Obsah

1 Úvod

2 Teorie k řešené problematice

2.1 Reprezentace znalostí

2.2 Popis obrázků

2.3 Sémantika

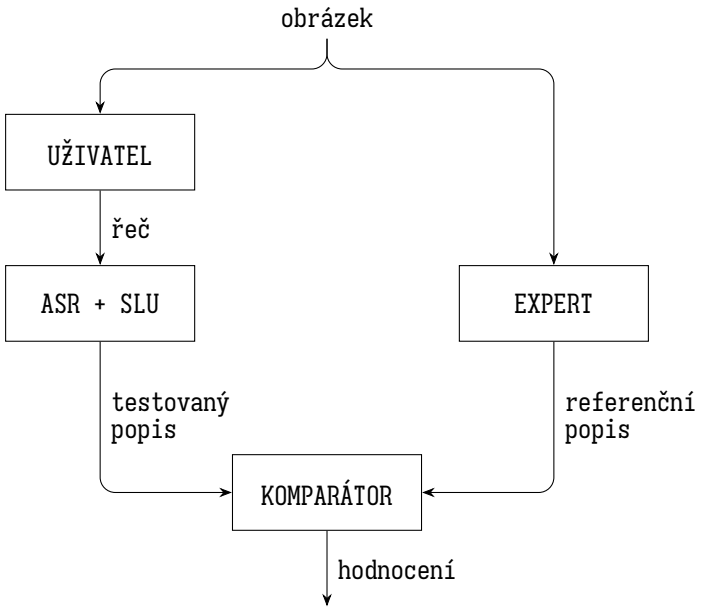
3 Návrh a architektura systému

3.1 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 (⇒ testovaný popis)
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 (ASR). TODO[1] Z tohoto přepisu je extrahována sémantická informace (SLU), 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 1.



Obrázek 1: Schéma obecné architektury systému

[1] Jak/kde vysvětlit zkratky?

Pro tuto práci bylo rozhodnuto, že základem bude expertní přístup. 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. TODO[2]

3.2 Reprezentace znalostí

Pro porovnání obrázku s jeho popisem v přirozené řeči bylo nutné zvolit či navrhnout nějakou 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. TODO[3]

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.

[2] zmínit, že referenční popis není třeba pokaždé tvořit znovu, ale lze udělat „offline“ předem?

[3] zmínit, že expertní přístup umožňuje lepší kontrolu nad obsahem/kvalitou než automat/statistika?

- **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 nejširší 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 4. TODO[4]

3.2.1 Objekty ve scéně

Cokoli, co lze v obrázku ohraničit rámečkem (angl. bounding-box) TODO[5] a při separaci od zbytku scény (obrázku) neztratí nebo zásadně 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. „obličej“ 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.

[4] jak a kde se správně používá emph pro zdůraznění pojmu/termínu?
[5] jak s anglikanismy?

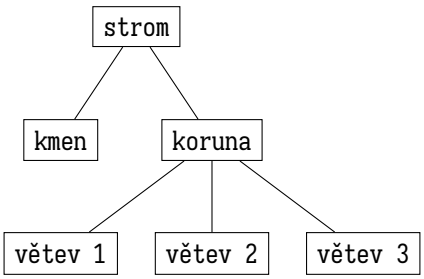
V jednom obrázku lze definovat různé množiny objektů, podle toho, jak moc detailní popis expert vytvoří. Například pokud by byl na obrázku člověk, lze jej popsat jedním objektem jako „člověk“ nebo „osoba“, ale také by šlo definovat ještě mnoho dalších objektů, například pro jednotlivé části těla nebo 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. TODO[6]

3.2.2 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 2.



Obrázek 2: 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*. Aby byl popis jednoznačný, tak bylo rozhodnuto, že vazby musejí být definované v referenčním popisu

[6] vyměnit dlouhé popisy za jeden ukázkový příklad?

oboustranně. Tím je myšleno to, že pokud objekt A specifikuje objekt B jako svého rodiče, pak musí i objekt B specifikovat A jako svého potomka. Pokud je hierarchická vazba definována pouze jednostranně, měl by být referenční popis implementací systému odmítnut jako invalidní.

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. TODO[7]

3.2.3 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*.

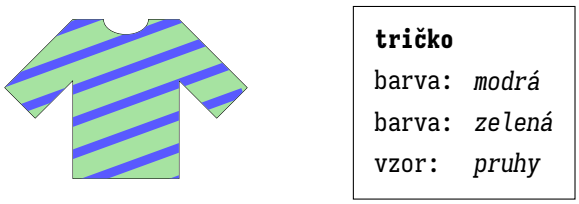
Atributem je možné popsat jakoukoli informaci o objektu, která není závislá na jiném objektu. Jinými slovy, pokud bychom objekt izolovali od zbytku scény (obrázku), tak všechny vlastnosti, které se tím nezmění nebo nezaniknou, lze popsat pomocí atributů. Typickým příkladem je barva nebo tvar objektu 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 víc stejnojmenných atributů s různými hodnotami. Typickým příkladem může být vícebarevný objekt, kde atribut s názvem „barva“ bude vícekrát, pro různé konkrétní barvy. Příklad

[7] zdůvodňovat? dávat příklad? přirovnávat k OOP dědičnost vs kompozice?

jednoho takového popisu je na Obrázku 3, kde název objektu je „tričko“ a definované jsou tři atributy: dvě barvy a vzor. TODO8 TODO9



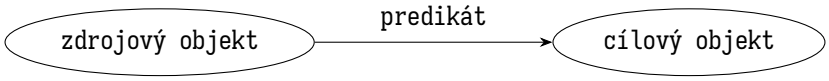
Obrázek 3: Příklad použití atributů pro popis objektu

Pro složitější objekty se opět naskýtá 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 3.2.1 definující objekty. A i zde platí to, že popis byl záměrně vytvořen tak, aby byl univerzální a volba atributů do referenčního závisí na expertovi a konkrétní aplikaci.

3.2.4 Vazby mezi objekty

Posledním 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. TODO10 Obecné schéma jednoho tripletu je na Obrázku 4.



Obrázek 4: 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

[8] hezčí tabulka pro popis objektu?

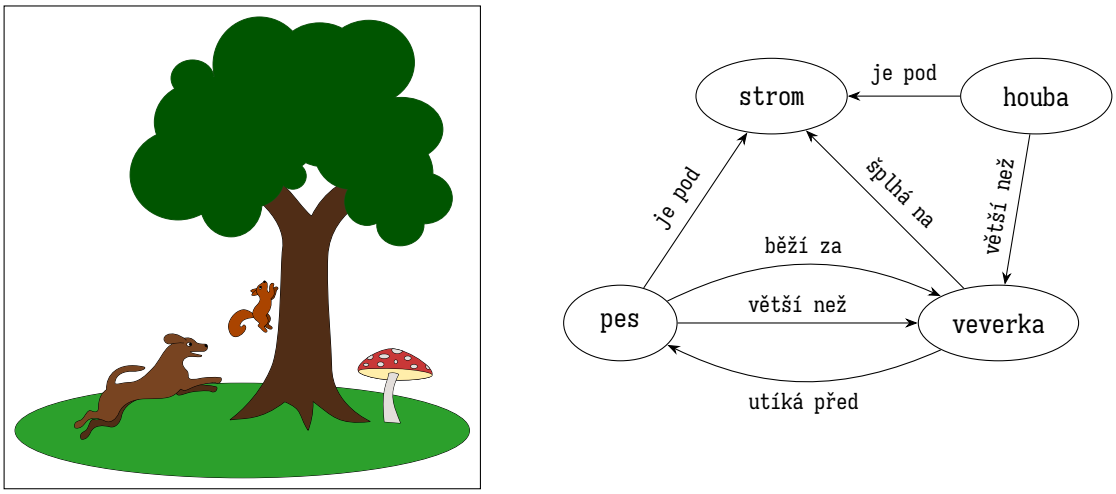
[9] přidat šipky a popisky do tabulky co je co?

[10] doplnit reference!

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 naskýtá 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 5.



Obrázek 5: Příklad použití tripletů pro popis vztahů mezi objekty

TODO 11

[11] příklad s kompletním popisem obrázku (všechno dohromady)?

3.3 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. TODO[12] 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 (ASR). TODO[13] Problematika rozpoznání řeči a převodu audia do textu je nad rámec této práce a předpokládá se, že bude v praxi řešena nějakou již existující implementací. TODO[14] Ve zbytku práce bude tedy pro zjednodušení rovnou uvažovaným vstupem text. TODO[15]

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 označkovaná část textu je pak označována jako *sémantická entita.* TODO[16] Konkrétní syntaxe, použití a funkčnost těchto gramatik bude popsán později spolu s implementací v části 4.1.

[12] ozdrojovat?

[13] existují metody co to dělají rovnou z audia?

[14] zmínit SpeechCloud jako existující implementaci?

[15] doplnit, že jsem při vývoji používal ruční přepisy + odkud? Nebo až později u implementace?

[16] nemá „sémantická entita“ jiný význam a můžu takhle přetížit definici?

Množina všech těchto sémantických entit, které byly ze vstupního textu extrahované, pak tvoří testovaný popis.

3.3.1 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 3.2) 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.

3.3.2 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 $\mathcal{O}$:

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

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 TODO[17] .

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.

[17] doplnit referenci na budoucí kapitolu

3.3.3 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 popsat 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 TODO[18] .

[18] doplnit referenci na budoucí kapitolu

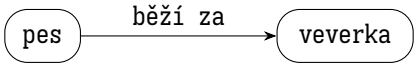
3.3.4 Extrakce sémantiky - vazby mezi objekty

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 detekovat 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 při 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 a bude detailně adresován v sekci TODO[19] .

Je vhodné zmínit, že ačkoli byly v kapitole 3.2 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.

[19] doplnit referenci na budoucí kapitolu

3.4 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: TODO[20]

$$\textit{referenční popis obrázku} = \left(\mathcal{O}_R, \mathcal{A}_R, \mathcal{V}_R\right)$$
$$\textit{testovaný popis obrázku} = \left(\mathcal{O}_T, \mathcal{A}_T, \mathcal{V}_T\right)$$

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.

[20] definovat symbol pro popis obrázku? použít pro definice jiný symbol než =?

3.4.1 Chybějící objekty

Pokud byl v referenčním popise 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:

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

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

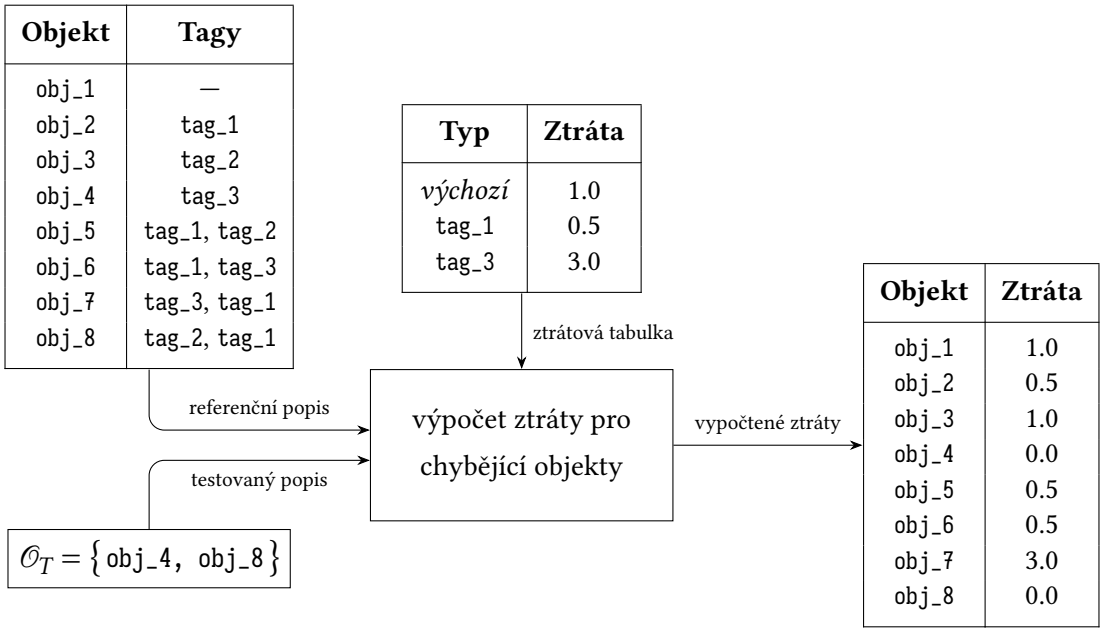
Lze očekávat, že různé objekty budou ve scéně různě důležité a bylo by vhodné, aby byl tento fakt zohledněn při výpočtu ztráty. Proto lze přiřadit objektům *tagy*. Jako *tag* je v tomto kontextu chápána nějaká značka, která říká, že objekt patří do dané skupiny. Například objekt „`tričko`“ může mít přiřazen 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 6. [TODO 21]

[21] vyměnit obecné názvy za konkrétní?



Obrázek 6: Jednoduchá ukázka ztráty na chybějících objektech

Na příkladu zobrazeném na Obrázku 6 je možné vidět, že objekt `obj_4`, 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 `obj_6` a `obj_7` 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 6, šel by tento finální krok znázornit Obrázkem 7.

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.

Objekt	Tagy	Ztráta
obj_1	—	1.0
obj_2	tag_1	0.5
obj_3	tag_2	1.0
obj_4	tag_3	0.0
obj_5	tag_1, tag_2	0.5
obj_6	tag_1, tag_3	0.5
obj_7	tag_3, tag_1	3.0
obj_8	tag_2, tag_1	0.0

→

Chybějící objekty	Ztráta
všechny	6.5
objekty s tag_1	4.5
objekty s tag_2	1.5
objekty s tag_3	3.5

Obrázek 7: Finálního krok při počítání ztrát na chybějících objektech

3.4.2 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. TODO[22]

3.4.3 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.

[22] dát konkrétní příklad včetně tabulek, výpočtů ...?

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. TODO[23]

3.4.4 Atributy s chybnou hodnotou

Pokud v 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:

$$\mathscr{A}_R = \{\texttt{tričko: barva = modrá}\}$$
$$\mathscr{A}_T = \{\texttt{tričko: barva = zelená}\}$$

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 tabulku shodnou s Tabulkou 2. Tu je možné číst následujícím způsobem:

[23] dát konkrétní příklad včetně tabulek, výpočtů ...?

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 1.

Atribut	**Ztráta**
výchozí	1.0
`barva`	**Hodnoty** / **Ztráta** *výchozí* / 0.5 `žlutá`, `oranžová` / 0.3 `červená`, `modrá`, `zelená` / 0.8
`tvar`	**Hodnoty** / **Ztráta** *výchozí* / 1.5

Tabulka 1: 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 2: Výpočtu ukázkových ztrát chybných hodnot atributů podle Tabulky 1

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.

3.4.5 Výstup hodnotícího algoritmu

Finálním výstupem hodnotícího algoritmu je množina označených číselných hodnot, které jsou akumulované ztráty pro různé druhy chyb. Přirozenou datovou strukturou, která by odpovídala formátu, by bylo asociativní pole (někdy také označované jako hash-tabulka).

Výstup hodnotícího algoritmu, jak byl popsán v předchozích částech, obsahuje vždy celkovou ztrátu pro daný typ chyby a poté akumulované ztrátové hodnoty stejného typu, ale rozdělené ještě podle jednotlivých kategorií. TODO[24] 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

[24] příklad?

- 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 různé konkrétní nasazení mohou být různé 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 použít pro implementaci alternativních způsobů hodnocení, nebo případně rovnou použít jako výstup, pokud by daná aplikace benefitovala z podrobnější analýzy výsledků.

4 Implementace a testování

4.1 Sémantické parsování pomocí gramatik

5 Závěr