Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

Predição de categorias de lugares baseado em dados geolocalizados

Trabalho de Graduação

Felipe Henrique de Almeida Bormann

Recife

Dezembro de 2018

Universidade Federal de Pernambuco

Centro de Informática

Graduação em Ciência da Computação

Predição de categorias de lugares baseado em dados geolocalizados

Trabalho apresentado ao Programa de Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Aluno: Felipe Henrique de Almeida Bormann

Orientador: Cleber Zanchettin

Recife, dezembro de 2018

"Forecasting is the art of saying what will happen, and then explaining why it didn't"
Anônimo

Agradecimentos

Gostaria, primeiramente, de agradecer a todos que fazem parte do Centro de Informática da Universidade Federal de Pernambuco, professores, funcionários e alunos.

Agradeço ao professor Cleber Zanchettin, pela orientação neste trabalho e pelas discussões de ideias e problemas que encontrei durante o desenvolvimento, sempre disponível e claro nas dúvidas.

Agradeço a In Loco, por permitir o uso do conjunto de dados nos experimentos deste trabalho, e pelo grande desenvolvimento profissional que me proporcionaram.

Agradeço à minha irmã e cunhado, por serem meus alicerces nessa grande busca por conhecimento e privação de noites e noites sem dormir, além de suportarem meu gênio difícil.

Agradeço à minha mãe e pai, por proverem o necessário para sobreviver e me dar tempo e espaço suficiente para que o crescimento como pessoa pudesse ocorrer.

Agradeço à todos os colegas de trabalho na Universidade, que, de uma forma ou de outra me ensinaram algo sobre esta enorme área da computação.

Resumo

Geolocalização é a forma de referenciar um ponto no espaço da terra utilizando coordenadas geográficas. Existem diversas formas de localização, sendo a mais famosa o GPS (*Global Positioning System*), onde são utilizados pelo menos 3 satélites para definir a posição de um objeto na terra. Este tipo de localização é eficiente para objetos em ambientes *outdoor*, ou seja, ambientes externos como campos, praças, ruas e avenidas. Contudo, detectar que a localização de um objeto que está no lado direito de uma sala ou mesmo dentro dela, neste caso, um ambiente interno ou *indoor,* é um problema bem mais complexo no qual a tecnologia GPS tem dificuldade em tratar.

A In Loco, sediada em Recife - PE, é uma empresa especializada em dados de Geolocalização e que possui capacidade para definir a localização *indoor* de um objeto com uma margem de erro de menos de 2 metros. Esta tecnologia permite, por exemplo, detectar que um usuário está dentro de uma "loja" em um shopping. Ou seja, é possível verificar que um cliente "visitou" a loja com base em sua Geolocalização.

Neste trabalho, a partir de uma base de dados de Geolocalização *indoor* disponibilizada pela empresa In Loco, foram construídos diferentes modelos de inferência para prever o comportamento de visita e deslocamento dos usuários. O conceito de visita utilizado associa uma Geolocalização, em determinado instante de tempo, de um usuário à um local físico que possui uma categoria associada, como por exemplo: *gym, beauty_saloon, car_repair* ou *restaurant*.

Os modelos analisados foram LSTM (*Long Short Term Memory*), MLP (*Multi Layer Perceptron*) e RNN (*Recurrent Neural Network*). Os resultados obtidos mostram que a LSTM obteve melhores resultados comparado aos outros modelos quando treinada para cada usuário individualmente. Contudo, quando os modelos foram treinados em toda a base, a performance máxima obtida pela melhor configuração dos modelos foram equivalentes entre si.

Palavras-chave: Predição de categorias, geolocalização, visita, séries temporais.

Abstract

GeoLocation is the way that we refer to a point in space on earth based on geographic coordinates. There are many ways of localization, the most famous being GPS (*Global Positioning System*), where at least 3 satellites are used to define the position of an object on Earth.

This kind of localization is efficient for objects in outdoors environments such as parks,, streets or avenues. But to that an object is on the right corner side of a room or even inside of it, in a indoor environment, is a much more complex problem that the GPS isn't the proper technology to be used.

The In Loco, a company based in Recife, is a company specialized in geolocation data and that possess the capability to pinpoint an indoor location within a margin error of 2 meters. This technology allows, per example, to detect that an user is inside a store in a shopping mall. Which means that is possible to "verify" that a user visited a store based on its geolocation.

In this work, from a indoor geolocation dataset provided by In Loco, different machine learning models were built to predict the behavior of visits and locomotion of users. The concept of visit used is when a geolocation of an user is associated, in a given timestamp, to physical place that possess a category associated with it, such as: gym, beauty_saloon, car_repair or restaurant.

The models analyzed were LSTM (Long Short Term Memory), MLP (Multi Layer Perceptron), e RNN (Recurrent Neural Network). The results obtained were that the LSTM model achieved better results compared to the other models when trained for each user individually. Although, when the models where trained against the whole database, the maximum performance achieve by their best configuration was equivalent.

Keywords: Category Prediction, geolocation, visit, time series.

Sumário

Capítulo 1 **8**
Introdução 8

Capítulo 2 **10**
Fundamentos 10
2.1 Séries Temporais 10
2.3 Redes Neurais Artificiais 10
2.3.2 RNN (Recurrent Neural Networks) 10
2.3.1 LSTM (Long Term Short Memory) 12
2.4 Predição de categorias em Séries Temporais 15
2.4.1 Definição do Problema 15
2.5 Trabalhos relacionados 15

Capítulo 3 **17**
Materiais e Métodos 17
3.1 Conjunto de dados 17
3.2 Processamento dos dados 19
3.3 Anonimização dos dados 22
3.4 Pré-processamento dos dados 22
3.5 Técnicas Investigadas 24
3.6 Metodologia 25
3.8 Hardware Utilizado 27

Capítulo 4 **28**
Experimentos 28
4.1 Treinamento e avaliação 28
4.2 Resultados 31
4.2.1 Variação na acurácia a partir da modificação na quantidade de elementos no vetor de entrada 31

Capítulo 5 **37**
Conclusão 37

Referências **39**

Anexo A - Listagem de Categorias **41**

Anexo B - Pacotes Instalados **44**

Capítulo 1

Introdução

O comportamento de usuários de tecnologia hoje em dia é capaz de gerar um número gigantesco de informações variadas, como qual aplicativo ele está utilizando, em qual hora está utilizando, ou para que ele está utilizando aquele aplicativo, desta forma o comportamento do usuário se torna uma rica fonte de dados sobre ele mesmo. Isto gerou e continua gerando novos mercados, como o de propaganda, que ao aprender sobre os hábitos e preferências do usuário faz com que empresas como a Google[1], Facebook[2] e In Loco[3] sejam capazes de servir propagandas à estes usuários de forma que seja relevante para ele e que de fato modifique seu comportamento. Por exemplo, a In Loco é capaz de ao detectar que um usuário está dentro da loja de um concorrente de um cliente, enviar uma propaganda de promoção para este usuário de forma que a venda que ele iria fazer no concorrente seja convertida para a loja do seu cliente.

Um dos fatores de complexidade deste problema é ser capaz de entregar uma propaganda que seja, de fato, relevante para o usuário e que se converta em uma visita ou click. O conceito de visita utilizado neste trabalho é quando um usuário passa tempo suficiente em um local que se possa assumir que ele realmente visitou o local.

A visita é gerada através do algoritmo proprietário de localização da In Loco, e eles são armazenados na cloud da Amazon, no serviço de armazenamento chamado S3, o formato utilizado para armazenar os dados hoje em dia é o Apache Parquet (Apache Parquet, 2018), um formato de armazenamento colunar, multiplataforma e binário.

O objetivo deste trabalho é explorar os dados geolocalizados da In Loco de forma a gerar modelos que sejam capazes de predizer a probabilidade de um usuário visitar determinado local. Para isto, foram estudados modelos baseados em redes neurais artificiais, principalmente o modelo LSTM (*Long Short-Term Memory*), para predizer qual a categoria de lugar visitado pelo o usuário.

[1] "Google." https://www.google.com.br/. Acessado em 12 dez. 2018.
[2] "Facebook." https://pt-br.facebook.com/. Acessado em 12 dez. 2018.
[3] "In Loco." https://inloco.com.br/. Acessado em 12 dez. 2018.

Este tipo de classificação é difícil pois o contexto do usuário influencia muito, apesar de o usuário ter feito o mesmo trajeto de visitas, o classificador deve levar em consideração outros fatores que são importantes e nem sempre estão associados aos eventos de visita por si só, como qual o dia da semana que está acontecendo a visita, se o usuário visita o estabelecimento somente uma vez por mês. Esses padrões latentes são difíceis de capturar pois eles possuem dependências muito distantes entre si dado que os usuários possuem rotinas semanais bem comuns mas com algumas variações entre si (Jing et al., 2018).

O documento está estruturado da seguinte maneira: O segundo capítulo introduz os fundamentos para o trabalho, como a definição do que é uma série temporal, redes neurais artificiais, além dos trabalhos relacionados. No terceiro capítulo, o desenvolvimento do projeto é elucidado, apresentando com detalhes o conjunto de dados, o processamento deles, os cuidados com a anonimização e a limpeza. No quarto capítulo, os experimentos são apresentados e por último, no quinto capítulo as conclusões e resultados são listadas, comparando os modelos gerados e os resultados alcançados.

Capítulo 2

Fundamentos

Neste capítulo serão apresentados importantes fundamentos para o entendimento do trabalho. Inicialmente, explicarei o conceito de Série Temporais e Redes Neurais Artificiais. Por último, definirei o problema de predição de categorias em Aprendizagem de Máquina.

2.1 Séries Temporais

Séries temporais são uma série de dados indexados pela ordem temporal (Li Youru et. al, 2018). Ou seja, todo evento destes dados são associados a um ponto no tempo, como por exemplo: " joão comeu sushi no restaurante GoTemaki no dia 23/10/2018 às 19h30". Com a informação, "joão comeu sushi no restaurante GoTemaki", associada à um ponto no tempo, "dia 23/10/2018 às 19h30". Um conjunto sucessivo desses eventos separados por um intervalo de tempo variável é o que chamamos de série temporal.

2.3 Redes Neurais Artificiais

Redes Neurais são processadores distribuídos paralelamente feitos de pequenas unidades simples, que possuem uma tendência natural de armazenar experiências e torná-las disponíveis para uso. Elas relembram o cérebro de duas maneiras:

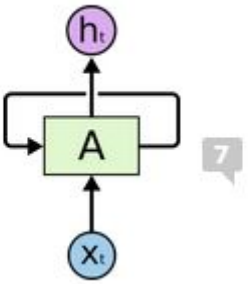
1. O conhecimento é adquirido pela rede a partir do seu ambiente através de um processo de aprendizagem.
2. Pesos das conexões entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar os conhecimentos adquiridos. (Haykin, Simon)

2.3.2 RNN (*Recurrent Neural Networks*)

Redes Neurais recorrentes foram propostas para resolver o problema de persistência de informação na rede, ou seja, que ela seja capaz de utilizar informações previamente vistas

para aprender sobre algo novo. Elas conseguem isso através da adição de um loop, como demonstrado na Figura 1.

Figura 1. Neurônio de uma RNN



Fonte: Understanding LSTM Networks, 2018

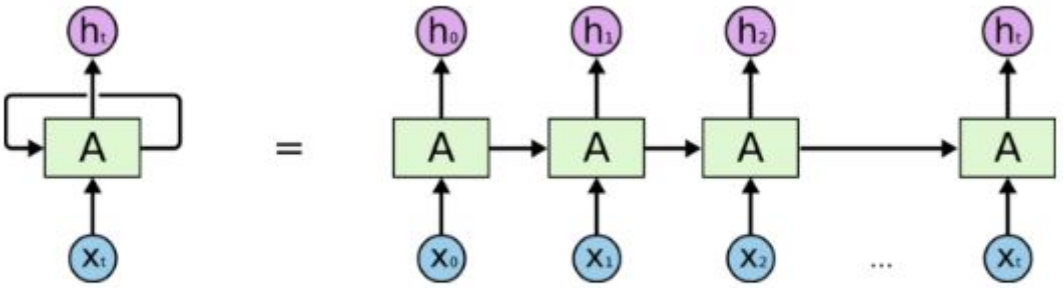
No diagrama acima, A é uma célula da rede neural, Xt é uma entrada e Ht a saída do neurônio A. O que o loop encontrado no neurônio adiciona à fórmula de predição de uma RNN é a capacidade de levar o que já foi aprendido por A com outros inputs para a classificação do output Ht. Todo neurônio possui um vetor h, que é atualizado toda vez que um novo input é transformado.

Em relação à aprendizagem uma RNN não é nada mais do que um conjunto de matrizes de transformações que é aplicado à um vetor de entrada e retorna um vetor de saída.

Para aprender sobre os dados, todo neurônio de uma rede RNN possui um estado interno que é atualizado a cada input fornecido ao modelo, isto só é permitido graças a autoconexão existentes em cada célula da RNN.

Uma maneira fácil de entender a capacidade de uma RNN é pensar como se ela fosse uma sequência de redes neurais, onde cada uma passa o output para a próxima, e os pesos de das redes sejam os estados após cada input de uma sequência, por exemplo, uma sequência das categorias visitadas por um usuário:

Figura 2. RNN visualizada como uma sequência de redes neurais



An unrolled recurrent neural network.

Em relação ao algoritmo de aprendizagem, geralmente chamado de *BackPropagation Through Time* (BPTT), este algoritmo é uma aplicação do algoritmo BackPropagation através da camada RNN para cada input recebido, é fácil entender sua aplicação quando observamos uma RNN como uma sequência de redes dependentes entre si como na Figura 2, e que o erro é propagado desde a última rede até a primeira.

O problema de utilizar esta técnica em várias camadas ou em inputs muito grandes é o chamado *Vanishing Gradient Problem,* ele ocorre pois o impacto do erro do último input recebido no vetor de entrada vai perdendo força com a quantidade de aplicações de gradientes consecutivas.

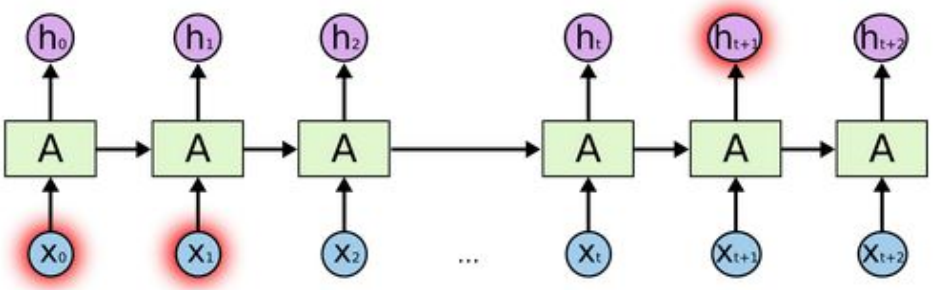
2.3.1 LSTM (*Long Term Short Memory*)

Um dos pontos de interesse deste trabalho foi explorar o uso de um tipo de rede neural artificial chamada LSTM. Esta rede foi escolhida especialmente pela sua capacidade de aprender padrões temporais melhor do que outros modelos de Aprendizagem de Máquina convencionais (Schmidhuber et al, 2005). Apesar de ser um modelo antigo, foi concebido em 1997, ainda é muito utilizado, o que torna uma adição interessante ao projeto.

A LSTM é uma RNN que possui algumas peculiaridades no seu neurônio que uma RNN comum não possui de forma que ela seja capaz de resolver um problema que a RNN comum não consegue: Utilizar informações “distantes”, ou pouco dependentes entre si, no tempo, para predizer o valor atual, isto acontece por conta do *Vanishing Gradient Problem* mencionado na seção anterior.

O diagrama apresentado na Figura 2 exemplifica o problema.

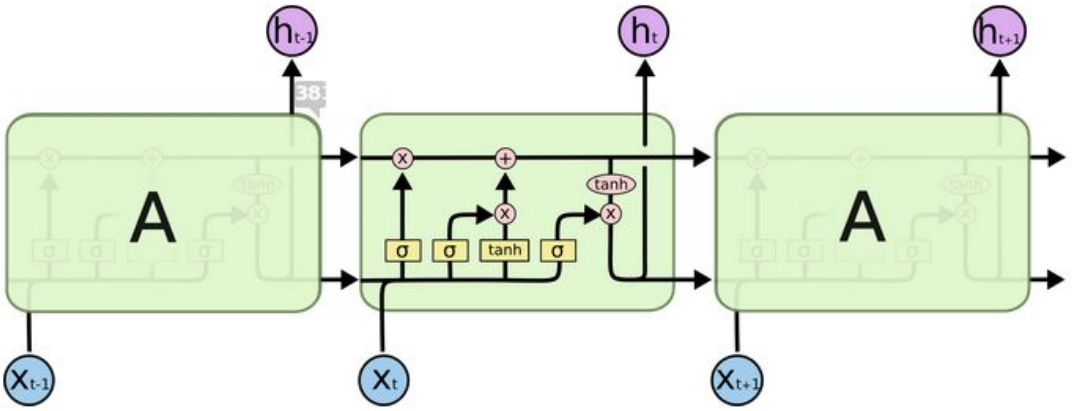
Figura 2. Exemplificando o problema de dependências distantes



Fonte: Understanding LSTM Networks, 2018

Apesar das RNNs serem capazes de conseguir utilizar informações previamentes vistas para predizer uma informação futura, ela não é capaz de detectar dependências que sejam distantes entre si. A LSTM resolve este problema modificando a estrutura do neurônio da seguinte forma, ao invés de possuir apenas um loop com uma função de ativação, o neurônio de uma LSTM possui outros caminhos de ativação, chamados “portões”. Todo neurônio numa LSTM possui um “estado”, este estado está suscetível a diferentes transformações que são ativadas ou não pelos portões, onde um portão pode estar completamente ou parcialmente aberto, ou seja, com valor 1 ou próximo dele, ou fechado, igual à 0. Um gráfico representando o modelo é apresentado na Figura 3.

Figura 3. Arquitetura de um neurônio em uma LSTM



Fonte: Understanding LSTM Networks, 2018

Neste modelo, o primeiro passo é decidir qual informação nós iremos remover do estado da célula, essa decisão é tomada pelo "portão do esquecimento". Ela olha para o valor de Ht-1 e Xt e expele uma saída entre 0 e 1 onde 1 representa "manter o valor atual completamente" enquanto 0 representa "não utilize nada do valor".

O próximo passo é definir qual informação nova o modelo vai guardar no estado da célula. Este passo é dividido em duas partes:

1. Primeiro uma camada sigmoid chamada "portão da camada de entrada" decide quais valores serão atualizados
2. Em seguida, uma camada com a função tanh cria um vetor de novos valores candidatos que podem ser adicionados ao estado.

Basicamente, um estado Ct-1 é multiplicado pelo portão da camada de entrada e em seguida os novos candidatos são multiplicados por valores que escala a importância de cada um deles para formar o novo estado Ct.

O último passo é decidir a saída da célula LSTM, a saída é baseada no estado da célula, primeiro, o estado passa por uma camada com a função sigmoid que decide quais partes do estado da célula serão expelidos. Após isto, o estado da célula é transformado por uma função tanh para que os valores fiquem entre -1 e 1 e é multiplicado pela saída da camada sigmoid, de forma que só as partes escolhidas pela camada sigmoid serão parte da saída.

2.4 Predição de categorias em Séries Temporais

2.4.1 Definição do Problema

Dado um dataset de eventos de visita, onde cada evento v representa a associação de uma localização com um ou mais lugares Xi que possuem um conjunto de categorias y com um certo nível de confiança C, onde C pode ser qualquer valor dentro do conjunto composto pelos valores LOW, MEDIUM e HIGH. Onde todo evento v é associado à um ponto T no tempo, ou seja, todo usuário possui uma lista consecutivas de eventos de visita Vi ao longo do tempo, o que compõe uma série temporal.

Com isto, o problema se constitui em ser capaz de, dado um histórico de visitas e os lugares associados com suas respectivas categorias, predizer qual a próxima categoria será visitada por um usuário.

2.5 Trabalhos relacionados

Existem poucos trabalhos relacionados à predição de categorias de lugares visitados, em grande parte deles, o dataset utilizado é público, como em (Anastasios et al, 2011)., o que torna difícil comparar a performance do modelo dado que as bases possuem natureza e qualidades diferentes.

Mas relacionado ao problema de predição de valores categóricos em séries temporais, já existem alguns trabalhos que são úteis, como, por exemplo, em Chen et al., (2016). Neste trabalho foi possível predizer o intervalo entre eventos de localização na rede social Foursquare (Foursquare, 2018), o que é uma grande ajuda para entender sobre a natureza dos usuários que ali estão.

Um trabalho que possui uma descrição do problema bem parecida com a do meu trabalho é o de (He et. al, 2018). O objetivo do artigo é a recomendação de próximo lugar de interesse para o usuário, uma das partes que utilizei desse trabalho foi a remoção de usuários com uma quantidade mínima de visitas, pois a degradação na performance do modelo de previsão a partir de um certo valor é muito alta. Os autores chegam a conclusão de que o padrão da população em geral é muito parecida, mas todos possuem transições latentes específicas que podem ser aprendidas com um modelo de base geral que seja adaptado aos

usuários especificamente, de forma a capturar essas variantes da grande rotina semelhante que a maioria possui.

Em busca de soluções que fossem específicas para o problema, encontrei uma arquitetura de rede neural chamada LSTM, o artigo original dela foi que serviu de base para o entendimento. (Greff et al., 2017) explicam muito bem como o modelo LSTM progrediu ao longo do tempo, contudo, sem grandes avanços para datasets em geral, apenas alguns datasets específicos. Com *fine tuning* de vários parâmetros do modelo foi possível perceber diferença entre as diversas versões de LSTM (Greff et al., 2017)

Capítulo 3

Materiais e Métodos

Este capítulo explica como o conjunto de dados foi coletado, quais características dos dados são importantes para o problema, como ele foi anonimizado e quais decisões foram tomadas para pré-processar os dados e por último quais técnicas utilizadas para modelagem.

3.1 Conjunto de dados

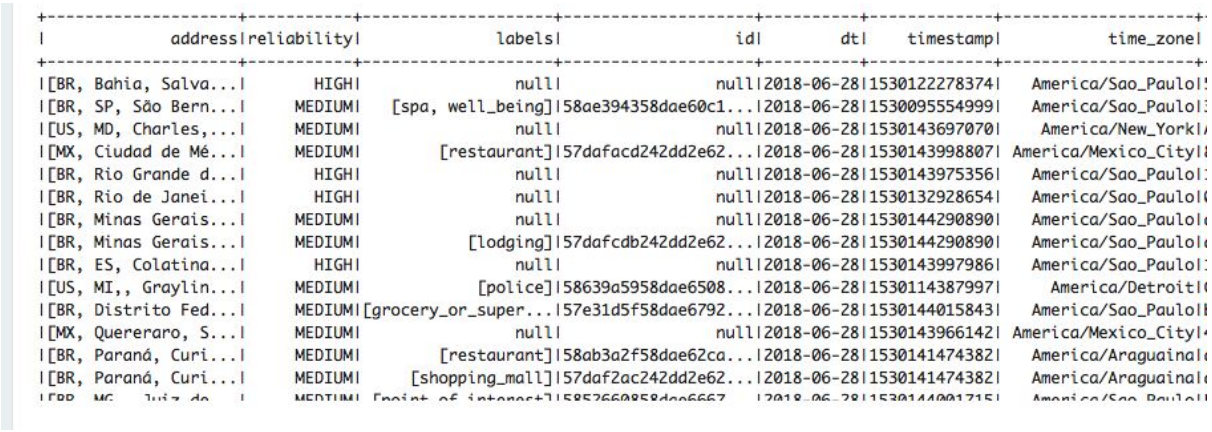
O conjunto contém informações que foram adquiridas a partir de dados de geolocalização coletados passivamente, ou seja, nenhum usuário explicitou qualquer informação a partir de um *site* ou aplicativo. Logo, nota-se que neste trabalho os dados descrevem o comportamento *offline* do usuário, como ele interage no mundo físico sem precisar se engajar em uma ação *online*.

A coleta das visitas é feita da seguinte maneira:

1. Com o SDK da In Loco instalado em algum aplicativo, a localização do usuário ativada no dispositivo móvel e a permissão concedida pelo usuário.
2. O SDK monitora todos os eventos de localização enviados pelo sistema para o app.
3. Ao receber o evento de localização, o scan de Wi-Fi e outros atributos do celular, o servidor da In Loco é capaz de determinar o evento de visita, ou seja, associar a localização do usuário com um lugar na base de dados e configurar uma “visita”.

A base de dados utilizada foi dos eventos de visitas entre Maio e Julho de 2018 de todos os dias da semana. Os dados de visitas possuem diversos campos, como mostra a figura abaixo:

Figura 4. Exemplos de dados de visita



address	reliability	labels	id	dt	timestamp	time_zone
[BR, Bahia, Salva...	HIGH	null	null	2018-06-28	1530122278374	America/Sao_Paulo
[BR, SP, São Bern...	MEDIUM	[spa, well_being]	58ae394358dae60c1...	2018-06-28	1530095554999	America/Sao_Paulo
[US, MD, Charles,...	MEDIUM	null	null	2018-06-28	1530143697070	America/New_York
[MX, Ciudad de Mé...	MEDIUM	[restaurant]	57dafacd242dd2e62...	2018-06-28	1530143998807	America/Mexico_City
[BR, Rio Grande d...	HIGH	null	null	2018-06-28	1530143975356	America/Sao_Paulo
[BR, Rio de Janei...	HIGH	null	null	2018-06-28	1530132928654	America/Sao_Paulo
[BR, Minas Gerais...	MEDIUM	null	null	2018-06-28	1530144290890	America/Sao_Paulo
[BR, Minas Gerais...	MEDIUM	[lodging]	57dafcdb242dd2e62...	2018-06-28	1530144290890	America/Sao_Paulo
[BR, ES, Colatina...	HIGH	null	null	2018-06-28	1530143997986	America/Sao_Paulo
[US, MI,, Graylin...	MEDIUM	[police]	58639a5958dae6508...	2018-06-28	1530114387997	America/Detroit
[BR, Distrito Fed...	MEDIUM	[grocery_or_super...	57e31d5f58dae6792...	2018-06-28	1530144015843	America/Sao_Paulo
[MX, Quererarо, S...	MEDIUM	null	null	2018-06-28	1530143966142	America/Mexico_City
[BR, Paraná, Curi...	MEDIUM	[restaurant]	58ab3a2f58dae62ca...	2018-06-28	1530141474382	America/Araguaina
[BR, Paraná, Curi...	MEDIUM	[shopping_mall]	57daf2ac242dd2e62...	2018-06-28	1530141474382	America/Araguaina

Fonte: O Autor

Um fator importante a destacar é que várias dessas informações foram removidas para que a privacidade do usuário fosse mantida, dado que a localização de um usuário pode indicar a o endereço de sua residência, por exemplo. Visto que podem existir várias visitas naquele local toda semana durante um período.

Existem 97 valores possíveis valores para labels (Anexo A), em relação à confiabilidade (coluna reliability), um fator importante pode ser destacado. A "confiabilidade" possui três valores possíveis: "HIGH" (alta), "MEDIUM" (média) e "LOW" (baixa). Ele determina o quão confiante o algoritmo da In Loco está de que a visita ocorreu no local indicado.

A assertividade possui um papel muito importante na utilidade daquela visita, pois indica a certeza de que esta visita ocorreu em certo local a partir das informações existentes sobre ela.

Tabela 1. Informação de assertividade nas visitas

Usuário	Assertividade	Locais
X	Alta	A
Y	Média	A, B, C
Z	Baixa	A, B, C, D, E, F

Fonte: O Autor

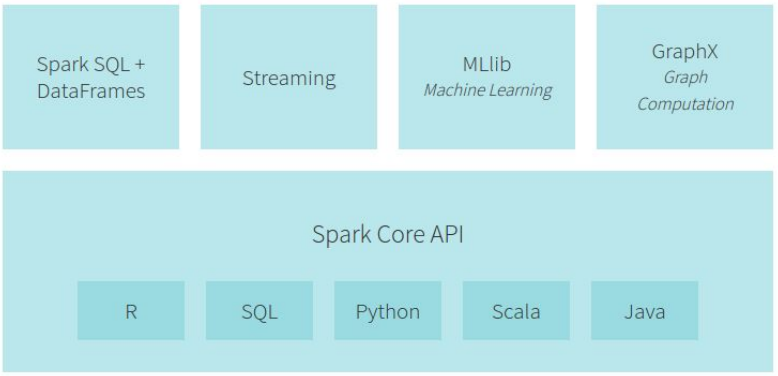
A Tabela 1 exemplifica os casos possíveis. No caso da primeira visita, o usuário X visitou o lugar A com alta assertividade, ou seja, existe uma certeza associada que é bastante

alta. Em toda visita que possui alta assertividade, existe apenas um local associado. No caso da segunda visita, Y realizou uma visita, porém existe uma grau de dúvida se ela se deu nos locais A, B ou C. A assertividade média indica que existem poucos lugares possíveis, mas a quantidade de informação não foi suficiente para garantir em qual desses lugares a visita ocorreu. No último caso, Z realizou uma visita, entretanto existem várias possibilidades de onde ela pode ter ocorrido, logo a assertividade foi baixa.

3.2 Processamento dos dados

O Apache Spark[4] é um framework de processamento de Big Data que oferece velocidade e facilidade de uso. Possui módulos para processamento de dados em *batch* e *streaming*, podendo ler diversos tipos de dados, como *parquet* e csv, de maneira intuitiva. Apesar de ter um impacto maior ao rodar de forma distribuída, com alta performance, também pode rodar *standalone*, oferecendo também o Spark-shell, que provê uma interface de linha de comando interativa.

Figura 5. Módulos do Apache Spark



Fonte: https://databricks.com/spark/about

O Spark SQL é o módulo que permite trabalhar com dados estruturados, em que são utilizados DataFrames. Um DataFrame é uma coleção de dados organizada de forma colunar, em que cada coluna é tipada e possui um nome, se assemelhando a uma tabela de um banco de dados relacional.

Outra parte importante foi o processamento local que foi feito para que os dados se adequassem à modelagem de série temporal, o código utilizado foi da figura abaixo,

[4] https://spark.apache.org/

basicamente ele é capaz de criar séries temporais de tamanho arbitrário dado um DataFrame de features e labels:

Figura 6. Código para criação do vetor de entrada

```
def create_input_data(dataset, look_back=1, flatten=False):
    X_data, Y_labels = [], []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i+look_back)]
        X_data.append(a)
        Y_labels.append(dataset[i + look_back])
    return X_data, Y_labels
```

Fonte: O Autor

Outro ponto importante foi que, para o primeiro tipo de experimento realizado (detalhado na seção 4.1), foi necessário criar o código de forma que ele consumisse a menor quantidade de memória possível de forma que os testes fossem realizados com maior quantidade de dados disponível. Em resumo, o que ele faz é criar um processo separado do processo central, mover todos os recursos para esse processo (variáveis, constantes), executar o código, retornar o resultado da função para o processo principal e se remover pelo garbage collector.

Figura 7. Código para previnir *Out of Memory*

```
import os
import sys
import traceback
from functools import wraps
from multiprocessing import Process, Queue


def processify(func):
    '''Decorator to run a function as a process.
    Be sure that every argument and the return value
    is *pickable*.
    The created process is joined, so the code does not
    run in parallel.
    '''

    def process_func(q, *args, **kwargs):
        try:
            ret = func(*args, **kwargs)
        except Exception:
            ex_type, ex_value, tb = sys.exc_info()
            error = ex_type, ex_value, ''.join(traceback.format_tb(tb))
            ret = None
        else:
            error = None

        q.put((ret, error))

    # register original function with different name
    # in sys.modules so it is pickable
    process_func.__name__ = func.__name__ + 'processify_func'
    setattr(sys.modules[__name__], process_func.__name__, process_func)

    @wraps(func)
    def wrapper(*args, **kwargs):
        q = Queue()
        p = Process(target=process_func, args=[q] + list(args), kwargs=kwargs)
        p.start()
        ret, error = q.get()

        if error:
            ex_type, ex_value, tb_str = error
            message = '%s (in subprocess)\n%s' % (ex_value.message, tb_str)
            raise ex_type(message)

        return ret
    return wrapper
```

Fonte: O Autor

Com isso, ao anotar funções como *create_input_data* com o decorator *processify*, é possível que ao finalizar uma rodada do método e logo após chamá-lo novamente e os dados serem limpos do interpretador *python*, dado que o processo é removido neste momento.

3.3 Anonimização dos dados

Para anonimizar os dados e prevenir que ao sair das redes da In Loco a informação não pudesse ser trackeada de volta ao usuário original de forma que um terceiro pudesse identificá-lo, realizei os seguintes procedimentos:

1. Para cada *mad_id* associei um número randômico gerado pela biblioteca java.util.Random.
2. Apliquei a cifra md5 à concatenação do valor de *mad_id* e o número randômico gerado no passo 1.
3. Removi a coluna *mad_id* dos dados.

Desta forma, é possível garantir que a identidade do usuário é preservada, dado que não é possível fazer o caminho de volta para o *mad_id* do mesmo.

3.4 Pré-processamento dos dados

A limpeza dos dados ocorreu da seguinte forma:

1. Todas as visitas onde a latitude ou longitude fossem nulas foram removidas. Isto pode acontecer por diversos fatores, mas para nosso trabalho se uma visita não está associada à uma localização, ela é considerada inválida.
2. Todas as visitas onde o campo “reliability” é diferente de “HIGH” foram removidas. Se um lugar não é classificado com “HIGH”, o algoritmo de classificação de lugares não possui certeza da visita, o que, dado a quantidade de dados existentes, dificulta o pré-processamento e a análise.
3. Todos os lugares visitados possuem uma lista de categorias associadas à ele na base da In Loco, em ordem de especificidade, desta forma, escolhemos a mais específica sempre.
4. O timestamp teve de ser dividido por 1000, pois a maioria dos algoritmos espera o *timestamp* em milisegundos.

Ao final desta etapa, o conjunto de dados é formado pelas informações apresentadas na Figura 8.

Figura 8. Esquema dos dados pré-processados

new_id	address	reliability	labels	id	dt	timestamp	time_zone
cde8be3aaa1b8949c8d62a3a	{'country': 'BR', 'admin_area': 'São Paulo', '...	MEDIUM	[restaurant]	58b84ce658dae658c2172e9d	2018-07-07	1530974866722	America/Sao_Paulo
16ab453a4b9a06cf23277d1b	{'country': 'BR', 'admin_area': 'Acre', 'sub_a...	HIGH	None	None	2018-07-01	1530459729029	America/Manaus
16ab453a4b9a06cf23277d1b	{'country': 'BR', 'admin_area': 'Acre', 'sub_a...	MEDIUM	None	None	2018-06-28	1530220674935	America/Manaus
16ab453a4b9a06cf23277d1b	{'country': 'BR', 'admin_area': 'Acre', 'sub_a...	MEDIUM	None	None	2018-07-02	1530565496644	America/Manaus
16ab453a4b9a06cf23277d1b	{'country': 'BR', 'admin_area': 'Acre', 'sub_a...	MEDIUM	None	None	2018-07-02	1530561893535	America/Manaus

Fonte: O Autor

Onde,

- *new_id* = é uma representação criptografada do *id* dado ao usuário por toda a base da In Loco;
- *address* = é o endereço da visita registrada, composto por alguns subcampos:
    - *reliability* = confiança que temos de que o lugar foi o lugar visitado;
    - *id* = evento de visita, essencial para contagem não ser duplicada;
    - *dt* = data em que houve o evento de visita;
- *timestamp* = tempo exato em milisegundos que a visita é detectada no servidor;
- *time_zone* = região de horário setado pelo device na hora da visita.

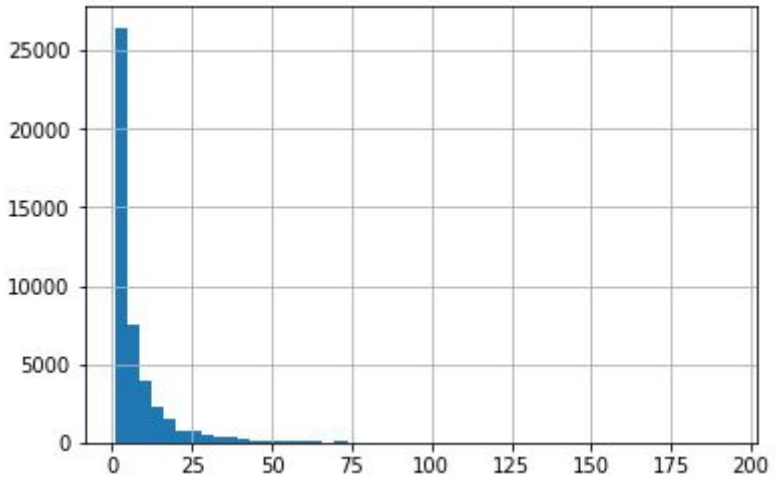
Ao final, o conjunto de dados utilizado para as análises possui as seguintes características:

- 732,938 visitas;
- 45,336 usuários únicos;
- Em média 7 visitas por usuário com desvio padrão de 10 visitas;
- 25% dos usuários possuem no máximo 1 visita;
- Somente 25% possuem mais do que 10 visitas;

- O máximo de visitas é 192.

A distribuição de visitas é mostrada no gráfico da Figura 9.

Figura 9. Distribuição de Visitas por usuário no conjunto de dados



Fonte: O Autor

3.5 Técnicas Investigadas

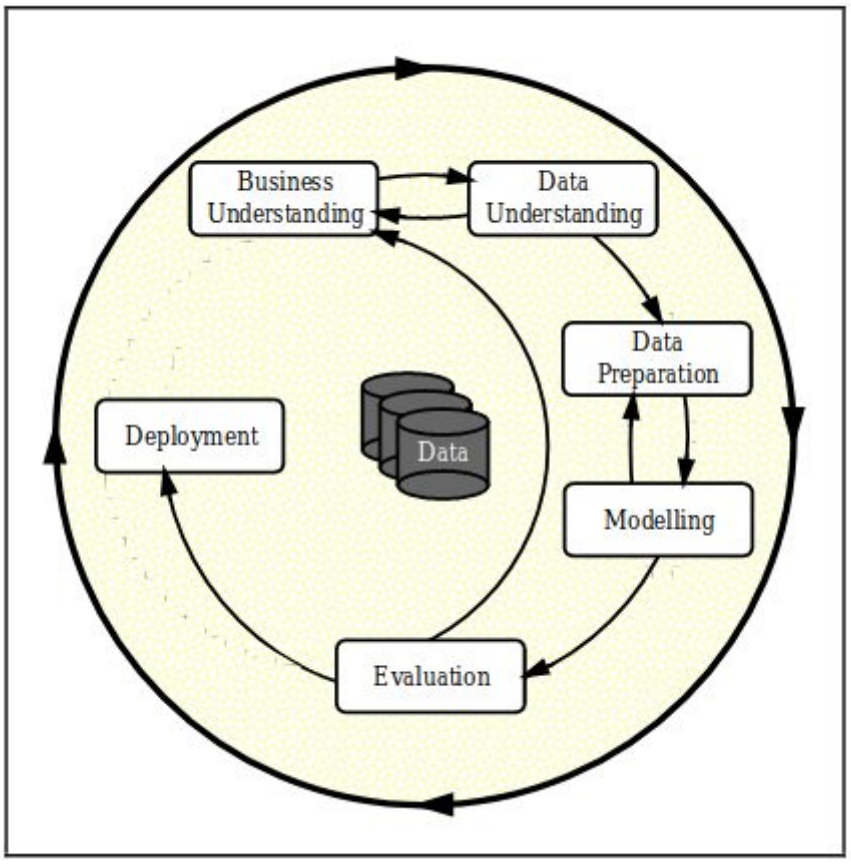
As técnicas investigadas neste projeto foram o CRISP-DM (*Cross Industry Industry Standard Process for Data Mining*), uma ferramenta teórica criada por um consórcio de empresas como IBM, Microsoft e outras para a organização de projetos de *Data Mining*. Esta estratégia foi utilizada para a elaboração do plano de projeto e cronograma. Nós investigamos as três arquiteturas de redes neurais artificiais:

1. MultiLayer Perceptron (MLP);
1. Long Short Term Memory (LSTM);
2. Recurrent Neural Network (RNN);

3.6 Metodologia

A metodologia seguida para o desenvolvimento do modelo de aprendizagem foi o CRISP-DM. Na Figura 9 é apresentado um diagrama exemplificando a metodologia.

Figura 10. Representa o processo de modelagem CRISP-DM



Fonte: Wirth et. al (2000)

O processo CRISP-DM é dividido em 6 grandes blocos:

Entendimento do negócio (*Business Understanding*): Neste passo o objetivo é entender o porque do modelo está sendo criado. Qual seu objetivo do ponto de vista de negócio, ou seja, qual métrica ele irá ajudar alcançar.

Entendimento dos dados (*Data Understanding*): Se busca neste estágio conhecer mais sobre as peculiaridades do conjunto de dados, como a distribuição dos seus valores, a quantidade de dados nulos ou valores que podem ser *outliers*.

Preparação dos dados (*Data Preparation*): É a fase final de preparação dos dados. Nesta fase estão incluídos, por exemplo, a limpeza dos dados, transformações e construções de novos atributos.

Modelagem (*Modelling*): Nesta fase, é onde várias técnicas de modelagem são testadas e seus parâmetros são ajustados para melhor atender o problema. Como existe uma dependência entre a entrada requerida pelo modelo, a parte de modelagem está muito atrelada à preparação dos dados, muitas vezes inclusive existe uma interação entre as duas.

Avaliação (*Evaluation*): Nesta fase é onde os modelos criados na fase de modelagem são avaliados e há uma checagem de que eles alcançam as necessidades do negócio e se de fato alcança a qualidade esperada através de testes estatísticos.

Deploy (*Deployment*): Ao final do projeto, o conhecimento adquirido deve ser apresentado ao cliente, seja na forma de relatório, API (*Application Programming Interface*) ou qualquer outro entregável acordado entre o analista e o cliente.

Apesar deste modelo apresentar o fluxo completo de desenvolvimento, não segui todos os passos, dado que não coloquei o modelo em produção.

3.7 Linguagem utilizada

A linguagem utilizada no projeto foi Python 3.6, utilizando o gerenciador de pacotes Conda (Anaconda Cloud, 2018). Dessa forma, o arquivo gerado pelo ambiente Conda (Anexo B - Pacotes Instalados) torna fácil a reprodução do estudo. Segue no anexo B os principais pacotes utilizados.

A métrica utilizada em todas as avaliações foi a *categorical_accuracy* encontrada nos métodos de avaliação da biblioteca Keras. O código para este método de avaliação é apresentado na Figura 10.

Figura 11. Implementação da função categorical_accuracy

```
def categorical_accuracy(y_true, y_pred):
    return K.cast(K.equal(K.argmax(y_true, axis=-1),
                          K.argmax(y_pred, axis=-1)),
                  K.floatx())
```

Fonte: https://github.com/keras-team/keras/blob/master/keras/metrics.py

Basicamente, este código verifica se a saída da classe do vetor predito (*y_pred*) e do vetor original (*y_true*) são iguais para o valor de maior ativação da rede neural artificial.

3.8 Hardware Utilizado

O hardware utilizado para testes possui as seguintes características:

1. Processador: Intel Core i5 5200U 2.20GHz
2. Placa de Vídeo: GeForce 910M
3. Sistema Operacional: Ubuntu 64 bit
4. RAM: 8GB
5. Armazenamento: SSD de 240GB

Capítulo 4

Experimentos

Este capítulo é constituído por duas seções, a primeira referente à explicação de como os modelos foram treinados e como eles também foram avaliados, além de como foi constituído cada grupo de teste.

Em seguida, apresento os resultados encontrados nos experimentos.

4.1 Treinamento e avaliação

Após construir os modelos de acordo com a metodologia definida, avaliamos a distribuição das acurácias alcançadas entre eles e comparamos os seus resultados para determinar se existe diferença significativa entre eles.

Os modelos e formas de treinamento avaliados foram divididos em dois grupos:

1. Um modelo dos tipos LSTM, RNN e MLP foi treinado para todos os usuários. Os seguintes parâmetros foram variados: taxa de aprendizagem, arquitetura da rede e quantidade de elementos que formavam o vetor de entrada.
    a. Este modelo foi testado contra todos os usuários que formam o 1ª grupo de teste, este 1ª grupo é formado por uma parcela da base total, em torno de 30% dos usuários.
    b. A modificação da arquitetura das redes neurais utilizadas como modelos foi feita de forma manual e arbitrária, o único parâmetro válido para escolher uma sobre outra foi a diferença de acurácia, sendo a de maior acurácia escolhida.

2. Neste segundo conjunto, os modelos são treinados e testados em cima de cada usuário da base, ou seja, é criado um modelo para cada usuário e ele testado diretamente sobre aquele usuário. A divisão dos dados para os usuários foi feita da seguinte maneira:
    a. Para cada usuário, as suas visitas são ordenadas em relação ao tempo e as 70% primeiras são utilizadas para treinamento e as 30% restantes são utilizadas para avaliação, este grupo de usuários utilizados é que chamo de 2ª grupo de teste.

b. A modificação da arquitetura das redes neurais utilizadas como modelos foi feita de forma manual e arbitrária, o único parâmetro válido para escolher uma sobre outra foi a diferença de acurácia, sendo a de maior acurácia escolhida.

Outro ponto importante do projeto foi a utilização da biblioteca Keras (Keras, 2018), que tornou bem mais simples a criação de modelos. Nas Figura 11, 12 e 13 são apresentados exemplos de algumas redes neurais utilizadas e o código-fonte utilizado para criar suas camadas (cada camada foi adicionada na sequência em que o código é executado).

Figura 12. Código para a criação do modelo LSTM

```
def build_LSTM_model(array_size=97, use_dropout=False, input_shape = (1,1), optimizer= Adam()):
    model = Sequential()
    model.add(LSTM(40, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.6))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.2))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.2))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.2))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.4))
    model.add(LSTM(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(TimeDistributed(Dense(97)))
    model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=[categorical_accuracy])
    return model
```

Fonte: O Autor

Figura 13. Código para a criação de uma MLP

```
def build_model(array_size):
    #parameters
    batch_size = 20
    epochs = 10

    model = Sequential()
    model.add(Dense(120, input_dim=array_size, activation='relu'))
    model.add(Dense(80, input_dim=97, activation='relu'))
    model.add(Dense(97, input_dim=97, activation='sigmoid'))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[categorical_accuracy])


    return model
```

Fonte: O Autor

Figura 14. Código para a criação de uma RNN totalmente conectada

```
def build_RNN_model(array_size=97, use_dropout=False, input_shape = (1,1)):
    model = Sequential()
    model.add(SimpleRNN(40, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.6))
    model.add(SimpleRNN(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(SimpleRNN(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.2))
    model.add(SimpleRNN(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(SimpleRNN(30, input_shape=input_shape, return_sequences=True, activation="softmax"))
    model.add(Dropout(0.2))
    model.add(TimeDistributed(Dense(97)))
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[categorical_accuracy])
    return model
```

Fonte: O Autor

4.2 Resultados

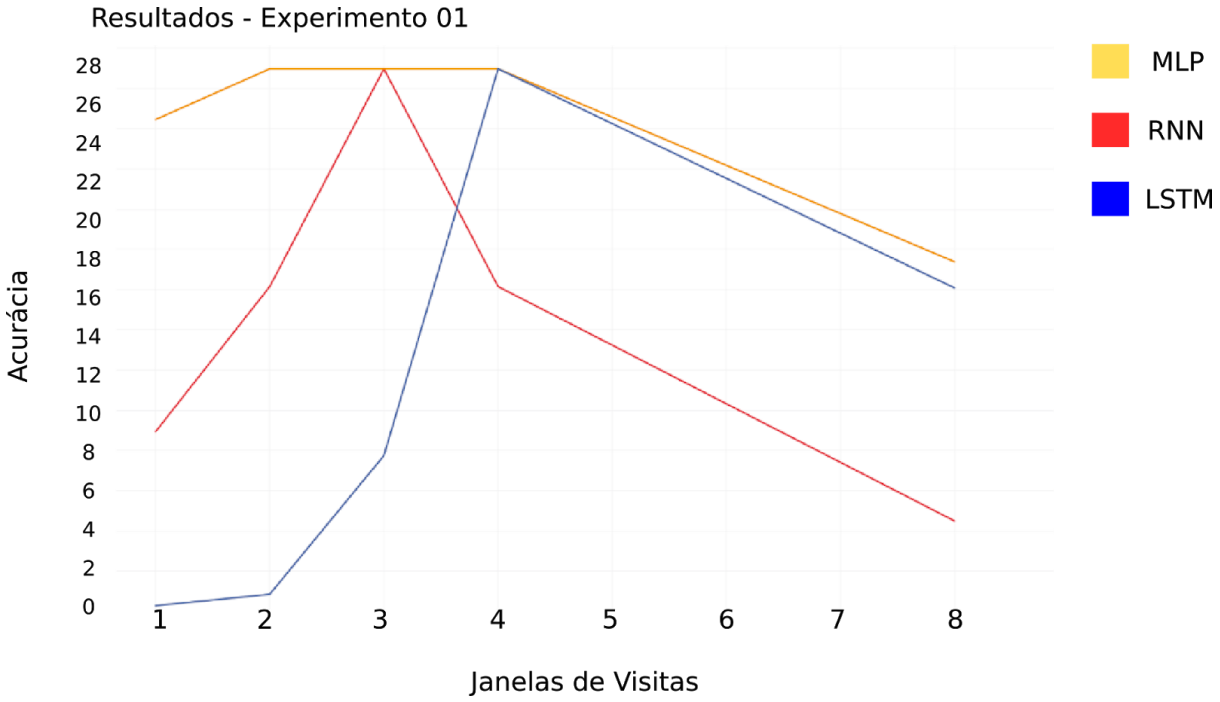
Os resultados para 1ª grupo foi o mesmo para todos os modelos dado a melhor configuração encontrada para cada, a acurácia encontrada foi de 26%, mesmo modificando a taxa de aprendizagem de todos os modelos na melhor arquitetura, nenhum consegue passar dessa marca.

Foi verificado que a variação na taxa de aprendizagem não ajuda em nada em relação a acurácia, o único impacto da variação na taxa de aprendizagem é a quantidade de épocas que demora para que o modelo se ajuste e chegue à acurácia máxima, se estabilize.

4.2.1 Variação na acurácia a partir da modificação na quantidade de elementos no vetor de entrada

Nesta seção, as figuras 15 demonstra o impacto do tamanho no vetor de entrada, ou seja, da janela de tempo utilizada, a quantidade de visitas associadas com a próxima visita, então, se o tamanho do vetor for igual à 3 significa utilizar informações de 3 visitas anteriores como entrada para a informação da visita sendo classificada.

Figura 15. Performance dos modelos



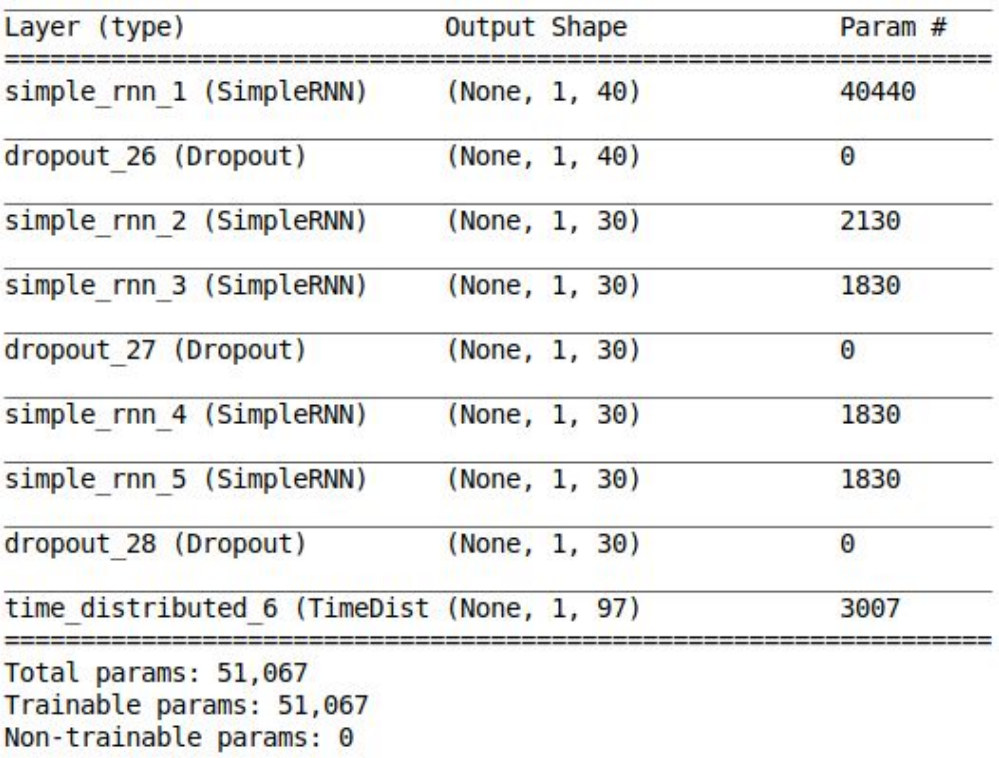
Fonte: O Autor

É possível perceber que a performance da MLP degrada com o aumento do tamanho do vetor enquanto a performance da LSTM cresce. O mais interessante é perceber que a qualidade e limitação da RNN é percebida no gráfico acima, sua performance possui o pico em um tamanho de vetor pequeno, enquanto a LSTM possui uma vantagem sobre os modelos quando o tamanho do vetor de entrada aumenta. O pico de performance de todos os modelos é quase o mesmo, 26% de acurácia, o que eu não consegui comprovar é se é uma limitação dos dados ou da arquitetura dos modelos que utilizei, logo em seguida é possível checar as configurações "ótimas" encontradas manualmente, a variação de camadas, quantidade de neurônios e tipos de camadas foi baseada nos testes em cima de usuários com vetor de tamanho 4.

Em relação ao tempo de treinamento, aumentar a quantidade de neurônios por camada para um número maior que 30 só ajudou no modelo MLP. Na LSTM e na RNN, a variável que realmente afetou a performance dos modelos foi sua profundidade, não consegui observar o porquê do impacto dessas variações na aprendizagem dos modelos.

Segue a arquitetura das redes neurais que obtiveram melhores resultados no primeiro grupo:

Figura 16. Rede RNN - Arquitetura que entregou o melhor resultado

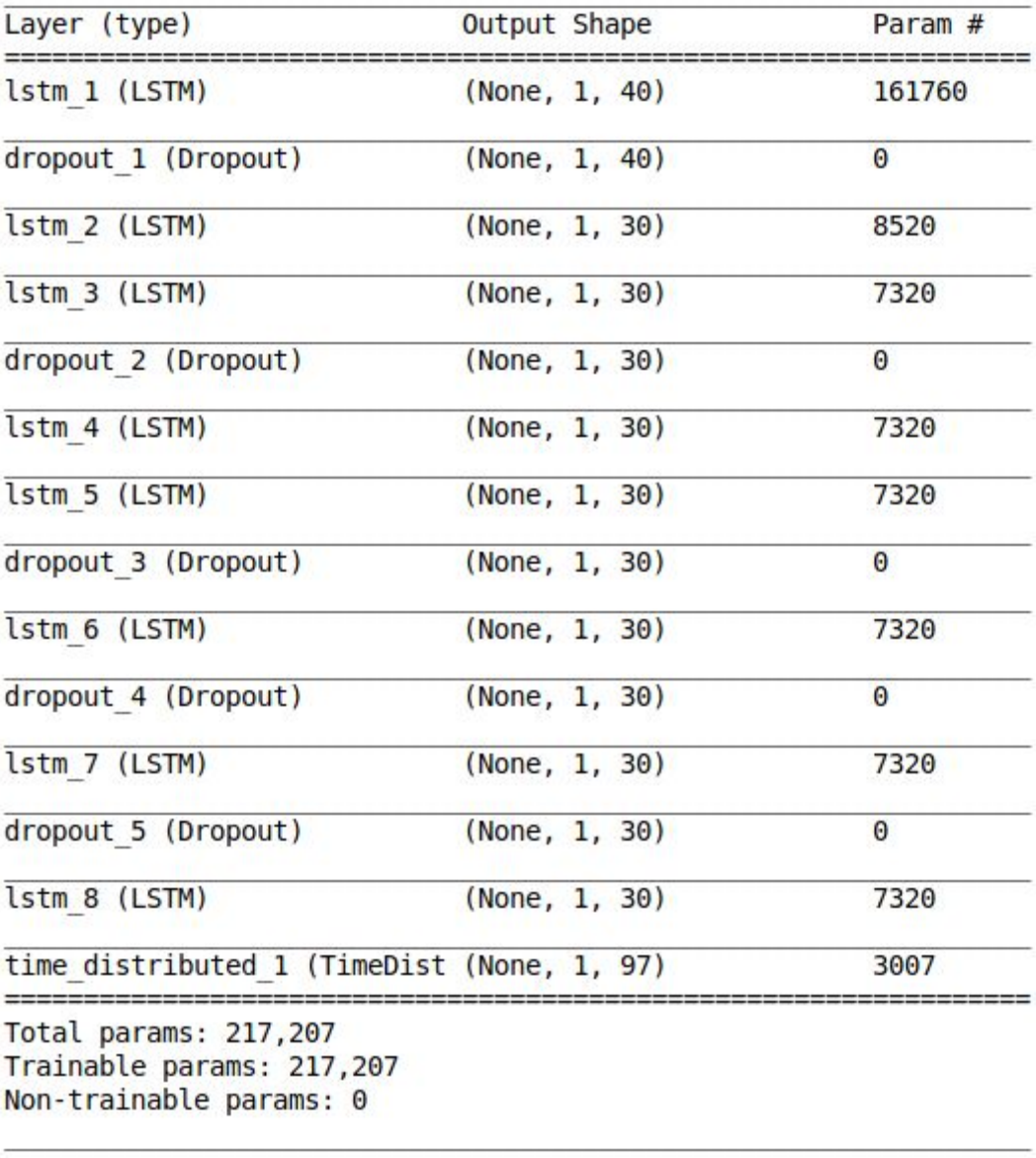


Layer (type)	Output Shape	Param #
simple_rnn_1 (SimpleRNN)	(None, 1, 40)	40440
dropout_26 (Dropout)	(None, 1, 40)	0
simple_rnn_2 (SimpleRNN)	(None, 1, 30)	2130
simple_rnn_3 (SimpleRNN)	(None, 1, 30)	1830
dropout_27 (Dropout)	(None, 1, 30)	0
simple_rnn_4 (SimpleRNN)	(None, 1, 30)	1830
simple_rnn_5 (SimpleRNN)	(None, 1, 30)	1830
dropout_28 (Dropout)	(None, 1, 30)	0
time_distributed_6 (TimeDist	(None, 1, 97)	3007

Total params: 51,067
Trainable params: 51,067
Non-trainable params: 0

Fonte: O Autor

Figura 17. Arquitetura do Modelo LSTM - Arquitetura que entregou o melhor resultado



Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 1, 40)	161760
dropout_1 (Dropout)	(None, 1, 40)	0
lstm_2 (LSTM)	(None, 1, 30)	8520
lstm_3 (LSTM)	(None, 1, 30)	7320
dropout_2 (Dropout)	(None, 1, 30)	0
lstm_4 (LSTM)	(None, 1, 30)	7320
lstm_5 (LSTM)	(None, 1, 30)	7320
dropout_3 (Dropout)	(None, 1, 30)	0
lstm_6 (LSTM)	(None, 1, 30)	7320
dropout_4 (Dropout)	(None, 1, 30)	0
lstm_7 (LSTM)	(None, 1, 30)	7320
dropout_5 (Dropout)	(None, 1, 30)	0
lstm_8 (LSTM)	(None, 1, 30)	7320
time_distributed_1 (TimeDist	(None, 1, 97)	3007

Total params: 217,207
Trainable params: 217,207
Non-trainable params: 0

Figura 18. Arquitetura do Modelo MLP - Arquitetura que entregou o melhor resultado

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 120)               11760
_________________________________________________________________
dense_12 (Dense)             (None, 80)                9680
_________________________________________________________________
dense_13 (Dense)             (None, 97)                7857
=================================================================
Total params: 29,297
Trainable params: 29,297
Non-trainable params: 0
_________________________________________________________________
```

O resultado dos experimentos para o 2ª Grupo, detalhado na seção 4.2, são apresentados na Tabela 2.

Tabela 2. Resultados no 2º grupo de teste

	LSTM	RNN	MLP
Média	18,35	14,52	30.94
Variância	286	197	260
Desvio Padrão	16,91	14	16

Fonte: O Autor

A média de acurácia alcançada pela LSTM foi de 18% de acerto com uma variância de 286 pontos enquanto a RNN alcançou uma acurácia de 14,52% de acerto com variância de 197, o T-test demonstra que a diferença de acurácia entre os dois modelos não é significativa, p-value igual à 0.0855. Enquanto isto, a MLP obteve o melhor rendimento, com 30,94% de acurácia média.

Capítulo 5

Conclusão

A partir dos experimentos realizados e os resultados obtidos, demonstrados na Tabela 2, foi verificado que, em geral, a LSTM possui uma melhor performance do que outras redes neurais profundas, como RNN, dado uma liberdade de 10% de precisão, porém os testes nos do 2ª grupo revelam que há uma vantagem da MLP sobre as redes neurais profundas quando não há variação de tamanho de vetor pois o seu desempenho médio é pouco dependente do tamanho do vetor, como demonstrado nas figuras 15 à 17 do primeiro grupo de teste.

A variação da quantidade de elementos utilizados no vetor de entrada é extremamente significante na melhora ou piora de cada classificador e o valor ideal não é único para toda base. Por exemplo, paga alguns usuários, um vetor de tamanho 5, ou seja, possuindo 5 visitas anteriores à label da posterior é bem melhor do que 10, por exemplo, o que significa que alguns usuários possuem ciclos de rotina maiores enquanto que outros ciclos menores. O que tornou mais difícil mensurar a capacidade de cada modelo, dado que utilizei um número fixo de tamanho de vetor de entrada para todos os usuários ao invés de testar alguns valores para achar o ótimo para cada usuário.

Nos experimentos foi possível avaliar que a LSTM só passou a ser melhor que os outros modelos a partir de um tamanho de vetor alto, entre 7 e 9, o que dificultou os testes pela quantidade de memória exigida para criar os datasets de treinamento e teste.

Em suma, utilizar a LSTM requer um *fine tuning* de parâmetros que eu não pude explorar por completo neste trabalho, o mesmo vale para os outros modelos testados como RNN e MLP. Além disso, o dataset não possui tantos dados de qualidade para todos os usuários de sua base, a métrica de teste melhorou muito ao filtrar os usuários que possuíam pelo menos 10 visitas, de forma que fui capaz de entender um pouco mais sobre o que era rotina e como ela aparecia no comportamento dos usuários.

Outro ponto importante é que nos dados existem *missing data* relacionados às visitas dos usuários, seja porque eles desligaram a localização do celular ou o próprio dispositivo. Ou seja, pode ser que a performance seja melhorada caso os usuários estejam dispostos a

manter o device ligado ininterruptamente durante o dia por um período contínuo de tempo, 1 mês, por exemplo.

Para futuros projetos e estudos com esta base de localização, acho ser essencial a utilização de dados de usuários que possuam pouca quantidade de *missing data* pois é um problema muito difícil o preenchimento do mesmo. Outro ponto importante é a exploração de como aprender o valor ideal do "tamanho da rotina do usuário".

Uma limitação importante do trabalho está relacionado à ferramenta utilizada, ela não possui uma forma simples de manipular os portões utilizados nos neurônios de uma LSTM. Esta limitação dificulta a manipulação da rede neural para o problema específico, então uma troca de ferramenta ou modificação da mesma para permitir tais manipulações pode ser um grande avanço para projetos futuros que utilizem LSTM como principal tipo de rede neural.

Referências

Anaconda Cloud, Disponível em: https://anaconda.org/, Acesso em 06 Dez. 2018

Apache Parquet, Disponível em: https://parquet.apache.org/. Acesso em 06 Dez. 2018

Fokianos, K., & Kedem, B. (2003). Regression theory for categorical time series. *Statistical science*, *18*(3), 357-376.

Graves, Alex, and Jürgen Schmidhuber. "Framewise phoneme classification with bidirectional LSTM and other neural network architectures." *Neural Networks* 18.5-6 (2005): 602-610.

Greff, Klaus, et al. "LSTM: A search space odyssey." *IEEE transactions on neural networks and learning systems* 28.10 (2017): 2222-2232.

Haykin, Simon. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Keras, Disponível em: https://keras.io/, Acesso em 10 Dez. 2018

Li, Youru, et al. "EA-LSTM: Evolutionary Attention-based LSTM for Time Series Prediction." *arXiv preprint arXiv:1811.03760* (2018).

Metrics - Keras Documentation, Disponível em : https://keras.io/metrics/#custom-metrics. Acesso em: 03 Dez. 2018.

Noulas, Anastasios, et al. "An empirical study of geographic user activity patterns in foursquare." *ICwSM* 11.70-573 (2011): 2.

Understanding LSTM Networks, Disponível em: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ Acesso em: 03 Dez. 2018

Wirth, Rüdiger, and Jochen Hipp. "CRISP-DM: Towards a standard process model for data mining." *Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining*. Citeseer, 2000.

Chen, Wang, Qiang Gao, and Huagang Xiong. "Temporal Predictability of Online Behavior in Foursquare." *Entropy* 18.8 (2016): 296.

Foursquare, Disponível em: https://pt.foursquare.com/. Acesso em 10 Dez. 2018.

He, Jing, et al. "Personalized Next Point-of-Interest Recommendation via Latent Behavior Patterns Inference." *arXiv preprint arXiv:1805.06316* (2018).

Anexo A - Listagem de Categorias

Categoria em Português	Chave
Contabilidade	accounting
Aeroportos	airport
Parques de Diversão	amusement_park
Aquários	aquarium
Galerias de Arte	art_gallery
Caixas Eletrônicos	atm
Padarias	bakery
Bancos	bank
Bares	bar
Lojas de Bicicletas	bicycle_store
Boliche	bowling_alley
Estações de Ônibus	bus_station
Empresariais	business_center
Açougueiros	butcher
Cafés	cafe
Acampamentos	campground
Revendedores de carros	car_dealer
Locadoras de Carros	car_rental
Oficinas de carros	car_repair
Lava-Jatos	car_wash
Cassinos	casino
Prefeituras	city_hall
Lojas de Roupas	clothing_store
Centros Comunitários	community_center
Penitenciárias	correctional_facility

Tribunais	courthouse
Locais Culturais	cultural_venue
Auto Escolas	driving_school
Eletricistas	electrician
Embaixadas	embassy
Corpo de Bombeiros	fire_station
Floriculturas	florist
Praças de Alimentação	food_court
Jardins	garden
Postos de Combustível	gas_station
Supermercados	grocery_or_supermarket
Academias	gym
Lojas de Material de Construção	hardware_store
Lojas de Artigos para o Lar	home_goods_store
Agências de Seguro	insurance_agency
Joalherias	jewelry_store
Pontos Turísticos	landmark
Lavanderias	laundry
Advogados	lawyer
Bibliotecas	library
Lojas de Bebidas	liquor_store
Repartições Governamentais	local_government_office
Chaveiros	locksmith
Hospedagem	lodging
Marinas	marina
Delivery de Refeições	meal_delivery
Refeições para Viagem	meal_takeaway
Lojas de Motocicletas	motorcycle_store
Locadoras de Filmes	movie_rental

Cinemas	movie_theater
Empresas de Mudanças	moving_company
Museus	museum
Bancas de Jornal	news_stand
Boates	night_club
Escritórios	office
Oculistas	optician
Pintores	painter
Estacionamentos	park
Parques	parking
Lojas de Animais	pet_store
Farmácias	pharmacy
Locais Religiosos	place_of_worship
Encanadores	plumber
Pontos de Interesse	point_of_interest
Polícia	police
Correios	post_office
Transporte Público	public_transportation
Restaurantes de Fast Food (QSR)	qsr
Imobiliárias	real_estate_agency
Restaurantes	restaurant
Telhados	roofing_contractor
RV Parks	rv_park
Escolas	school
Lojas de Sapatos	shoe_store
Shoppings	shopping_mall
Spas	spa
Centros Esportivos	sports_center
Estádios	stadium

Armazenamento	storage
Lojas	store
Estações de Metrô	subway_station
Piscinas	swimming_pool
Pontos de Táxi	taxi_stand
Teatros	theater
Circuitos	track
Estações de Trem	train_station
Estações de Trânsito	transit_station
Agências de Viagem	travel_agency
Universidades	university
Veterinários	veterinary_care
Bem-Estar	well_being

Anexo B - Pacotes Instalados

1. Python 3.6[5]
2. Pandas[6]
3. Keras[7]
4. Jupyter notebook[8]

[5] "Python.org." https://www.python.org/. Acessado em 12 dez. 2018.
[6] "Python Data Analysis Library — pandas: Python Data Analysis Library." https://pandas.pydata.org/. Acessado em 12 dez. 2018.
[7] "Keras Documentation: Home." https://keras.io/. Acessado em 12 dez. 2018.
[8] "Jupyter Notebook." https://jupyter.org/. Acessado em 12 dez. 2018.