Limpando dados do OpenStreetMap com Python e mongoDB

Márcio Ozório de Jesus

Introdução

O OpenStreetMap (OSM) é um projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo. Traduzindo para português o nome significa Mapa Aberto de Ruas.

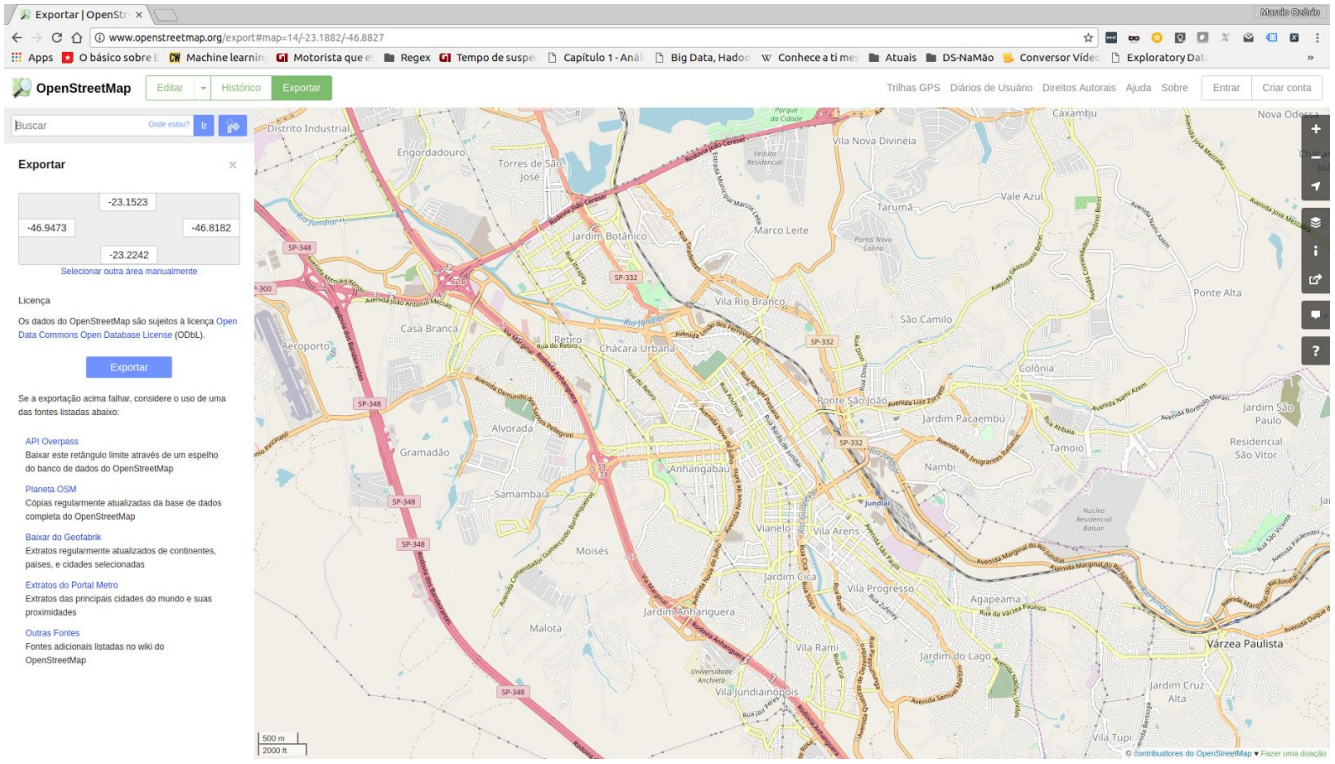
Diversas pessoas do mundo adicionam novos mapas geográficos ou realizam correções e melhoria das informações já existentes. O formato padrão para envio e recebimento de informações é o XML. O site oficial do OpenStreetMap é o http://www.openstreetmap.org. Para saber mais informações sobre o conteúdo, padrões, etc, acesse o site http://wiki.openstreetmap.org. O endereço http://wiki.openstreetmap.org/wiki/Pt:Main_Page tem parte do conteudo traduzido, guia para iniciantes e informações sobre a comunidade do OpenStreetMap no Brasil.

Área do mapa: Cidade de Jundiaí e Região, São Paulo, Brasil.

Escolhi esta localização, pois, nasci e moro atualmente em Jundiaí e achei interessante saber como estão seus dados no OpenStreetMap nesta região, além da maior facilidade em realizar análises, uma vez que já conheço os detalhes da região. Caso existam muitos problemas com os dados, poderia até contribuir com melhorias.

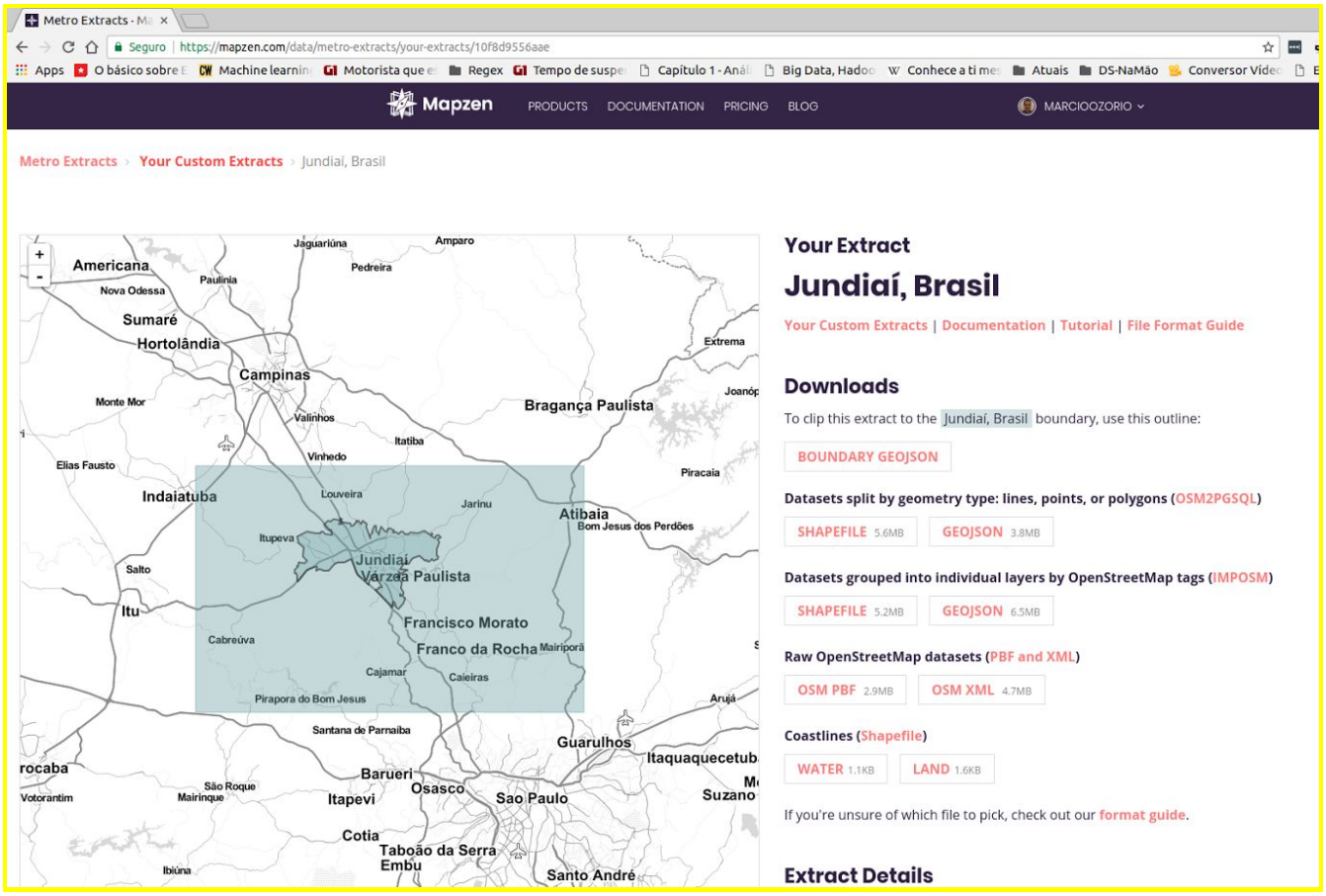
Para conhecimento da base de dados, baixei uma área reduzida da Cidade de Jundiaí diretamente do OpenStreetMap:

http://www.openstreetmap.org/export#map=14/-23.1882/-46.8827



O mapa completo foi extraído do MapZen utilizando um recorte considerando a cidade de Jundiaí e região:

https://mapzen.com/data/metro-extracts/your-extracts/10f8d9556aae



1. Problemas encontrados no Mapa
    a. Nomes de cidades incorretos
    b. Telefones fora do formato padrão
    c. Códigos postais (CEP) fora do formato padrão
    d. Tipos de ruas fora do padrão e com caracteres especiais
2. Limpando os dados através de programação
3. Preparação do arquivo e importação no mongoDB
4. Visão geral dos dados (Data Overview)
5. Explorações adicionais usando queries no mongoDB
6. Sugestões de Melhoria
7. Conclusão

1. Problemas encontrados no Mapa

a. Nomes de cidades incorretos

Segue abaixo as cidades incorretas destacadas abaixo juntamente com as quantidades de ocorrências, identificadas pelo programa: 1.a. - audit_cidades.ipynb

```
Araçariguama: 3
Atibaia: 167
Botujuru: 1
Caieiras: 4
Cajamar: 13
Campo Limpo Paulista: 11
Campo0 Limpo Paulista: 1
Francisco Morato: 9
Franco da Rocha: 86
Indaiatuba: 1
Itupeva: 13
Jarinu: 1
Jundiai: 11
JundiaÌ: 2
Jundiaí: 350
JUndiaí: 1
Jundiái: 1
Louveira: 1
Mairipora: 1
Mairiporã: 382
Mariporã: 2
Pirapora do Bom Jesus: 2
São Bernardo do Campo: 2
São Paulo: 11
Vinhedo: 4
Várzea Paulista: 18
Várzea Paulista, SP: 1
```

b. Telefones fora do formato padrão

Na documentação do OpenStreetMap (que pode ser acessada clicando aqui), podemos encontrar o formato padrão para telefones.

Segue o resultado dos telefones inconsistentes identificados através do programa: 1.b. - audit_telefones.ipynb

```
node tag {'k': 'phone', 'v': '11 4038-2655'}
node tag {'k': 'phone', 'v': '+55 11 45272373'}
node tag {'k': 'phone', 'v': '(11) 4522-3149'}
node tag {'k': 'phone', 'v': ': 11 3917-0751'}
way tag {'k': 'phone', 'v': '+55 11 4531 0082;+55 11 4531 0083'}
way tag {'k': 'phone', 'v': '+55 11 44466767'}
way tag {'k': 'phone', 'v': '+55 11 4526-1246;+55 11 4588-9446'}
way tag {'k': 'phone', 'v': '+55 11 45841402'}
way tag {'k': 'phone', 'v': '+ 55 11 4525 5000'}
way tag {'k': 'phone', 'v': '(11) 4496-2754'}
```

c. Códigos postais (CEP) fora do formato padrão

Conforme a documentação do OpenStreetMap (acesse aqui) o formato do código postal depende do local do mapa. Como estamos no Brasil, utilizei como referência a definição realizada Correios, que pode ser consultada clicando aqui.

Utilizando o programa 1.c - audit_codigo_postal.ipynb, foram identificadas as inconsistências abaixo, juntamente com as quantidades:

```
07600000: 129
12940700: 1
12942540: 1
12942655: 10
12943000: 2
12943310: 7
12943320: 13
12943330: 31
12943340: 38
12943350: 16
12943370: 13
12943380: 3
12943500: 12
12947452: 8
13203280: 1
13221550: 1
13240000: 6
3221-390: 1
```

d. Tipos de logradouros fora do padrão ou com caracteres especiais

Segue alguns exemplos de logradouros identificados como fora do padrão. Na fase de limpeza eles serão verificados, e corrigidos se necessário. O programa utilizado para a auditoria foi o 1.d. - audit_nome_das_ruas.ipynb (onde você poderá encontrar lista completa dos logradouros).

```
{'k': 'addr:street', 'v': 'estrada Tahira Eki'}
{'k': 'name', 'v': 'Rua " 30 "'}
{'k': 'name', 'v': 'Rua " 26 "'}
{'k': 'name', 'v': 'Rua "1"'}
{'k': 'name', 'v': 'Rua "04"'}
{'k': 'name', 'v': 'Avenidas das Aves Marinhas'}
{'k': 'name', 'v': 'Rau Diogo Alveres correia'}
{'k': 'name', 'v': 'rua Raul Breesane Malta'}
{'k': 'name', 'v': 'José Pereira da Silva'}
{'k': 'name', 'v': 'Rod. Manoel Silvério Pinto'}
{'k': 'name', 'v': 'Rua Vitoria,'}
{'k': 'name', 'v': 'Rua Manoel Pinto Rodrigues,'}
{'k': 'name', 'v': 'estrada da bucolica'}
{'k': 'name', 'v': 'Av.enida João Casarotto'}
{'k': 'name', 'v': 'Av Jeronimo de Camargo'}
```

2. Limpando os dados

Para a limpeza dos dados, foi criado um programa para correção dos problemas identificados na auditoria, seja Cidade, CEP, Telefone ou Rua/Logradouros.

Após a execução do programa 2. - limpando os dados.ipynb, é exibido um log com todas as correções realizadas.

Segue abaixo alguns exemplos dos caso que foram identificados e corrigidos:

Nomes de cidades

```
Mairipora  =>  Mairiporã
Várzea Paulista, SP  =>  Várzea Paulista
Jundiai  =>  Jundiaí
JundiaÌ  =>  Jundiaí
JundiaÌ  =>  Jundiaí
Campo0 Limpo Paulista  =>  Campo Limpo Paulista
Mariporã  =>  Mairiporã
Mariporã  =>  Mairiporã
```

Telefones:

```
11 4038-2655  =>  +55 11 4038-2655
+55 11 45272373  =>  +55 11 4527 2373
(11) 4522-3149  =>  +55 11 4522-3149
: 11 3917-0751  =>  +55 11 3917-0751
+55 11 4531 0082;+55 11 4531 0083  =>  +55 11 4531 0082;+55 11 4531 0083
+55 11 44466767  =>  +55 11 4446 6767
+55 11 4526-1246;+55 11 4588-9446  =>  +55 11 4526-1246;+55 11 4588-9446
+55 11 45841402  =>  +55 11 4584 1402
+ 55 11 4525 5000  =>  +55 11 4525 5000
(11) 4496-2754  =>  +55 11 4496-2754
```

Código Postal (CEP):

```
07600000  =>  07600-000
13240000  =>  13240-000
3221-390  =>  13221-390
13221550  =>  13221-550
13203280  =>  13203-280
12943310  =>  12943-310
12943370  =>  12943-370
12943340  =>  12943-340
12940700  =>  12940-700
12943350  =>  12943-350
12943500  =>  12943-500
12947452  =>  12947-452
```

Tipos de logradouro / Ruas:

```
estrada Tahira Eki  =>  Estrada Tahira Eki
Rua " 30 "  =>  Rua 30
Rua "1"  =>  Rua 1
Rod. Manoel Silvério Pinto  =>  Rodovia Manoel Silvério Pinto
Rua Vitoria,  =>  Rua Vitoria
estrada da bucolica  =>  Estrada da bucolica
Av.enida João Casarotto  =>  Avenida João Casarotto
R. Jurandir Rodrigues de Castro  =>  Rua Jurandir Rodrigues de Castro
AvenidaAssembléia de Deus Ministério de Belém  =>  Avenida Assembléia de Deus Ministério de
Belém
Av Jeronimo de Camargo  =>  Avenida Jeronimo de Camargo
```

3. Preparação do arquivo e importação no mongoDB

a. Preparando o arquivo para importação

O motivo pela escolha do banco de dados mongoDB é que já tenho uma grande experiência com banco de dados SQL. Por isso, gostaria de ganhar experiência em um banco de dados NOSQL (Not Only SQL) que é amplamente utilizado para trabalhar com dados não estruturados.

Para realizar a importação das informações no banco de dados mongoDB, iremos utilizar o comando **mongoimport**. Mas antes é necessário transformar o formato do arquivo de XML para o formato JSON.

Além disso, é importante realizar algumas modificações, permitindo que cada nó (node ou way) receba um ID, tornando-se assim assim um documento distinto dentro do mongoDB.

Para realizar essa transformação, foi criado o programa 3.a. - XML to JSON - OpenStreetMap.ipynb.

O programa irá ler o arquivo com estrutura XML chamado *output_jundiaieregiao.osm* e criará o arquivo *output_jundiaieregiao.osm.json*.

b. Importação no banco de dados mongoDB

Conforme dito anteriormente, para realizar a importação do arquivo no banco de dados mongoDB, utilizaremos o comando **mongoimport** que deve ser executado diretamente na linha ou prompt de comando. Veja a execução do comando (ambiente linux) e log de execução:

```
$ mongoimport -d examples -c jundiaieregiao --file output_jundiaieregiao.osm.json
--type=json --drop
```

Foram importados **344.958** documentos.

4. Visão geral dos dados (Data Overview)

Detalhes dos arquivos utilizados:

Nome do arquivo	Tamanho	Comentário
jundiai-small.osm	6 MB	Arquivo inicial utilizado para estudo
jundiai_e_regiao_map_zen.osm	66 MB	Arquivo original extraído do MapZen
output_jundiaieregiao.osm	67 MB	Arquivo com realização da limpeza
output_street.osm.json	116 MB	Arquivo convertido para o formato JSON

Obs.: Os comandos abaixo foram executados diretamente no aplicativo do mongoDB. Também podem executados diretamente no Python através da API pymongo.

Número de documentos:

```
> db.jundiaieregiao.find().count()
344958
```

Número de logradouros/ruas identificados:

```
> db.jundiaieregiao.find({"$or" : [{"$and" : [{"name" : {"$exists" : 1}},
{"highway" : {"$exists" : 1}}]}, {"address.street" : {"$exists" :
1}}]}).count()

14543
```

Número de usuários únicos:

```
> db.jundiaieregiao.distinct('created.user').length
328
```

Top 5 usuários que mais contribuíram:

```
> db.jundiaieregiao.aggregate([{"$group":{"_id" : "$created.user", "count" :
{"$sum" : 1}}}, {"$sort" : {"count" : -1}}, {"$limit" : 1}])

{ "_id" : "AjBelnuovo", "count" : 88960 }
{ "_id" : "natinacio", "count" : 43120 }
{ "_id" : "patodiez", "count" : 22154 }
{ "_id" : "Louis Goyard", "count" : 16178 }
{ "_id" : "Roberto Costa", "count" : 13975 }
```

Número de usuários que aparecem apenas uma vez:

```
> db.jundiaieregiao.aggregate([{"$group":{"_id" : "$created.user", "count" :
{"$sum":1}}}, {"$group":{"_id" : "$count", "num_users" : {"$sum":1}}},
{"$sort"  : {"_id":1}}, {"$limit":1}])

{ "_id" : 1, "num_users" : 62 }
```

Cidades e quantidade respectiva de documentos encontrados nesta base de dados:

```
> db.jdiregfinal.aggregate([{"$match" : {"address.city" : {"$exists" : 1}}},
{"$group":{"_id" : "$address.city", "count" : {"$sum" : 1}}}, {"$sort" :
{"count" : -1}}])

{ "_id" : "Mairiporã", "count" : 383 }
{ "_id" : "Jundiaí", "count" : 364 }
{ "_id" : "Atibaia", "count" : 167 }
{ "_id" : "Franco da Rocha", "count" : 85 }
{ "_id" : "Várzea Paulista", "count" : 19 }
{ "_id" : "Itupeva", "count" : 13 }
{ "_id" : "Cajamar", "count" : 13 }
{ "_id" : "Campo Limpo Paulista", "count" : 12 }
{ "_id" : "São Paulo", "count" : 11 }
{ "_id" : "Francisco Morato", "count" : 9 }
{ "_id" : "Caieiras", "count" : 4 }
{ "_id" : "Vinhedo", "count" : 4 }
{ "_id" : "Araçariguama", "count" : 3 }
{ "_id" : "São Bernardo do Campo", "count" : 2 }
{ "_id" : "Pirapora do Bom Jesus", "count" : 2 }
{ "_id" : "Mariporã", "count" : 2 }
{ "_id" : "Louveira", "count" : 1 }
{ "_id" : "Indaiatuba", "count" : 1 }
{ "_id" : "Jarinu", "count" : 1 }
{ "_id" : "Botujuru", "count" : 1 }
```

5. Explorações adicionais usando queries no mongoDB:

Cinco tipos de restaurantes mais populares:

```
> db.jundiaieregiao.aggregate([{"$match" : {"cuisine" : {"$exists" : 1},
"amenity" : "restaurant"}}, {"$group" : {"_id" : "$cuisine", "count" : {"$sum"
: 1}}}, {"$sort" : {"count" : -1}}, {"$limit" : 5}])

{ "_id" : "regional", "count" : 15 }
{ "_id" : "pizza", "count" : 6 }
{ "_id" : "italian", "count" : 6 }
{ "_id" : "japanese", "count" : 4 }
{ "_id" : "steak_house", "count" : 3 }
```

Igrejas com maior quantidade (por denominação):

```
> db.jundiaieregiao.aggregate([{"$match" : {"denomination" : {"$exists" :
1}}}, {"$group":{"_id" : "$denomination", "count" : {"$sum" : 1}}}, {"$sort" :
{"count" : -1}}, {"$limit" : 10}])

{ "_id" : "roman_catholic", "count" : 38 }
{ "_id" : "catholic", "count" : 15 }
{ "_id" : "pentecostal", "count" : 11 }
{ "_id" : "neo-pentecostal", "count" : 7 }
{ "_id" : "baptist", "count" : 5 }
{ "_id" : "mormon", "count" : 2 }
{ "_id" : "presbyterian", "count" : 2 }
{ "_id" : "adventist", "count" : 1 }
{ "_id" : "sunni", "count" : 1 }
{ "_id" : "Batista", "count" : 1 }
```

Anos que tiveram maiores quantidades de contribuição:

Podemos notar que os últimos anos foram os que mais tivemos contribuições, havendo um grande pico no ano de 2015.

```
> db.jundiaieregiao.aggregate([{"$project" : {"year" : {"$substr" :
["$timestamp", 0, 4]}}}, {"$group" : {"_id" : "$year", "count" : {"$sum" :
1}}}, {"$sort" : {"count": -1}}, {"$limit" : 5}])

{ "_id" : "2015", "count" : 109871 }
{ "_id" : "2017", "count" : 68979 }
{ "_id" : "2016", "count" : 61444 }
{ "_id" : "2013", "count" : 31429 }
{ "_id" : "2012", "count" : 30101 }
```

Locais comuns nas cidades com as respectivas quantidades:

```
> db.jundiaieregiao.aggregate([{"$match" : {"$or" : [{"amenity" : "hospital"},
{"amenity" : "police"}, {"amenity" : "prison"}, {"amenity" : "bank"},
{"amenity" : "cinema"}, {"amenity" : "university"}, {"amenity" : "teatre"},
{"amenity" : "marketplace"}]}}, {"$group" : {"_id" : "$amenity", "count" :
{"$sum" : 1}}}, {"$sort" : {"count": 1}}])

{ "_id" : "cinema", "count" : 2 }
{ "_id" : "university", "count" : 3 }
{ "_id" : "marketplace", "count" : 4 }
{ "_id" : "prison", "count" : 7 }
{ "_id" : "police", "count" : 16 }
{ "_id" : "hospital", "count" : 26 }
{ "_id" : "bank", "count" : 66 }
```

Cidades com maior número de Escolas:

```
> db.jundiaieregiao.aggregate([{"$match" : {"amenity" : "school"}}, {"$match"
: {"address.city": {"$exists" : 1}}}, {"$group":{"_id" : {"cidade":
"$address.city", "escola": "$amenity"}, "count" : {"$sum" : 1}}}, {"$sort" :
{"count" : -1}}, {"$limit" : 3}])

{ "_id" : { "cidade" : "Jundiaí", "escola" : "school" }, "count" : 34 }
{ "_id" : { "cidade" : "Franco da Rocha", "escola" : "school" }, "count" : 16
}
{ "_id" : { "cidade" : "Várzea Paulista", "escola" : "school" }, "count" : 7 }
```

Cidades com maior número de Hospitais:

```
> db.jundiaieregiao.aggregate([{"$match" : {"amenity" : "hospital"}},
{"$match" : {"address.city": {"$exists" : 1}}}, {"$group":{"_id" : {"cidade":
"$address.city", "hospital": "$amenity"}, "count" : {"$sum" : 1}}}, {"$sort" :
{"count" : -1}}, {"$limit" : 3}])

{ "_id" : { "cidade" : "Jundiaí", "hospital" : "hospital" }, "count" : 6 }
{ "_id" : { "cidade" : "Franco da Rocha", "hospital" : "hospital" }, "count" :
3 }
{ "_id" : { "cidade" : "Caieiras", "hospital" : "hospital" }, "count" : 2 }
```

6. Sugestões de melhoria

a. Melhoria das informações de endereço a partir do código postal (CEP)

Os Correios (empresa responsável pelo envio e recebimento de correspondências) disponibiliza de forma gratuita uma funcionalidade via Webservice onde é possível consultar o endereço a partir de um determinado número de código postal (CEP).

A partir do número do CEP, o webservice retorna as seguintes informações:

- Endereço
- Complemento 1
- Complemento 2
- Bairro
- Cidade
- UF

Algumas melhorias que podem ser conseguidas utilizando as informações retornadas pelos Correios. Podemos verificar cada informação existente, como Logradouro, Bairro, Cidade, UF ou então adicioná-las individualmente caso estejam faltando.

Um exemplo de uma simples solução de consulta dos dados dos Correios pode ser acessada neste endereço: http://www.eduardorizo.com.br/2014/12/04/correios-webservice-para-consulta-de-enderecos-a-partir-de-um-cep/

Com o intuito de evitar novos problemas, qualquer tipo de alteração dos dados devem ser profundamente estudados. Um cuidado que se deve ter por exemplo são os casos onde utiliza-se CEPs únicos (CEPs gerais), onde não há distinção de logradouros. Para testes casos, geralmente os três últimos dígitos são zeros. Para maiores detalhes, clique aqui.

b. Prevenção para que dados já validados não sejam sobrepostos

A modificação dos dados é algo constante. No entanto, poderia ser criadas estratégias com o objetivo de evitar com que uma informação que já passou por um processo de validação (seja ele através dos correios ou de outra fonte) seja sobreposta indevidamente.

Uma tag adicional poderia ser adicionada (no item anterior *"a."* por exemplo) indicando que o dado já passou por um processo de validação. Assim poderia ser definida uma política diferenciada para poder realizar modificações, ou seja, se a informação já estiver validada, o cuidado na realização de qualquer modificação deverá ser muito maior.

c. Validação da existência de locais utilizando outras fontes de dados

Com a finalidade exclusiva de validação da existência das informações, para melhoria da qualidade dos dados, poderia ser realizado uma validação cruzada com outros bancos de dados.

Os locais que forem encontrados correspondentes em outros banco de dados, podem receber uma tag adicional semelhante a relatada no item (b), onde passa a ser um dado mais confiável, portanto, sua sobreposição deve ser controlada.

Um problema que pode ocorrer tanto para este item, quanto para o item b. é o nível de confiança. O quão confiável é a classificação de que a informação foi validada?

Uma sugestão é a criação de uma tabela de classificação de confiança. Por exemplo:

- Validação através de código postal (CEP) = *Confiança nível 1*.
- Validação de existência de local através do Google Maps: *Confiança nível 2*.

- Validação de local através de outros bancos de dados*: Confiança nível 3*.

Dependendo de cada nível de confiança, uma política de tratativa de sobreposição (por exemplo) pode ser definida.

É importante ressaltar que antes de utilizar qualquer banco de dados externo, é necessário verificar os “termos de uso”, como por exemplo no caso do Google Maps.

d. Aumentar o engajamento dos usuários

Com a finalidade de aumentar o engajamento dos usuários, poderia ser criada uma área de ranking no site do OpenStreetMap. Poderia ser por região por exemplo, estimulando assim a competição entre os usuários. Esse conceito é comumente conhecido como Gamification ou Ludificação, e pode ser utilizado de diversas formas. Seguindo com o exemplo que estamos utilizando, o usuário que estiver em uma boa colocação, além de estar a frente no desafio, também ganharia maior destaque sendo uma ótima oportunidade de se fazer conhecido pela comunidade.

Regras bem definidas devem ser estabelecidas para a colocação do ranking garantindo o interesse dos usuários pela disputa de posições. Exemplo: Qual é a pontuação pelo envio de novos mapas? Qual é a pontuação pela correção ou por adicionar novas informações do mapa já existente? etc.

e. Preenchimento de informações faltantes utilizando informações de outros nós

Uma forma de melhorar os dados, tornando-os mais completos é realizando buscas de informações dentro da própria base de dados. Em resumo, podemos prreencher as informações faltantes em um nó utilizando dados de outros nós. Um exemplo é o fato de que rua pode ser utilizada em vários locais. Em um local ela pode estar com a informação bairro sem preenchimento, mas em outro local pode estar preenchido. Isso é muito comum.

Para a realização desta melhoria, devem haver divesos cuidados. Um deles é em relação à abrangência da área a ser verificada. A lógica para realizar a pesquisa em outros nós deve considerar a cidade, uma vez que o mesmo nome de rua pode existir em duas cidades diferentes. Portanto, uma análise detalhada deve ser realizada antes de utilizar as informações de outros nós.

7. Conclusão

Através das pesquisas realizadas, foi contatado uma grande diversidade de informações referentes ao local do mapa que podem ser adicionadas e que existe uma variação significativa dependendo de região para região. Cada país realiza adequações nas informações para a utilização da estrutura.

É preciso se familiarizar com os termos utilizados para classificação das informações, pois estão todos na língua inglesa e pode não refletir a necessidade de todos os locais do planeta.

Para facilitar a padronização das informações, existem grupos de usuários do OpenStreetMap de acordo com a localidade, que trocam informações e se comunicam afim de criarem uma espécie de boas práticas para utilizarem um formato o mais padronizado possível.

No Brasil, temos uma comunidade com o foco ainda em algumas grandes capitais. Em contato com esta comunidade, pude conversar e tiver algumas dúvidas através de um grupo no telegram.

A ferramenta mais utilizada atualmente pelos membros da comunidade é chamada JOSM. Trata-se de um editor criado em java com o foco na manutenção das informações do OpenStreetMap.

Comentei da realização deste projeto com o objetivo da limpeza das informações e acharam interessante, no entanto, se mostraram preocupados com iniciativas deste tipo, pois é algo que precisa ser muito bem discutido, planejado, verificado e aceito. Disseram que é comum alguém realizar correções e logo depois outro usuário realiza a sobreposição com dados incorretos novamente. E se houver atualizações em massa, fica mais difícil de identificar o que está causando o problema. Neste link segue o código de conduta utilizado pela comunidade.

Apesar dos erros encontrados, a maior parte das informações da região extraída estão com uma boa qualidade. Ainda há muito o que melhorar, principalmente no que refere-se a quantidade de informações da região. Por exemplo, na ultima consulta que fizemos em “Locais comuns nas cidades”, identificamos somente 4 supermercados. E posso assegurar que somente em Jundiaí existem pelo menos 30 supermercados. Apesar disso, pudermos ver através das explorações realizadas na base de dados, o crescente aumento de contribuições nos últimos anos. Isto significa que os dados do OpenStreetMap estão cada dia mais completos.

Referências:

Mapas
http://www.openstreetmap.org
https://mapzen.com

OpenStreetMap - Wikipedia
https://pt.wikipedia.org/wiki/OpenStreetMap

OpenStreetMap - Wiki
http://wiki.openstreetmap.org/wiki/Main_Page

Comunidade OpenStreetMap Brasil no Telegram
https://web.telegram.org/#/im?p=@OSMBrasil_Suporte

Correios - Formato do código postal (CEP)
https://www.correios.com.br/para-voce/precisa-de-ajuda/o-que-e-cep-e-por-que-usa-lo/estrutura-do-cep

Documentação Python
https://docs.python.org/3/

MongoDB
https://docs.mongodb.com/

Expressões Regulares
https://docs.python.org/2/howto/regex.html
https://pythex.org/
https://tableless.com.br/o-basico-sobre-expressoes-regulares/

Converter XML para Json (mongoDB)
https://github.com/bestkao/data-wrangling-with-openstreetmap-and-mongodb/blob/master/data-wrangling-with-openstreetmaps.ipynb

Posts sobre XML, Python, Expressões regulares e MongoDB
https://stackoverflow.com

Ludificação (Gamification)
https://pt.wikipedia.org/wiki/Ludifica%C3%A7%C3%A3o

Código de Endereçamento Postal (CEP)
https://pt.wikipedia.org/wiki/C%C3%B3digo_de_Endere%C3%A7amento_Postal

CEP para áreas rurais
https://www.correios.com.br/para-voce/precisa-de-ajuda/o-que-e-cep-e-por-que-usa-lo/cep-para-areas-rurais