UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

TIAGO PAULA E SILVA DE HOLANDA CAVALCANTI

AGRUPANDO GEOLOCALIZAÇÕES PARA INFERIR RESIDÊNCIA

RECIFE

2017

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

TIAGO PAULA E SILVA DE HOLANDA CAVALCANTI

AGRUPANDO GEOLOCALIZAÇÕES PARA INFERIR RESIDÊNCIA

Monografia apresentada ao Centro de Informática (CIN) da Universidade Federal de Pernambuco (UFPE), como requisito parcial para conclusão do Curso de Engenharia da Computação, orientada pelo professor Silvio de Barros Melo.

RECIFE

2017

UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE INFORMÁTICA

CURSO DE BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

TIAGO PAULA E SILVA DE HOLANDA CAVALCANTI

AGRUPANDO GEOLOCALIZAÇÕES PARA INFERIR RESIDÊNCIA

Monografia submetida ao corpo docente da Universidade Federal de Pernambuco, defendida e aprovada em 14 de Julho de 2017.

Banca Examinadora:

Orientador

Silvio de Barros Melo

Doutor

Examinador

Carlos Ferraz

Doutor

AGRADECIMENTOS

Agradeço em primeiro lugar a minha família. Aos meus pais, Teófilo e Ana Cristina de Holanda Cavalcanti, que não mediram esforços para garantir que eu tivesse a melhor educação disponível. À minha irmã, Mariana, que sempre me apoiou e admirou, e ao meu irmão Túlio, que seguiu passos semelhantes aos meus e hoje estuda neste Centro.

Em segundo lugar agradeço a meus colegas de curso: Renê Leite, Moiseis Gauthier, Gustavo Braynner, Saulo Pereira, André Buarque, Rodrigo Santos, Tasso Luís, Thiago Suzart, Rennason Carneiro, entre tantos outros, que estiveram ao meu lado durante essa caminhada. Agradeço novamente a Gustavo, Renê e Moiseis por terem topado participar da Maratona de Programação comigo e assim sermos o primeiro time da UFPE a participar de forma independente da competição.

Em terceiro lugar aos professores dessa Universidade. Em especial Silvio Melo e Carlos Ferraz, que toparam orientar e avaliar este trabalho, respectivamente. Agradeço também a Pedro Manhães de Castro, que ministrou encontros do grupo GEOALGO, onde ocorreram trocas de conhecimento valiosas para a minha formação.

E por último, aos companheiros da In Loco, a maior startup em linha reta do mundo! A todos aqueles que fizeram ou fazem parte do time de dados: Alan Gomes, Tiago Lima, Emanuel Ferreira, Guilherme Peixoto, Caio César, Eduardo Rocha, Rafael Aguiar (RIP), e a todos que ajudaram na concepção do nosso produto de verificação de residência, que hoje é tema do meu trabalho. Deixo também um agradecimento especial a Pedro Augusto Bello, que concebeu o algoritmo que serviu de inspiração para o apresentado aqui.

Obrigado!

“Se cheguei até aqui foi porque me apoiei
nos ombros de gigantes. ”
Isaac Newton

RESUMO

A localização da residência do usuário é uma informação que pode auxiliar sistemas de publicidade, recomendação e mobilidade urbana a serem mais assertivos em suas previsões. Pode auxiliar também sistemas bancários a se protegerem de fraudes, principalmente com a transição de serviços para o mundo online. Proponho neste trabalho um método de inferência da localização da residência do usuário baseado em seu histórico de geolocalizações. Esse método conta com um algoritmo de agrupamento criado para este trabalho e uma estratégia já consolidada de escolha de locais como a residência. Algoritmos de agrupamentos encontrados na literatura não satisfazem os requisitos semânticos do problema. Em um teste com 200 amostras, nosso método teve uma taxa de acerto de 94%.

Palavras-chave: *clustering*, inferência de residência, geolocalização do usuário

ABSTRACT

User home location information can aid marketing, recommendation e urban mobility systems in getting more accurate results. This info can also aid banking systems on protecting themselves against fraud. We propose on this work a method for home location inference based on the user's geolocation history. This method is composed of a clustering algorithm created for this work and a consolidated strategy to evaluate places and choose the most probable home. Algorithms found on the literature usually fail at some semantic points. On a test with 200 samples, our method showed a 94% accuraccy rate.

Keywords: clustering, home location inference, user geolcation

Sumário

1. Introdução

1.1. Motivação

A localização da residência do usuário é uma informação bastante valiosa. Em sistemas de recomendação de anúncios pode ser usada para recomendar propagandas de um mercadinho apenas para as pessoas que moram perto dele; em sistemas de planejamento urbano pode ser útil para estudar padrões de mobilidade; e em sistemas bancários pode ser útil para detectar fraudes. Pode ser útil até para inferir estilos de vida, rotinas e fazer estimativas de renda.

É por tantos motivos como esses que o problema da inferência de residência é bastante estudado, nos mais diversos níveis de granularidade e sobre as mais diversas fontes de dados. A maior parte dos estudos se dão em cima de informações públicas dos usuários, disponíveis em redes sociais como *Twitter* e *Foursquare*. Alguns deles se baseiam nas informações de *checkin*, enquanto outros são capazes de usar informações como redes de amizades e textos publicados para extrair resultados [3]. No entanto, para a grande maioria deles, descobrir a localização da residência a nível de cidade é suficiente.

Sistemas de inferência de residência não são apenas úteis para descobrir a residência quando não se tem essa informação, mas também para validar se essa informação foi passada corretamente por outra fonte quando não se confia nela. É o caso de serviços que hoje exigem comprovante de residência de seus clientes.

Serviços como esses e que operam no mundo online (ou que estão fazendo essa transição), hoje se deparam com um grande desafio na hora de obter essa comprovação por meios tradicionais. O cliente é obrigado a tirar uma foto do comprovante de residência, que passa por complexos sistemas de visão computacional para comprovar a validade do documento e em seguida verificar o endereço escrito na imagem. Sistemas tão complexos tem baixa taxa de confiabilidade e alto custo de manutenção. Assim, grande parte desses comprovantes são levados à revisão manual. O que não é escalável e gera ainda mais custos.

Um grande exemplo atual são os bancos que estão começando a disponibilizar abertura de contas online. E eles sofrem com a perda de clientes que desistem de abrir suas contas apenas pela ineficiência desse processo.

Mas graças ao advento do smartphone com serviços de geolocalização, hoje é possível obter essa comprovação por meios mais modernos, como analisar o histórico de localizações do usuário (desde que autorizado por ele). Essa alternativa usa algoritmos muito mais simples e por isso são mais confiáveis e fáceis de manter. Além, é claro, de ser bastante escalável.

1.2. Objetivos

Assim sendo, este trabalho tem como objetivo propor um método para inferir a localização da residência de um usuário dado o seu histórico de localizações. Visto que geolocalização é uma informação de alta resolução (granularidade), podemos realizar esta inferência a nível de dezenas de metros.

Esse problema, então, pode ser dividido em duas partes: o problema de agrupar corretamente os pontos de localização em locais visitados; e o problema de escolher qual desses locais é a mais provável residência, baseado em suas rotinas de visitas. A segunda parte do problema já é bem resolvida: das abordagens encontradas na literatura, duas são as mais presentes: máximo número de visitas e máximo número de visitas à noite [3]. Usaremos neste trabalho a segunda abordagem, por acreditar que as pessoas geralmente dormem em casa e que se usarmos o dia inteiro como referência podemos aumentar o ruído devido a outros lugares frequentes do usuário, como trabalho ou escola. Sabemos, porém, que a abordagem escolhida pode aumentar o erro sobre a parcela da população que trabalha à noite, mas acreditamos que essa parcela é desprezível em relação ao todo.

Então o foco deste trabalho está em analisar os algoritmos de *clustering* (agrupamento) usados para tal propósito e propor um novo algoritmo para resolver o problema. Com esse algoritmo, e usando a estratégia de local com mais visitas à noite, nós conseguimos ter uma acurácia de 94% na verificação de residência de uma base de 200 usuários.

1.2. Estrutura do trabalho

No capítulo 2 analisaremos os algoritmos de *clustering* usados pela literatura para para agrupar pontos de localização antes de fazer a inferência da localização da residência. No capítulo 3 é apresentado o novo algoritmo criado para resolver esse problema e a explicação teórica de sua corretude. No capítulo 4 mostraremos os experimentos feitos e seus resultados, e no capítulo 5 concluímos o trabalho discutindo possíveis melhorias futuras.

2. Estado da Arte

Nesta seção descreveremos um pouco as abordagens usadas por diversos autores para agrupar pontos de localização em locais visitados, e onde cada uma delas não atende tão bem às especificações do problema de agrupar para inferir a residência do usuário.

2.1. Grid

Começaremos pela abordagem de visualizar o mapa em um *grid*. Não é um algoritmo de *clustering* propriamente dito, mas foi uma técnica utilizada por alguns autores [1, 2, 3] para agrupar pontos antes de inferir a residência do usuário. Consiste em dividir o mapa em células quadradas de igual tamanho, definindo como cluster todos os pontos dentro da mesma célula. Os centroides podem ser tanto a média desses pontos ou apenas o centro do quadrado. A maior parte dos autores usa essa técnica para inferir residências com baixo nível de granularidade, como em [1, 2], que usam células de 25 por 25km. Mas ela também é usada por inferências granularidade maior, como mostrado em *Hu et al.*[3], que usa células de 100 por 100 metros.

Essa técnica funciona muito bem com as inferências de baixa granularidade, como é o caso de quando é suficiente descobrir a cidade que alguém mora. Já não funciona do mesmo jeito com altas granularidades, pois quanto menor forem as células, maior a probabilidade da nuvem de pontos que representa um local ser dividida entre células. Isso pode comprometer por completo os resultados do agrupamento.

2.2. Single-pass clustering

Uma abordagem baseada no algoritmo de agrupamento *Single-pass* [9] é usada por *Gu et al.*[4]. Nessa abordagem inicialmente não existem clusters. Então se varre a lista de localizações do usuário sequencialmente, e para cada localização se acha o centroide de cluster mais próximo dela. Se a distância entre eles for maior do que um limiar L ou se não houverem clusters, é criado um novo cluster com apenas essa nova localização. Caso contrário, essa nova localização se junta ao cluster mais próximo a ela.

Essa abordagem tem a vantagem de ser linear, mas pode facilmente resultar em uma distribuição errônea de clusters. Ela só funcionará perfeitamente se a distância máxima entre pontos do mesmo cluster D_{in}^{max} for menor do que o limiar L e esse limiar for menor do que a distância mínima entre pontos de diferentes clusters D_{out}^{min}. Acontece que em dados reais

D_{out}^{min} geralmente é menor do que D_{in}^{max}, impossibilitando essa condição. Assim, sendo L menor do que D_{in}^{max}, pode acontecer de pontos que deveriam estar no mesmo cluster serem divididos pelo algoritmo em clusters diferentes, enquanto se L for maior do que D_{out}^{min} pode acontecer de pontos que deveriam estar em clusters diferentes serem agrupados pelo algoritmo.

2.3. *Clustering* hierárquico

Essa abordagem tem diversas variantes nos trabalhos analisados. Em sua forma geral, consiste em iniciar o algoritmo com um cluster para cada ponto de localização. Em seguida, os clusters mais próximos uns dos outros se juntam para se tornar um novo cluster. Essa etapa se repete iterativamente, até se deparar com um critério de parada, que geralmente é uma distância mínima entre os clusters. O que diferencia os diversos algoritmos hierárquicos é principalmente a forma com a qual é calculada a distância entre os clusters.

Liu et al. [5] usa a estratégia *single-linkage* [10], onde a distância entre dois clusters *A* e *B* é igual à distância entre os dois pontos mais próximos um do outro, sendo um de *A* e o outro de *B*. *Liu et al.* usa a distância máxima de 150m entre clusters como critério de parada, mas diz que distâncias entre 100m e 500m funcionam bem na prática. Já *Krumm et al.* [6] usa menos que 14m de distância entre clusters como critério de parada, mas alega que o ruído de seu equipamento de GPS tem desvio padrão de apenas 4m, bem abaixo do encontrado em smartphones atuais. *Hoh et al.* [7] também usa uma abordagem hierárquica, apesar de batizar seu algoritmo de *k-means pairwise*. Seu critério de parada é quando todos os centroides têm todos seus elementos dentro de um certo raio *R* na média. Em suas simulações, ele usou um valor de 100m para *R*.

Das abordagens apresentadas aqui, essa é provavelmente a que melhor modela o agrupamento de pontos de localização a fim de definir locais visitados. Mas mesmo ela apresenta suas falhas. As maiores críticas quanto ao uso de algoritmos de *clustering* hierárquicos é a facilidade de juntar clusters que não deveriam ser um só, principalmente usando a estratégia *single-linkage*. Clusters claramente distintos, mas que possuem uma pequena intersecção são agrupados usando essa técnica. Isso é intuitivamente ruim e ainda pode resultar em possíveis clusters alongados, modelo que não reflete muito bem a realidade de residências.

2.4. DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) [11] é um dos algoritmos usados em *Poulston et al.* [8] para agrupar pontos de localização. O DBSCAN forma clusters baseado em regiões de alta densidade. Dada uma distância máxima de vizinhança ε e um parâmetro de densidade *minPts*, os pontos são classificados da seguinte maneira:

- *core* – pontos que tem pelo menos *minPts* vizinhos em um raio ε;
- *reachable* – pontos que não são *core*, mas são vizinhos de pontos *core*;
- *outliers* – pontos que não são nem *core* nem vizinhos de *core*.

Dada a classificação dos pontos, definimos um cluster como sendo todos os pontos *core* vizinhos entre si mais os pontos *reachables* vizinhos deles.

Este algoritmo é muito conhecido por ser capaz de identificar clusters não convexos. Mas no nosso caso, isso não é uma grande vantagem; clusters convexos resolvem. A fraqueza desse algoritmo para a nossa aplicação é ter que depender do parâmetro de densidade *minPts*. Escolher erroneamente esse parâmetro pode fazer locais serem classificados como *outliers*, se *minPts* for alto demais, ou pode agrupar clusters indevidamente, caso *minPts* seja baixo demais (assim como o algoritmo hierárquico de *single-linkage*). Além disso, o parâmetro ótimo para um usuário pode não ser tão bom para outro. Ele pode depender da quantidade de tempo de dados coletados para o usuário e da frequência com que são coletadas as requisições de localização, que por sua vez pode variar com a frequência de uso do smartphone e das permissões de coleta de dados. Como é difícil achar esse *fine-tuning*, e como ele é diferente para cada usuário, DBSCAN também talvez não seja a melhor escolha para nossa aplicação.

2.5. *K-means* e *Expectation–maximization*

Poulston et al. [8] também usa os algoritmos de *clustering K-means* [12] e *Expectation–maximization* [13] para agrupar pontos de localização. O *K-means* funciona da seguinte maneira: dado um número *K* de clusters desejados, centroides de tais clusters são inicializados de forma aleatória tomando *K* pontos do conjunto de localizações. A partir daí se atribui a cada ponto o cluster com o centroide mais próximo. Depois os centroides são recalculados como a média entre os pontos do cluster. Esses dois últimos passos se repetem até a convergência (até que a diferença entre novos e antigos centroides seja menor que um limiar).

Já o algoritmo de *Expectation-maximization* pode ser visto como uma versão *soft-clustering* do *K-means*: dados *K* componentes gaussianos iniciais, duas etapas se repetem até a convergência. A primeira etapa, *Expectativa*, consiste em avaliar a verossimilhança dos parâmetros estimados, enquanto a segunda, *Maximização*, procura maximizar a verossimilhança do passo anterior.

Apesar desses algoritmos conseguirem minimizar a variância de cada cluster, dependem fortemente de uma boa escolha do valor de *K*. Como o número de locais visitados de cada usuário não é conhecido a princípio e variam de usuário para usuário, esses algoritmos não se tornam muito vantajosos para nossa aplicação.

3 *Maximal Density Clustering Algorithm*

Como visto na seção anterior, todos os métodos de *clustering* usados na literatura para agrupar pontos de localização dependem de parâmetros que não dispomos a priori, como o número de clusters desejados ou a densidade mínima de cada um. Assim, inspirados por conceitos do DBSCAN e do EM, criamos um novo algoritmo que, dado alguns pressupostos, depende apenas da acurácia do equipamento de medição de GPS. Nós o chamaremos de *Maximal Density Clustering Algorithm*.

3.1 Visão Geral

O *Maximal Density Clustering Algorithm* (MDCA) é um algoritmo de *clustering* baseado em centros e densidade ao mesmo tempo. Ele é dividido em 3 etapas:

1. Cálculo dos *scores* de densidade;
2. Escolha dos pontos de máximo *score* locais como centroides;
3. Atribuição de cada ponto ao cluster de máxima verossimilhança.

```
function MaximalDensityClustering(Dataset, R)
      //inicialização
      for each x in Dataset do:
            score[x] = 0
            isCentroid[x] = True
            clusterOf[x] = null

      //cálculo dos scores de densidade
      for each x in Dataset do:
            for each y in Dataset do:
                  score[x] += normalDist(mean=y, stdDev=R)(x)

      //escolha dos pontos de máximo score locais como centroides
      for each x in Dataset do:
            for each y in getNeighbors(Dataset, x, R) do:
                  if score[y] > score[x]:
                        isCentroid[x] = False
                        break
                  if score[y] == score[x] and isCentroid[y] == True:
                        isCentroid[x] = False
                        break

      Centroids = [x in Dataset where isCentroid[x] == True]

      //atribuição de cada ponto ao cluster de máxima verossimilhança
      for each x in Dataset do:
            clusterOf[x] = c in Centroids that maximizes:
                  score[c] * normalDist(mean=c, stdDev=R)(x)

      return Centroids, clusterOf, score
```

Algoritmo 1 MDCA: *Maximal Density Clustering Algorithm*

Esse algoritmo recebe como entrada um parâmetro *R* que é para ser o desvio padrão do erro do instrumento de GPS usado, também chamado de acurácia ou precisão.

Cálculo dos *scores* de densidade – Nessa etapa são calculados os *scores* de densidade de cada ponto. Para cada ponto *P* do *Dataset*, seu *score* de densidade é definido como a soma das distribuições normais centradas em cada ponto *Q* do *Dataset*, de desvio padrão igual a *R*, avaliadas em *P*.

Escolha dos pontos de máximo *score* locais como centroides – Dado que os scores de densidade já estão calculados, é definido como o centroide de cluster todo ponto que tem o *score* de densidade maior do que o de todos os seus vizinhos em *R* (pontos a uma distância *R*). Caso dois ou mais vizinhos compartilhem o mesmo score máximo (algo bastante improvável), um deles é escolhido aleatoriamente, de forma que não existam dois centroides vizinhos.

Atribuição de cada ponto ao cluster de máxima verossimilhança – Agora que os centroides estão definidos, basta associar cada ponto ao seu cluster. Essa associação é feita escolhendo o cluster de máxima verossimilhança. A verossimilhança entre um ponto *P* e um cluster de centroide *C* é proporcional ao *score* de *C* multiplicado pela componente gaussiana centrada em *C*, com desvio padrão *R*, avaliada em *P*.

3.2 Porque ele funciona?

Esse algoritmo funciona quando se leva em consideração as seguintes modelagens do mundo real, que nem sempre são verdadeiras, mas que se adequam bem ao nosso propósito de achar a residência do usuário.

1. Como colocado na página do *Android*, podemos modelar a função de distribuição de probabilidade da localização real do usuário, dada uma medição de GPS, como uma distribuição normal centrada na medição e de desvio padrão igual à precisão do instrumento.
2. Simetricamente, podemos modelar a função de distribuição de probabilidade das medições de GPS do usuário, dada sua localização real, como uma distribuição normal centrada na posição real e de desvio padrão igual à precisão do instrumento.
3. Assumindo que a área de imprecisão de um instrumento de GPS pode ser de uma ordem de grandeza maior que a área de uma residência, podemos modelar um local visitado como um ponto adimensional.

A terceira suposição pode parecer irreal para alguns, mas a precisão de um instrumento de GPS comum de smartphone geralmente é de dezenas, podendo chegar a centenas de metros.

Uma acurácia de 30m gera uma área circular de mais de 2700m^2, ordens de grandeza acima da média do tamanho das residências de hoje em dia, não passa dos 80m^2.

Isso claramente não é válido para locais como parques, shoppings e aeroportos, então o algoritmo provavelmente não irá se comportar corretamente nesses lugares. Mas como a nossa aplicação está mais interessada em residências, podemos desconsiderar esses erros.

Dadas as considerações acima, formaremos o seguinte raciocínio: esteja um usuário parado em um local (que estamos considerando como um ponto adimensional), e seu aparelho fizer várias medições de GPS, é esperado que essas medições sejam distribuídas normalmente ao redor desse local. Agora se somarmos as diferentes gaussianas normalizadas geradas por esses pontos de localização, é esperado que a função resultante, quando normalizada, seja semelhante à convolução entre gaussianas, que também é uma gaussiana.

Assim, é esperado que a soma das gaussianas de todos os pontos de localização, gerados a partir de diversos locais diferentes visitados pelo usuário, se assemelhe a soma de gaussianas centradas nos locais em si. A partir daí, estimar onde ficam os locais fica fácil: basta achar os máximos locais dessa resultante.

Achar esses máximos locais de forma analítica pode não ser uma tarefa tão fácil. Pensando nisso, decidimos avaliar a função resultante em um conjunto finito de pontos a fim de encontrá-los. Escolhemos então, por comodidade, os próprios pontos de localização que foram dados como entrada do algoritmo como os pontos onde a função será avaliada.

3.3 Complexidade de tempo

A complexidade de tempo do algoritmo como foi descrito é, devido à etapa de cálculo de *scores* de densidade, $O(n^2)$. Mas ela pode melhorar bastante se considerarmos que a função gaussiana é desprezível depois de 3 desvios padrões. Assim podemos limitar a avaliação da função de soma apenas para pontos distantes no máximo 3 desvios padrões do ponto em questão. Somando a isso o uso correto de um índice espacial, como uma Quad-Tree, essa complexidade de tempo pode chegar a $O(n log n)$.

3.4 Melhorando o algoritmo e EM

Dado que os centroides escolhidos pelo algoritmo podem não ser ótimos, já que eles são obrigados a pertencer ao conjunto de pontos iniciais, podemos melhorar o algoritmo fazendo o seu resultado passar por iterações do algoritmo de *Expectation-maximization*. Essa

operação garantirá centroides mais confiáveis. No fim das contas, o MDCA pode ser usado como um bom inicializador de componentes gaussianas do EM.

4 Experimentos

Nesta sessão testamos o MDCA em conjunto com a estratégia de número máximo de visitas à noite para descobrir o quão eficiente é essa abordagem para resolver o problema de inferência de residência.

4.1 A base de dados

Durante a formulação deste trabalho, tive acesso a uma base de dados de 6202 usuários anônimos, contendo o histórico de localizações dos mesmos e o endereço de onde alegam morar. Assumimos, para os propósitos deste experimento, que todos eles foram verdadeiros nessa alegação.

4.2 Descrição do experimento

Usamos o Google Geocoder para converter os endereços fornecidos pelos usuários em pontos de localização. Comparamos esses pontos, para cada usuário, com a localização do centroide do cluster resultante do MDCA com mais visitas no período das 21h e 7h. Se a distância entre eles for menor que 100 metros, dizemos que acertamos a inferência.

4.3 Primeiros resultados

Após rodar o experimento, nosso algoritmo só acertou a residência de 65% dos usuários. Esse valor nos pareceu muito abaixo do esperado, então fomos investigar a origem de tantos erros. Após uma análise profunda, descobrimos que a maior parte das falhas se dá na etapa de conversão do endereço para a geolocalização (*geocoding*).

4.4 Análise dos erros de *geocoding*

Ao analisar os dados mais profundamente, descobrimos que parte dos erros é devido a erros dos *geocoders*, e outra parte é devido ao falho sistema de endereçamentos brasileiro. Fizemos testes também com outros provedores de *geocoding*, além o Google, mas todos os testados falham nos mesmos quesitos.

Erros dos *geocoders*:

- Não acharam localização para 20% dos endereços enviados,
- Às vezes não acha a rua devido a abreviações ou erros de escrita. Quando isso acontece o sistema pode retornar um ponto aleatório no bairro ou na cidade.
- Muitas vezes confundem ruas com nomes parecidos.
- Quando acham a rua, geralmente erram o número. Esse erro pode, muitas vezes, ser maior que 500 metros.

Erros do sistema de endereçamento:

- Muitas favelas não têm serviço de endereçamento. Então alguns usuários colocam como endereço no sistema o endereço da entrada da favela, que pode ser bem distante de sua casa.
- Uma situação similar ocorre com condomínios. O *geocoder* retorna a geolocalização da entrada do mesmo, essa podendo ser distante da casa do usuário.
- Em regiões mais pobres ruas podem não ter nomes e casas podem não ter números.
- Em muitos casos também foram encontradas numerações na rua feita de forma quase aleatória: não seguiam uma ordem crescente ou decrescente, podendo até ter casas com o mesmo número, na mesma rua, em locais não vizinhos!

4.5 Novo base de *ground truth*

Devido a todos os problemas que tivemos com os *geocoders*, decidimos, a fim de analisar a corretude de nosso algoritmo, refazer o experimento inicial apenas com uma base de dados na qual podíamos confirmar a geolocalização verdadeira da casa do usuário. Para isso fizemos verificações de forma manual, com o auxílio do Google Street View, de 200 endereços.

Muitos endereços no meio do caminho nós não fomos capazes de verificar, ou porque o carro do Google nunca passou lá, ou porque não tínhamos como descobrir ou inferir o número da casa. Só paramos de trabalhar nessa verificação quando alcançamos os 200 endereços. Não contamos quantos endereços não conseguimos verificar até atingir tal meta.

4.6 Segunda rodada de resultados

Com essa nova base de endereços, nossos resultados foram muito melhores: subiram para 94%, algo semelhante, e até um pouco superior, ao encontrado na literatura. *Gu et al.* [4] obteve 92% de assertividade rodando o algoritmo dele sobre apenas os usuários que possuem dados de *checkin*. Mas como a base que usamos para testar o nosso método é muito pequena (200 amostras), esse resultado de 94% pode ser muito volátil.

5 Conclusões e Trabalhos Futuros

Apresentamos nesse trabalho um método para inferir a localização da residência de um usuário quando se tem seu histórico de localizações. Este método conta com um algoritmo de *clustering* próprio, o *Maximal Density Clustering Algorithm* (MDCA), criado para este fim. O método como um todo teve um bom resultado na identificação das residências: inferiu corretamente 94% de 200 endereços verificados.

Em seguida eu apresento ideias de possíveis melhoramentos no MDCA, possível material para trabalhos futuros.

5.1 Considerar diferentes acurácias de medida

O MDCA considera uma acurácia única do equipamento de medição de GPS na hora de fazer seus cálculos. Mas sabemos que a acurácia não é uma variável atrelada ao instrumento em si, mas à medição. Cada medição pode ter uma acurácia diferente.

Fatores que influenciam na acurácia incluem a quantidade de satélites disponíveis na hora de cada requisição. Outros instrumentos que auxiliam na localização, como torres de telefonia e roteadores WiFi, também tem grande influência sobre a precisão. Pretendemos em trabalhos futuros remodelar a solução para levar em consideração também as diferentes acurácias das medições.

5.2 Considerar locais como corpos extensos

Outra consideração do algoritmo é tratar os locais como pontos sem dimensão. Isso funciona muito bem quando estamos falando de casas e escritórios, mas quando usamos o mesmo método para fazer agrupamentos em shoppings, parques ou aeroportos, os resultados já não são tão bons. Pretendemos criar uma modelagem para tais casos no futuro.

5.3 Considerar horários das visitas na hora de atribuir pontos a locais

A consideração mais importante é, no entanto, levar em consideração o *timestamp* das visitas dos usuários nos locais na hora do agrupamento. Podemos considerar que as pessoas geralmente seguem rotinas, como sair de manhã para trabalhar, ter uma pausa para o almoço,

voltar a trabalhar e então de retornar ao lar. Isso nos dá valiosas informações importantes na hora de decidir à que cluster pertence um ponto de localização.

Veja o exemplo: para uma pessoa que almoça perto do trabalho, pontos de localização que ficam na fronteira entre as nuvens de pontos de um local e outro vão poder ser classificados com maior assertividade se a hora da visita for considerada. Podemos assumir que um ponto que esteja no meio dos dois é mais provável que pertença a um restaurante se tiver sido coletado ao meio dia e mais provável de ser do trabalho se coletado às 3 da tarde.

6 Bibliografia

[1] Cho, Eunjoon, Seth A. Myers, and Jure Leskovec. "Friendship and mobility: user movement in location-based social networks." *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011.

[2] Scellato, Salvatore, et al. "Socio-spatial properties of online location-based social networks." *ICWSM* 11 (2011): 329-336.

[3] Hu, Tian-ran, et al. "Home location inference from sparse and noisy data: models and applications." *Frontiers of Information Technology & Electronic Engineering* 17.5 (2016): 389-402.

[4] Gu, Yulong, et al. "We know where you are: Home location identification in location-based social networks." *Computer Communication and Networks (ICCCN), 2016 25th International Conference on*. IEEE, 2016.

[5] Liu, Hao, et al. "Mining checkins from location-sharing services for client-independent ip geolocation." *INFOCOM, 2014 Proceedings IEEE* (2014): 619-627.

[6] Krumm, John, and Dany Rouhana. "Placer: semantic place labels from diary data." *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*. ACM, 2013.

[7] Hoh, Baik, et al. "Enhancing security and privacy in traffic-monitoring systems." *IEEE Pervasive Computing* 5.4 (2006): 38-46.

[8] Poulston, Adam, Mark Stevenson, and Kalina Bontcheva. "Hyperlocal Home Location Identification of Twitter Profiles." *Proceedings of the 28th ACM Conference on Hypertext and Social Media*. ACM, 2017.

[9] William B Frakes, Ricardo Baeza-Yates, *Information retrieval: data structures and algorithms*, 1992.

[10] S. C. Johnson. Hierarchical Clustering Schemes. PSYCHOMETRIKA, vol.32, no.3, pp.241-254, 1967.

[11] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, and others. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In Kdd, Vol. 96. 226–231.

[12] James MacQueen and others. 1967. Some methods for classification and analysis of multivariate observations. In Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Vol. 1. Oakland, CA, USA., 281–297.

[13] Arthur P Dempster, Nan M Laird, and Donald B Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. Journal of the royal statistical society. Series B (methodological) (1977), 1–38.