BAKALÁŘSKÁ PRÁCE

Zuzana Svobodová

Generování textového popisu trasy v mapě

Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Rudolf Rosa, Ph.D.

Studijní program: Informatika

Studijní obor: IOI
Na tomto místě děkuji svému vedoucímu Rudolfu Rosovi za podněty a rady, za srozumitelné a přátelské vedení práce.

A svým blízkým za podporu.
Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

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

V .................. dne ...................... podpis
Název práce: Generování textového popisu trasy v mapě

Autor: Zuzana Svobodová

Katedra / Ústav: Ústav formální a aplikované lingvistiky

Vedoucí bakalářské práce: Mgr. Rudolf Rosa, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Výsledkem práce je program, který pro pěší trasu v mapě vytvoří textovou navigaci, která má být snadno srozumitelná člověku. Soudobé aplikace pro textový popis trasy (které jsou součástí například mapy.cz či google.maps.com) uživateli informují hlavně o vzdálenostech a směrech, podle čehož není vždy snadné se orientovat. Když trasu popisuje člověk, většinou při popisu používá význačné objekty v krajině a také spojuje jednotlivé informace do složitějších. V našem programu se tedy snažíme o tento více lidský popis, který by chodci umožnil následovat trasu jen s pokyny programem vygenerovanými.

Klíčová slova: mapa, trasa, navigace, generování textu

Title: Generating text descriptions of journeys in a map

Author: Zuzana Svobodová

Department: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Rudolf Rosa, Ph.D., Ústav formální a aplikované lingvistiky

Abstract: This thesis aims to present the key aspects of a program developed for the purpose of improving orientation in maps by generating text description of routes. Even though such tools are already available and integrated within the most widely used internet map engines (e.g. maps.google.com and mapy.cz), they are not particularly user-friendly, as they rely on directions and distances. People are, on the other hand, more inclined to use landmarks such as significant buildings for orientation and synthetize simple information into more complex one. The program presented in this thesis addresses this issue and offers more intuitive route descriptions enabling its user to reach his/her destination potentially faster and more reliably.

Keywords: map, route, navigation, text generation
Obsah

1 Úvod ............................................................................................................. 1

2 Předchozí výzkum .......................................................................................... 2
   2.1 Popisy cest v on-line mapách ................................................................. 2
   2.2 Jak na uživatelsky přívětivější popis ..................................................... 3
   2.3 Alternativní aplikace pro popis cesty ................................................... 3

3 Řešení a cesta k němu ..................................................................................... 5
   3.1 Základní pojmy ....................................................................................... 5
      3.1.1 Data pro popis z openstreetmap.org ............................................. 5
      3.1.2 Trasy k popisu z mapy.cz .............................................................. 7
      3.1.3 Význačné objekty .......................................................................... 7
   3.2 Popis řešení ............................................................................................ 8
      3.2.1 Schéma programu ........................................................................ 8
      3.2.2 Parsování vstupu .......................................................................... 8
      3.2.3 Generování popisu cesty ............................................................... 9
      3.2.4 Důkladněji k sběru a rozčlenění dat a tvorbě výstupu ................... 11
      3.2.5 Uživatelský výstup ..................................................................... 11
      3.2.6 Qgis ............................................................................................. 12

4 Technická implementace ............................................................................. 13
   4.1 Názvosloví pro tuto kapitolu ................................................................. 13
   4.2 Popis programu ..................................................................................... 13
      4.2.1 Parsování vstupu ........................................................................ 13
      4.2.2 Sběr dat ....................................................................................... 14
      4.2.3 Rozčlenění dat ............................................................................ 16
      4.2.4 Tvorba výstupu .......................................................................... 16
   4.3 Křižovatky ............................................................................................. 17
      4.3.1 Sjednocení dvou typů ways v křižovatce ...................................... 17
      4.3.2 Výběr správné cesty ................................................................... 18
   4.4 Význačné objekty .................................................................................. 19
   4.5 Soubory s kódem a testovací data ........................................................ 20
      4.5.1 Soubory s kódem ....................................................................... 20
      4.5.2 Testovací data ........................................................................... 21

5 Zhodnocení řešení ......................................................................................... 22
   5.1 Za jakých okolností program funguje ................................................... 22
      5.1.1 Křižovatka, kde na sebe cesty přímo navazují ................................ 22
      5.1.2 Úseky bez odboček ..................................................................... 23
      5.1.3 Pokud jsou OSM přesné ............................................................... 24
      5.1.4 Nezvyklé značení ways blízko křižovatky ....................................... 24
      5.1.5 Příklady dobře popsaných tras .................................................... 26
   5.2 Za jakých okolností program nefunguje ................................................ 30
      5.2.1 Pokud je trasa zaznačena nečekaně vzhledem k datům z OSM ........ 30
      5.2.2 Velké plochy v křižovatce ............................................................ 31

6 Možnosti vylepšení ...................................................................................... 33
   6.1 Navigační a popisová vylepšení ............................................................ 33
      6.1.1 Navigování do křižovatky .............................................................. 33
      6.1.2 Pořadí objektů v cestě ................................................................. 33
      6.1.3 Výběr význačných objektů ............................................................ 34
   6.2 Technické nápady .................................................................................. 34
      6.2.1 Automatické získání mapy ............................................................. 34
6.2.2 Větší využití hotových nástrojů pro práci s OSM..........................34
7 Jak požívat Route-to-description...............................................................35
  7.1 Jak získat vstupní data........................................................................35
  7.2 Jak spustit program...............................................................................35
    7.2.1 Bez parametrů..............................................................................35
    7.2.2 S parametry................................................................................35
  7.3 Co je výstupem....................................................................................36
8 Závěr........................................................................................................37
9 Seznam použité literatury..........................................................................38
10 Seznam použitých zkratek......................................................................39
11 Přílohy......................................................................................................40
  11.1 Zdrojové kódy..................................................................................40
  11.2 Testovací data...................................................................................40
1 Úvod


Program, který je výsledkem této bakalářské práce, převádí trasu na textový popis, který je oproti výše zmíněným rozšířen o snahu informovat uživatele o význačných objektech v krajině a shrnovat jednoduché instrukce do komplexnějších, aby napodobil, jak trasu popisuje člověk. Vzhledem k různorodosti dat z map je nelehké připravit program na všechny situace, které může chodec cestou potkat, v této práci jsme tedy pokryli jen část z nich.

Vzniklý program jsme pojmenovali Route-to-description (dále budeme nazývat většinou zkratkou RTD) a jeho použití je následovné: uživateli si najde v internetové mapě trasu, kterou se chce dát, a mapu, která pokrývá tuto trasu, a předá je našemu programu. Program vygeneruje jako výstup textový popis oné trasy, který by měl být dostatečně vyčerpávající, aby jen s pomocí tohoto textu mohl chodec trasu následovat a trefit do cíle.
2 Předešlý výzkum

V této kapitole si stručně shrneme, jak vypadají aktuální automaticky gene-
rované popisy v on-line mapách a obsahy dvou vědeckých článků, zabývající se při-
rozeným popisem trasy, které slouží jako částečná inspirace pro náš RTD.

2.1 Popisy cest v on-line mapách

Automatické generování textového popisu trasy nabízí internetové mapy
maps.google.com a mapy.cz. V závěru práce bude jejich popis porovnán s popisem
vygenerovaným pomocí RTD.

Mapy.cz nabízejí přibližně takovéto instrukce:
• „Odbočte ostře vlevo směrem na Bechyňova a pokračujte 174 m.“
• „Rovně směrem na chodník a pokračujte 399 m.“
• „Odbočte vlevo směrem na zpevněnou cestu a pokračujte 136 m.“
• „Odbočte vlevo směrem na zelenou a pokračujte 448 m.“

Pokyny, kudy se dát, jsou tedy založeny především na názvech ulic (Bechyňova)
či povaze cest (chodník, zpevněná vozovka), směrech (ostře vlevo) a vzdálenostech
(174 metrů). Zajímavým prvkem je zmínění turistických značek (směrem na zelenou,
tedy na zelenou turistickou značku).

Maps.google.com generují následující typy instrukcí:
• „Jděte na sever směrem na Vítězné nám (5 m).“
• „Pokračujte po Zengrova (140 m) (použijte schody).“

Jde tedy o velmi podobný koncept, obohacený občas o nějakou informaci navíc,
například že na Zengerově ulici budou schody.

Tyto příklady textových popisů mohou sloužit dobře v oblastech, kde jsou ulice
důsledně pojmenovány, či má-li chodce nástroje k přesnému odhadnutí vzdálenosti.
2.2 Jak na uživatelsky přívětivější popis

Inspirací pro návrh uživatelsky přívětivějšího popisu se stal mimo jiné článek „Generating More Natural Route Descriptions“[3], jehož autoři jsou Robert Dale, Sabine Geldof a Jean-Philippe Prost. V článku se rozebírá, co je pro člověka v orientaci podstatné a co automatické systémy (dále AS) pro generování popisu trasy obvykle opomíjejí či řeší jinak:

1. Některé informace se lidem zdají zjevné, a proto je do popisu nezahrnou, zatímco AS je zmíní, protože nerozpoznají jejich (ne)důležitost. (Člověk zmíní pouze nepřehlédnutelné objekty jako kostel či most, AS oproti tomu neví, co je „nepřehlédnutelné“, a může zmínit i nevýrazné objekty.)
2. Lidé pro navigaci používají především význačné body v krajině, zatímco AS používají vzdálenosti a směry. (Člověk: „Za mostem zabočte kolem hřbitova doprava“, AS: „Jděte deset metrů rovně a potom zabočte doprava“)
3. Lidé jednotlivé instrukce dávají do souvislosti a spojují je do složitějších pokynů, zatímco AS přiřadí typicky každou instrukci do nové věty. (Člověk: „Projdete rovně přes dvě křižovatky“, AS: „Pokračujte rovně 20 m. Pokračujte rovně 50 m.“)

Podle této klasifikace odpovídají popisy generované pomocí mapy.cz a maps.google.com více tvorbě automatických systémů pro generování trasy. V této práci se zaměřujeme na to, aby výsledný popis cesty byl pro uživatele více názorný, tedy chce-me napodobit, jak by trasu popsal člověk. Proto je zde snahou pro navigaci používat významné body v krajině a spojovat informace dohromady.

2.3 Alternativní aplikace pro popis cesty

Michael Roth a Anette Frank v článku „A NLG-based Application for Walking Directions“[4] popisují aplikaci, která má za úkol (stejně jako náš RTD) vytvořit uživatelsky přívětivý popis trasy, a to v anglickém jazyce. Aplikace získá hotovou trasu a význačné body z několika zdrojů, mezi nimi i openstreetmap.org[5] (používané v RTD). Důkladně se tady věnují lingvistické stránce věcí, aby se popis co nej-
více podobal lidskému. Neobjevili jsme žádnou už existující aplikaci, která by popis tvořila v češtině, a to dělá RTD.
3 Řešení a cesta k němu

3.1 Základní pojmy

V této podkapitole stručně vysvětlíme, s jakými daty RTD pracuje.

3.1.1 Data pro popis z openstreetmap.org

Pro tuto práci jsme využívali internetovou mapu openstreetmap.org[5] (dále nazýváme OSM), ze které je snadně exportovat část mapy se všemi potřebnými informacemi pro popis trasy. V OSM jsou objekty z reálného světa ukládány jako takzvané uzly (nodes), cesty (ways) a relace (relations). Každá tato entita může mít tagy, které upřesňují její vlastnosti.


Obrázek č. 1: Uzel představující lavičku ve formátu jak ho přijímá náš program. Zdroj: openstreetmap.org
Cesta je skupina uzlů a reprezentuje nejčastěji věci, které se na mapě jeví jako křivky (silnice, chodníky, zídky, potoky) a mnohoúhelníky (budova, zahrada). Máme-li v mapě třeba cestu značící kostel, může například obsahovat jednotlivé tagy s informacemi, jak se kostel jmenuje, jakému přísluší náboženství a odkaz na stránku na wikipedia.org, tomuto kostelu příslušející. Viz obrázek č. 2.

Relace zastupuje v mapě komplexní objekty, popisující typické skupinu objektů, jako například městská část (obsahující domy, ulice, …), komplex budov, náměstí (obsahující kašnu, chodníky, obchody, …), park (obsahující lavičky, stromy, různé povrchy, …). Relace se tak skládá z uzlů a cest, a i z jiných relací. Pro příklad tagů si vezmeme Druhé nádvoří na Pražském hradě, o kterém tagy prozrazují mimo jiné jeho název v různých jazycích, že má dlážděný povrch a že je turisticky atraktivní.
3.1.2 Trasy k popisu z mapy.cz

Trasy, pro které RTD vygeneruje textový popis, jsme získali z internetových map mapy.cz, kde je snadné trasu naplánovat a poté exportovat. Trasa je exportována jako soubor GPX, obsahující seznam bodů, jak jdou v trase po sobě, kde každý je charakterizován svou zeměpisnou šířkou a délkou. Překvapivě může tento soubor obsahovat dva identické body po sobě.

![Diagram GPX souboru trasy](image)

**Obrázek č. 4: Dva body z trasy, jak jsou zaznačeny ve vyexportovaném souboru. Zdroj: mapy.cz**

3.1.3 Význačné objekty


Přesný seznam najdete v sekci Technická implementace – Význačné objekty na straně 19.
3.2 Popis řešení

3.2.1 Schéma programu

RTD schématicky odpovídá diagramu na obrázku č. 5. K jednotlivým částem diagramu se budeme detailněji vyjadřovat v následujících podkapitolách.

![Diagram schématické znázornění běhu programu](image)

3.2.2 Parsování vstupu

Popisovaná část programu v této kapitole odpovídá první buňce z diagramu na obrázku č. 5 – „Parsování vstupu“.

Na vstupu dostane RTD trasu (získanou z mapy.cz ve formátu souboru gpx) a mapu (získanou z openstreetmap.org ve formátu souboru OSM).

Soubor s trasou obsahuje v podstatě jen body dané zeměpisnými souřadnicemi, které trasu určují a jsou seřazeny podle směru trasy. Tyto body si pro další snazší manipulaci uložíme do seznamu.

Soubor s mapou obsahuje mnoho dat, které nebudou v naší verzi RTD potřeba, vybereme si z ní jen některé *uzly* a *cesty*, které jsou dostatečně blízko trasy. Cesty budou potřeba pro rozpoznání křižovatek a informaci, do jakého směru zatáčíme a jakou ulicí se dáme, má-li jméno. Některé *uzly* také slouží k rozpoznání křižovatek a jiné zastupují význačné body z krajiny, které budeme chtít nahlásit chodci.

Pro další snadnější manipulaci si tato vybraná data z mapy upravíme tak, že ke každému *uzlu* vytvoříme objekt `ExtendedNode`, který obsahuje tento *uzel* a spolu
s ním počet cest uzlem procházejících a seznam těchto cest. Když bude RTD potom v budoucnu hledat v blízkosti naší trasy křižovatku, stačí prozkoumat ty ExtendedNode, kterými prochází více než jedna cesta. Z mapy tedy získáme seznam s těmito ExtendedNode.

**Obrázek č. 6:** Vlevo nahoře vidíme červeně vyznačený node, který bude zařazen mezi ExtendedNodes, protože jím prochází hned tři ways, vyznačené na dalších částech obrázku. Je vidět, že v tomto místě se opravdu nachází křižovatka. Zdroj: openstreetmap.org

### 3.2.3 Generování popisu cesty

Nyní (bez technických detailů) popíšeme, jakým způsobem náš RTD naviguje chodce.

Následující podkapitoly vysvětlují základní principy, které jsou použity v programu, podrobnější popis s vysvětlivkami k samotnému kódu je v kapitole Technická implementace na straně 13.

Z parsování máme k dispozici dva seznamy, jeden s trasovými body a druhý s blízkými ExtendedNode vzhledem k trase.
RTD postupně prochází seřazené body z trasy a sleduje, co se kolem těchto bodů v mapě nachází. Pokud je tam něco, co je užitečné sdělit uživateli, tak to udělá. Nědůležité informace se snaží vynechat.

Na trase rozlišujeme dvě situace, a to křižovatku nebo delší úsek bez možnosti odbočení. Delším úsekem myslíme úsek delší než 250 metrů. (Soudíme totiž, že do vzdálenosti zhruba 250 metrů chodec nutně nepočítá s dalším odbočením, ale pro delší úseky je lepší ho upozornit, že teď chvíli odbočovat nebude, aby neměl obavu, že odbočku minul.)

Pokud trasa vede přes křižovatku, tak RTD upozorní, že jsme na křižovatce, řekne, jakou cestou se dát, a je-li v blízkosti křižovatky nějaký význačný bod, tak ho zmíní. Význačné body rozdělí do čtyř oblastí kolem křižovatky, podle toho, jestli je oblast z pohledu chodce za či před křižovatkou, a dále podle toho, jestli je napravo či nalevo od cesty. Podle pozice objektu a zatáčení cesty z pohledu chodce RTD zformuluje navigační instrukci. Příklad instrukci:

| Na další křižovatce odbočte doleva ulicí „Zámecká“, kde napravo uvidíte restautaci jménem „Tre colori“. |

Ukázkou snahy dávat informace do souvislosti je, že je-li na trase více křižovatek za sebou, na kterých má jít chodec rovně, RTD nekomentuje každou křižovatku zvlášť, ale shrne je do jedné věty, která říká, že uživatel má projít několik křižovatek rovně, případně zmínit významné objekty, které jsou po cestě, jsou-li takové. Příklad:

| Další 2 křižovatky projdete rovně. |

Pokud máme na trase delší úsek bez odboček, RTD uživateli sdělí, jakou vzdálenost teď má ujít bez možnosti odbočení a jaké významné objekty cestou potká.

| Nyní pokračujte asi 1.5 kilometrů bez možnosti odbočení. |

Zvlášť se řeší úsek cesty od poslední křižovatky do cíle, zde RTD jen řekne, jak daleko to uživatel od poslední křižovatky k cíli má.

| Po 42 metrech dojdete do cíle. |
3.2.4 Důkladněji k sběru a rozčlenění dat a tvorbě výstupu

Předchozí kapitolu popíšeme teď více z pohledu, jak je generování rozděleno na části v programu, aby bylo vidět souvislost s diagramem na obrázku č. 5 v úvodu. Takže pro každou zbývající buňku z diagramu popíšeme, co se děje v té části programu, která jí přísluší.

- „Sběr dat“: RTD prochází seznam trasových bodů a hledá důležité události trasy (křižovatky a dlouhé úseky bez odbočení) a pro každou nalezenou událost vytvoří objekt obsahující informace, které se dál budou rozvíjet pro uživatelský výstup (příklady těchto informací: zda jde o křižovatku nebo úsek bez odbočení, jestli je to poslední úsek cesty, je-li to křižovatka, tak nás zajímá „cesta“, kterou chodce má pokračovat a směr kam odbočí). Tyto objekty se uloží do seznamu.

- „Rozčlenění dat“: Vzniklý seznam s obecnými objekty je převzat další částí programu, která každý obecný objekt transformuje na konkrétnější, který pak dokážeme zpracovat na textový výstup. Vznikají zde čtyři druhy objektů: samostatná křižovatka, skupina křižovek, delší úsek bez možnosti odbočení, poslední úsek. Vzniklé konkrétní objekty jsou opět uloženy do seznamu a předány dál.

- „Tvorba výstupu“: Poslední část programu vytvoří z konkrétních objektů textové řetězce – instrukce pro uživatele. Pro každý typ konkrétního objektu je funkce, která z informací v objektu uložených vytvoří navigační instrukci, tedy větu podobnou příkladům z minulé kapitoly.

3.2.5 Uživatelský výstup

Instrukce pro uživatele jsou tvořeny na základě několika šablon, jak může věta s pokynem vypadat. Například pokud cesta zatáčí doprava a zároveň je vpravo před křižovatkou významný objekt A, věta pro uživatele začne: „Kolem objektu A zatočte doprava...“. Pokud není před křižovatkou vpravo žádný významný objekt a není tedy kolem čeho zatáčet, začátek věty bude: „Na další křižovatce odbočte doprava...“ . Pokud je před křižovatkou nějaký objekt B vlevo, instrukce bude znít: „Poté, co
vlevo minete objekt B, odbočte doprava…“.

V některých místech věty je za účelem větší variability výstupu náhodně vybráno z několika možných výrazů. Například tam, kde má být sloveso vyjadřující změnu směru chůze na křižovatce, jsou na výběr možnosti: „zatočte“, „odbočte“, „jděte“. Nebo na místě slovesa vyjadřujícího, že chodec může vidět nějaké význačné objekty, se objeví například „minete“ nebo „uvidíte“.

3.2.6 Qgis

Pro vizualizaci dat jsme během vývoje programu používali program Qgis. Sloužil především při kontrole, jestli program dělá to co má. Dají se v něm přehledně zobrazit data z OSM, vybírat z nich určité skupiny objektů a podobně.
4 Technická implementace

V této kapitole se důkladněji podíváme na to, jak je RTD technicky implementováno a rozebereme důkladněji jeho hlavní části.

4.1 Názvosloví pro tuto kapitolu

V této části textu budeme uzly, cesty a relace nazývat jejich původními anglickými názvy, tedy jako node(s), way(s), relation(s), protože takto se s nimi pracuje i v programu. Nodes, ways a relations mají tedy původ v OSM datech, tedy v soubořu s mapou na vstupu. Dále pojmem bod trasy rozumíme objekt vytvořený z elementu <trkpt>... </trkpt>, tedy bod náležící trase, kterou popisujeme, který má původ v gpx souboru s trasou na vstupu. Mluvíme-li o nějaké funkci z programu, značíme ji závorkami za jejím názvem, například get_points_from_route().

4.2 Popis programu

Připomeneme si schéma programu, a popíšeme jeho části více technicky.

Vstup (trasa,mapa)

Parsování vstupu

Sběr dat

Rozčlenění dat

Tvorbá výstupu

Výstup (text)

Parsování vstupu

Vytvoření seznamů obsahujících potřebná data z mapy a tras.

Sběr dat

Průchod trasou a zaznamenání významných částí trasy.

Rozčlenění dat

Významné části trasy rozděleny podle zásobu zpracování na výstup.

Tvorbá výstupu

Sestavení instrukcí k navigaci do částech trasy.

Obrázek č. 7: Opakování diagramu – schématické znázornění běhu programu

4.2.1 Parsování vstupu

Pro tuto část programu jsou klíčovými částmi kódu třída Map a funkce get_points_from_route().

Program na vstupu dostane soubor s trasou (GPX Exchange format) a soubor
s mapou (OSM data). Pro jejich parsování se používá knihovna lxml.etree, která po každý node nebo way z OSM souboru a pro každý trkpt (zkratka ze slov track point) z GPX souboru vytvoří příslušný objekt.

Třída Map naparsuje OSM soubor do několika seznamů, kde každý seznam obsahuje nějaký typ dat z OSM. Momentální verze programu používá dál pouze seznam s ExtendedNodes, kde jsou body, které mohou být zajímavé pro výstup, tedy ty, které by mohly být křižovatkou (tedy se v nich křižuje dvě a více cest) nebo ty, které mají aspoň jeden tag (mohly by tedy být význačným bodem v krajině).

Protože máme v programu několik objektů podobného charakteru, které vyjadřují, že jde o bod v mapě, sjednotíme je do wrapper třídy Point. Instancí třídy Point může být node z mapy, bod trasy, nebo instance třídy ExtendedNode. Třída Point má například metody vracející zeměpisnou délku a šířku zadaného bodu, metodu, která zjistí, jestli obsahuje zadaný bod určitý tag, jestli jsou dva nodes dost blízko sebe a podobně.

Funkce get_points_from_route() získá ze souboru GPX s trasou seznam bodů z trasy.

4.2.2 Sběr dat

Se vzniklými seznamy (body z trasy a ExtendedNodes) pracuje funkce get_important_situations_during_route().

V této funkci je klíčový cyklus, který prochází body z trasy tak, jak jdou po sobě, a tvoří objekty s informacemi o důležitých navigačních situacích na cestě. Na zveme ho hlavní cyklus.

Představme si, že jsme v nějaké iteraci tohoto hlavního cyklu, a jsme tedy na nějakém konkrétním bodě z trasy, nazveme ho A.

Nejdříve si vybereme jeho blízké extendedNodes, které mají tag s klíčem amenity nebo tourist, co má hodnotu, která je zajímavá pro chodce (například hospody, lékárny, kontejnery, turistické informace, vyhlídky) a uložíme si je do seznamu. Tyto vybrané extendedNodes budeme nazývat významné. ExtendedNodes hledáme do vzdálenosti 20 metrů od bodu z trasy.

Konstanty 20 a 11 metrů byly zvoleny na základě zkušenosti, že na vyzkoušených vstupech s nimi RTD dobře fungovalo.

Případný nalezený central_node_of_crossroads a významné nodes odstraníme z centrálního seznamu s extendedNodes, abychom je už určitě nezmínili podruhé, kdyby byly blízko i dalším bodům z trasy.

Pokud jsme nalezli central_node_of_crossroads, tak víme, že jsme na křížovatce a chceme zjistit, jakou cestu z křížovatky doporučit chodci.

Pro tento účel potřebujeme tři význačné body z trasy:
1. ten který je nejblíž bodu central_node_of_crossroad a který nazveme actual (nemusí to být stejný bod z trasy jako ten, který je aktuální vzhledem k hlavnímu cyklu, naopak je často v cestě někde dále)
2. bod z trasy který je v trasovém pořadí hned za bodem actual a nazveme ho descendant
3. bod z trasy, který je v trasovém pořadí hned před actual a nazveme ho ancestor.

Tyto tři body z trasy vytvoří instanci třídy PointsAroundCrossroads a tuto instanci nazveme threee_points_of_crossroads. Detaily o tom, jak se vybere správná cesta, najdeme v kapitole Křižovatky na straně 17.

Z úhlu mezi three_points_of_crossroads dále určíme, jak moc bude chodec v křížovatce zahýbat doprava či doleva, nebo půjde-li rovně. Tato informace bude taky zmíněna na výstupu.

Všechny informace o důležitém úseku trasy se uloží do instance třídy NavigationInformation a pole všech těchto NavigationInformation se předá v programu dá.

15
4.2.3 Rozčlenění dat

Funkce `sort_by_type_of_important_situation()` se stará o rozdělení významných situací na trase, aby se daly zpracovat na textový výstup.

Takto vznikají instance tříd:

1. `NavigationInformationAboutOneCrossroad` (pokud je to informace jen o jedné křižovatce)

2. `NavigationInformationAboutManyCrossroads` (pokud je to informace o několika křižovatkách, které jsou v trase po sobě, kde má jít chodec vždy rovně, a proto se dají snadno shrnout do jedné instrukce)

3. `NavigationInformationAboutPartWithoutCrossroads` (pokud to bude delší úsek bez odbočení)


Každá tato instance obsahuje už jen informace, které potřebuje pro svůj typ výstupu, na rozdíl od obecného `NavigationInformation`, které obsahovalo všechno.

Dohromady tedy bude navigačních instrukcí po tomto procesu stejně, nebo méně, pokud se nějaká skupina křižovatek shrne do jedné instrukce. Seznam těchto konkrétních instrukcí jde dále ke zpracování na výstup.

4.2.4 Tvorba výstupu

Textové řetězce s pokynem kam jít a co lze vidět, které budou vytisknuty na výstupu, tvoří funkce `convert_instructions_into_strings_for_output()`.

Projede v cyklu všechny instrukce, a pro každou zavolá příslušnou funkci, která podle informací z instrukce vytvoří textový řetězec.

Nejsložitější problém tvorba textu v případě, že se zpracovává instance třídy `NavigationInformationAboutOneCrossroad`, tedy jedna křižovatka. Popíšeme proto více tuto situaci, ostatní jsou podobné.

Funkce `prepare_instruction_one_crossroads_for_output()`, která je součástí třídy `NavigationInformationAboutOneCrossroad`, má na starosti popis jedné křižovatky.
Pokyn je víceeméně rozdělen na tři části:

1. nejdříve se mluví o tom co je před křižovatkou,
2. poté o křižovatce,
3. a nakonec o tom co je za křižovatkou.

První nebo třetí část může chybět, pokud není nic zajímavého ke zmínění před nebo za křižovatkou.

První část se liší podle toho, jestli jde uživatel na křižovatce rovně nebo zahýbá. Pokud zahýbá, pokyn vydá funkce `get_string_when_turn_left_or_right()`; takový pokyn je složitější. Druhá část popisu je kratší, do pokynu se přidá směr a cesta, kterou má chodec jít, pokud tato cesta má název. Třetí část textu vytvoří funkce `describe_objects_after_crossroads()`, kde se už jen dodá, co by měl uživatel vidět hned po průchodu křižovatkou.

Používá se zde ještě několik dalších pomocných funkcí, například `return_objects_define_by_amenity_as_string_for_print()`, která přetvoří seznam význačných bodů na textový řetězec a podobně.

4.3 Křižovatky

V této kapitole si řekneme více o výběru správné cesty na křižovatce.

Jsme tedy v situaci, kdy RTD objevil v blízkosti `bodu z trasy` takový `node`, kterým prochází alespoň tři `ways`, a tedy je pravděpodobně ve středu křižovatky. Tento `node` nazýváme `central_node_of_crossroads`.

Víme, že se jedná o `ExtendedNode`, tedy obsahuje u sebe rovnou i informaci kolik `ways` jím prochází a jejich seznam. Tyto `ways` byly předem vybrány tak, že jsou to právě ty `ways`, které obsahují `tag` s klíčem `highway`. Cesty které mají tento `tag` reprezentují právě různé druhy cest, ulic, chodníků, silnic a podobně.

4.3.1 Sjednocení dvou typů `ways` v křižovatce

Některé z těchto `ways` v křižovatce končí (takové budeme nazývat `půlcesty`) - takže `central_node_of_crossroads` je na některém konci `way`, tedy v pořadí `nodes`, které `way` obsahuje, je jako první nebo poslední.

Pokud mám nåkterou way, která tímto central_node_of_crossroads prochází, rozdělíme ji na dvě půlcesty. Provedeme to tak, že vytvořím dvě kopie původní way, a půlcesty z nich utvořím tak, že z každé kopie odstraníme nodes, které ji nenáleží – tedy pro jednu půlcestu odstraníme ty, které se v seznamu nodes oné way nachází před central_node_of_crossroads, a pro druhou půlcestu odstraníme zase nodes za central_node_of_crossroads, přičemž jelikož central_node_of_crossroads nebyl na žádném z krajů toho seznamu, v obou případech se nějaké nodes opravdu smažou.

Pokaždé, když nějakou way rozdělíme na dvě, upravíme také počet cest v central_node_of_crossroads, aby odpovídal aktuálnímu počtu ways jím procházející, tedy tento počet zvýšíme o jedna.

Tento proces sjednocení dvou typů cest na křižovatce probíhá sice už když se hledá v blízkosti bodu z trasy nějaký central_node_of_crossroads, ale pro přehlednost to popisujeme až zde.

4.3.2 Výběr správné cesty

Víme, že k dispozici máme tři význačné body z trasy: actual (který je z trasových bodů nejblíž bodu central_node_of_crossroad), ancestor (trasový bod před bodem actual) a descendent (trasový bod následující za bodem actual).

S jejich pomocí vybere funkce choose_the_right_way_to_go() správnou cestu, a to následovně: Správná cesta je většinou ta, která z křižovatky vychází co nejpodobnějším směrem jako naplánovaná trasa. Ve většině případů to funguje dobře, avšak vzhledem k rozmanitosti dat, se kterými pracujeme, se nedá zaručit, že to vždy. Takže funkce choose_the_right_way_to_go() najde a vrátí way takovou, že úhel (daný body descendent, central_node_of_crossroad a druhým
node oné way – tj. node, který je částí oné way a je vzdálenostně druhý v pořadí od central_node_of_crossroad) je minimální. Tato vybraná way bude tedy uživateli nahlášena jako ta, kterou se má dál vydat. Příklad postupu u výběru křižovatky vidíme na obrázku č. 8.

4.4 Význačné objekty

Jako objekty, pomocí kterých uživateli upřesníme cestu, jsme vybrali některé druhy nodes, a to takové nodes, které mají tag amenity nebo tag tourist s povolenou hodnotou (podle wiki.openstreetmap.org[6] značka amenity „slouží k mapování prvků občanské vybavenosti, sloužící obyvatelům a návštěvníkům oblasti“ a značka tourist je „značka pro mapování míst a objektů, které jsou zajímavé pro turisty“; na základě naší zkušenosti vystihují tyto značky velkou část objektů, které chceme zmínit uživateli).

Povolené hodnoty tagu amenity v RTD jsou následující: restaurant, bar, cafe, pub, fast_food, ice_cream, nightclub, fountain, clock, bench, shelter, atm, toilets, recycling, waste_basket, bicycle_parking, telephone, post_box, school, bank, theatre, pharmacy, police, library. Povolené hodnoty tagu tourist v RTD jsou následující: guest_house, hotel, hostel, information, museum, viewpoint, alpine_hut.
V programu jsou povolené hodnoty uloženy ve slovníku, kde navíc každá u sebe má český překlad v takových pádech, které budeme potřebovat na výstupu. Když chceme pak například uživatele upozornit na node který má amenity s hodnotou recycling v sedmém pádě, v našem slovníku program objeví, že má vytisknout slovo „odpadkovými kontejnery“.

Snaha byla vybrat objekty, které by při popisu potenciálně zmínil člověk. Pokud by se v budoucnu ukázalo, že některé k popisu vhodné objekty tam chybí, dají se doplnit do existujícího nebo nového seznamu.

### 4.5 Soubory s kódem a testovací data

#### 4.5.1 Soubory s kódem

Pro fungování programu je třeba mít nainstalované následující moduly pro Python verze 3: geopy, overpy, random, copy, lxml.etree, math. Pro spuštění unit testů je třeba mít navíc moduly unittest a filecmp.

Program je rozdělen do deseti souborů s kódem v Pythonu verze 3.

Soubor „main.py“ obsahuje hlavní funkci, která volá čtyři důležité funkce (nazveme je centrální), kde každá odpovídá svou činností jedné buňce z diagramu na obrázku č. 7 (jak je popsáno v kapitole Technická implementace – Popis programu na straně 13).

Tyto centrální funkce jsou uloženy v souboru „functions.py“. V tomto souboru jsou navíc třídy, které tvoří konkrétní objekty s informacemi souvisejícími s diagramovými buňkami „Rozčlenění dat“ a „Tvorba výstupu“ z obrázku č. 7.

Dalších sedm souborů obsahuje pomocné funkce pro hlavní funkci a centrální funkce. Jmenovitě jsou to soubory:

- „functions_parsing_input_map_and_route.py“ (část kódu, která se zabývá parsováním vstupních dat, tedy souvisí s buňkou z diagramu „Parsování vstupu“ z obrázku č. 7)
- „functions_important_objects_and_lingvistics.py“ (část kódu se seznamy důležitých objektů, které chceme hlásit uživateli, a se seznamy typů cest s jejich
českými gramatickými tvary a souvisejícími funkcemi)
• „functions_structure_for_ancestor_and_descendant.py“ (zde jsou uloženy třídy pro práci s třemi význačnými body křižovatky, dříve zmíněnými ancestor, actual a descendant)
• „functions_sorting_points.py“ (funkce pro řazení bodů podle cesty)
• „functions_angles_computing.py“ (geometricky zaměřené funkce počítající úhly a vektory)
• „functions_for_ways.py“ (funkce pro práci s ways)
• „functions_structure_for_points.py“ (obsahuje třídy Point a ExtendedNode, ve třídě Point je hodně funkcí pro práci s nodes)

Poslední soubor „tests.py“ obsahuje unit testy.

4.5.2 Testovací data

V příloze bakalářské práce jsou dva testovací vstupy a obrázky map, které jim odpovídají.

Jeden vstup testuje trasu v Praze v oblasti Bubeneč. Pro spuštění programu jsou tedy k dispozici OSM soubor s mapou „praha_bubenec.osm“ a GPX soubor s trasou „praha_bubenec.gpx“, pro kontrolu výsledného popisu trasy, kterou RTD vydá, je tu obrázek mapy „praha_bubenec.png“.

Druhý vstup je z přírodní oblasti Staré Hamry – Gruň a obsahuje obdobně soubory „grun.osm“, „grun.gpx“, „grun.png“.
5 Zhodnocení řešení

Route-to-description umí správně popsat cestu jen za některých podmínek. Data z OSM jsou rozmanitá, podobně jako reálný svět, který jimi modelujeme. Proto jsme v této práci vyřešili jen některé situace, které se na cestě mohou objevit. Dále pokud v blízkosti trasy nejsou žádné objekty, které hlásíme uživateli, tak ve vygenerovaném popisu nebude žádné zmíněny, a popis z RTD se pak příliš neliší od popisů získaných z mapy.cz nebo maps.google.com.

5.1 Za jakých okolností program funguje

RTD je připraveno na následující situace:

5.1.1 Křižovatka, kde na sebe cesty přímo navazují

To znamená, že se skládá z ways, které obsahují tag s hodnotou highway, a tyto ways všechny obsahují společný node uprostřed křižovatky (jako v kapitole Křižovatky). Jsou to tedy křižovatky, které jsou v OSM zaznačeny tak, že jsou křížením pouze cest a není v nich žádný větší objekt, například náměstí nebo kruhový objezd.

Obrázek č. 9: Příklady správně popsaných křižovatek programem RTD. Zdroj: mapy.cz
5.1.2 Úseky bez odboček

O delších úsecích bez odboček RTD sdělí jejich délku a vyjmenuje význačné objekty kolem.

Obrázek č. 10: Příklady správně popsaných křížovek programem RTD. Zdroj: mapy.cz

Obrázek č. 11: Příklad popsání rovného úseku delšího než 250 metrů. Zdroj: openstreetmap.org

Na další křížovatce zatočte doprava. Na další křížovatce jděte rovně.
Další křížovatku projdete rovně, cestou "Bezručova".

Nyní pokračujte asi 282 metrů bez možnosti odbočení. Cestou uvidíte turistické informace jménem "Pomník valašské ovci".
5.1.3 Pokud jsou OSM přesné

Protože RTD u navigace na správné cesty v křižovatkách i u popisu význačných objektů čerpá z OSM, jeho výstup je správný, pokud jsou tato data přesná. Pokud například nějaká cesta v OSM zaznačena není, v důsledku toho může RTD vynechat informaci o křižovatce. To platí i pro význačné body: pokud je u trasy zaznačen význačný bod, který tam ve skutečnosti není, RTD to pochopitelně neodhalí. Spolehlivost RTD je tedy závislá na přesnosti OSM dat.

5.1.4 Nezvyklé značení ways blízko křižovatky

Tento bod byl dříve problémový, proto ho zde zvlášť zdůrazníme.

Program chyboval, pokud byly cesty v blízkosti křižovatky zaznačeny útržkovitě, jak je tomu na příkladu na obrázku č. 12 na další straně. Místo aby zde byly například čtyři ways končící právě v křižovatce, jedna z nich přečnívá za křižovatku a druhá je rozdělena blízko křižovatky na tři části. Tehdy se všechna tato rozdělení hlásila jako křižovatka, takže RTD v tomto místě postupně nahlásilo tři křižovatky.

V aktuální verzi programu jsou ihned vyřazeny ze seznamu potenciálních křižovatek ty křižovatky, ze kterých vycházejí jen dvě cesty, proto už tato rozdělení ways nejsou zbytečně zmíňovaná na výstupu.
Dříve problematická křižovatka

Horizontální cestu značí tři "ways, na obrázku červené.

Vertikální ulice je značena dvěma "ways", z nichž jedna přesahuje přes křižovatku a druhá naopak nedosahuje ke křižovatce.

Obrázek č. 12: Křižovatka s nezvykle rozdělenými cestami. Zdroj: openstreetmap.org
5.1.5 Příklady dobře popsaných tras

U následujících dvou příkladů uvádíme pro srovnání vždy nejdříve obrázky cesty z mapy.cz a z maps.google.com, potom popis, vygenerovaný z RTD, a nakonec pro porovnání také slovní popisy z mapy.cz a maps.google.com

5.1.5.1 První příklad

1. Trasa na maloměstě ve Frýdlantě nad Ostravicí, z autobusového nádraží do ulice Hamernická. Pro srovnání uvádíme i popisy z mapy.cz a z maps.google.com.


Obrázek č. 14: Trasa vyznačena v openstreetmap.org, ze kterých máme data k popisu. Zdroj: openstreetmap.org
Další 2 křížovatky projdete rovně cestou "Nádražní". Poté, co vlevo minete lavičku a stojan na kola, jděte doprava cestou "Hlavní". Další 2 křížovatky projdete rovně cestou "Hlavní". Cestou vpravo uvidíte: bankovní automat a lékárnu jménem "Lékárna Frýdlant". Na další křížovatce odbočte doleva ulicí "Smetanova". Na další křížovatce jděte rovně ulicí "Smetanova", kde napravo spatříte restauraci jménem "Imrvere". Na další křížovatce jděte doprava ulicí "Hamernická". Po 42 metrech dojde do cíle.

_Obrázek č. 15: Popis vygenerovaný pomocí RTD_

5.1.5.2 Druhý příklad

2. Trasa v hornaté krajině na hřebenu Gruně, z chalupy na autobus:


Obrázek č. 18: Trasa vyznačena v openstreetmap.org, ze kterých máme data k popisu. Zdroj: openstreetmap.org
Na další křižovatce zatočte doleva, kde vpravo spatříte odpadkový koš.
Nyní pokračujte asi 449 metrů bez možnosti odbočení. Cestou uvidíte turistické informace jménem "Školisko".
Na další křižovatce zatočte mírně doprava.
Další 2 křižovatky projdete rovně. Cestou vpravo uvidíte: vysokohorskou chatu jménem "hotel Charbulák". Nalevo potkáte: turistické informace s názvem "Gruň (hot.Charbulák)".
Na další křižovatce zatočte mírně doprava.
Na další křižovatce jděte rovně.
Nyní pokračujte asi 475 metrů bez možnosti odbočení.
Na další křižovatce zatočte mírně doprava.
Nyní pokračujte asi 328 metrů bez možnosti odbočení.
Na další křižovatce jděte mírně doleva.
Na další křižovatce jděte rovně.
Nyní pokračujte asi 1.5 kilometrů bez možnosti odbočení.
Na další křižovatce pokračujte rovně, kde vpravo spatříte turistické informace jménem "Jamník (bus)".
Po 41 metrech dojdete do cíle.

Obrázek č. 19: Popis vygenerovaný pomocí RTD

5.2 Za jakých okolností program nefunguje

Rozmanitých případů, kde program udělá chybu, která může chodce zmást, je mnoho, uvedeme si zde některé.

5.2.1 Pokud je trasa zaznačena nečekaně vzhledem k datům z OSM

Trasu pro program čerpáme ze stránek mapy.cz, zatímco data pro určení křížovatek a význačných objektů ze stránek openstreetmap.org. To proto, že jsme neodhalili snadný způsob, jak z openstreetmap.org stáhnout trasu. Pokud se pak trasa z mapy.cz nějakým způsobem odchyluje od pozice, kde by měla být v OSM mapách, může to způsobit chyby v popisu RTD. Čím více cesty (a tedy i celá trasa) v mapy.cz odpovídají cestám (a pak i popisu trasy) v openstreetmap.org, tím je pravděpodobnější, že popis od RTD bude správný. Pokud se cesty nějak odchylují, nastane nejspíš v popisu chyba.

Obrázek č. 21: Mapy.cz navigují chodce na křížovatce nejdříve trochu doprava a pak až doleva – v cestě se vytvoří jakoby „zub“. RTD ho přeloží tak, že chodce navádí nejdříve doprava do ulice Thunovská, a poté až správně doleva. Zdroj: mapy.cz, openstreetmap.org
5.2.2 Velké plochy v křižovatce

Program špatně naviguje u jiných typů křižovatek než jsou popsány v předchozí kapitole. Například jde-li o náměstí, složitý dopravní uzel a podobně. Nachází se zde zpravidla hodně cest, program je všechny zmíní, zatímco člověk by informaci shrnul do jedné věty.

Obrázek č. 22: Trasa přes Staroměstské náměstí, které obsahuje velké množství ways, některé označují jeho různé části, některé cesty které do něj ústí, a RTD tady vygeneruje hodně z pohledu člověka redundantních instrukcí. Zdroj: mapy.cz


Obrázek č. 23: Popis k trase na Staroměstském náměstí. RTD zmiňuje všechny křižovatky vzhledem k cestám, co ústí do náměstí. Pro člověka praktičtější je například oznámit, že dojde na náměstí a má pokračovat ulicí v protějším roku.

Na další křižovatce jděte doleva, ulicí "Dejvická".
Další 2 křižovatky projdete rovně, cestami "Dejvická", "Čs. armády". Cestou vpravo uvidíte: poštovní schránku a stojan na kola.
Na další křižovatce odbočte mírně doleva, na silnici "Jugoslávských partyzánů". 
Na další křižovatce odbočte ostře doleva, ulicí "Šolínova".
Na další křižovatce zatočte doprava.
Po 65 metrech dojde te do cíle.

Obrázek č. 25: Popis k trase na Vítězném náměstí. Člověk by pravděpodobně jen řekl, že na velkém kruhovém objezdu půjdeme druhým výjezdem
6 Možnosti vylepšení

V této kapitole si řekneme o věcech, které by se na RTD daly vylepšit, nebo zcela změnit, kdyby to v budoucnu někdo chtěl napsat celé znova a lépe.

6.1 Navigační a popisová vylepšení

6.1.1 Navigování do křižovatky

Když RTD dává uživateli pokyn, kam má jít na křižovatce, navádí ho pomocí směru (ostře doprava, doprava, mírně doprava, rovně, mírně doleva, doleva, ostře doleva), popřípadě pomocí názvu ulice. U městských křižovatek to docela dobře stáčí, protože tam jsou málokdy více než čtyři cesty vedoucí z křižovatky.

V přírodě a krajině to ale ukazuje jako nedostatečné, protože jsou tam cesty nepojmenované, může jich být hodně v jedné křižovatce, a bývá více cest, které na příklad subjektivně zatáčí doprava, a pro chodce nemusí být zjevné, která z nich zatáčí doprava nebo mírně doprava. Proto by bylo vhodné přidat v těchto situacích nějakou informaci navíc, například, jděte druhou cestou doprava.

6.1.2 Pořadí objektů v cestě

Při vývoji RTD jsme si nevšimli, že by tento bod způsoboval chyby u našich zkoušených vstupů, ale není vyloučeno, že by někdy mohl, proto ho zde taky zmíníme.

Když máme nějaké význačné objekty u cesty, je třeba je seřadit podle toho, jak jsou po sobě, když tou cestou jdeme. V RTD se používá pro seřazení těchto objektů jednoduchý algoritmus, který pro bod trasy A seřadí jeho blízké objekty vzestupně podle vzdálenosti k A, a poté pro bod B, který následuje po A, sestupně podle vzdálenosti k B. Tento algoritmus většinou body seřadí optimálně, ale existují případy, kdy ne. Pro větší záruku spolehlivosti by se tedy mělo nahradit toto řazení nějakým, které je správné za všech okolností.
6.1.3 Výběr význačných objektů

Význačné objekty RTD vybírá pouze v blízkosti bodů z trasy – oproti tomu, kdyby je například vybíralo v pásu dané šířky kolem celé trasy. Pokud bude trasa získaná z mapy.cz mít třeba velkou mezeru mezi nějakými dvěma body, a byly by tam nějaké zajímavé objekty pro popis, RTD je může takto minout.

Původně jsme mysleli, že význačné objekty jsou stejně potřeba hlavně v místě odbočování (a tam body z trasy většinou jsou, takže RTD zde objekty pravděpodobně objeví), a při delší rovné cestě, kde mohou body z trasy chybět, to nemá takovou navigační důležitost (a v mnoha případech tomu tak je), ale s vývojem práce si myslíme, že by i nalezení více objektů na rovných kusech trasy mohlo mít svůj účel.

6.2 Technické nápady

6.2.1 Automatické získání mapy

Jedním z plánů při vývoji v RTD bylo zařídit, aby si program na základě trasy, kterou dostane na vstupu, sám stáhl potřebnou mapu (a nemusel to dělat uživatel). Při pokusu tohle zařídit se bohužel vyskytly na výstupu chyby, jejichž původ se nepodařilo objasnit, takže RTD tuto funkčnáliitu nakonec nemá.

Možná cesta by mohla vést přes dotazování pomocí Overpass API[7] (pomocí kterého jsme to zkoušeli i u RTD, ale nepodařilo se nám to dotáhnout ke zdárnému konci). Výhodou je zde, že se dá stáhnout jen oblast mapy, která je blízko trasy (na rozdíl od toho, když si stahujeme mapu z OSM ručně, tak musíme vybrat vždy obdélník, který obsahuje i spoustu dat, která nevyužijeme), takže by se chod programu v parsovací části urychlil.

6.2.2 Větší využití hotových nástrojů pro práci s OSM

Část programu se zabývá parsováním xml dokumentu, ve kterém je zachycena mapa, ze které získáváme význačné objekty a další data. Tuto část by mohlo být elegantnější nahradit prací s nějakou knihovnou, vytvořenou přímo pro zpracování dat z OSM, a i v dalších částech programu pak s touto knihovnou pracovat.
7 Jak požívat Route-to-description

7.1 Jak získat vstupní data

Jako vstupní data potřebujete soubor s trasou (*GPX Exchange format*) a soubor s mapou (*OSM data*).

Soubor s trasou získáte tak, že na stránkách *Mapy.cz* naplánujete trasu, a poté kliknete na tlačítko vpravo *Exportovat*. Trasu uložíte a přesunete do složky, kde máte zdrojové kódy s programem.

Soubor s mapou získáte tak, že si na stránkách *OpenStreetMap* najdete oblast, ve které jste si naplánovali trasu, poté vlevo nahoře kliknete na tlačítko *Export*, a vyberete možnost *Ručně vybrat jinou oblast*, a vyberete oblast která pokrývá celou vaši trasu. Poté potvrdíte modrým tlačítkem *Export* a trasu uložíte. Přesuňte ji opět do složky, kde jsou zdrojové kódy k RTD.

7.2 Jak spustit program

Pro spuštění programu jsou potřeba následující moduly pro python: *geopy, overpy, random, copy, lxml.etree, math* a python ve verzi alespoň 3.

Chcete-li spustit RTD, otevřete si příkazovou řádku a pomocí příkazu *cd* se přesuňte do složky, kam jste si uložili zdrojové kódy programu. Poté můžete spustit RTD bez parametrů nebo s parametry následovně:

7.2.1 Bez parametrů

Spustíte-li v příkazové řádce příkaz:

```
python3 main.py
```

pak se RTD spustí na dvě testovací trasy, jejichž obrázky map jsou přiloženy.

7.2.2 S parametry

Pokud chcete vytvořit popis vlastní trasy, spusťte RTD se třemi parametry:

1. jméno souboru s mapou
2. jméno souboru s trasou

3. jméno výsledného textového souboru s popisem trasy

Například, mám-li uloženou mapu v souboru se jménem brno_jih.osm a trasu v souboru se jménem do_obchodu.gpx a chci pojmenovat v budoucnu vzniklý soubor s popisem cesty jako cesta_na_nakup.txt, tak příkaz, který zadám do příkazové řádky bude:

```
python3 main.py brno_jih.osm do_obchodu.gpx cesta_na_nakup
```

7.3 Co je výstupem

Výstupem je textový soubor, který se objeví ve stejné složce, kde máte uložené zdrojové kódy s RTD, a obsahuje textový popis trasy.
8 Závěr

V práci jsme se zabývali průzkumem možností a nástrah při navigaci chodce na základě dat z openstreetmap.org, a samotným vývojem programu, který generuje člověkem snadno následovatelný popis cesty.

Program by se dal v budoucnu vylepšit k větší spolehlivosti, aby dokázal chodce navigovat ve více rozmanitých situacích. Taky by se dal rozšířit o audio navigaci.
9 Seznam použité literatury

## 10 Seznam použitých zkratek

<table>
<thead>
<tr>
<th>AS</th>
<th>Automatické systémy</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSM</td>
<td>openstreetmap.org</td>
</tr>
<tr>
<td>RTD</td>
<td>Route-to-description</td>
</tr>
</tbody>
</table>
11 Přílohy

11.1 Zdrojové kódy

Kód program je rozdělen do deseti souborů

1. main.py
2. tests.py
3. functions.py
4. functions_parsing_input_map_and_route.py
5. functions_important_objects_and_lingvistics.py
6. functions_structure_for_ancestor_and_descendant.py
7. functions_sorting_points.py
8. functions_angles_computing.py
9. functions_for_ways.py
10. functions_structure_for_points.py

11.2 Testovací data

Program může být otestován na dvou trasách. Pro každou jsou zde tři soubory:

- soubor s mapou, s příponou .osm
- soubor s trasou, s příponou .gpx
- soubor s vizualizací trasy z mapy.cz, pro porovnání s výstupem programu

Seznam souborů:

1. praha_bubenec.osm (zdroj: openstreetmap.org)
2. praha_bubenec.gpx (zdroj: mapy.cz)
3. praha_bubenec.png (zdroj: mapy.cz)
4. grun.osm (zdroj: openstreetmap.org)
5. grun.gpx (zdroj: mapy.cz)
6. grun.png (zdroj: openstreetmap.org)