VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2016 Tomáš Caha

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

SEPARACE INFORMACÍ Z WEBOVÝCH STRÁNEK
EXTRACTION OF INFORMATION FROM WEB PAGES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Tomáš Caha
AUTHOR

VEDOUCÍ PRÁCE doc. Ing. Dan Komosný, Ph.D.
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Tomáš Caha ***ID:*** 164249
Ročník: 3 ***Akademický rok:*** 2015/16

NÁZEV TÉMATU:

Separace informací z webových stránek

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi odhadu geografické pozice IP adres pomocí geolokačních databází. Zaměřte se na volně dostupné geolokační databáze. Zhotovte program v jazyce Python pro automatický odhad polohy zadaných IP adres. Program sestavte tak, aby současně umožňoval získávat polohu vzdáleně z webových stránek databází a lokálně ze stažených databázových souborů.

DOPORUČENÁ LITERATURA:

[1] PUŽMANOVÁ, R. TCP/IP v kostce. 2. vyd. Kopp, 2009. 620 s. ISBN: 978-80-7232-388-3.

[2] Linux Dokumentační projekt. 4. vyd. Computer Press, 2008. 1336 s. ISBN: 978-80-251-1525-1.

[3] POESE, I., UHLIG, S., KAAFAR, M., DONNET, B., GUEYE, B. IP Geolocation Databases: Unreliable? ACM SIGCOMM Computer Communication Review, 2011, roč. 41, č. 2, s. 53-56. ISSN: 0146-4833.

Termín zadání: 1.2.2016 ***Termín odevzdání:*** 1.6.2016

Vedoucí práce: doc. Ing. Dan Komosný, Ph.D.
Konzultant bakalářské práce:

doc. Ing. Jiří Mišurec, CSc., *předseda oborové rady*

ABSTRAKT

Tato práce se věnuje problematice separace informací z webových stránek vybraných geolokačních služeb. Jsou shrnuty používané metody geografické lokalizace síťových zařízení a množství údajů poskytovaných vybranými volně dostupnými geolokačními databázemi. Především jsou popsány způsoby získávání informací o IP adresách z rozhraní jednotlivých databází. V práci jsou představeny způsoby, jakých bylo využito při programování systému na automatický odhad geografické polohy zadaných IP adres načítaných ze zdrojového souboru a porovnání získaných údajů s referenčními daty. Vytvořený systém v jazyce Python poskytuje jednoduchý způsob ověření informací o zadaných IP adresách celkem v pěti volně dostupných geolokačních databázích. Dále bylo také provedeno vyhodnocení přesnosti získávaných dat a srovnání jednotlivých volně dostupných geolokačních databázích.

KLÍČOVÁ SLOVA

geolokace, IP, odhad, Python, skript, systém

ABSTRACT

This thesis deals with the separation of information from websites of selected geolocation services. Methods of geographical location of network devices and amount of available data provided by chosen freely accessible geolocation databases. The data are summarized with focus on methods of obtaining information about IP addresses from APIs of particular databases. In the paper there are also presented ways used to develop the system for automatic estimation of geographic location of IP addresses specified in source file and process of comparing retrieved data with reference data. The system is created in Python and provides a simple way for verifying information about given IP addresses in five freely accessible databases. Furthermore, accuracy of the retrieved data is evaluated and five geolocation databases is compared.

KEYWORDS

estimate, geolocation, IP, Python, script, system

CAHA, Tomáš *Separace informací z webových stránek*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 51 s. Vedoucí práce byl doc. Ing. Dan Komosný, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Separace informací z webových stránek“ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu doc. Ing. Danu Komosnému, Ph.D. za odborné vedení, konzultace a podnětné návrhy k práci.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

ÚVOD

Seznámil jsem se s možnostmi odhadu pozice IP adres pomocí volně dostupných geolokačních databází a způsobů, kterými lze polohu odhadovat, což jsem popsal v kapitole 1. Zadáním práce je zhotovit systém pro automatický odhad polohy zadaných IP adres pomocí volně dostupných geolokačních databází, a to jsem detailně popsal v kapitole 2. Lze se tam dočíst o návrhu celého systému, o možnostech získávání údajů z jednotlivých databází, způsoby, které a z jakých důvodů jsem zvolil a navržených vylepšeních. V kapitole 3 popisuji vytvořený statistický systém a porovnávám přesnost jednotlivých volně dostupných databází nad údaji o uzlech experimentální sítě PlanetLab. Na závěr diskutuji výhody mého systému a slabiny geolokačních databází.

1 GEOGRAFICKÁ LOKALIZACE SÍŤOVÝCH ZAŘÍZENÍ

Geografická lokalizace je v dnešní době Internetu, který každý den používá velká část lidstva, velmi důležitá. Využívá se v aplikacích, kde je potřebné nebo výhodné (především z hlediska uživatelského komfortu) znát fyzickou polohu uživatele (resp. polohu jeho síťového zařízení). Díky znalosti geografické polohy uživatele jsme schopni přesně cílit reklamu (např. nabízet obchody a jejich služby nebo produkty v blízkém okolí). Na internetových stránkách se zprávami, s kulturními akcemi nebo předpověďmi počasí můžeme automaticky nabízet lokalitu, ve které se uživatel právě nachází. Jelikož Internet na rozdíl od států nemá žádné pevně stanovené hranice, je velmi složité zjistit pozici koncového zařízení připojeného k internetu[3].

1.1 Metody geografické lokalizace síťových zařízení

Metod pro geografickou lokalizaci známe hned několik. Dělíme je do dvou základních skupin: pasivní a aktivní metody. V mé práci se v podstatě využívá pasivní metody geolokace na základě IP adresy síťového zařízení.

1.1.1 Pasivní metody geografické lokalizace

Pasivní metody využívají především informací o síťovém zařízení (IP adresa, DNS) uložených v komerčních nebo veřejných databázích. Každé zařízení připojené k internetu má IP adresu, resp. skoro vždy je zjistitelná (případně lze využít IP adresy výchozí brány). Známe-li IP adresu, jsme schopni ji porovnat s databází organizace pro přidělovaní adres IANA (Internet Assigned Numbers Authority), resp. s databázemi koordinačních středisek, které pod ni spadají. Jednotlivé databáze poskytují různé množství informací s různou přesností a aktuálností informací.

Určování polohy síťového zařízení ze znalosti jeho DNS záznamu nám dá pouze velmi hrubou a v mnoha případech nepřesnou představu o tom, kde se zařízení nachází. Využijeme-li reverzního DNS záznamu, kdy se nám IP adresa přeloží na doménové jméno, vzhledem k povaze doménových jmen, která jsou strukturovaná a hierarchická, zjistíme například dle národní domény 1. řádu přibližnou zemi, kde by se zařízení mohlo nacházet. Je nutné upozornit na to, že zařízení s určitou IP adresou a doménovým jménem, jehož doména 1. řádu je například .CZ, vůbec nemusí být umístěno v České republice, ale kdekoliv na světě. Pokud doménové jméno nepatří

pod národní domény, ale pod generické, tak přibližnou pozici nejsme schopni odhadnout vůbec (koncovky .COM, .NET nebo nové koncovky .WEBCAM, .TRAVEL a mnoho dalších).

1.1.2 Aktivní metody geografické lokalizace

Aktivní metody odhadují fyzickou polohu síťového zařízení měřením nejčastěji latence (zpoždění), tedy doby, za kterou se jeden datový segment přenese od zdroje k cíli a zpět – RTT (obousměrné zpoždění – Round Trip Time). Zpoždění je ovlivněno mnoha okolnostmi, mezi které patří aktuální vytížení linky, přenosová rychlost vedení nebo geografická vzdálenost mezi jednotlivými uzly.

1.2 Údaje poskytované vybranými volně dostupnými geolokačními databázemi

V této kapitole se budu věnovat jednotlivým volně dostupným geolokačním databázím, které ve svém skriptu využívám. Databáze pracují na principu sdružování IP adres do bloků a evidování informací k těmto blokům[1].

1.2.1 DB-IP/IPtoCity

Databáze DB-IP obsahuje více než 10 milionů bloků IP adres IPv4 a IPv6, což ji řadí mezi jednu z nejobsáhlejších databází. K dispozici je několik verzí, které se liší přesností záznamů, počtem záznamů, množstvím poskytovaných informací a cenou předplatného (některé verze jsou i zdarma).[1]

1.2.2 HostIP

HostIP je komunitní projekt na geografickou lokalizaci IP adres. Informace o IP adresách a jejich fyzických pozicích, resp. opravy fyzických pozic mohou zadávat všichni uživatelé této služby, protože se jedná o otevřený projekt. Odkud pochází základní data o IP adresách jsem na internetových stránkách tohoto projektu nenalezl, ale vzhledem k faktu, že reklamují možnost získávání přesnějších informací z komerční geolokační databáze MaxMind, tak předpokládám, že určitá část databáze pochází právě odtud.[2]

[1] `https://db-ip.com/db/`

[2] `http://www.hostip.info/about.html`

1.2.3 freegeoip.net

Freegeoip.net je jeden z dalších komunitních projektů, které poskytují webové API (rozhraní pro programování aplikací – Application Programming Interface) pro získávání informací o IP adresách. K dispozici jsou informace jako je stát, kraj, město, poštovní směrovací číslo, GPS (globální polohovací systém – Global Positioning System) pozice nebo časová zóna. Jedná se o bezplatnou a open source službu a je povoleno maximálně 10 000 dotazů za hodinu.[3]

1.2.4 MaxMind/GeoLite2

Geolokační databáze GeoLite2 od společnosti MaxMind je poskytována zdarma, avšak je méně přesná oproti komerční databázi GeoIP2. K databázi GeoLite2 společnost neposkytuje žádnou podporu a vzhledem k licenční politice je povinnost při využívání této databáze uvádět zpětný odkaz na stránky společnosti. Databáze GeoLite2 je poskytována ve dvou verzích – Country a City, já budu využívat verzi City, protože kromě informací o státu, kde IP adresa leží, obsahuje i další informace o městech a krajích.[4]

1.2.5 IP2Location/DB11.LITE

Společnost IP2Location poskytuje několik variant svých geolokačních databází v závislosti na tom, jakou přesnost a jaké množství dat potřebujeme. Opět jsou k dispozici jak placené, tak neplacené varianty. V našem případě připadá do úvahy databáze alespoň typu DB5, protože poskytuje informace o státě, kraji, městě a GPS souřadnicích.[5]

[3] `http://freegeoip.net/`
[4] `http://dev.maxmind.com/geoip/geoip2/geolite2/`
[5] `http://lite.ip2location.com/databases`

2 VYTVOŘENÝ SYSTÉM PRO HROMADNÝ ODHAD POZIC IP ADRES

Zadání práce na vytvoření systému pro automatický odhad polohy zadaných IP adres pomocí volně dostupných geolokačních databází je poměrně obecné a bylo by možné jej zpracovat různými způsoby. Po dohodě s mým vedoucím práce, a poté co jsem se seznámil s jeho představami a požadavky, jsme stanovili stěžejní body systému. Systém musí být:

- spustitelný na operačních systémech na bázi Linuxu (odpadají platformě závislá řešení jako C#),
- napsaný v přehledném skriptovacím jazyce (ne kompilovaná binárka) z důvodu jednoduchých příp. úprav (odpadají řešení jako C++ nebo Java),
- použitelný ideálně bez nutnosti instalovat velké množství softwaru (odpadají řešení jako PHP),
- konfigurovatelný – vše podstatné musí být volitelné parametry při spuštění skriptu.

Jako skriptovací jazyk byl tedy vybrán Python, který splňuje výše uvedené – jedná se o platformě nezávislý skriptovací jazyk, jehož interpret bývá ve většině případů předinstalovaný na linuxových systémech[2]. V jazyce Python jsem nikdy předtím nepracoval, jedná se tedy o můj první výtvor.

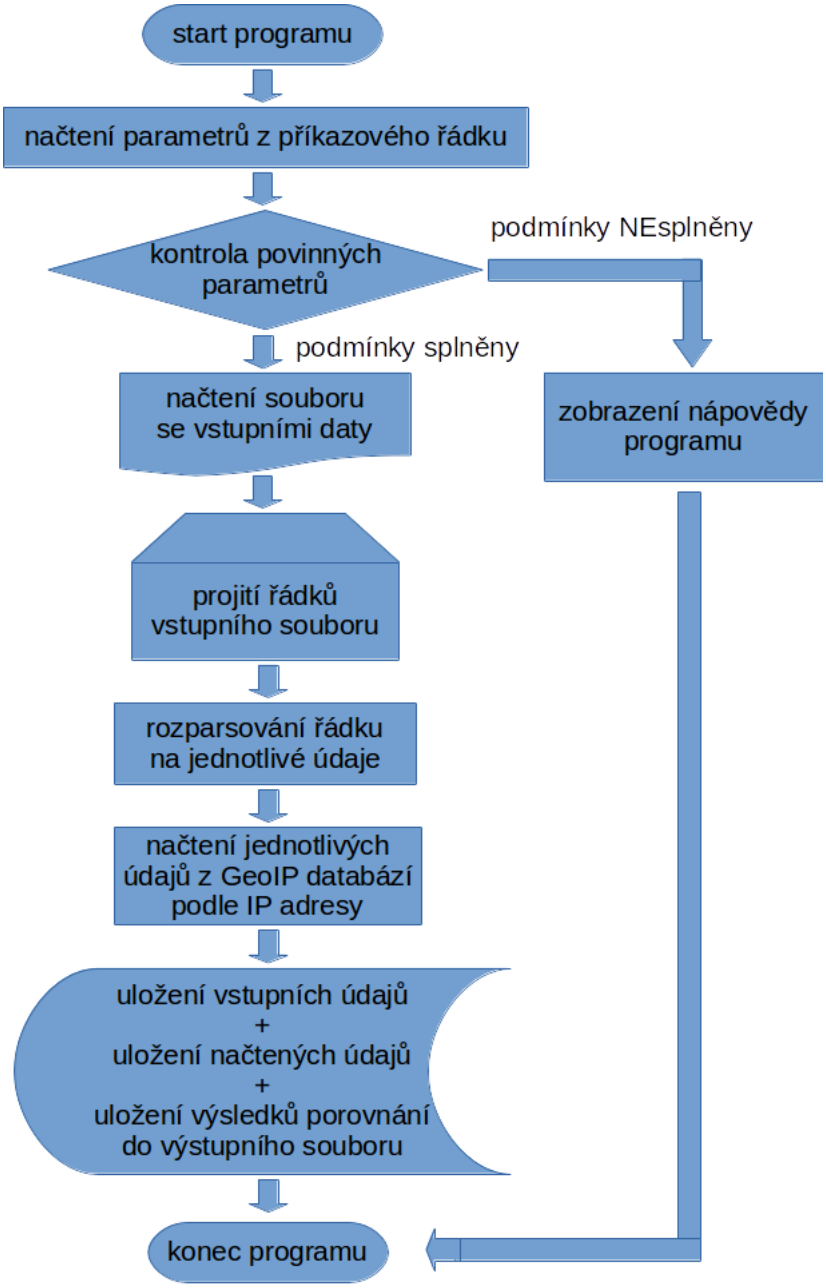
2.1 Návrh systému

Nejdůležitějším krokem při vytváření systému je jeho pečlivý a promyšlený návrh. Navrhl jsem tedy hlavní funkci, která jako argument přijímá argumenty z příkazového řádku. Nejdříve dojde k inicializaci proměnných na výchozí hodnoty. Následně dojde k pokusu o parsování argumentů z příkazového řádku a v případě úspěchu dojde k načtení do patřičných proměnných. Dále dojde ke kontrole povinných parametrů, bez kterých není možné pokračovat. V případě problémů dojde k vypsání nápovědy skriptu a ukončení jeho běhu. Dojde také k otevření souboru se vstupními daty pro čtení a k otevření souborů pro výstupní data k zápisu.

Cyklem se projde vstupní soubor řádek po řádku a prožene se funkcí na parsování řádku na vstupní data. Poté dojde k získání dat z vybraných geolokačních databází, která se proženou funkcí na porovnávání dat. Řádek ze vstupního souboru se doplní o získané údaje a výsledky porovnání a zapíšou se do patřičného výstupního souboru. Po skončení procházení vstupního souboru dojde k uzavření všech souborů a ukončení skriptu.

Skript tedy celkem obsahuje funkce, které zajišťují:

- parsování informací z řádku ze vstupního souboru,
- získávání informací z geolokačních databází,
- porovnávání informací ze vstupního souboru vůči informacím z geolokačních databází,
- výpis nápovědy,
- provedení automatického odhadu poloh zadaných IP adres.



Obr. 2.1: Zjednodušený vývojový diagram systému pro hromadný odhad pozic IP adres

2.2 Parsování vstupních údajů

Skript načítá vstupní údaje ze zadaného souboru, který obsahuje data na řádcích v následujícím formátu:

```
1 id ip dns continent country region city something something
    latCor lonCor something
```

id:	identifikační kód stanice
ip:	IPv4 nebo IPv6 adresa
dns:	hostitelský název stanice, resp. reverzní DNS záznam
continent:	dvoupísmenný kód kontinentu
country:	dvoupísmenný kód státu
region:	název kraje, pokud obsahuje mezery a oddělovač je také mezera, tak musí být v uvozovkách
city:	název města, pokud obsahuje mezery a oddělovač je také mezera, tak musí být v uvozovkách
something:	může obsahovat cokoliv
latCor:	latitude – zeměpisná šířka
lonCor:	longitude – zeměpisná délka

Jednotlivé údaje jsou odděleny mezerami nebo tabulátorem. Jelikož jsem v dodaných vstupních datech našel i údaje, které nesplňují výše uvedené podmínky (především to, že víceslovné názvy krajů a měst musí být v uvozovkách), musel jsem se s tím určitým způsobem vypořádat.

Funkce na parsování údajů z jednoho řádku, který je jedním z parametrů funkce, nejdříve zjistí použitý oddělovač. V případě mezery se prvních šest údajů získá pomocí regulárního výrazu, který bylo poměrně složité sestavit a odladit. Pokud víceslovný název kraje nebo města není v uvozovkách, tak se s ním naloží tak, že se použije pouze první slovo, zbytek je zahozen. Ze stejného důvodu není možné jednoduše získat zeměpisnou šířku a délku, resp. také dvě pole před tím. Toto jsem obešel tak, že celý řádek je rozdělen podle specifikovaného oddělovače do pole a poslední čtyři údaje se získávají z tohoto pole počítáním prvků od konce. V případě tabulátoru je získání poměrně jednoduché, celý řádek se rozdělí podle tabulátoru do pole.

2.3 Získávání údajů z jednotlivých databází

Geolokační databáze podporují různé způsoby získávání dat přes webové rozhraní. V této kapitole popíšu u každé databáze, jaké způsoby podporují, dále také jaký způsob jsem zvolil a jeho implementaci v mém skriptu. Velmi důležitým prvkem

funkcí je návratová hodnota, kterou jsem zvolil následovně: jedná se o uspořádanou šestici obsahující IP adresu, dvoupísmenný kód státu, název kraje, název města a souřadnice zeměpisné šířky a délky. Jestliže nějaká informace chybí, tak v případě kódu státu je nahrazena písmeny XX a v ostatních případech pomlčkou. Tato unifikace návratových kódů je velmi důležitá pro další zpracovávání dat.

2.3.1 DB-IP/IPtoCity

API této služby je postaveno na pricipech REST (Representational State Transfer). Informace z API se získávají jednoduchými HTTP (Hypertext Transfer Protocol) požadavky (samotný návrh REST umožňuje požadavky GET, POST, PUT a DELETE), toto API umožňuje pouze GET požadavky.

Z API lze získávat informace následovně:

Příklad požadavku:

```
1 http://api.db-ip.com/addrinfo?api_key=
    bdf6930fac585dd37b9dc225e0864327c999a011&addr=147.229.2.90
```

Odpověď:

```
1 {"address":"147.229.2.90","country":"CZ","stateprov":"South
    Moravian Region","city":"Brno (Brno-st\u0159ed)"}
```

API využívající principů návrhu REST pracují především ve formátu JSON (JavaScriptový objektový zápis – JavaScript Object Notation). S formátem JSON jsem obeznámen z prací na mnoha jiných projektech a osobně mi vyhovuje, protože informace jsem schopen z odpovědi ověřit letmým pohledem. Práce s JSON objekty je v mnoha jazycích poměrně jednoduchá a pohodlná a jinak tomu není ani v programovacím jazyce Python.

Níže si rozebereme funkci z mého skriptu, která ověřuje IP adresu zadanou v parametru vůči databázi DB-IP/IPtoCity. Nejdříve funkce vypíše informaci o provádění dotazu, poté se provede HTTP GET požadavek na zadanou URL (jednotná adresa zdroje – Uniform Resource Locator) adresu včetně patřičného klíče a zadané IP adresy. V případě, že API vrátí regulérní odpověď, tak HTTP hlavička obsahuje stav 200 OK, v ostatních případech je vyhozena výjimka. Dále se obsah odpovědi převede na JSON objekty.

Jelikož databáze v této variantě neposkytuje GPS pozice, implementoval jsem alespoň základní získání těchto souřadnic pomocí Open Street Map Nominatim, kde se vyhledají souřadnice řetězce „město, kraj stát“. Funkce vrací uspořádanou šestici dat.

```
1 def geo_dbiptocity(ip):
```

```

    # header
    print 'Checking ' + str(ip) + ' against DB-IP/IPtoCity'

    # process request
    request = requests.get('http://api.db-ip.com/addrinfo?
       api_key=bdf6930fac585dd37b9dc225e0864327c999a011&addr=
       ' + urllib.quote(ip), timeout=62)

    # sleep for a second
    time.sleep(1)

    # check for HTTP errors
    if request.status_code != 200:
        raise Exception('Bad HTTP response')

    # parse content
    content = request.content
    content = json.loads(content)

    # check for API error
    if 'error' in content:
        raise Exception('Bad API response')

    # get lat/lon from OSM
    g = geocoder.osm(content['city'] + ', ' + content['
       stateprov'] + ' ' + content['country'], timeout=62)
    g = g.json

    # check for errors in geocoder
    if g['status_code'] == 200:
        content['lat'] = unicode(g['lat'])
        content['lon'] = unicode(g['lng'])
    else:
        content['lat'] = '-'
        content['lon'] = '-'

    # return IP, state, region, city, lat, lon
    return (content['address'], content['country'], content['
       stateprov'], content['city'], '-', '-')
```

2.3.2 HostIP

K API této služby se přistupuje běžnými HTTP GET požadavky. API umožňuje vracet následující data:

(a) dvoupísmenný kód země

Příklad požadavku:

```
http://api.hostip.info/country.php?ip=147.229.2.90
```

Odpověď:

```
CZ
```

(b) dvoupísmenný kód země, název země, název města, GPS souřadnice a zadanou IP adresu ve formátu plaintext (obyčejný text)

Příklad požadavku:

```
http://api.hostip.info/get_html.php?ip=147.229.2.90&position
    =true
```

Odpověď:

```
Country: CZECH REPUBLIC (CZ)
City: Brno

Latitude: 49.2
Longitude: 16.6333
IP: 147.229.2.90
```

(c) dvoupísmenný kód země, název země, název města, GPS souřadnice a zadanou IP adresu ve formátu JSON (JavaScriptový objektový zápis – JavaScript Object Notation)

Příklad požadavku:

```
http://api.hostip.info/get_json.php?ip=147.229.2.90&position
    =true
```

Odpověď:

```
{"country_name":"CZECH REPUBLIC","country_code":"CZ","city":
    "Brno","ip":"147.229.2.90","lat":"49.2","lng":"16.6333"}
```

(d) dvoupísmenný kód země, název země, název města, GPS souřadnice a zadanou IP adresu ve formátu XML (rozšiřitelný značkovací jazyk – Extensible Markup

Language)

Příklad požadavku:

```
http://api.hostip.info/?ip=147.229.2.90
```

Odpověď:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<HostipLookupResultSet version="1.0.1" xmlns:gml="http://www.opengis.net/gml" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="http://www.hostip.info/api/hostip-1.0.1.xsd">
<gml:description>This is the Hostip Lookup Service</gml:description>
  <gml:name>hostip</gml:name>
    <gml:boundedBy>
      <gml:Null>inapplicable</gml:Null>
    </gml:boundedBy>
    <gml:featureMember>
    <Hostip>
      <ip>147.229.2.90</ip>
      <gml:name>Brno</gml:name>
      <countryName>CZECH REPUBLIC</countryName>
      <countryAbbrev>CZ</countryAbbrev>
      <!-- Co-ordinates are available as lng,lat -->
      <ipLocation>
        <gml:pointProperty>
          <gml:Point srsName="http://www.opengis.net/gml/srs/epsg.xml#4326">
            <gml:coordinates>16.6333,49.2</gml:coordinates>
          </gml:Point>
        </gml:pointProperty>
      </ipLocation>
    </Hostip>
  </gml:featureMember>
</HostipLookupResultSet>
```

Ve svém skriptu jsem zvolil variantu C, tedy odpovědi API ve formátu JSON. Níže je funkce z mého skriptu, která ověřuje IP adresu zadanou v parametru vůči databázi HostIP. Funkce jednoduše vypíše informaci o provádění dotazu na databázi, následně se provede HTTP GET požadavek na patřičnou URL adresu, kde se specifikuje zadaná IP adresa. Odpověď je zkontrolována na HTTP hlavičku 200 OK,

protože jiná hlavička by znamenala, že server vrátil neplatnou odpověď (v tomto případě je vyhozena výjimka). Následně se JSON řetězec z odpovědi převede na objekt. Abych zajistil jednotný formát výstupních dat z jednotlivých databází, musí se data obsažená v JSON zkontrolovat a případně upravit, protože může nastat situace, kdy zadaná IP adresa neexistuje (API ji vrátí jako privátní adresu), případně databáze neobsahuje určité údaje o konkrétní IP adrese (nezná stát nebo město nebo GPS souřadnice). Následně funkce vrací uspořádanou šestici dat.

```
1  def geo_hostip(ip):
2
3      # header
4      print 'Checking ' + str(ip) + ' against HostIP'
5
6      # process request
7      request = requests.get('http://api.hostip.info/get_json.
          php?position=true&ip=' + urllib.quote(ip), timeout=62)
8
9      # sleep for a second
10     time.sleep(1)
11
12     # check for HTTP errors
13     if request.status_code != 200:
14         raise Exception('Bad HTTP response')
15
16     # parse content
17     content = request.content
18     content = json.loads(content)
19
20     # format data
21     if content['country_name'] == '(Private Address)':
22         content['ip'] = 'XXX.XXX.XXX.XXX'
23     if content['country_code'] == 'XX':
24         content['country_code'] = '-'
25     if content['city'] == '(Unknown City?)':
26         content['city'] = '-'
27     if content['city'] == '(Unknown city)':
28         content['city'] = '-'
29     if content['city'] == '(Private Address)':
30         content['city'] = '-'
31     if content['lat'] == None:
32         content['lat'] = '-'
```

```
    if content['lng'] == None:
        content['lng'] = '-'

    # return IP, state, region, city, lat, lon
    return (content['ip'], content['country_code'], '-',
       content['city'], content['lat'], content['lng'])
```

2.3.3 freegeoip.net

K API této služby se přistupuje jako v předešlých případech HTTP GET požadavky. Z API lze získávat veškerá data v několika formátech následovně:

(a) CSV (hodnoty oddělené čárkami – Comma-Separated Values)

Příklad požadavku:

```
http://freegeoip.net/csv/147.229.2.90
```

Odpověď: (CSV soubor, který obsahuje následující data oddělená čárkou)

```
147.229.2.90,CZ,Czech Republic,JM,South Moravian,Brno,614
    00,Europe/Prague,49.20,16.63,0
```

(b) XML (rozšiřitelný značkovací jazyk – Extensible Markup Language)

Příklad požadavku:

```
http://freegeoip.net/xml/147.229.2.90
```

Odpověď:

```
<?xml version="1.0" encoding="UTF-8"?>
<Response>
  <IP>147.229.2.90</IP>
  <CountryCode>CZ</CountryCode>
  <CountryName>Czech Republic</CountryName>
  <RegionCode>JM</RegionCode>
  <RegionName>South Moravian</RegionName>
  <City>Brno</City>
  <ZipCode>614 00</ZipCode>
  <TimeZone>Europe/Prague</TimeZone>
  <Latitude>49.2</Latitude>
  <Longitude>16.6333</Longitude>
  <MetroCode>0</MetroCode>
</Response>
```

(c) JSON (JavaScriptový objektový zápis – JavaScript Object Notation)
Příklad požadavku:

```
http://freegeoip.net/json/147.229.2.90
```

Odpověď:

```
{"ip":"147.229.2.90","country_code":"CZ","country_name":"
    Czech Republic","region_code":"JM","region_name":"South
    Moravian","city":"Brno","zip_code":"614 00","time_zone":"
    Europe/Prague","latitude":49.2,"longitude":16.6333,"
    metro_code":0}
```

(d) JSONP (JSON with Padding)
Příklad požadavku:

```
http://freegeoip.net/json/147.229.2.90?callback=mojeFunkce
```

Odpověď:

```
mojeFunkce({"ip":"147.229.2.90","country_code":"CZ","
    country_name":"Czech Republic","region_code":"JM","
    region_name":"South Moravian","city":"Brno","zip_code":"
    614 00","time_zone":"Europe/Prague","latitude":49.2,"
    longitude":16.6333,"metro_code":0});
```

Použil jsem opět formát odpovědi ve formátu JSON. Funkce na získávání informací o IP adresách v mém skriptu si jsou všechny velice podobné. Funkce opět vypíše informaci o získávání dat o IP adrese z databáze, poté se zašle HTTP GET požadavek na vybranou URL adresu. Vyčká se na vypršení požadavku – celkem až 64 vteřin. Zkontroluje se odpověď na stavový kód 200 OK (jiný kód by znamenal buď neplatnou odpověď serveru, anebo vyčerpání počtu dotazů, a v tomto případě bude vyhozena výjimka). Obsah odpovědi se převede na JSON objekt a provede se unifikace chybějících informací v odpovědi. Funkce opět vrací uspořádanou šestici dat.

```
def geo_freegeoip(ip):

    # header
    print 'Checking ' + str(ip) + ' against freegeoip.net'

    # process request
    request = requests.get('http://freegeoip.net/json/' +
        urllib.quote(ip), timeout=62)
```

```
8
9       # sleep for a second
10      time.sleep(1)
11
12      # check for HTTP errors
13      if request.status_code != 200:
14          raise Exception('Bad HTTP&API response')
15
16      # parse content
17      content = request.content
18      content = json.loads(content)
19
20      # format data
21      if content['country_code'] == '':
22          content['country_code'] = '-'
23      if content['region_name'] == '':
24          content['region_name'] = '-'
25      if content['city'] == '':
26          content['city'] = '-'
27      content['latitude'] = unicode(content['latitude'])
28      content['longitude'] = unicode(content['longitude'])
29
30      # return IP, state, region, city, lat, lon
31      return (content['ip'], content['country_code'], content['
           region_name'], content['city'], content['latitude'],
           content['longitude'])
```

2.3.4 MaxMind/GeoLite2

Společnost MaxMind má velmi slušně zpracované API k poskytování informací. V případě placených variant je možné využívat webového rozhraní, v případě neplacené varianty lze využívat čtení z lokálně staženého databázového souboru ve formátech CSV (hodnoty oddělené čárkami – Comma-Separated Values) nebo MMDB (MaxMind binární formát). K datům se přistupuje nepřímo, ale využívají se dodávané knihovny, které jsou dostupné pro nepřeberné množství programovacích jazyků (Python, PHP, .NET C#, C, JAVA nebo Perl), a dále také knihovny třetích stran pro další jazyky (Lua, Ruby, Node.js a další). Uvádět tedy příklady volání každého API nemá vzhledem k počtu rozmanitých verzí smysl. Níže se podíváme, jak jednoduše lze získat informace o IP adrese ze City databáze v Pythonu:

Příklad požadavku:

```
>>> import geoip2.database
>>> reader = geoip2.database.Reader('GeoLite2-City.mmdb')
>>> response = reader.city('147.229.2.90')
>>> print response
```

Odpověď: (dekódované unicode znaky)

```
geoip2.models.City({u'city': {u'geoname_id': 3078610, u'names':
    {u'ru': u'\u0411\u0440\u043d\u043e', u'fr': u'Brno', u'en': u
    'Brno', u'de': u'Br\xfcnn', u'pt-BR': u'Brno', u'ja': u'\
    u30d6\u30eb\u30ce', u'es': u'Brno'}}, 'traits': {'ip_address'
    : '147.229.2.90'}, u'country': {u'geoname_id': 3077311, u'
    iso_code': u'CZ', u'names': {u'ru': u'\u0427\u0435\u0448\
    u0441\u043a\u0430\u044f \u0420\u0435\u0441\u043f\u0443\u0431\
    u043b\u0438\u043a\u0430', u'fr': u'R\xe9publique tch\xe8que',
     u'en': u'Czech Republic', u'de': u'Tschechien', u'zh-CN': u'
    \u6377\u514b\u5171\u548c\u56fd', u'pt-BR': u'Rep\xfablica
    Checa', u'ja': u'\u30c1\u30a7\u30b3\u5171\u548c\u56fd', u'es'
    : u'Rep\xfablica Checa'}}, u'registered_country': {u'
    geoname_id': 3077311, u'iso_code': u'CZ', u'names': {u'ru': u
    '\u0427\u0435\u0448\u0441\u043a\u0430\u044f \u0420\u0435\
    u0441\u043f\u0443\u0431\u0438\u043a\u0430', u'fr': u'R\
    xe9publique tch\xe8que', u'en': u'Czech Republic', u'de': u'
    Tschechien', u'zh-CN': u'\u6377\u514b\u5171\u548c\u56fd', u'
    pt-BR': u'Rep\xfablica Checa', u'ja': u'\u30c1\u30a7\u30b3\
    u5171\u548c\u56fd', u'es': u'Rep\xfablica Checa'}}, u'
    subdivisions': [{u'geoname_id': 3339536, u'iso_code': u'JM',
    u'names': {u'ru': u'\u042e\u0436\u043d\u043e\u043c\u043e\
    u0440\u0430\u0432\u0441\u043a\u0438\u0439 \u043a\u0440\u0430\
    u0439', u'fr': u'Moravie-du-Sud', u'en': u'South Moravian', u
    'de': u'S\xfcdm\xe4hrische Region', u'pt-BR': u'Mor\xe1via do
     Sul', u'ja': u'\u5357\u30e2\u30e9\u30f4\u30a3\u30a2\u5dde',
    u'es': u'Regi\xf3n de Moravia Meridional'}}, {u'geoname_id':
    3078609, u'iso_code': u'622', u'names': {u'en': u'Mesto Brno'
    }}], u'location': {u'latitude': 49.2, u'time_zone': u'Europe/
    Prague', u'longitude': 16.6333}, u'postal': {u'code': u'614
    00'}, u'continent': {u'geoname_id': 6255148, u'code': u'EU',
    u'names': {u'ru': u'\u0415\u0432\u0440\u043e\u043f\u0430', u'
    fr': u'Europe', u'en': u'Europe', u'de': u'Europa', u'zh-CN':
     u'\u6b27\u6d32', u'pt-BR': u'Europa', u'ja': u'\u30e8\u30fc\
    u30ed\u30c3\u30d1', u'es': u'Europa'}}}, ['en'])
```

Funkce pro získávání geografických informací o IP adrese z databáze MaxMind/GeoLite2 je odlišná od předchozích, protože využívá knihovny geoip2. Funkce má kromě povinného parametru specifikujícího IP adresu také druhý parametr určující cestu k MMDB souboru. Funkce nejdříve vypíše informaci o získávání dat o IP adrese, dále se načte databázový soubor a vyhledají se informace o konkrétní IP adrese. Jelikož se o každé IP adrese podaří získat různé množství informací, musí se opět provést unifikace výstupních dat na uspořádanou šestici. V případě, že by během nějaké činnosti při práci s knihovnou geoip2 došlo k nějaké výjimce, je tato výjimka odchycena a opětovně je vyhozena výjimka s obecným popisem.

```
1  def geo_maxmindgeolite2city(ip, database_file_path):
2
3      # header
4      print 'Checking ' + str(ip) + ' against MaxMind/GeoLite2'
5
6      # try catch for API errors
7      try:
8          # process request
9          request = geoip2.database.Reader(database_file_path)
10
11         # content
12         res = request.city(ip)
13         content = {}
14
15         # format data
16         content['ip'] = res.traits.ip_address
17         if not res.country:
18             content['country_code'] = 'XX'
19         else:
20             content['country_code'] = res.country.iso_code
21         if not res.subdivisions:
22             content['region_name'] = '-'
23         else:
24             content['region_name'] = res.subdivisions[0].
                   names['en']
25         if not res.city.names:
26             content['city'] = '-'
27         else:
28             content['city'] = res.city.names['en']
29         if not res.location:
30             content['latitude'] = '-'
```

```
31                content['longitude'] = '-'
32            else:
33                content['latitude'] = unicode(res.location.
                     latitude)
34                content['longitude'] = unicode(res.location.
                     longitude)
35
36            # return IP, state, region, city, lat, lon
37            return (content['ip'], content['country_code'],
                 content['region_name'], content['city'], content['
                 latitude'], content['longitude'])
38        except:
39            raise Exception('Bad geoip2 API response')
```

2.3.5 IP2Location/DB11.LITE

Společnost IP2Location má obdobně jako MaxMind velmi slušně zpracované API k poskytování informací. K datům v databázových souborech se jednoduše přistupuje za pomoci knihoven pro konkrétní programovací jazyky (Perl, Python, Pascal, PHP, Ruby a další), které jsou zdarma k dispozici, nebo pomocí knihoven placených (.NET, C#, VB.NET nebo JAVA). Dále je také možnost využít programů pro Windows nebo Linux, resp. multiplatformního Perl skriptu. Uvedu příklad, jak získat informace o IP adrese ve formátu XML pomocí Perl skriptu (který mi doporučil vedoucí práce):

Příklad požadavku:

```
1 $./ip2location.pl -format xml -datafile ./DB5.BIN -ip
    147.229.2.90
```

Odpověď:

```
1 <xml>
2 <ip2location>
3   <ip>147.229.2.90</ip>
4   <countryshort>??</countryshort>
5   <countrylong>This record is unavailable in demo version.
       Please subscribe.</countrylong>
6   <region>N/A</region>
7   <city>N/A</city>
8   <latitude>N/A</latitude>
9   <longitude>N/A</longitude>
```

```
10    <zipcode>N/A</zipcode>
11    <timezone>N/A</timezone>
12    <ispname>N/A</ispname>
13    <domainname>N/A</domainname>
14    <netspeed>N/A</netspeed>
15    <iddcode>N/A</iddcode>
16    <areacode>N/A</areacode>
17    <weatherstationcode>N/A</weatherstationcode>
18    <weatherstationname>N/A</weatherstationname>
19  </ip2location>
```

Získávání geografických informací o IP adrese z databáze IP2Location probíhá obdobně jako u ostatních funkcí, funkce má 3 povinné parametry – IP adresu, cestu k souboru ip2location.pl a cestu k databázovém souboru ve formátu BIN. Funkce vypíše informaci o získávání dat o IP adrese, spustí se podproces s Perl skriptem včetně potřebných parametrů. Následně se ověří návratový kód (cokoliv jiného než 0 znamená chybu v běhu skriptu). Výsledek se zpracuje rozšířením na XML data, provede se unifikace návratových dat a vrátí se uspořádaná šestice.

```
1  def geo_ip2locdb11lite(ip, cmd, database_file_path):
2
3      # header
4      print 'Checking ' + str(ip) + ' against IP2Location/DB11.
          LITE'
5
6      # process request and wait
7      request = subprocess.Popen(cmd + ' -format xml -datafile
          ' + database_file_path + ' -ip ' + ip, shell=True,
          stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
8      request.wait()
9
10     # check for CMD errors
11     if request.returncode != 0 or request.stderr != None:
12         raise Exception('Bad exit code from IP2Location
              script')
13
14     # parse content
15     content = request.stdout.read().strip()
16     xmldoc = minidom.parse(StringIO(unicode(content)))
17
18     # format data
```

```
    content = {}
    content['ip'] = xmldoc.getElementsByTagName('ip')[0].firstChild.data
    if xmldoc.getElementsByTagName('countryshort')[0].firstChild.data == 'N/A' or xmldoc.getElementsByTagName('countryshort')[0].firstChild.data == '??':
        content['country_code'] = '-'
    else:
        content['country_code'] = xmldoc.getElementsByTagName('countryshort')[0].firstChild.data
    if xmldoc.getElementsByTagName('region')[0].firstChild.data == 'N/A':
        content['region_name'] = '-'
    else:
        content['region_name'] = xmldoc.getElementsByTagName('region')[0].firstChild.data
    if xmldoc.getElementsByTagName('city')[0].firstChild.data == 'N/A':
        content['city'] = '-'
    else:
        content['city'] = xmldoc.getElementsByTagName('city')[0].firstChild.data
    if xmldoc.getElementsByTagName('latitude')[0].firstChild.data == 'N/A':
        content['latitude'] = '-'
    else:
        content['latitude'] = xmldoc.getElementsByTagName('latitude')[0].firstChild.data
    if xmldoc.getElementsByTagName('longitude')[0].firstChild.data == 'N/A':
        content['longitude'] = '-'
    else:
        content['longitude'] = xmldoc.getElementsByTagName('longitude')[0].firstChild.data

    # return IP, state, region, city, lat, lon
    return (content['ip'], content['country_code'], content['region_name'], content['city'], content['latitude'], content['longitude'])
```

2.4 Návrh metody na porovnávání vstupních a získaných údajů

Proces porovnávání vstupních dat a získaných údajů z geolokačních databází obstarává funkce k tomu určená, která přijímá dvakrát šest parametrů reprezentujících IP adresu, kód státu, název kraje, název města, zeměpisnou šířku a zeměpisnou délku a k tomu ještě parametry specifikující oddělovač, který bude použitý ve výstupním souboru. Sobě odpovídající si údaje se porovnají a v případě shody vstupního a získaného údaje se k výstupnímu řetězci doplní klíčové slovo *YES*, v případě, že se slova neshodují, tak se doplní klíčové slovo *NO*. Pokud nastane varianta, kdy jeden z údajů není znám (je reprezentován jako -), doplní se klíčové slovo *UNK*. Údaje o zeměpisné šířce a délce se neporovnávají, ale pokud jsou všechny čtyři údaje dostupné, tak se za pomoci Vincentova vzorce založeného na elipsoidickým modelu Země vypočítá vzdálenost zadaných souřadnic v kilometrech, v opačném případě se místo vzdálenosti vyplní *UNK*.

2.5 Návrh metody na zvýšení přesnosti

Při testování se zjistilo, že porovnávání vstupních dat a získaných údajů není úplně ideální a u informací o názvu kraje nebo města dochází k nesprávnému vyhodnocení shody. Problém však nebyl způsoben technickým provedením, ale jednoduchostí porovnávání.

Příklady špatně vyhodnoceného porovnání:

- název francouzského kraje *Île-de-France* ve vstupních datech obsahoval národní znaky, avšak údaj získaný z geolokační databáze poskytl název převedený na ASCII znaky *Ile-de-France*,
- ve vstupních datech bylo uvedeno město *Praha*, avšak údaj získaný z geolokační databáze obsahoval zpřesnění v závorce *Praha (Praha 6)*,
- ve vstupních datech bylo uvedeno město *Praha*, avšak údaj získaný z geolokační databáze obsahoval přesný název *Hlavní město Praha*,
- ve vstupních datech bylo uvedeno město *Praha*, avšak údaj získaný z geolokační databáze obsahoval anglický název *Prague*,
- ve vstupních datech byl uveden kraj *Bratislavský kraj*, avšak údaj získaný z geolokační databáze obsahoval pouze *Bratislavský*,

Z výše uvedených příkladů vyplývá, že jednoduché porovnávání pro přesné výsledky není dostačující, a proto došlo k rozšíření původní funkce na porovnávání, která byla doplněna o argumenty obsahující seznam slov k ořezu a překladový slovník. Při porovnávání jednotlivých údajů je poté volána funkce pro „chytré“ porov-

nání údajů, která provádí převod na malá písmena, odstranění slov v závorkách včetně závorek, ořez slov dle seznamu, nahrazení slov dle překladového slovníku a nakonec dojde k odstranění diakritiky. Funkce vrací dva údaje – informaci o shodě (True nebo False) a vstupní řetězec po provedených úpravách, který se doplní ve výstupním souboru místo původního údaje získaného z geolokační databáze.

3 PŘESNOST ZÍSKANÝCH ÚDAJŮ A POROVNÁNÍ JEDNOTLIVÝCH DATABÁZÍ

Celý systém pro automatický odhad polohy zadaných IP adres pomocí volně dostupných geolokačních databází bylo potřeba otestovat na větším množství dat a poté bylo možné přesnost jednotlivých databází vzájemně porovnat. Systém byl otestován na seznamu celkem 991 IP adres uzlů experimentální sítě PlanetLab. Jelikož výstupem systému je soubor, který obsahuje informace o získaných údajích z geolokačních databází a údaje o shodě se vstupními daty, bylo nutné vytvořit druhý systém, který poskytne souhrnné statistické údaje o tomto výstupním souboru.

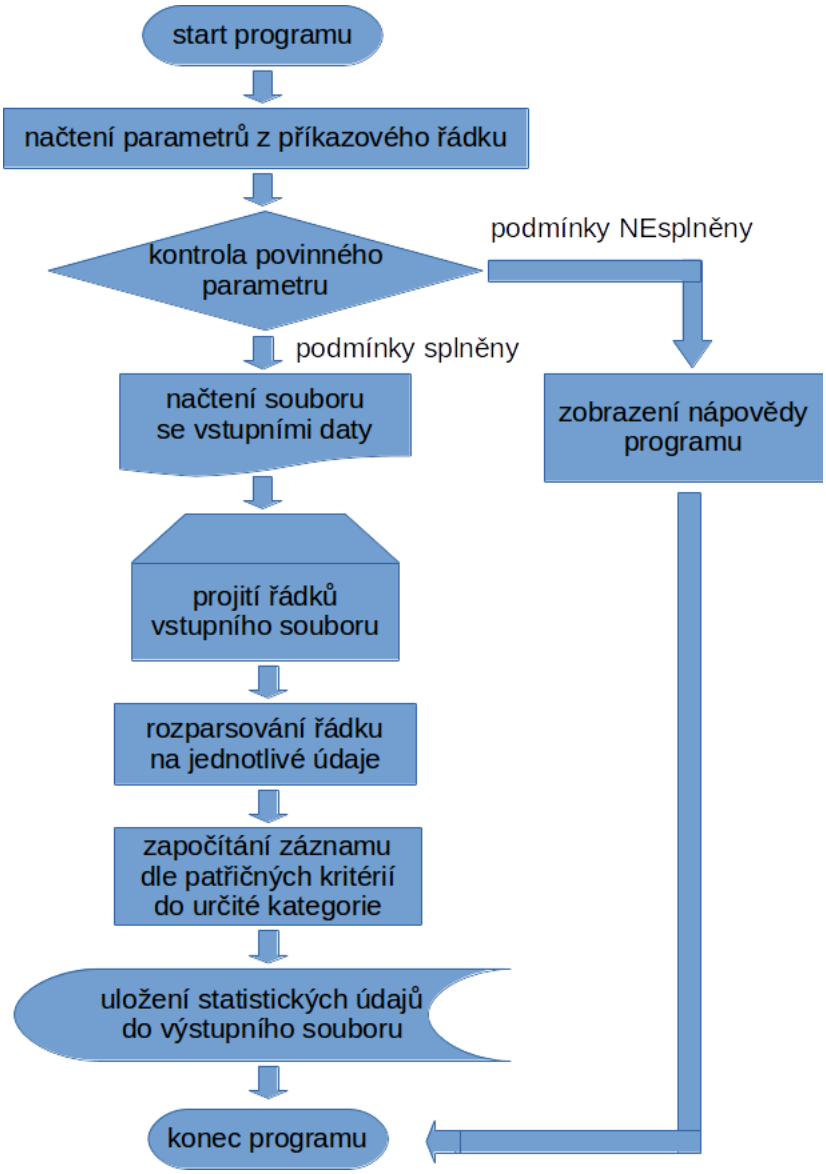
3.1 Systém na zpracování výsledků

Pro rychlé a jednoduché získání statistických údajů o zpracovaných IP adresách jsem navrhl a vytvořil statistický systém opět v jazyce Python. Statistický systém pracuje obdobně jako systém pro hromadný odhad pozic IP adres. Hlavní funkce přijímá argumenty z příkazového řádku, dojde k inicializaci několika proměnných a následně dojde k pokusu o parsování argumentů z příkazového řádku a v případě úspěchu dojde k načtení do patřičných proměnných. Dále dojde ke kontrole povinného parametru – cestě k souboru s výsledky ze systému pro hromadný odhad pozic IP adres. V případě problému dojde k vypsání nápovědy skriptu a ukončení jeho běhu. Dojde také k otevření souboru se vstupními daty pro čtení a k otevření souboru pro výstupní statistická data k zápisu.

Cyklem se projde vstupní soubor řádek po řádku a prožene se funkcí na parsování řádku na vstupní data. Poté dojde k započítání záznamu dle patřičných kritérií do určité kategorie. Na závěr dojde k vypsání statistických údajů do výstupního souboru ve formátu CSV, které lze následně zpracovat libovolným tabulkovým procesorem do grafů atp. Před ukončením skriptu samozřejmě dojde k uzavření všech souborů.

Skript tedy celkem obsahuje funkce, které zajišťují:

- parsování informací z řádku ze vstupního souboru,
- výpis nápovědy,
- provedení statistických výpočtů.

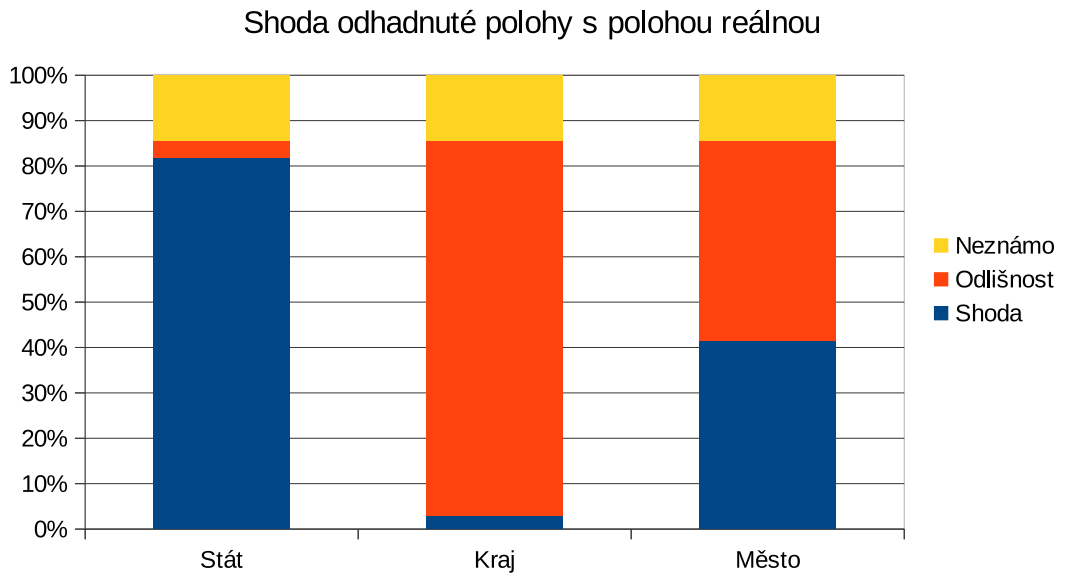


Obr. 3.1: Zjednodušený vývojový diagram statistického systému

3.2 Přesnost získaných údajů z databází

3.2.1 DB-IP/IPtoCity

Z grafu na obr. 3.2 je patrné, že nejvyšší procento shody je u položky *Stát*. U položky *Město* databáze dosahuje shody v přibližně 40 % případech. U všech sledovaných položek je konstantní procento neznámých výsledků odhadu, což je způsobené chybami při procesu získávání údajů, kdy databáze z neznámého důvodu vrací chybový stav (při ruční kontrole však údaje vrátí). Toto je pravděpodobně způsobené bezpečnostními mechanismy nebo výkonnostními limity na straně rozhraní databáze.

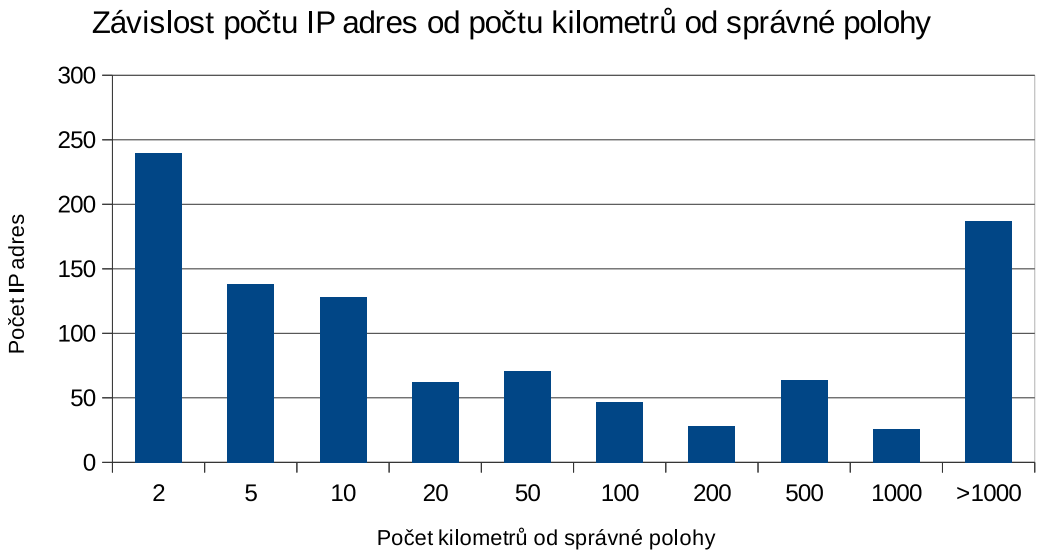


Obr. 3.2: Relativní přesnost odhadu polohy databáze DB-IP/IPtoCity

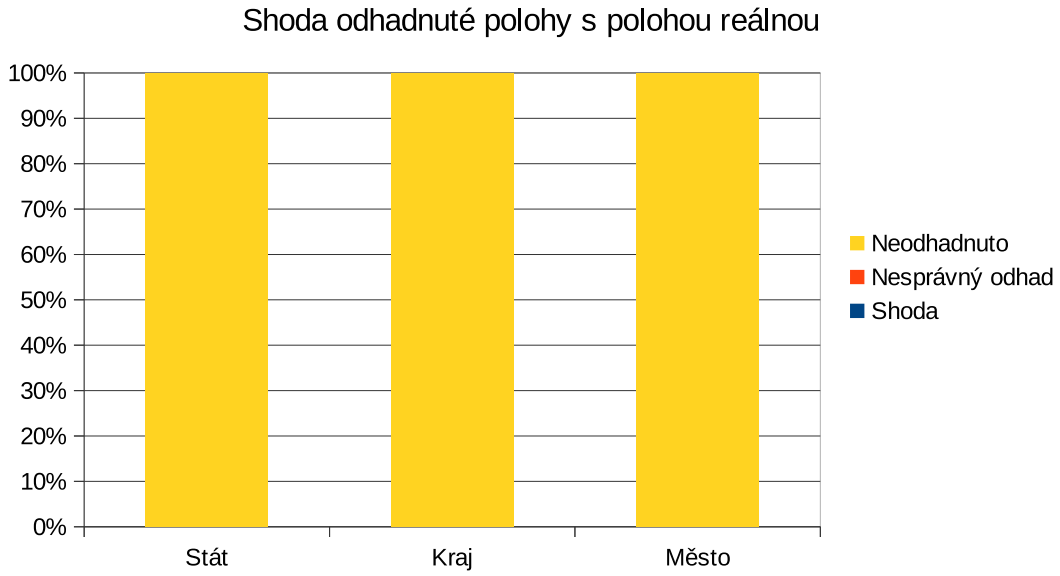
Na obr. 3.3 můžeme vidět, že z celkového počtu 991 testovaných IP adres u necelých 250 byla poloha určena s přesností do 2 km od správné polohy. Do počtu IP adres s odchylkou větší než 1 000 km jsou taktéž započítány všechny IP adresy (cca 140), u kterých se vyskytla chyba během procesu získávání údajů z databáze nebo během porovnávání. U této konkrétní databáze jsou však do výsledků vneseny chyby, kvůli kterým se přesnost databáze může jevit horší nebo dokonce lepší než ve skutečnosti je. Databáze neposkytuje údaje o GPS souřadnicích, souřadnice jsou získávány ze služby Open Street Map Nominatim. Absolutní přesnost znázorněna v tomto grafu tedy závisí ještě na (ne)přesnosti další služby.

3.2.2 HostIP

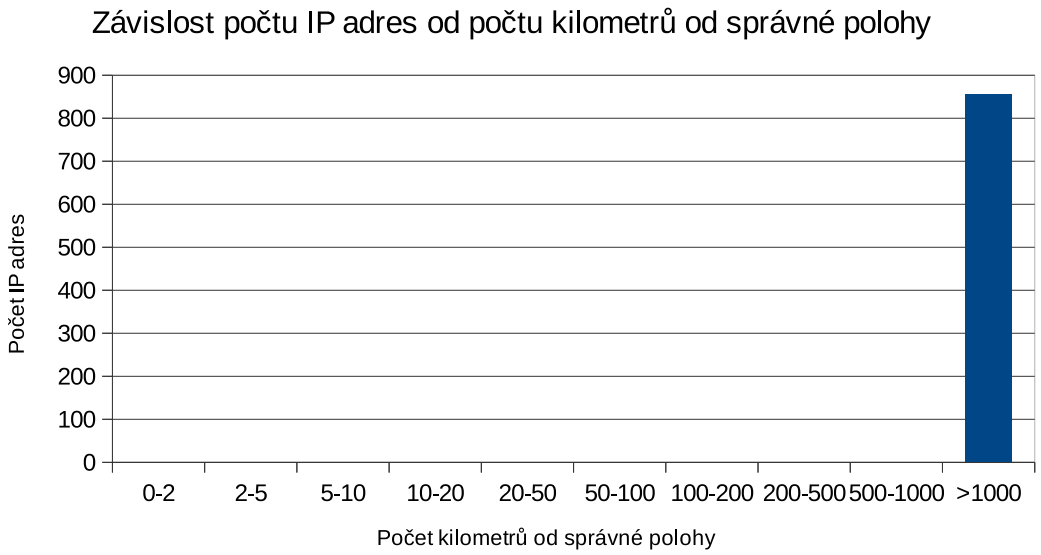
Z obr. 3.4 a obr. 3.5 by se mohlo na první pohled zdát, že přesnost databáze HostIP není žádná. Můj systém pro automatický odhad polohy zadaných IP adres jsem oproti API této databáze odladil a vše fungovalo bez problému. Bohužel než jsem dospěl do fáze, kdy jsem celý systém mohl otestovat na větším množství IP adres, tak rozhraní této databáze přestalo poskytovat výsledky a údaje je nyní možné získávat pouze zadáváním do textového pole na webu. Grafy tedy reprezentují skutečnost, že API této databáze je v současné době nedostupné (minimálně od února 2016).



Obr. 3.3: Absolutní přesnost odhadu polohy databáze DB-IP/IPtoCity



Obr. 3.4: Relativní přesnost odhadu polohy databáze HostIP



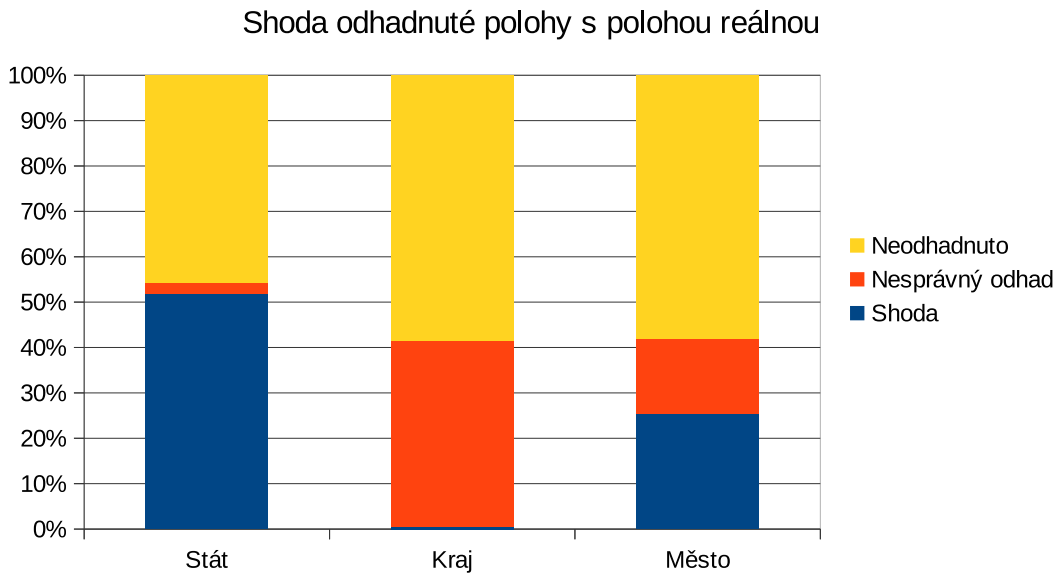
Obr. 3.5: Absolutní přesnost odhadu polohy databáze HostIP

3.2.3 freegeoip.net

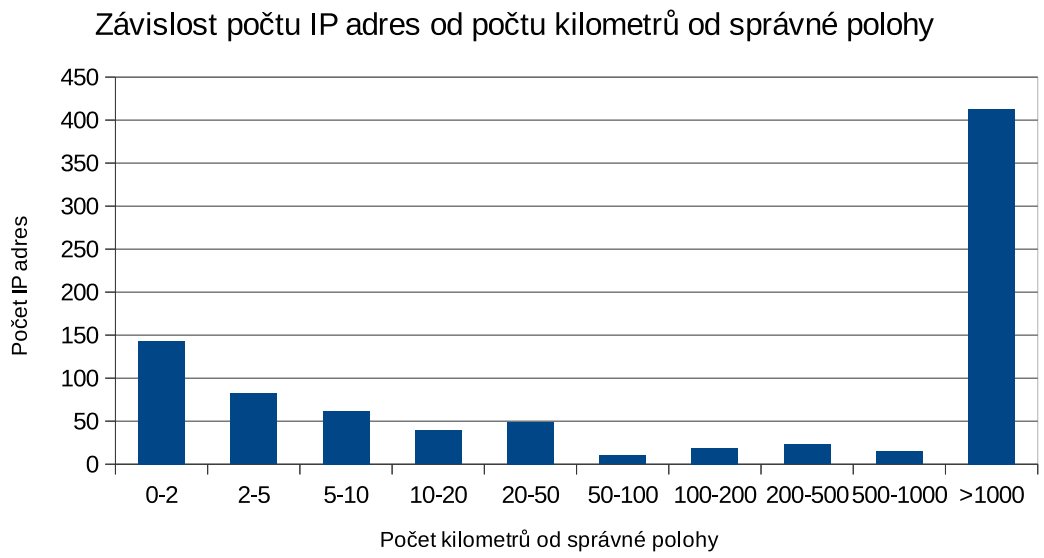
Databáze se potýká s obdobnými technickými problémy jako rozhraní HostIP. Služby poskytované freegeoip.net jsou nedostupné přibližně od dubna 2016. Výhodou této databáze je, že celý systém je dostupný na GitHubu[1], takže kdokoliv této služby potřebuje využít, může si systém rozjet na vlastním serveru. Před odstavením této služby jsem testoval nekompletní seznam uzlů experimentální sítě PlanetLab, který čítal 855 IP adres (tedy o 136 méně než finální verze seznamu). Rozhodl jsem se tato data použít, protože se jedná o poměrně velké množství IP adres.

Na obr. 3.6 je graf, ze kterého můžeme usuzovat poměrně malou přesnost odhadu této databáze. Přibližně 50% shoda u položky *Stát* a zhruba 25% shoda u položky *Město* je velmi málo. Nepřekvapí tedy ani téměř nulová shoda u položky *Kraj*. Graf na obr. 3.7 potvrzuje, že i absolutní přesnost odhadu této databáze je velmi malá. Dá se říci, že databáze buď odhadne polohu s přesností do 50 km nebo vůbec.

[1] `https://github.com/fiorix/freegeoip`



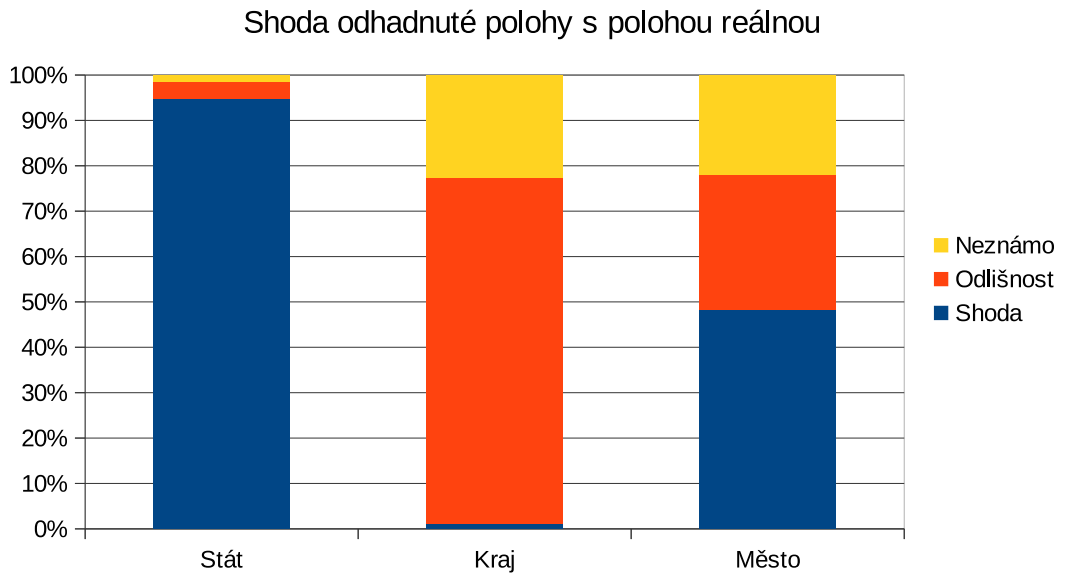
Obr. 3.6: Relativní přesnost odhadu polohy databáze freegeoip.net



Obr. 3.7: Absolutní přesnost odhadu polohy databáze freegeoip.net

3.2.4 MaxMind/GeoLite2

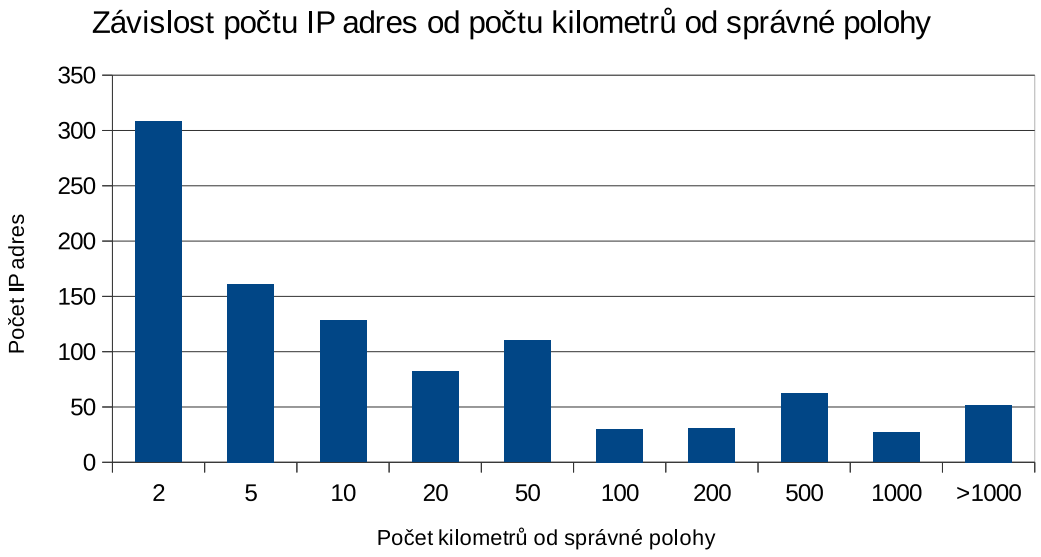
Na obr. 3.8 lze vidět, že přesnost odhadu *Státu* je velmi vysoká, přes 90 %. S přesností odhadu *Města* je to horší, ale stále je procentuální shoda téměř 50 %. Ovšem shoda v odhadu *Kraje* je téměř nulová. Graf na obr. 3.9 nám ukazuje, že databáze vykazuje poměrně velkou absolutní přesnost, téměř dvě třetiny IP adres bylo odhadnuto s přesností do 10 km.



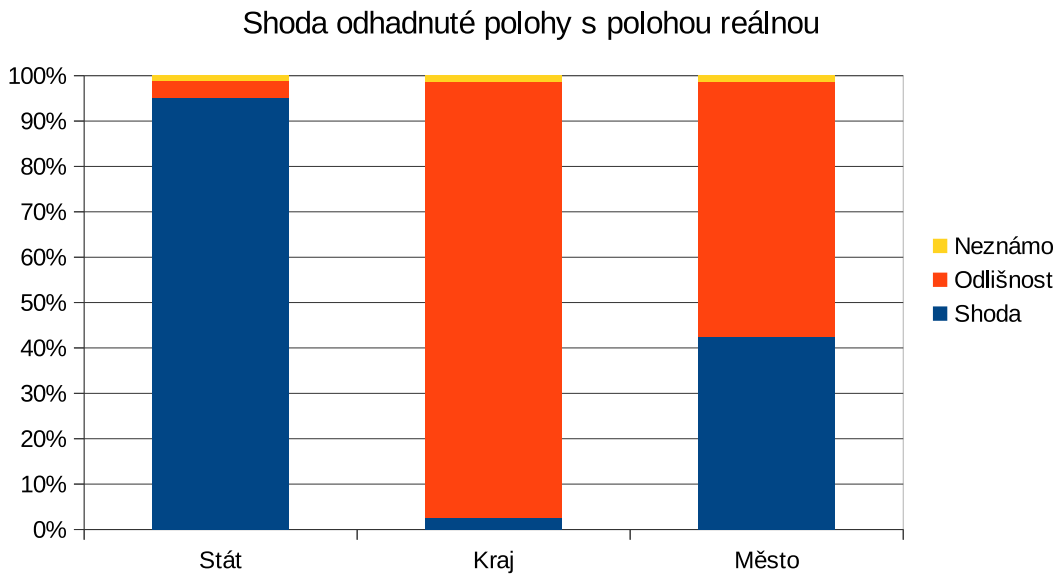
Obr. 3.8: Relativní přesnost odhadu polohy databáze MaxMind/GeoLite2

3.2.5 IP2Location/DB11.LITE

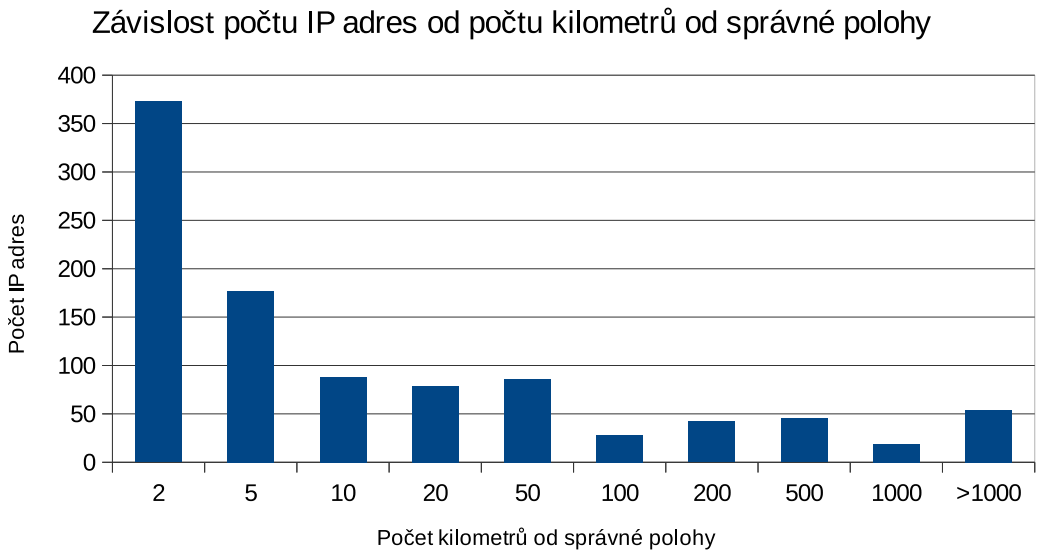
Z grafu na obr. 3.10 je patrné, že nejvyšší procento shody je u položky *Stát* a přibližně poloviční procentní shoda je u položky *Město*. Položka *Kraj* opět vykazuje téměř nulovou procentní shodu. Ve srovnání s ostatními databázemi je však z grafu patrné minimální procento neznámých odhadů. Na obr. 3.11 můžeme vidět poměrně velmi vysokou absolutní přesnost odhadu. U více než třetiny ze zadaných IP adres byla poloha odhadnuta s přesností do 2 km.



Obr. 3.9: Absolutní přesnost odhadu polohy databáze MaxMind/GeoLite2



Obr. 3.10: Relativní přesnost odhadu polohy databáze IP2Location/DB11.LITE



Obr. 3.11: Absolutní přesnost odhadu polohy databáze IP2Location/DB11.LITE

3.3 Porovnávání přesnosti jednotlivých databází

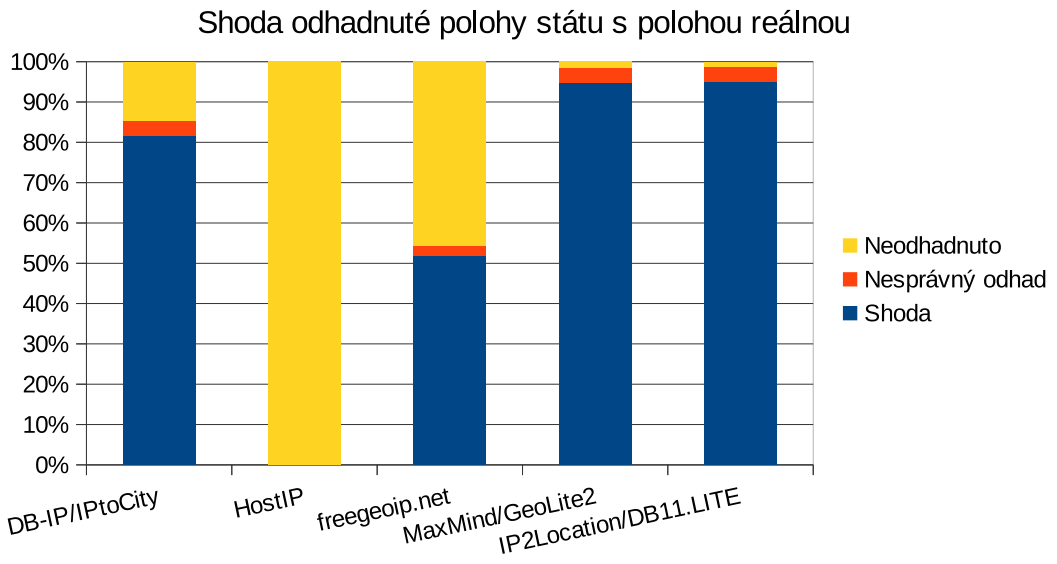
3.3.1 Relativní chyba odhadu

Stát

Z tabulky tab. 3.2 vyplývá, že databáze DB-IP/IPtoCity, MaxMind/GeoLite2 a IP2Location/DB11.LITE dosahují velmi vysoké shody v odhadu státu. Databáze freegeoip.net dosahuje poloviční hodnoty.

	Shoda	**Odlišnost**	**Neznámo**
DB-IP/IPtoCity	810	37	144
HostIP	0	0	855
freegeoip.net	444	20	391
MaxMind/GeoLite2	939	37	15
IP2Location/DB11.LITE	942	37	12

Tab. 3.1: Relativní přesnost jednotlivých databází v odhadu státu



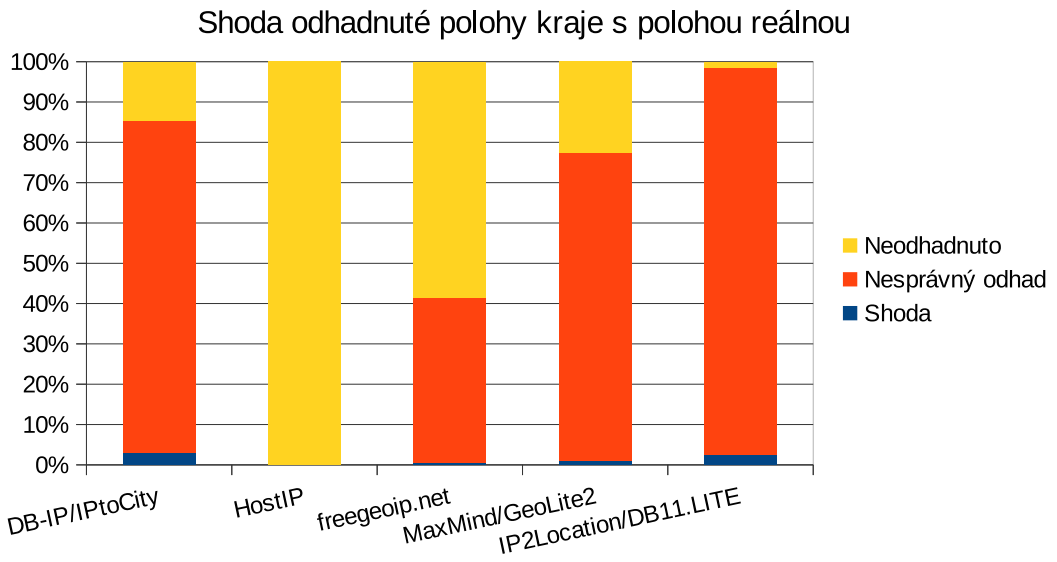
Obr. 3.12: Relativní přesnost jednotlivých databází v odhadu státu

Kraj

Na grafu na obr. 3.13 můžeme vidět, že všechny databáze jsou na tom velmi podobně co se týče shody v odhadu kraje. Když se ale podíváme na odlišnosti v odhadu, tak nejvíce údajů získáme z databáze IP2Location/DB11.LITE. Důvodů, proč se získaný údaj neshoduje, může být hned několik. Může se jednat o jiné pojetí termínu *Kraj* na straně databáze (např. *Jihomoravský kraj* vs. *Morava*) nebo samozřejmě může jít o zcela špatný odhad.

	Shoda	**Odlišnost**	**Neznámo**
DB-IP/IPtoCity	29	818	144
HostIP	0	0	855
freegeoip.net	4	351	500
MaxMind/GeoLite2	10	757	224
IP2Location/DB11.LITE	25	952	14

Tab. 3.2: Relativní přesnost jednotlivých databází v odhadu kraje



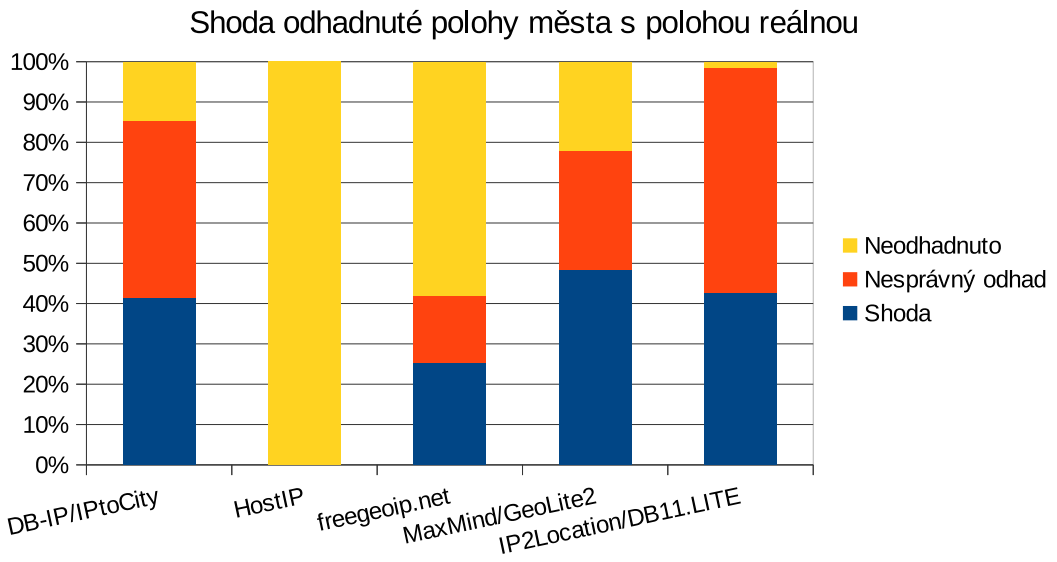
Obr. 3.13: Relativní přesnost jednotlivých databází v odhadu kraje

Město

Z tabulky tab. 3.3 vyplývá, že největší počet shody v odhadu města má databáze MaxMind/GeoLite2 následovaná databázemi IP2Location/DB11.LITE a DB-IP/IPtoCity. Databáze freegeoip.net má přibližně poloviční počet správně odhadnutých měst.

	Shoda	**Odlišnost**	**Neznámo**
DB-IP/IPtoCity	411	436	144
HostIP	0	0	855
freegeoip.net	216	143	496
MaxMind/GeoLite2	479	294	218
IP2Location/DB11.LITE	422	555	14

Tab. 3.3: Relativní přesnost jednotlivých databází v odhadu kraje



Obr. 3.14: Relativní přesnost jednotlivých databází v odhadu města

3.3.2 Absolutní chyba polohy

V tabulce tab. 3.4 můžeme vidět velmi vysokou absolutní přesnost odhadu databáze IP2Location/DB11.LITE, kdy více než třetina zadaných IP adres byly odhadnuty s přesností do 2 km. Další databáze s velkým množstvím zadaných IP adres odhadnutých s přesností do 2 km je MaxMind/GeoLite2 následována databází DB-IP/IPtoCity.

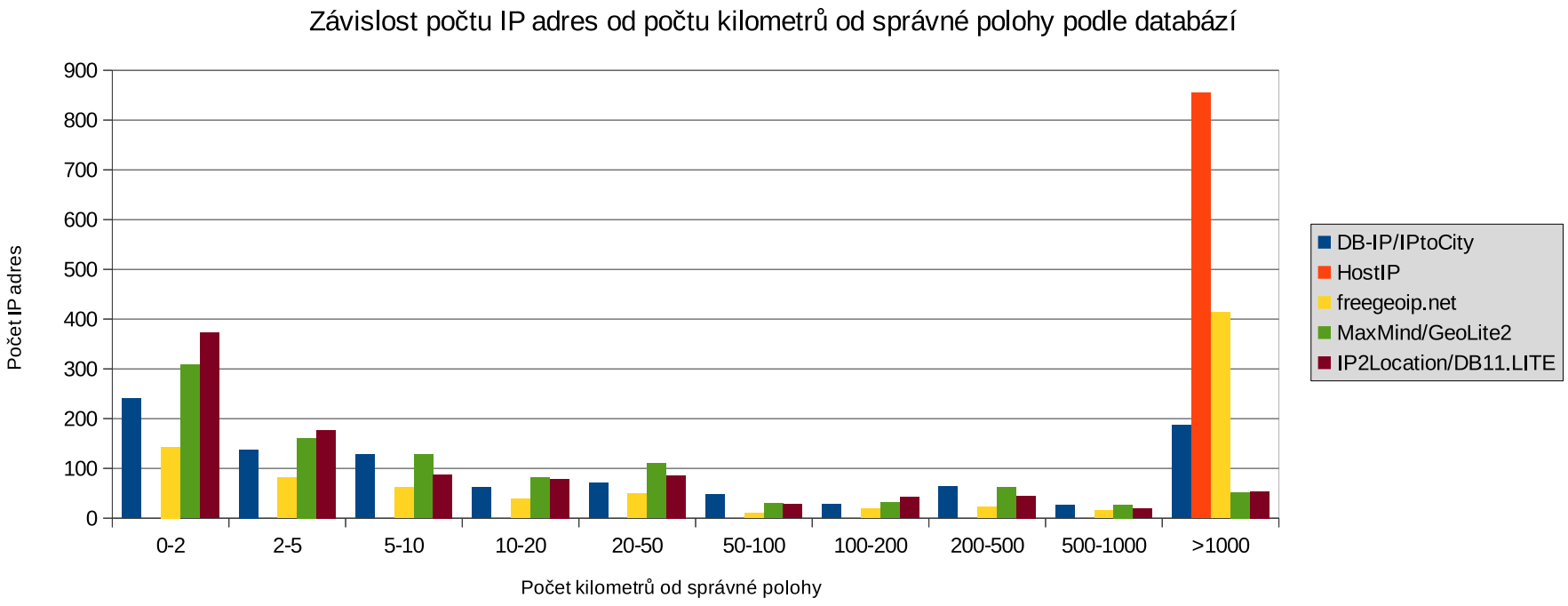
Z tabulky tab. 3.5 vyplývá, že nejlepšího mediánu počtu kilometrů od správné polohy dosáhla databáze IP2Location/DB11.LITE s hodnotou 3,65 km následována databázemi freegeoip.net (5,54 km), MaxMind/GeoLite2 (5,62 km) a DB-IP/IPtoCity (5,85 km). Nejlepšího průměrného počtu kilometrů od správné polohy dosáhla databáze MaxMind/GeoLite2 s hodnotou 204,81 km, poté následují databáze freegeoip.net (211,71 km), DB-IP/IPtoCity (302,46 km) a IP2Location/DB11. LITE (316,59 km).

Počet kilometrů od správné polohy	DB-IP/IPtoCity	HostIP	freegeoip.net	MaxMind/GeoLite2	IP2Location/DB11.LITE
0-2	240	0	143	308	373
2-5	138	0	82	161	177
5-10	128	0	62	128	88
10-20	62	0	39	82	79
20-50	71	0	49	110	86
50-100	47	0	10	30	28
100-200	28	0	19	31	42
200-500	64	0	23	62	45
500-1000	26	0	15	27	19
>1000	187	855	413	52	54

Tab. 3.4: Absolutní přesnost jednotlivých databází v odhadu polohy

	DB-IP/IPtoCity	HostIP	freegeoip.net	MaxMind/GeoLite2	IP2Location/DB11.LITE
Medián	5,85		5,54	5,62	3,65
Aritmetický průměr	302,46		211,71	204,81	316,59
Směrodatná odchylka	1314,42		914,68	921,78	1386,27

Tab. 3.5: Statistické údaje o absolutní přesnosti jednotlivých databází v odhadu polohy



Obr. 3.15: Absolutní přesnost jednotlivých databází v odhadu polohy

3.3.3 Srovnání výsledků s dalšími publikacemi

Procentuální shoda relativních odhadů poloh států je obecně velmi vysoká, publikace[4] uvádí shodu přes 80 %, což moje výsledky potvrzují (shoda přes 80 % vyjma databáze freegeoip.net). Zahraniční publikace neuvádějí relativní odhady poloh krajů. Procentuální shoda relativních odhadů poloh měst je v zahraniční publikaci[4] kolem 20 %. Mé výsledky dosahují přesnosti vyšší – až 50 %. Co se týče absolutní přesnosti odhadu polohy, mé výsledky korespondují se zahraničními publikacemi[3][4][5] (více než polovina IP adres byla určena s přesností do 20 km).

4 ZÁVĚR

Na začátku této práce jsem se seznámil se způsoby, kterými je možné odhadovat geografickou pozici IP adres, a s jednotlivými volně dostupnými geolokačními databázemi. Dále jsem popsal a vytvořil systém, který načte zadaný seznam IP adres spolu s dalšími údaji, získá informace z jednotlivých databází, porovná a uloží výsledky do souboru.

Zjistil jsem, že přesnost volně dostupných geolokačních databází je velmi různá a databáze od databáze se přesnost lokalizace velmi liší (IP adresa z Brna byla jednou databází lokalizována blízko Düsseldorfu v Německu, zatímco jiná databáze polohu určila v Brně s velkou přesností). Jednotlivé databáze také vrací data v různých jazycích, formátech, někdy také se speciálními znaky jazyka dané země, což jsem vyřešil zavedením seznamu slov k oříznutí a překladového slovníku.

Dále jsem vytvořený systém otestoval na IP adresách uzlů experimentální sítě PlanetLab a provedl porovnání přesnosti vstupních dat a získaných údajů. Databáze IP2Location/DB11.LITE, MaxMind/GeoLite2 a DB-IP/IPtoCity poskytují výsledky velmi podobné přesnosti. Všechny tři jmenované databáze dosahují více než 80% shody v odhadu státu a více než 40% shody v odhadu města. Kraj žádná z databází v podstatě nedokázala určit správně, resp. zřejmě bylo by potřeba obsáhlého překladového slovníku. Absolutní přesnost v odhadu polohy je také poměrně zajímavá, všechny databáze poskytly výsledky s mediánem počtu kilometrů od správné polohy do 6 km.

LITERATURA

[1] PUŽMANOVÁ, Rita. *TCP/IP v kostce*. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009, 619 s. ISBN 978-80-7232-388-3.

[2] *Linux: dokumentační projekt*. 4., aktualiz. vyd. Brno: Computer Press, 2007, 1334 s. ISBN 978-80-251-1525-1.

[3] POESE, Ingmar, Steve UHLIG, Mohamed Ali KAAFAR, Benoit DONNET a Bamba GUEYE. *IP Geolocation Databases: Unreliable?*. ACM SIGCOMM Computer Communication Review [online]. 2011, 41(2), 53- [cit. 2016-05-10]. DOI: 10.1145/1971162.1971171. ISSN 01464833. Dostupné z: `http://portal.acm.org/citation.cfm?doid=1971162.1971171`

[4] SHAVITT, Yuval a Noa ZILBERMAN. *A Geolocation Databases Study*. IEEE Journal on Selected Areas in Communications [online]. 2011, 29(10), 2044-2056 [cit. 2016-05-10]. DOI: 10.1109/JSAC.2011.111214. ISSN 0733-8716. Dostupné z: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6081357`

[5] HUFFAKER, Bradley a Marina FOMENKOV. *Geocompare: a comparison of public and commercial geolocation databases - Technical Report*.C ooperative Association for Internet Data Analysis (CAIDA) [online]. 2011 [cit. 2016-05-10]. Dostupné z: `https://www.caida.org/publications/papers/2011/geocompare-tr/`

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

IP	Internet Protocol
DNS	hiearchický systém doménových jmen – Domain Name System
IANA	Internet Assigned Numbers Authority
RTT	obousměrné zpoždění – Round Trip Time
API	rozhraní pro programování aplikací – Application Programming Interface
GPS	globální polohovací systém – Global Positioning System
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
JSON	JavaScriptový objektový zápis – JavaScript Object Notation
JSONP	JSON with Padding
XML	rozšiřitelný značkovací jazyk – Extensible Markup Language
CSV	hodnoty oddělené čárkami – Comma-Separated Values
URL	jednotná adresa zdroje – Uniform Resource Locator

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

- 164249.pdf – tato bakalářská práce ve formátu PDF
- geoipchecker.py – vytvořený systém pro automatický odhad polohy zadaných IP adres v jazyce Python (ověřena funkčnost s verzí 2.7.6)
- geoipstats.py – vytvořený statistický systém v jazyce Python (ověřena funkčnost s verzí 2.7.6)
- planetlab.node – vstupní soubor se seznamem uzlů experimentální sítě PlanetLab
- replace_dict.txt – použitý překladový slovník
- cutoff_list.txt – použitý seznam slov k ořezu