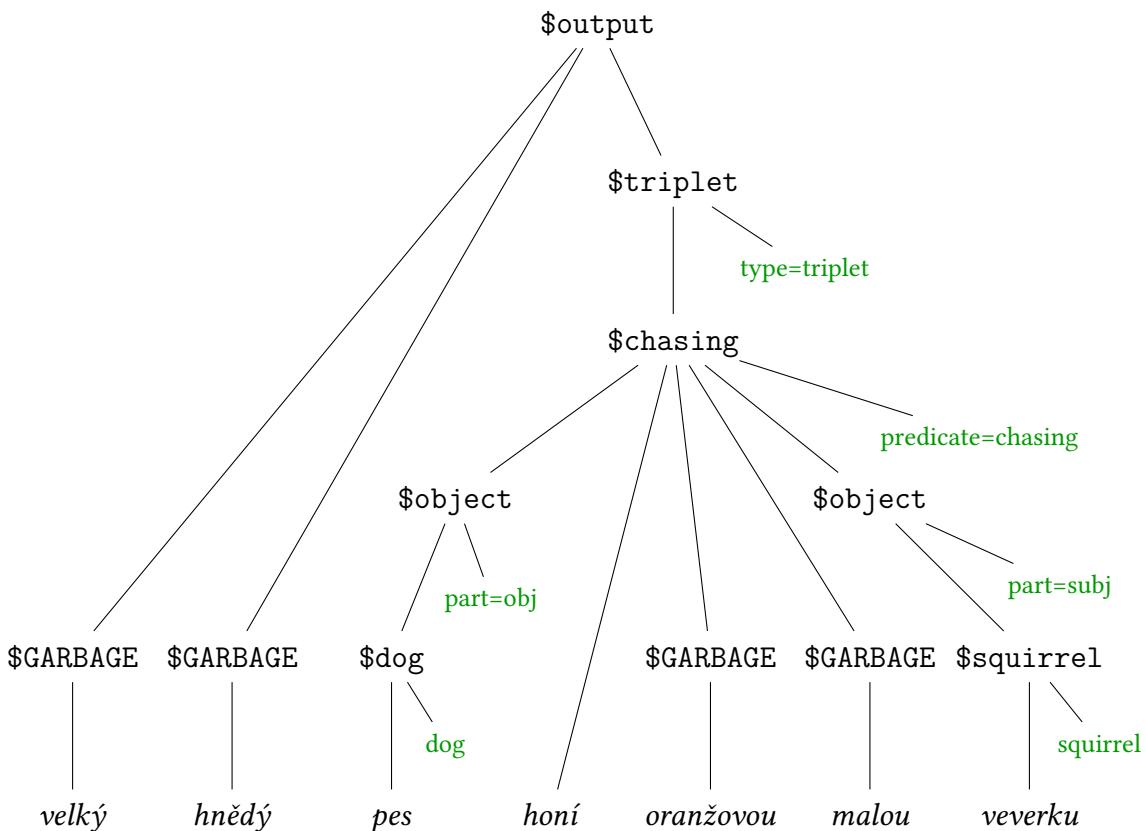
Zde navržený přístup klade požadavek na způsob, jakým jsou sestavené SPGF gramatiky, protože předpokládá, že výstupní derivační stromy budou mít takovou strukturu, aby při jejich procházení byly nalezené tagy v očekávaném pořadí. Neklade tedy omezení přímo na strukturu samotných derivačních stromů, ale pouze na relativní rozmístění tagů uvnitř derivačního stromu.

6.3.1 Algoritmus zpracování derivačních stromů

V první fázi algoritmus projde získaný derivační strom T strategií depth-first-post-order (dfpo) [27] a všechny navštívené tagy t_i ukládá do seznamu. Index i značí index tagu odpovídající jeho pořadí při dfpo procházení stromu T .

Příklad takového stromu je na Obrázku 29. Zde je možné vidět, že ve skutečnosti tagy jsou realizované jako samostatné uzly, které následují vždy jako poslední (zleva doprava) uzel ke svému rodičovskému uzlu. Při dfpo zpracování pak z tohoto stromu vznikne následující posloupnost tagů:

[dog, part=obj, squirrel, part=subj, predicate=chasing, type=triplet]



Obrázek 29: Příklad derivačního stromu s tagy pro zpracování

Získaný seznam tagů je postupně porovnáván s jednotlivými vzory, zda odpovídá nějaké předem definované sémantické informaci. Rozlišované jsou tři typy informací, které odpovídají struktuře referenčního a testovaného popisu:

1. Objekty:

Pro objekty stačí pouze identifikátor toho, že se jedná o objekt, a poté samotný název objektu. Očekávaný seznam tagů tedy musí mít následující strukturu:

[*name*, type=object]

kde *name* je libovolný řetězec, který bude považován za název objektu.

Například při zpracování derivačního stromu, ze kterého bude získána posloupnost tagů [dog, type=object], bude extrahován objekt „dog“, který bude poté přidán do testovaného popisu.

2. Atributy:

Pro kompletní určení atributu je třeba identifikátor toho, že se jedná o atribut, pak objekt, kterému je atribut přiřazen, následně název atributu a jeho hodnota. Pro extrakci atributů byly definovány dvě různé struktury, kterých může seznam tagů nabývat:

[*o_name*, part=obj, *a_val*, *a_name*, part=attr, type=attribute]
[*a_val*, *a_name*, part=attr, *o_name*, part=obj, type=attribute]

kde *o_name*, *a_val* a *a_name* jsou libovolné řetězce, které budou považované za název objektu, hodnotu atributu a název atributu.

Dvě různá očekávaná pořadí tagů vycházejí z toho, že v běžné mluvě lze specifikovat vlastnost objektu před objektem (např. „hnědý pes“) i po něm (např. „pes je hnědý“).

Například při zpracování derivačního stromu, ze kterého budou získány tagy

[dog, part=obj, brown, color, part=attr, type=attribute],

bude extrahován atribut „dog: color = brown“, který bude poté přidán do testovaného popisu.

3. Vazby mezi objekty:

Pro kompletní určení vazby mezi dvěma objekty je potřeba znát názvy obou objektů a název vazby mezi nimi. Pro extrakci vazeb mezi objekty je očekáváno následující pořadí tagů:

[*from*, part=obj, *to*, part=subj, predicate=pred, type=triplet]

kde *from*, *to* a *pred* jsou libovolné řetězce, které budou považované za název zdrojového objektu, cílového objektu a název vazby.

Například při zpracování derivačního stromu, ze kterého budou získány tagy

```
[dog, part=obj, squirrel, part=subj, predicate=chasing,  
type=triplet],
```

bude extrahován triplet „*dog chasing squirrel*“, který bude poté přidán do testovaného popisu.

Po extrakci objektů, atributů a tripletů jsou tyto získané sémantické informace profiltrovány, za cílem odstranění falešně detekovaných entit. Jak již bylo dříve zmíněno, filtrování probíhá tím způsobem, že pokud daný objekt, atribut, nebo triplet nejsou v referenčním popisu, jsou ze seznamu odstraněné. Výjimku tvoří atributy, kde je pouze chybná hodnota - tyto atributy jsou v seznamu ponechané a jsou uvažované v hodnocení, jak bylo popsáno v kapitole 5.4.

6.3.2 Číslování objektů

Jeden z problémů, na který byl během implementace objeven, je absence číslování objektů. Jak bylo popsáno v sekci 6.1, objekty mohou mít v rámci svého názvu číslování, kvůli jednoznačné rozlišitelnosti. Nelze předpokládat, že by se tato číselná informace jakkoli vyskytovala v přirozeném popisu. Naskýtá se tedy otázka, jak z přirozeného textu získat informaci o tom, o který objekt se konkrétně jedná (za předpokladu, že daný objekt je v referenčním popise číslován).

Pro tento účel rozlišení číslovaných objektů byl sestaven doplňkový algoritmus, který využívá unikátních vlastností objektů a jejich vazeb. Hlavní myšlenka spočívá v tom, že pokud se názvy vícero objektů liší pouze číslováním, pak je možné využít jejich atributy a vlastnosti k rozlišení, který objekt je který.

Například pokud ve scéně budou dva stromy „strom #1“ a „strom #2“ a právě jeden z nich bude součástí tripletu „veverka → šplhá_po → strom #1“, pak lze například z textu:

„na obrázku vidím veverku, co šplhá po stromě“

usoudit, že se jedná o „strom #1“, protože „strom #2“ nemůže být součástí daného tripletu. Je zde opět použit optimistický přístup, který předpokládá, že uživatel skutečně měl na mysli ten daný objekt (v tomto případě strom), který se vyskytuje v detekovaném atributu či tripletu.

Na tomto principu byl tedy implementován algoritmus, který iterativně prochází extrahované informace z derivačních stromů a postupně hledá, zda jsou v datech nějaké objekty, které by šly takto specifikovat. Pokud je nějaký takový objekt nalezen, pak je mu přidáno příslušné číslování nejen v daném atributu nebo tripletu, ale i ve všech ostatních extrahovaných datech, jelikož si vnitřní implementace udržuje informace o tom, ze které části textu který kousek sémantiky pochází.

Dále se může stát, že specifikování jednoho objektu umožní následovné specifikování jiného objektu, protože dojde k eliminaci ostatních možností. Tento iterativní algoritmus tedy prochází opakovaně všechna extrahovaná data až do chvíle, než je celý průchod beze změny - v ten okamžik proces končí.

6.4 Hodnotící algoritmus a ztrátová tabulka

Po získání testovaného popisu je potřeba jej srovnat s referenčním popisem a ohodnotit míru jejich shody. K tomuto účelu byl navržen a implementován hodnotící algoritmus, který byl detailně popsán v kapitole 5.4.

Vstupem je referenční popis získaný od experta, testovaný popis, jehož získání je popsáno v předchozích kapitolách a ztrátová tabulka, která udává závažnost různých chyb. Samotná ztrátová tabulka odpovídá svým formátem asociativnímu poli (též hash-tabulka nebo slovník), s předem danou strukturou. Jelikož je SPGF systém citlivý na velikost písmen a interpunkční znaménka, jsou texty před vstupem normalizované a zpracovávané větu po větě, výsledky pro jednotlivé věty jsou na konci sloučené.

Jak již bylo řečeno v kapitole 5.4, ztrátovou tabulkou sestavuje expert, stejně jako referenční popis a SPGF gramatiky. Z toho důvodu bylo nutné zvolit nějaký formát, který by byl čitelný člověkem a zároveň zpracovatelný strojem. Zvolen byl opět formát JSON, který představuje dobrý kompromis mezi oběma faktory.

Ukázka ztrátové tabulky je na Výpisu 6.

```

1  {
2      "missing_objects": 3,
3      "missing_attributes": 1,
4      "missing_triplets": 2,
5      "numberless_penalty": 0.5,
6      "wrong_values": 2,
7      "missing_objects_override": [
8          ["person", 5],
9          ["animal", 4],
10         ["environment", 1]
11     ],
12     "missing_attributes_override": [
13         ["action", 1.5],
14         ["state", 0.5]
15     ],
16     "missing_triplets_override": [["falling into", 5]],
17     "wrong_values_override": [
18         {
19             "attribute": "color",
20             "default": 0.5,
21             "overrides": [
22                 [[["white", "yellow", "pink"], 0.3],
23                  [[["red", "blue", "green", "yellow"], 0.8]
24                 ]
25             },
26             {
27                 "attribute": "action",
28                 "default": 1.5,
29                 "overrides": [
30                     [[["sitting", "reading"], 0.8],
31                     [[["fishing", "sitting"], 0.7]
32                 ]
33             }
34         }
35     ]
}

```

Výpis 6: Ztrátová tabulka použitá při testování

Význam jednotlivých položek byl detailně popsán v sekci 5.4, proto zde bude pouze stručně shrnuto, který klíč odpovídá jakému významu:

- `missing_objects` → výchozí ztráta pro chybějící objekty,
- `missing_attributes` → výchozí ztráta pro chybějící atributy,
- `missing_triplets` → výchozí ztráta pro chybějící triplety (vazby),

- `numberless_penalty` → ztráta pro informace obsahující objekty s chybějícím číslováním (viz sekce 6.3.2),
- `wrong_value` → výchozí ztráta pro atributy s chybnou hodnotou,
- `missing_objects_override` → seznam konkrétních ztrát pro chybějící objekty s daným tagem,
- `missing_attributes_override` → seznam konkrétních ztrát pro chybějící atributy (podle názvu atributu),
- `missing_triplets_override` → seznam konkrétních ztrát pro chybějící triplety (podle názvu vazby),
- `wrong_values_override` → seznam konkrétních ztrát pro daný atribut a případně i pro záměnu konkrétních hodnot.

Finálním výstupem hodnotícího algoritmu je množina označených číselných hodnot, získaných akumulací ztrát pro různé druhy chyb. Přirozenou datovou strukturou, která by odpovídala tomuto formátu, bylo asociativní pole.

Příkladem takového výstupu může být Výpis 7.

```

1 Score {
2     missing_objects: 125.5,
3     missing_attributes: 79.5,
4     missing_triplets: 164.0,
5     wrong_values: 0.0,
6     grouped_missing_objects: {
7         "group": 21.0,
8         "item": 24.0,
9         "person": 5.5,
10        "animal": 52.0,
11        "clothing": 24.0,
12        "environment": 17.0,
13        "background": 15.0,
14        ...
15    },
16    grouped_missing_attributes: {
17        "hairstyle": 1.0,
18        "pattern": 1.0,
19        "action": 5.0,
20        "color": 61.0,
21        "facial expression": 0.5,
22        ...
23    },
24    grouped_missing_triplets: {
25        "reading": 0.0,
26        "climbing": 2.0,
27        "throwing": 2.0,
28        "playing with": 6.0,
29        "chasing": 2.0,
30        "swimming in": 2.0,
31        "following": 10.0,
32        "running away from": 2.0,
33        "has child object": 70.0,
34        "sitting on": 8.0,
35        "wearing": 20.0,
36        ...
37    },
38    grouped_wrong_values: {
39        "state": 0.0,
40        "facial expression": 0.0,
41        "pattern": 0.0,
42        "color": 0.0,
43        "count": 2.0,
44        ...
45    },
46 }

```

Výpis 7: Ukázka výpisu hodnocení (zkráceno)

7 Vyhodnocení

Posledním krokem bylo ověření funkčnosti navrženého a implementovaného systému. K tomu bylo potřeba získat vhodná data, konkrétně obrázek a k němu přirozené popisy, pokud možnost v textové podobě. Tato data byla získána z projektu TAČR SIGMA_DC3 „*Telemedicínské samovyšetření řeči a paměti pro rychlou detekci kognitivních poruch metodami strojového učení*“.

Popisovanou scénou byla kresba zobrazená na Obrázku 19. K tomuto obrázku bylo k dispozici několik přepisů, které byly použité pro účely testování.

Referenční popis daného obrázku, stejně jako SPGF gramatika a ztrátová tabulka, byly vytvořeny autorem této práce a nikoli expertem z oboru, odkud pocházejí nasbíraná data.

Hodnoty ve ztrátové tabulce je tedy třeba chápat jako ilustrační a ve výsledcích je možné srovnávat pouze relativní ztráty, nelze dělat smysluplné závěry z konkrétních hodnot. To však nepředstavuje problém, protože cílem zde bylo ověřit funkčnost navrženého systému, na což ilustrační hodnoty postačují.

Vytvořený referenční popis Obrázku 19 obsahuje 69 objektů a 37 vazeb mezi nimi a je možné jej prozkoumat spolu se sestavenou SPGF gramatikou u zdrojových kódů. Ztrátová tabulka použitá při testování je na Výpisu 6.

V Tabulce 4 je několik vět z prvního přepisu, k nimž je vypsána množina extrahovaných objektů \mathcal{O} , množina extrahovaných atributů \mathcal{A} a množina extrahovaných vazeb \mathcal{V} . Jak je možné na těchto ukázkách vidět, tak navržený systém je schopen z přirozeného textu extrahovat očekávané sémantické informace. Dalším pozitivním faktem je to, že všechny extrahované informace jsou skutečně obsažené v původních promluvách a tudíž lze usoudit, že filtrační algoritmus pro odstranění falešných detekcí funguje správně.

Věta 1	<i>na břehu rybníka je pět osob a několik zvířat</i>		
Extrakty	Objekty \mathcal{O}_1	Atributy \mathcal{A}_1	Vazby \mathcal{V}_1
	water people animals	people: count = 5	—
Věta 2	<i>vlevo sedí žena na lehátku čte si knihu pod slunečníkem</i>		
Extrakty	Objekty \mathcal{O}_2	Atributy \mathcal{A}_2	Vazby \mathcal{V}_2
	woman lounger book parasol	—	woman reading book
Věta 3	<i>vedle sebe má rozprostřenou piknikovou deku s košíkem a s jídlem na piknik</i>		
Extrakty	Objekty \mathcal{O}_3	Atributy \mathcal{A}_3	Vazby \mathcal{V}_3
	blanket basket	—	—
Věta 4	<i>je to děvče s culíkem a v plavkách a chlapec s kšiltovkou a pruhovaným tričkem a kráťasy</i>		
Extrakty	Objekty \mathcal{O}_4	Atributy \mathcal{A}_4	Vazby \mathcal{V}_4
	girl, shorts, swimsuit #1 boy, cap tshirt	girl: hairstyle = ponytail tshirt: pattern = striped	girl wearing swimsuit #1 boy wearing cap

Tabulka 4: Příklad konkrétních vět z nich extrahovaných entit

V Tabulce 4 lze dále pozorovat, že nejpočetnější kategorií ve větách jsou objekty, jejichž zachycení je nejjednodušší a především jich věty také obsahují nejvíce. Dále lze pozorovat, že detekovaný objekt *swimsuit #1* ve čtvrté větě má u sebe přiřazené číslování (v popise se vyskytují dvoje plavky, jedny červené na dívce v pozadí a jedny bílé na ženě na lehátku). To je významné z toho důvodu, že to značí schopnost systému rozlišit objekty stejného typu, jak bylo popsáno v sekci 6.3.2.

Předpokladem pro kvalitní výstup je kvalitní referenční popis a SPGF gramatika. Oba tyto faktory jsou zcela závislé na expertovi, který vytváří jak referenční popis, tak gramatiku použitou pro extrakci sémantiky z textu.

To může představovat problém ve smyslu vysokých počátečních nákladů, protože tvorba referenčního popisu a SPGF gramatiky vyžaduje znalost zde navrženého formátu a může být pro lidského experta časově náročná. Naopak výhodou zvoleného přístupu je ale možnost upravovat a vylepšovat jak referenční popis, tak SPGF gramatiku i v budoucnu, protože oba tyto vstupy jsou navržené tak, aby byly čitelné člověkem a bylo možné je v případě potřeby kdykoli upravit.

Dále tento expertní přístup a manuální správa vstupních dat umožňuje přesnou kontrolu, na rozdíl od jiných přístupů, například neuronových sítí, jejichž funkce je spíše black-box a potřeba provést detailní změny nebo přímo kontrolovat některé aspekty procesu jsou velmi obtížné.

Celé přepisy použité pro další testování a získání ilustračních výstupů jsou následující:

1. Přepis 1:

Na břehu rybníka je pět osob a několik zvířat. Vlevo sedí žena na lehátku, čte si knihu pod slunečníkem. Vedle sebe má rozprostřenou piknikovou deku s košíkem a s jídlem na piknik. Opodál si hrají děti s míčem, hází si. Je to dívče s culíkem a v plavkách a chlapec s kšiltovkou a pruhovaným tričkem a kraťasy. Na pravé straně sedí rybář, který chytá ryby, má nahrozený prut a vedle sebe podběrák. Před ním skáče žába z kamene do rybníka. Mezi ženou a rybářem je uprostřed dítě, které vypadá, že padá do rybníka, doufám, že ho někdo zachrání, že si ho někdo všimne. A vpředu vpravo muž plave.

2. Přepis 2:

Vidím koupající se lidi. Pán plave kraula, malé dítě na břehu vypadá, že zakoplo a zrovna padá do vody. Na kraji u rákosí vpravo sedí pán s čepicí, s kostkovanou košílí a nějakou vestou chytá asi ryby. Na druhé straně, na levé straně obrázku je paní, která pod deštníkem si čte nějakou zajímavou knížku, usmívá se. Na pozadí si děti házejí míčem. Úplně vzádu vpravo nahore běhá pejsek, kterej honí neverku, která před ním leze, utíká na strom. Celkově svítí sluníčko, na nebi letí letadlo. Vidím tady čtyři stromy. S tím, že na vodní hladině ještě se tady vyskytuje zvířata jako je žába, kapr, kachna s malými kachňátky. Paní vypadá, že má dovolenou, protože kromě toho, že si čte knihu tak tam má ještě připravenou deku, nějaký piknik. Vypadá to, že má něco dobrého.

3. Přepis 3:

Na obrázku vidím řeku a pobřeží, pobřeží řeky. V řece plavou kačeny. Matka s pěti malými káčátky. Vyskakuje asi ryba. Je v ní plavec. Padá do ní dítě malé. Maminka možná sedí na lehátku, čte si knížku, takže si ho asi moc nevšímá. Vedle maminky leží deka. Na ní leží košík, je to jakýsi piknik. Má tam láhev s pitím, skleničku. Dítě tam má nějaké dvě, dva kyblíčky, lopatičku a vypadá to, dítě má na sobě možná jednorázovou plenu, že si asi hrálo s kyblíčkem a lopatičkou a teď nějakým záhadným způsobem padá do vody. Upadlo, maminka se nedívá. Takže to může být asi problém. Vedle sedí rybář, který se snaží lovit ryby. Vedle něho je kámen, z kterého skáče žába, asi ji vyplašil. Rybář se usmívá, což je zvláštní, protože tam vedle něho padá to dítě do vody, to mi přijde zvláštní. Potom za, v pozadí si hrají dvě takové už vzrostlejší děti, holka v plavkách, kluk v tričko, šortky.

4. Přepis 4 (pacient):

Tady si děti házej míčem. Pes honí neverku. Rybář chytá ryby. Tajdle plave řákej. Tamhle si čte paní. Kačeny tajdle jsou. Jsou tam stromy, no a sluníčko. Tady dítě se namáčí. Tam je žába, tajdle kapr. Tamhle letadlo, ptáci.

5. Přepis 5 (pacient):

Takže hrajou volejbal. Pes honí kočku. Tady se někdo opaluje. Tady si děcko hraje ve vodě. Rybář. Tady plave někdo. Tady je ryba, kachna s káčaty. To je asi všechno. Žába. Já bych řekla, že to je všechno. Tady řákej pták na stromě. Svítí sluníčko. Letí letadlo.

Dále bylo pro srovnání vygenerováno ještě několik „umělých“ popisu fiktivních obrázků, které jsou záměrně sémanticky zcela rozdílné. Generování těchto popisů bylo realizováno pomocí volně dostupného softwaru ChatGPT² se vstupní úlohou vygenerovat popis obrázku, který zobrazuje scénu na farmě, poté scénu v lese a nakonec scénu v rušném městě.

²<https://chatgpt.com/>

Tyto uměle generované popisy jsou následující:

1. (GPT3.5) Popis 1:

prompt: „vygeneruj popis obrázku, který zobrazuje scénu na farmě“

Na obrázku farmy je široká zelená pastvina s hejny ovcí, které se volně pasou. Vedle pastviny stojí starý dřevěný stodola se šindelovou střechou, která má patrné známky stáří. U stodoly je starší traktor s přívěsem, který je zřejmě používán k práci na polích. Na obzoru je vidět nízký řádek hor a nebe je jasné s několika obláčky. Slunce svítí a vytváří teplé odstíny na krajinu. Okolo farmy jsou roztroušené stromy a keře. Celkově působí scéna klidně a idylicky, zobrazující život na venkově.

2. (GPT3.5) Popis 2:

prompt: „vygeneruj popis obrázku, který zobrazuje scénu lese“

Obrázek zachycuje hustý les v letním období. V popředí jsou vidět vysoké stromy s bujným listovím, které tvoří stíněnou a chladivou atmosféru. Mezi stromy probíhá jemně vyšlapaná cesta, která se ztrácí dál do lesa. Na okrajích cesty rostou divoce rostoucí květiny a kapradiny. Sluneční paprsky pronikají mezi listy stromů, vytvářejíce hravé světelné skvrny na zemi. Vzduch je naplněn zvuky lesa, šeleštěním listí, štěkotem ptáků a občasným šustěním zvířat.

3. (GPT3.5) Popis 3:

prompt: „vygeneruj popis obrázku, který zobrazuje rušné město“

Obrázek zobrazuje rušné město plné života a pohybu. V popředí jsou rušné ulice s různými lidmi, kteří spěchají do práce nebo nakupují. Na chodnících se proplétají chodci a cyklisté. Všude kolem je mnoho automobilů, které se pomalu pohybují vpřed v hustém dopravním provozu. Vedle ulic jsou vysoké budovy s různými obchody, kancelářemi a byty. Některé z těchto budov mají zajímavou architekturu a moderní design, což přidává pestrost do urbanistického prostředí. Z vrcholů budov se tyčí reklamní panely a billboardy, které svítí vsemi barvami a lákají pozornost chodců. Na obzoru jsou vidět další výškové budovy a mrakodrapy, které vytvářejí panoramatickou linii města. Vzduch je plný zvuků - hluk automobilů, hovorů lidí, zvuků pouliční hudby a zvonění tramvají. Hlavní dominantou města může být ikonická stavba nebo památník, který zaujmě veškerou pozornost. Celkově působí obrázek jako dynamická a živá scéna, která představuje hektický rytmus moderního městského života.

Pro výše zmíněné přepisy bylo podle ztrátové tabulky (viz Výpis 6) vypočteno hodnocení, jehož část (zkráceno kvůli délce) je vidět v Tabulce 5. Kompletní výpis je možné prozkoumat spolu se zdrojovými kódy online.

Z Tabulky 5 je možné vidět, že pro jednotlivé testované popisy se výstupní ztráty skutečně liší, obzvláště v prvních několika hodnotách, které udávají ztráty pro celkově všechny chybějící objekty, atributy a triplety. Jak již bylo zmíněno dříve, referenční popis obrázku a ztrátová tabulka byly sestavené ne-expertem, především pro účely testování. Interpretace výsledků je tudíž obtížná, protože se může podstatně lišit od reálného nasazení, kde se předpokládá, že referenční popis a ztrátová tabulka budou pečlivě sestavené expertem z oboru.

I přes tyto komplikace je ale možné učinit několik pozorování:

1. Popisy č. 4 a 5, které jsou ve své celkové délce výrazně kratší a používají velmi strohé nerozvité věty, mají vyšší ztrátu pro chybějící vazby (triplety). To odpovídá tomu, že v těchto popisech je většina obsahu pouze vyjmenovávání objektů z obrázku a téměř žádná pozornost není věnována jejich vzájemným vazbám a vztahům mezi nimi.
2. Dalším znatelným rozdílem ve výstupních hodnotách, který je možné sledovat, je značný rozdíl ztráty pro chybějící objekty s tagem „*person*“. To odpovídá faktu, že zmínky osob se v jednotlivých popisech skutečně výrazně liší, konkrétně pak popisy 4 a 5 zmiňují omezené množství osob z obrázku ve srovnání s prvními třemi popisy.
3. Obdobně je možné v Tabulce 5 pozorovat, že přepisy č. 1 a 5 mají relativně vysoké ztráty pro objekty s chybějícím tagem „*animal*“. Opět to odpovídá tomu, že tyto texty neobsahují tolik zmínek o zvířatech v obrázku. Kontrastem může být text č. 2, který má tuto ztrátu nejnižší a skutečně je možné se přesvědčit, že v tomto textu je značná pozornost věnována popisu zvířat.
4. Dále je možné pozorovat, že vysoké hodnoty některých ztrát jsou ve sloupcích pro uměle generované popisy **G1**, **G2**, **G3**. Především je možné pozorovat výrazně vyšší ztráty pro chybějící objekty a atributy, způsobené patrně tím, že v příslušných textech se nevyskytují popisy sledovaných objektů.

Typ ztráty		Číslo přepisu							
		1	2	3	4	5	G1	G2	G3
chybějící objekty	chybějící objekty	125.5	97.0	108.05	141.0	161.0	197.0	191.0	197.0
	chybějící atributy	79.0	76.0	77.5	80.5	79.0	80.5	82.0	84.0
	chybějící triplety	164.0	156.5	164.00	173.0	175.0	217.0	217.0	217.0
	atributy s chybnou hodnotou	2.0	0.0	4.7	0.0	0.0	0.0	0.0	0.0
chybějící obj. s tagem	animal	52.0	18.5	26.5	28.0	40.0	60.0	52.0	60.0
	background	15.0	8.5	15.0	6.0	10.0	11.0	13.0	15.0
	clothing	24.0	24.0	27.0	36.0	36.0	36.0	36.0	36.0
	environment	17.0	11.5	16.0	15.0	14.0	13.0	15.0	18.0
	group	21.0	16.0	21.0	17.0	30.0	30.0	26.0	25.0
	item	24.0	33.0	30.0	42.0	45.0	45.0	45.0	45.0
	person	5.5	10.0	5.5	20.0	26.0	40.0	40.0	35.0
chybějící atr. s tagem	action	5.0	4.5	5.0	6.0	5.5	7.5	7.5	7.5
	color	61.0	56.5	57.5	59.0	59.0	58.0	59.0	61.0
	count	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
	facial expression	0.5	1.0	0.5	1.0	0.5	1.0	1.0	1.0
	hairstyle	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
	pattern	1.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
	style	0.5	0.0	0.5	0.5	0.0	0.0	0.5	0.5
chybějící vazby typu	catching	4.0	4.0	4.0	4.0	4.0	4.0	4.0	4.0
	chasing	2.0	0.0	2.0	0.0	2.0	2.0	2.0	2.0
	climbing	2.0	0.0	2.0	2.0	2.0	2.0	2.0	2.0
	falling into	0.0	0.0	0.0	5.0	5.0	5.0	5.0	5.0
	following	10.0	10.0	10.0	10.0	10.0	10.0	10.0	10.0
	playing with	6.0	6.0	6.0	6.0	6.0	6.0	6.0	6.0
	reading	0.0	0.0	0.0	2.0	2.0	2.0	2.0	2.0
	sitting on	8.0	8.0	6.0	8.0	8.0	8.0	8.0	8.0
	swimming in	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0
	wearing	20.0	24.0	22.0	24.0	24.0	24.0	24.0	24.0
chybné atr.	action	0.0	0.0	0.7	0.0	0.0	0.0	0.0	0.0
	color	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	count	2.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0
	state	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Tabulka 5: Ztrátové hodnoty testovaných přepisů (zkráceno)

Na závěr bylo použito všech 55 dostupných přepisů z projektu TAČR SIGMA_DC3 „*Telemedicínské samovyšetření řeči a paměti pro rychlou detekci kognitivních poruch metodami strojového učení*“. Jedná se o popisy ilustrace zobrazené na Obrázku 19, které byly přepsány do textové podoby.

V těchto datech se vyskytovalo 30 popisů nasbíraných od osob, u kterých nebyla diagnostikována žádná kognitivní porucha. Tato data jsou označena hlavičkou **N01** až **N30**. Dále data obsahovala 25 popisů od osob, u nichž byla diagnostikována nějaká kognitivní porucha. Tato data jsou označena hlavičkou **P01** až **P30**.

Cílem tohoto finálního experimentu bylo získání širšího přehledu o funkci navržených a implementovaných algoritmů. Podle toho byly do výsledné tabulky vybrány pouze ty typy ztrát, u kterých byly hodnoty rozdílné pro různé osoby.

Pro získání zobrazených hodnot byl použit stejný referenční popis, jako během testování, tedy vytvořený autorem této práce a nikoli expertem z oboru. Dále je třeba upozornit na to, že ztrátová tabulka byla zrovna tak sestavena pouze pro ilustrační a testovací účely a samotné numerické hodnoty tedy není možné interpretovat jako skutečná data, která by byla použita při reálném nasazení.

Výstup je vizuálně zobrazen v Tabulce 6. Barevná škála je vždy vztažená pouze k dané ztrátě (tedy odpovídajícímu rádku, přes obě tabulky).

I přes zmíněná omezení je ale možné pouhým okem sledovat, že spodní část Tabulky 6, která obsahuje ztrátové hodnoty popisu od diagnostikovaných pacientů, má dle očekávání celkově vyšší hodnoty ztrát, což odpovídá předpokladu, že osoby s kognitivní poruchou budou mít větší problém popsat obrázek zobrazující komplexní scénu. Lze tedy usoudit, že navržený systém funguje očekávaným způsobem a při pečlivém vytvoření kritických vstupních dat lze očekávat, že výstupní hodnoty budou ještě lépe prezentovat sledované rozdíly v kognitivních schopnostech.

Detailnější testování a analýza reálných, interpretovatelných výsledků je plánována jako součást další budoucí práce, v rámci projektu TAČR SIGMA_DC3 „*Telemedicínské samovyšetření řeči a paměti pro rychlou detekci kognitivních poruch metodami strojového učení*“. Pro toto další pokračování je předpokládáno, že budou vytvořena vstupní data (obrázek, referenční popis, gramatika a ztrátová tabulka) pečlivě lidským expertem a výstupní hodnoty tak bude možné interpretovat v kontextu daného nasazení.

	N01	N02	N03	N04	N05	N06	N07	N08	N09	N10	N11	N12	N13	N14	N15	N16	N17	N18	N19	N20	N21	N22	N23	N24	N25	N26	N27	N28	N29	N30
all_objects	112,0	97,0	145,5	130,0	133,0	134,5	152,5	144,0	131,0	129,5	132,0	130,0	151,0	114,0	155,0	136,5	118,5	128,5	127,0	108,0	109,0	138,0	131,0	128,0	145,5	134,0	99,5	129,5	101,5	92,0
all_attributes	77,5	76,0	79,0	78,5	78,5	78,0	80,0	82,0	79,5	77,5	79,0	76,5	82,5	80,0	80,5	77,5	79,0	81,0	78,0	77,5	77,5	79,5	79,5	80,5	78,0	79,0	75,5	74,5	75,5	74,5
all_triplets	206,0	198,5	213,0	217,0	213,0	215,0	217,0	212,0	213,0	210,0	215,0	217,0	210,0	204,0	217,0	211,0	208,0	215,0	217,0	206,0	213,0	217,0	215,0	209,0	217,0	215,0	206,0	213,0	217,0	211,0
attr_action	4,5	4,5	5,5	6,0	4,5	5,0	5,0	7,5	5,5	5,0	5,5	5,5	6,0	4,5	6,0	6,0	5,5	5,0	5,5	6,0	6,0	7,5	7,5	5,5	5,5	6,0	5,5	5,0	5,0	
attr_color	60,0	56,5	59,0	57,0	59,0	58,0	60,0	59,0	59,0	58,0	59,0	56,0	61,0	59,0	57,5	58,0	61,0	58,0	57,5	57,0	58,0	58,0	58,0	56,5	59,0	56,0	54,0	56,5	54,5	
attr_state	0,5	0,0	0,0	0,5	0,0	0,5	0,5	0,5	0,5	0,0	0,0	0,5	0,5	0,0	0,5	0,5	0,0	0,5	0,0	0,5	0,5	0,5	0,5	0,5	0,5	0,0	0,5	0,5	0,5	
obj_tag_animal	48,0	18,5	36,0	32,0	40,0	36,0	52,0	40,0	36,0	32,0	28,0	40,0	32,0	32,0	36,0	36,0	28,0	28,0	28,0	26,5	32,0	36,0	28,0	36,0	40,0	36,0	36,0	32,0	14,5	
obj_tag_background	15,0	8,5	10,0	4,0	13,0	9,0	12,0	14,0	9,0	5,0	6,0	9,0	11,0	6,0	13,0	11,0	6,0	8,0	6,0	15,0	5,0	7,0	5,0	10,0	9,0	9,0	8,0	5,0	9,0	
obj_tag_clothing	24,0	24,0	36,0	36,0	36,0	33,0	28,0	36,0	36,0	36,0	28,0	36,0	36,0	30,0	36,0	36,0	36,0	36,0	36,0	27,0	30,0	36,0	36,0	36,0	36,0	21,0	27,0	30,0	36,0	
obj_tag_environment	16,0	11,0	13,5	12,0	14,0	12,0	17,0	15,0	13,0	10,0	13,0	11,0	16,0	15,0	13,0	11,5	10,5	17,0	12,0	16,0	10,0	11,0	14,0	12,0	11,5	13,0	10,0	10,5	10,0	
obj_tag_group	16,0	16,0	26,0	21,0	30,0	21,0	25,0	30,0	25,0	16,0	17,0	24,0	21,0	16,0	21,0	28,5	11,0	12,0	21,0	21,0	16,0	25,0	17,0	25,0	23,5	25,0	24,0	21,0	24,5	25,0
obj_tag_item	21,0	33,0	39,0	30,0	30,0	33,0	30,0	30,0	30,0	36,0	39,0	30,0	42,0	27,0	42,0	33,0	39,0	42,0	27,0	30,0	21,0	30,0	33,0	24,0	33,0	33,0	27,0	36,0	18,0	21,0
obj_tag_person	0,0	10,0	21,0	20,0	10,0	20,5	25,5	20,0	16,0	15,5	16,0	21,0	25,0	10,0	25,0	20,0	5,0	5,5	16,0	5,5	16,0	25,0	20,0	25,0	16,0	5,5	20,0	11,0	10,5	
trip_chasing	2,0	0,0	0,0	2,0	0,0	2,0	2,0	0,0	2,0	0,0	0,0	2,0	0,0	0,0	2,0	0,0	2,0	0,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2,0	2,0	2,0	0,0	

	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25			
all_objects	188,0	167,0	142,5	131,0	138,0	150,0	126,5	161,0	155,0	146,0	170,0	142,0	152,0	164,5	163,0	151,0	167,0	157,0	145,0	190,0	112,0	141,0	141,0	164,0	188,0			
all_attributes	82,0	81,5	79,0	79,5	82,5	82,0	76,5	79,0	82,0	80,5	78,5	77,0	79,5	81,0	84,0	79,5	81,0	82,0	80,5	79,5	78,5	80,5	78,0	79,5	82,0			
all_triplets	217,0	215,0	215,0	215,0	208,0	217,0	217,0	217,0	215,0	217,0	211,0	215,0	217,0	217,0	217,0	217,0	217,0	215,0	217,0	217,0	215,0	210,0	217,0	217,0	217,0	217,0	217,0	
attr_action	7,5	5,5	4,5	5,5	6,0	7,5	5,5	5,5	7,5	6,0	5,5	5,0	5,5	5,0	7,5	5,5	7,5	7,5	6,0	5,5	5,5	6,0	5,5	5,5	7,5			
attr_color	59,0	61,0	59,0	59,0	61,0	59,0	56,5	59,0	59,0	59,0	58,0	57,0	59,0	61,0	61,0	59,0	58,0	59,0	59,0	59,0	58,0	59,0	59,0	59,0	59,0	59,0	59,0	59,0
attr_state	0,5	0,5	0,5	0,5	0,5	0,5	0,0	0,0	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
obj_tag_animal	52,0	48,0	40,0	28,0	36,0	40,0	18,5	40,0	32,0	40,0	48,0	40,0	44,0	48,0	32,0	44,0	56,0	36,0	40,0	60,0	32,0	28,0	44,0	48,0	48,0			
obj_tag_background	14,0	11,0	12,5	6,0	10,0	13,0	13,0	10,0	7,0	10,0	14,0	11,5	13,0	15,0	8,0	11,0	9,0	14,0	14,0	13,0	5,0	6,0	10,0	13,0	14,0			
obj_tag_clothing	36,0	36,0	36,0	36,0	36,0	33,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	36,0	31,0	36,0	36,0	36,0	36,0	36,0	
obj_tag_environment	15,0	17,0	13,5	14,0	15,0	15,0	14,0	14,0	15,0	14,0	13,0	11,5	14,0	18,0	18,0	15,0	14,0	15,0	15,0	14,0	15,0	15,0	14,0	14,0	14,0	16,0		
obj_tag_group	30,0	25,0	25,0	22,0	30,0	30,0	21,0	30,0	21,0	30,0	30,0	25,0	25,0	22,0	25,0	25,0	26,0	30,0	30,0	17,0	17,0	20,0	25,0	26,0				
obj_tag_item	42,0	45,0	30,0	27,0	36,0	39,0	39,0	45,0	42,0	36,0	39,0	36,0	39,0	39,0	42,0	30,0	36,0	42,0	36,0	45,0	24,0	42,0	36,0	42,0	45,0			
obj_tag_person	40,0	21,0	20,0	26,0	15,0	20,0	16,0	26,0	30,0	20,0	31,0	15,5	16,0	20,5	35,0	26,0	25,0	25,0	15,0	31,0	6,0	20,0	16,0	21,0	40,0			
trip_chasing	2,0	0,0	2,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	0,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	2,0	0,0	0,0	2,0	2,0	2,0	2,0	2,0	2,0

Tabulka 6: Vizuální znázornění rozdílných ztrát pro bez osoby bez diagnózy (sloupce N01 až N30) a diagnostikované pacienty (sloupce P01 až P25)

8 Závěr

Cílem této práce bylo navrhnut a implementovat systém pro hodnocení míry korespondence mezi obrázkem a jeho popisem v přirozeném řeči. Motivací pro takovýto systém bylo jeho využití pro usnadnění diagnostických testů pro detekci kognitivních poruch. Zvolené řešení bylo založeno na expertním přístupu, který byl pro toto použití shledán vhodným.

Během návrhu a realizace byl kladen důraz na to, aby všechny důležité procesy a subsystémy umožňovaly precizní kontrolu a možnost případných detailních úprav pro dosažení co nejlepších výsledků v reálném nasazení. Dále bylo během návrhu dbáno na modularitu a obecnost, aby bylo možné jednotlivé procesy nahradit za případné vhodnější alternativy pro konkrétní aplikace v různých oblastech.

Výsledná architektura celého systému se skládá z referenčního popisu obrázku, subsystému pro zpracování přirozené řeči a extrakce sémantické informace, která je v ní obsažena, a hodnotícího algoritmu, který porovná referenční popis se získanou sémantikou a na základě ztrátové tabulky vygeneruje vektor hodnot, které reprezentují míru sémantické shody mezi vzorovým a testovaným popisem. Tento výstupní vektor je pak možné použít jako výchozí bod pro další zpracování, například jako vektor příznaků pro klasifikační metody.

Od lidského experta v daném oboru je vyžadováno, aby vybral obrázek a sestavil jeho referenční popis, který udává, jak by měl vypadat ideální popis vybraného obrázku. Dále je nutné vytvořit gramatiku pro sémantické zpracování řeči a nakonec je na expertovi, aby vytvořil podle referenčního popisu ztrátovou tabulku, pomocí které specifikuje závažnost různých druhů chyb v popisu.

Tento expertní přístup s sebou přináší omezení ve smyslu vysokých počátečních nákladů a náročné přípravy, jelikož tvorba referenčního popisu, gramatik a ztrátové tabulky může představovat časově náročný úkol. Na druhou stranu ale zvolený přístup umožňuje velmi detailní a precizní kontrolu nad jednotlivými procesy a fázemi, právě díky tomu, že vstupní vzorová data mohou být pečlivě sestavena a nejsou generována automaticky pomocí statistických metod či neuronových sítí, jejichž výsledky nemusí být dostačující pro takto specializovanou problematiku.

V rámci implementace navržených systémů byl také navržen nový formát pro sémantické gramatiky, včetně implementace jeho parseru. Jedná se o jednou z klíčových částí zpracování přirozené řeči a extrakci sémantické informace ve zde prezentovaném systému. Tento nový formát, pojmenovaný „Semantic Parsing Grammar Format“, byl založen na existujícím standardu Speech Recognition Grammar Specification, oproti kterému nabízí rozšíření funkčnosti, především ve smyslu ergonomie použití a možnostech volby z více parsovacích strategií. Tento formát a k němu přidružený software je možné použít nezávisle na zbytku této práce i pro zcela jiné účely.

Během realizace se nevyskytly žádné významné technické komplikace. Jediným výraznějším problémem byla absence expertních znalostí z oblasti medicíny zaměřující se na kognitivní poruchy, která vedla k tomu, že vytvořený referenční popis obrázku a ztrátovou tabulku bylo možné brát pouze jako ilustrační a pro testovací účely. To však ve výsledku nepředstavovalo výrazný problém, jelikož účelem práce bylo navrhnout systém pro výše popsaný účel, na jehož otestování ilustrační vstupní data postačovala.

Výsledky prezentované v kapitole 7 ukazují, že navržený systém je schopen extrahat z přirozeného textu požadované sémantické informace, za předpokladu dobré navrženého referenčního popisu a především kvalitně sestavených gramatik. Vzhledem k již popsaným problémům s absencí expertem poskytnutých dat bylo vyhodnocení porovnávacího a hodnotícího algoritmu obtížnější, avšak dosažené výsledky nasvědčují tomu, že skutečně méně kvalitním popisům obrázku odpovídají vyšší hodnoty ztrát, což je v souladu s původním předpokladem.

Důkladnější testování hodnotícího algoritmu je plánováno v rámci již zmíněného projektu TAČR SIGMA_DC3 „*Telemedicínské samovyšetření řeči a paměti pro rychlou detekci kognitivních poruch metodami strojového učení*“, který umožní detailnější analýzu nad větším množstvím dat a spolupráci s experty z daného oboru. Dalším potenciálním rozšířením je otestování alternativních přístupů k extrakci sémantické informace, například s využitím dnes populárních velkých jazykových modelů (angl. large language models, LLMs).

Zdroje a použitá literatura

- [1] PALMER, F.R. *Semantics*. Cambridge University Press, 1981. Cambridge low priced editions. ISBN 9780521283762. Dostupné také z: <https://books.google.cz/books?id=UWJSaxH9GiMC>.
- [2] GEERAERTS, Dirk. *Theories of Lexical Semantics*. New York: Oxford University Press, 2010.
- [3] RAJMAN, M.; BESANÇON, R. *Speech and Language Engineering*. EPFL Press, 2007. Communication Sciences. ISBN 9780824722197. Dostupné také z: <https://books.google.cz/books?id=QMXjAAAAMAAJ>.
- [4] PORTNER, P.H.; PARTEE, B.H. *Formal Semantics: The Essential Readings*. Wiley, 2008. Linguistics: The Essential Readings. ISBN 9780470758182. Dostupné také z: <https://books.google.cz/books?id=ptgUWRtAkMC>.
- [5] RIEMER, N. *Introducing Semantics*. Cambridge University Press, 2010. Cambridge Introductions to Language and Linguistics. ISBN 9780521851923. Dostupné také z: <https://books.google.cz/books?id=tJjZ0FmNdzIC>.
- [6] FASOLD, R.; CONNOR-LINTON, J. *An Introduction to Language and Linguistics*. Cambridge University Press, 2006. ISBN 9781107717664. Dostupné také z: <https://books.google.cz/books?id=E85VAgAAQBAJ>.
- [7] BLACKBURN, P.; BOS, J. Computational Semantics. *Theoria: An International Journal for Theory, History and Foundations of Science* [online]. 2003, roč. 18, č. 1(46), s. 27–45 [cit. 2024-05-13]. ISSN 04954548, ISSN 2171679X. Dostupné z: <http://www.jstor.org/stable/23918435>.
- [8] BUNT, H.C.; MUSKENS, R.A. Computational semantics. In: *Computing meaning, Volume 1*. Ed. BUNT, H.C.; MUSKENS, R.A. Netherlands: Kluwer Academic Publishers, 1999, s. 1–32. Studies in linguistics and philosophy, č. 73. ISBN 0792361083. Pagination: 33.
- [9] MCTEAR, M.; CALLEJAS, Z.; GRIOL, D. *The Conversational Interface: Talking to Smart Devices*. Springer, 2016.
- [10] YANG, Jinbiao. *Rethinking Tokenization: Crafting Better Tokenizers for Large Language Models*. 2024. Dostupné z arXiv: 2403.00417 [cs.CL].

- [11] PSUTKA, J.; MÜLLER, L.; MATOUŠEK, J.; RADOVÁ, V. *Mluvíme s počítačem česky*. Prague: Academia, 2006. ISBN 80-200-1309-1.
- [12] JOSEF, Psutka; ŠMÍDL, Luboš. *Reprezentace znalostí* [Učební text KKY/UI]. 2021.
- [13] LEHMANN, Fritz. Semantic networks. *Computers & Mathematics with Applications*. 1992, roč. 23, č. 2, s. 1–50. ISSN 0898-1221. Dostupné z doi: [https://doi.org/10.1016/0898-1221\(92\)90135-5](https://doi.org/10.1016/0898-1221(92)90135-5).
- [14] SVÁTEK, Vojtěch. Ontologie a WWW. 2002. Dostupné také z: <http://nb.vse.cz/~svatek/onto-www.doc>.
- [15] LENAT, Douglas B. CYC: a large-scale investment in knowledge infrastructure. *Commun. ACM*. 1995, roč. 38, č. 11, s. 33–38. ISSN 0001-0782. Dostupné z doi: 10.1145/219717.219745.
- [16] JOLLEY, JL. The holotheme. In: *First Ottawa Conference on Conceptual Basis of the Classification of Knowledge, Ottawa*. 1971.
- [17] BURGER, Henry G. The wordtree: a transitive cladistic for solving physical & social problems: the dictionary that analyzes a quarter-million word-listings by their processes, branches them binarily to pinpoint the concepts, thus sequentially tracing causes to their effects, to produce a handbook of physical and social engineering. (*No Title*). 1984.
- [18] RDF 1.2 Schema. 2024. Dostupné také z: <https://www.w3.org/TR/rdf12-schema/>.
- [19] Resource Description Framework. 2014. Dostupné také z: <https://www.w3.org/RDF/>.
- [20] GRUBER, Ivan. *Convolutional Neural Networks* [Učební text KKY/NSES]. 2023.
- [21] SARWAR, Sohail; QAYYUM, Zia Ul; MAJEDD, Saqib. Ontology based Image Retrieval Framework Using Qualitative Semantic Image Descriptions. *Procedia Computer Science*. 2013, roč. 22, s. 285–294. ISSN 1877-0509. Dostupné z doi: <https://doi.org/10.1016/j.procs.2013.09.105>. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013.

- [22] MEIYU, Liang; JUNPING, Du; YINGMIN, Jia; ZENGQI, Sun. Image semantic description and automatic semantic annotation. In: *ICCAS 2010*. 2010, s. 1192–1195. Dostupné z DOI: 10.1109/ICCAS.2010.5669742.
- [23] LOWE, D.G. Object recognition from local scale-invariant features. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. 1999, sv. 2, 1150–1157 vol.2. Dostupné z DOI: 10.1109/ICCV.1999.790410.
- [24] *Graphviz*. Dostupné také z: <https://graphviz.org/>.
- [25] *Speech Recognition Grammar Specification*. 2004. Dostupné také z: <https://www.w3.org/TR/speech-grammar/>.
- [26] *TreeSitter*. Dostupné také z: <https://tree-sitter.github.io/tree-sitter/>.
- [27] KNUTH, Donald E. *The Art of Computer Programming, Vol. 1*. Third. Reading, Mass: Addison-Wesley, 1997. ISBN 9780201896831.