CrawMobi: Um web service para enriquecimento semântico de dados por meio da coleta de características de dispositivos móveis

Kaio Igor V. Alves[1], **Fabrício A. Silva**[1]

[1]Universidade Federal de Viçosa - Campus Florestal

{kaio.alves,fabricio.asilva}@ufv.br

Abstract. *In the context of mobile services, it is crucial that a company that develops these services knows its customers. One way to provide this knowledge is to profile users. An important aspect of such a profile concerns the characteristics of the mobile device used by the user. However, getting these features is not an easy task. In this work, a meta-data web service was developed to extract details of smartphones from trusted sites, returning a database with details about the type of device used. With the data returned, some testing and analysis was done to get the profile description of mobile users.*

Resumo. *No contexto dos serviços móveis, é fundamental para uma empresa que desenvolve tais tarefas conhecer os seus clientes. Uma das formas de prover esse conhecimento é a partir do perfil dos usuários. Um aspecto importante de tal perfil diz respeito às características do aparelho móvel utilizado pelo usuário. Porém, obter estas características não é uma tarefa fácil. Neste trabalho, foi desenvolvido um Web Service de metabusca de dados que extrai detalhes de dispositivos móveis de sites confiáveis na web, retornando uma base de dados com detalhes relvantes sobre o tipo de aparelho utilizado. Com os dados retornados, são feitos alguns testes e análises para levar a descrição do perfil de usuários móveis.*

1. Introdução

A evolução da tecnologia de serviços móveis fez com que cada vez mais aplicações surgissem nesse âmbito. Esse crescimento demanda um produto final de maior qualidade por parte das empresas desenvolvedoras destes serviços. Para obter resultados melhores é importante que uma empresa conheça bem seu cliente, podendo oferecer um serviço adequado para o perfil do mesmo. Uma forma de conhece-lô melhor, é analisando seu perfil como usuário móvel.

Existem diversos aspectos que definem o perfil de um usuário móvel. Dentre eles, um importante para tal tarefa diz respeito a conhecer as características do aparelho móvel do usuário. Tais características como o preço, memória de armazenamento, capacidade da bateria, tamanho da tela, entre outros, podem indicar algumas informações do proprietário deste smartphone. Estudos anteriores evidenciam diferenças sobre o padrão de uso de aplicações móveis de acordo com o usuário [Rahmati and Zhong 2013, H. Falaki and Estrin 2010]. A partir de análises de tais características dos aparelhos, podemos definir um padrão a partir do usuário, percebendo quais são seus interesses e serviços que tende a utilizar.

Além de seus interesses, é possível determinar também as necessidades de um usuário com base nesses estudos. Sabendo que a capacidade da bateria de um aparelho é alta, por exemplo, é possível inferir que seu proprietário demanda tal recurso por algum motivo. Podemos pensar também que, tendo uma câmera com ótima resolução, é provável que o usuário goste de fotografias e/ou filmagens de boa qualidade. Outro aspecto relevante, inferido a partir de análises das características dos aparelhos, é o perfil social. Pessoas com rendas diferentes tendem a utilizar serviços móveis distintos [Malmi and Weber 2016].

Utilizando as funcionalidades disponíveis nos sistemas operacionais móveis, os donos de aplicativos não conseguem muitas informações sobre o dispositivo utilizado pelo cliente. Portanto, é necessário realizar uma busca externa para coletar as características detalhadas do aparelho móvel utilizado. Entretanto, obter estas características não é uma tarefa trivial. Existe uma grande quantidade de aparelhos móveis e muitas fontes de informações sobre eles. Muitas das vezes, a ausência de dados ou mesmo informações errôneas dificultam o trabalho de extrair características coesas. Portanto, a coleta de informações detalhadas sobre o dispositivo móvel para enriquecer os dados dos usuários não é uma tarefa simples.

Pensando neste problema, o objetivo deste trabalho é extrair informações detalhadas de dispositivos móveis de algumas bases de dados reais. Para isso, foi desenvolvido um *Web Service*, chamado *CrawMobi*, que acessa sites confiáveis sobre telefonia móvel para obtenção das características de aparelhos desejados. Para avaliar o *Web Service* desenvolvido, o mesmo foi utilizado para criar uma base de dados com o intuito de oferecer estas ricas características dos aparelhos. Com o objetivo de comprovar os benefícios da base de dados oferecida, é apresentada uma análise dos dados coletados. Com isso, será possível conhecer o perfil de cada usuário, proprietário de algum aparelho listado, o classificando de acordo com as características da base de dados.

Este texto está organizado da seguinte forma. A seção 2 apresenta alguns conceitos utilizados para o desenvolvimento do *Web Service* e importantes para o entendimento do assunto. A seção 3 descreve a arquitetura do *CrawMobi*, detalhadamente por módulos. A seção 4 mostra testes e análises a partir da base de dados retornada pelo *CrawMobi*. Por último, as conclusões são apresentadas na seção 5.

2. Conceitos Relacionados

O desenvolvimento da aplicação demanda um estudo sobre alguns conceitos e técnicas. Estes serão abordados neste trabalho, juntamente de ferramentas utilizadas no desenvolvimento do projeto.

2.1. Recuperação de Informação

Para enriquecimento da base de dados, foi necessário a busca de informações na web. Esta tarefa é realizada graças à recuperação de informação, que é "o processo de recuperar documentos de uma coleção em resposta a uma consulta por um usuário"[Elmasri and Navathe 2010, p. 994].

Para introduzir o conceito de recuperação da informação é importante antes frisar a diferença entre dados estruturados e não estruturados. O primeiro trata-se de informações que se dão sempre em um padrão de organização, normalmente armazenadas em um

banco de dados relacional e nunca fugirão deste escopo pré-definido. Já em dados não-estruturados não é possível esperar um modelo formal bem definido para representação e argumento dos mesmos. Estes dados podem ser armazenados de diversas formas e estruturas. Um exemplo seria a locação de veículos. Este serviço pode seguir padrões diferentes de funcionamento de região para região, em um determinado país. Determinadas empresas neste ramo podem, por exemplo, exigir mais cláusulas no contrato, permitir o cadastro de mais de um número de telefone para contato, além de vários outros detalhes que a diferem das outras da mesma área. Tudo isso faz com que as informações geradas possam ter estruturas representativas bem diferentes. É importante destacar que dados não-estruturados normalmente seguem uma linguagem natural para serem representados.

Historicamente, o conceito abordado nesta seção é um contexto que "trata da estrutura, análise, organização, armazenamento, pesquisa e recuperação de informações" [Salton 1968]. Complementando este conceito é necessário ressaltar que a recuperação da informação se aplica a situações em que temos dados não-estruturados. Com a recuperação destes, suprimos as necessidades do usuário.

Existem, basicamente, três níveis de recuperação da informação: por tipos de usuários, tipos de dados e tipos de necessidade da informação [Elmasri and Navathe 2010]. Neste trabalho foi utilizado o segundo nível, que se refere a compor os sistemas de pesquisa em dados específicos. Com estes sistemas personalizados, recuperar informações da web sobre assuntos ou contextos específicos se torna mais eficiente. A busca é voltada para locais próprios para este conteúdo. A representação dos dados recuperados pode ser organizada de forma hierárquica de acordo com um assunto específico, facilitando a exploração do dados obtidos.

2.2. Mecanismo de Busca

Uma das formas práticas de realizar recuperação de informações é através de mecanismos de busca para grandes coleções de documentos. Dado o grande crescimento do número de dispositivos conectados à Internet, a quantidade de dados distribuídos na web é cada vez mais impactante. Isso fomentou o desenvolvimento de mecanismos de busca para recuperar diferentes tipos de dados em tempo real.

Habitualmente, a recuperação de informações retorna uma coleção de documentos como resposta. Estes documentos possuem dados não-estruturados contendo informações distribuídas por ele. Para retorno correto do documento buscado, os mecanismos de busca utilizam uma consulta especificando a demanda pela informação. Tal consulta é composta por um conjunto de palavras-chave para esta recuperação.

2.2.1. Crawler

O componente do mecanismo de busca responsável pela descoberta, análise e indexação do conteúdo web pesquisado é denominado *crawler*. Para descoberta do conteúdo interessante este componente vasculha a rede a procura de páginas, documentos, dentre outros, a partir de informações relevantes que ele recebe como parâmetro. Após descobrir um potencial documento, ele captura seu conteúdo e insere o seu endereço eletrônico em uma lista de links úteis encontrados. Com o conteúdo de uma página já capturado, o web *crawler* percorre seus elementos procurando em seu código HTML tags específicas que

possuam informações desejadas a serem recuperadas. O processo descrito anteriormente pode ser denominado como análise. Finalmente o componente realiza a indexação do conteúdo a partir dos links salvos. Por meio de cada link, o *crawler* pode navegar entre as páginas, sendo possível encontrar outros links importantes.

Esta importante parte da ferramenta de busca pode executar um pré-processamento dos dados obtidos fazendo com que as respostas das buscas por meio dos usuários sejam rápidas e eficazes. Porém, um *crawler* pode ser utilizado também para manutenção automatizada de páginas web e para obter dados específicos de uma ou mais páginas na rede. O sistema desenvolvido neste trabalho utilizou o *crawler* desta última forma em que ele visita páginas web específicas e captura determinados dados pré-estabelecidos para caracterização da base de dados.

2.3. Web Services

Com intuito de fazer com que a aplicação proposta neste trabalho fosse disponibilizada na Internet para facilitar o acesso remoto, foi criado um *Web Service*. *Web Service* é uma solução composta por vários métodos, que é invocada por outras aplicações utilizando tecnologias web. Assim, esta solução transmite dados utilizando protocolos de comunicação de rede para diferentes aplicações, independentemente de tecnologias e ferramentas utilizadas para sua criação. Um dos princípios de uma solução como esta é trazer agilidade para os processos de uma organização, tendo em vista que uma aplicação requer um conteúdo e o mesmo é retornado a ela via protocolos de rede. Portanto, ele traz eficiência na comunicação e integração entre serviços. Desde tarefas simples a complexas podem ser resolvidas para a aplicação que solicita o serviço, mesmo estando em plataformas diferentes.

O principal objetivo de um *Web Service* é prover a comunicação entre aplicações pela internet. Com isso ele facilita a integração entre diferentes aplicações de uma corporação. Este serviço também deve contar com uma facilidade em modificar e adicionar novas funcionalidades, com intuito de prover e facilitar a melhoria do mesmo. Utilizando um *Web Service* é notada uma redução no tempo de desenvolvimento de uma aplicação tendo em vista que um conjunto de funcionalidades necessárias já está disponível para utilização. Consequentemente, uma redução nos custos é observada já que eles tiram proveito de uma infraestrutura web já existente.

Entretanto, pelo fato de que as aplicações podem ter sido desenvolvidas em plataformas diferentes e com tecnologias distintas se faz necessária uma linguagem intermediária que possa interpretar e traduzir o que foi requisitado. Basicamente dois tipos de *Web Services* são utilizados para resolução da questão citada e para fazer uma integração dos sistemas de forma padronizada e reutilizável [Sampaio and Silva 2017].

O tipo SOAP (*Simple Object Access Protocol*), que como todo *Web Service*, é baseado em invocações remotas de métodos. Para isso é necessária a especificação do conteúdo, o nome do método a ser invocado e seus argumentos. Com esses dados ele cria um documento XML (*Extensible Markup Language*) estruturado que é transmitido, na maioria das vezes, via HTTP (*HyperText Transfer Protocol*). Ele não impõe uma semântica definida e específica para implementação, fazendo com que a comunicação funcione bem independentemente das diferenças entre as linguagens de programação utilizadas pela aplicação solicitante e o serviço. Abrangendo termos técnicos de redes de

computadores, o SOAP é um protocolo RPC (*Remote Procedure Call*) que utiliza o HTTP passando restrições básicas impostas por este protocolo suportando mensagens XML. Resumindo seu funcionamento, o SOAP envia um conteúdo XML através de um pedido HTTP e o cliente que solicita o serviço, deve validar e compreender a resposta.

O outro tipo utiliza arquitetura REST(*Representational State Transfer*), que quando utilizada para implementar *Web Services* é denominada RESTFul. "REST é um termo definido por Roy Fielding em sua tese de mestrado no qual ele descreve sobre um estilo de arquitetura de software sobre um sistema operado em rede"[Rondon 2010].

REST, em português, Transferência de Estado Representacional, como os outros *Web Services*, fornece interoperabilidade entre sistemas na Internet. Esta arquitetura possui um conjunto de restrições e propriedades definidos para comunicação via HTTP. Os sistemas que solicitam o serviço podem acessar e manipular recursos da Web de padronização e encapsulamento de dados (representados por textos, JSON, XML e outros), utilizando um conjunto de operações sem estado.

Para o desenvolvimento do *Web Service* proposto neste trabalho, foi escolhido o tipo com arquitetura REST visto a facilidade em recuperar a informação desejada. Isso faz com que o cliente consuma os dados dos servidores de maneira mais rápida, baixando somente o pacote requisitado. Além disso, o REST está sendo muito utilizado para tais serviços pois traz ótimos resultados.

A implementação do *Web Service* com arquitetura REST se deu utilizando a linguagem *Python*. Um serviço web é instalado em um servidor e normalmente demanda um *framework*. Essa ferramenta desempenha funções em tempo de execução, recepciona pedidos, realiza o processamento de mensagens, verifica os erros e a criação de representações dos pedidos, que são demandados ao serviço [Oliveira 2018]. O *framework* escolhido para tal tarefa foi o *Flask*. Ele é um *microframework* desenvolvido em *Python* e baseado nas bibliotecas *WSGI Werkzeug* e *Jinja2*. Ele é denominado assim por possuir uma base simples, porém podendo ser estendida. Ele não possui de forma pré-estabelecida vários dos componentes presentes em outros *frameworks* do mesmo tipo, como abstração para banco de dados, validações de formulário, controle de usuários, dentre outros [Ronacher 2018]. Porém ele suporta todas as extensões existentes na linguagem. Assim, a utilização deste *microframework* se torna relativamente fácil, trazendo uma economia de tempo na construção de aplicações web por possuir uma curva de aprendizado mais suave.

2.4. Análise de Dados

Após obter os dados retornados pela aplicação proposta, foram aplicadas algumas técnicas de Ciência de Dados com intuito de comprovar a robustez do *Web Service*. Portanto esta ciência e seus métodos foram estudados para embasar os estudos. Esta ciência, através de métodos como análises estatísticas por meio de algoritmos computacionais, realiza um estudo da capacidade de um determinado conjunto de dados se transformar em informações valiosas. "Compete à Ciência de Dados a aplicação de métodos de aprendizado de máquina, classificação, organização, clusterização, quantificação de incertezas, ciência computacional, mineração de dados, base de dados e visualização"[Samú 2018].

Para esta transformação de dados em conhecimento, uma análise criteriosa dos dados deve ser efetuada. A análise de dados possui três princípios básicos para guiar este

processo [Santos 2016]. O primeiro dos princípios é definir o foco da análise de dados. Para tal é necessário decidir o que será realizado, como realizar esse objetivo e saber quando a meta foi atingida, por meio de um indicador. O segundo princípio se baseia na definição das hipóteses para as análises. Uma hipótese é necessária pois é a partir dela que o problema é descoberto e entendido. Para compor ou refutar a hipótese é aplicado a análise dos dados. Por último, o terceiro princípio se dá em estruturar as perguntas para a análise. Estas perguntas devem abordar questões como o impacto dos dados e a frequência de acontecimentos, por exemplo. Portanto, quanto maior o entendimento sobre o problema melhor, pois assim várias perguntas poderão ser criadas fomentando o resultado final com muitas informações a partir das respostas pelas análises dos dados.

3. CrawMobi

Para solucionar o problema descrito na introdução deste trabalho, foi criado o *CrawMobi*. Ele é um *Web Service* implementado em linguagem *Python* que visita sites confiáveis na web com a finalidade de coletar várias características de aparelhos móveis. Para melhor entendimento do funcionamento deste *Web Service*, a Figura 1 apresenta sua arquitetura, contendo todos os seus componentes e o fluxo da informação desde a requisição do cliente até o retorno com a caracterização dos dados.

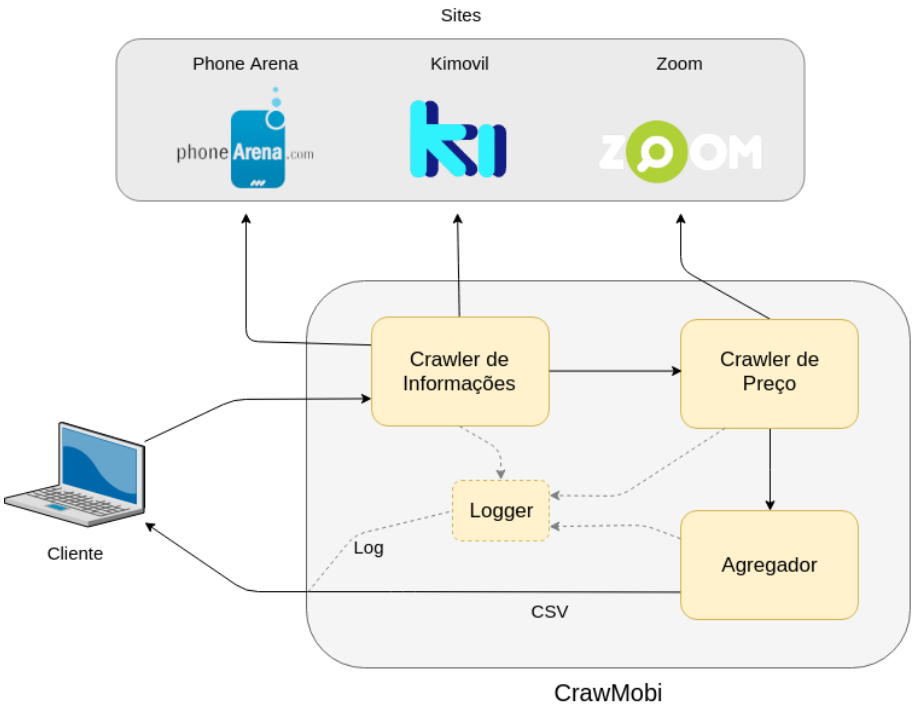


Figura 1. Arquitetura do CrawMobi

Em seguida será detalhado o funcionamento do *CrawMobi*, explicando como cada componente demonstrado na Figura 1 comporta-se no sistema.

Antes de explicar cada componente, é necessário entender como o cliente interage com o *CrawMobi*. O primeiro passo para o solicitante é criar um arquivo de extensão XLS que contenha os nomes comerciais dos aparelhos desejados para a coleta, descritos linha após linha na primeira coluna do arquivo. Os nomes comerciais devem ser compostos

pela marca do aparelho seguido de seu modelo. Com isso, o cliente entra em contato com o *Web Service* e em sua página principal e faz o *upload* deste arquivo. O serviço retorna um arquivo contendo 22 características dos aparelhos listados. Um critério para escolha destas características foi a importância delas para utilização em possíveis análises dos dados. Na Tabela 1 são apresentadas as características contidas no arquivo retornado pelo *CrawMobi*. Os aparelhos apresentados, como exemplo, e suas respectivas características são fictícios.

Tabela 1. Características dos Aparelhos

	Aparelho X	Aparelho Y
Marca	X	Y
Modelo	xx	yy
Capacidade da Bateria (mAh)	2500	3000
Memória RAM (GB)	3	4
Memória de Armazenamento (GB)	32	64
Versão do Bluetooth	4.1	4.2
NFC	SIM	NAO
Dual Chip	SIM	SIM
LTE (4G)	SIM	SIM
Resolução da Câmera (Mpx)	12	12
Peso (g)	180	195
Dimensões (mm)	74.7 x 152.7 x 7.3	73.8 x 150.5 x 7.1
Tamanho da Tela (”)	5.5	5
Sistema Operacional	SO X	SO Y
Versão do Sistema Operacional	8.1	10
Capacidade de Processamento (MHz)	2.0	1.8
Link da Fonte	www.kimovil.com	www.phonearena.com
Data de Coleta dos Dados	11/2018	11/2018
Data de Lançamento	06/2018	09/2018
Preço (R$)	1869.49	2500
Avaliação do Site	9.7	9.6
Avaliação Média dos Usuários	9.3	9.1

3.1. Crawler de Informações

Este é o primeiro módulo do *Web Service* a ser acionado. Como descrito na seção 2.2.1, ele é um *crawler* projetado para obter dados específicos de páginas na web. Utilizando a biblioteca *selenium* e o componente *webdriver* do *Google Chrome*, este *crawler* acessa dois sites específicos de comparação de smartphones: O *Kimovil* [Kimovil 2018] e o *Phone Arena* [Arena 2018]. Para cada aparelho descrito no arquivo de entrada, este componente executa as seguintes operações nos sites:

1. Percorre a página a procura do elemento HTML referente à barra de pesquisa do site. Após encontrá-la, o nome comercial do aparelho é escrito neste campo e a pesquisa é feita.
2. Com os resultados apresentados pelo site, o componente confere o nome de cada aparelho retornado. Sendo igual, ele clica automaticamente no elemento HTML

referente a este nome. Caso não encontre um nome idêntico, ele passa para o componente *Logger* que não encontrou o aparelho listado no determinado site, encerrando a execução para este.
3. Na página referente ao aparelho, o *crawler* percorre novamente os componentes HTML a procura dos elementos listados na tabela 1. Mesmo que algum atributo possua valor nulo no site, o *crawler* armazena este valor. Os dados coletados são retornados para o módulo principal do *CrawMobi* para posteriormente serem utilizados pelo componente *Agregador*.

Atualmente o *CrawMobi* está utilizando somente três sites para recuperação das informações referentes aos atributos do aparelho. Porém, uma de suas vantagens é que sua estrutura interna permite a inserção de inúmeros outros sites que contenham informações deste tipo. Isso se dá pois sua arquitetura é flexível, permitindo seu crescimento. Assim, basta definir o comportamento desempenhado pelo *crawler* para o novo site, agregando este componente.

3.2. Crawler de Preço

O atributo preço é fundamental para o propósito deste trabalho, pois a partir dele se torna possível fazer análises interessantes, obtendo informações para caracterização do perfil socio-econômico de usuário móvel. Pensando nisso, foi decidido coletar as informações de preços de um site nacional confiável, o *Zoom*. Assim, o preço recuperado será sempre o menor do mercado de forma coerente com os valores nacionais.

Similarmente ao componente anterior, o *Crawler de Preço* realiza basicamente os mesmos três passos para cada aparelho, salvo algumas modificações. Os passos seguidos são:

1. Vasculha a página do site *Zoom* à procura do elemento HTML referente a barra de pesquisa. Após encontrá-lo, o nome comercial do aparelho é escrito neste campo e a pesquisa é feita.
2. Com os resultados apresentados pelo site, o componente confere o nome de cada aparelho retornado. Sendo igual, ele clica automaticamente no elemento HTML referente a este nome. Caso não encontre um nome idêntico, ele passa para o componente *Logger* que não encontrou o aparelho listado no *Zoom*, encerrando a execução para este.
3. Na página referente ao aparelho, o *Crawler de Preço* vasculha novamente os componentes HTML. Porém, desta vez ele procura por um elemento específico, o preço. Ao encontrá-lo, este valor é retornado para o módulo principal do *CrawMobi* para ser posteriormente utilizado pelo componente *Agregador*. Caso o preço possua valor nulo na página, o valor retornado também é nulo, além de ser passado para o componente *Logger* esta informação.

3.3. Agregador

Este componente é responsável pela formatação, comparação e gravação dos dados recuperados pelo *crawler*.

A tarefa de formatação é importante pois é a partir dela que é feita uma limpeza dos dados recuperados pelo *Crawler de Informações*. Os atributos memória RAM, dimensões do aparelho, versões do sistema operacional e data de lançamento demandam

este serviço de formatação pois são recuperados dos sites de maneira não padronizada para agregação da base de dados. Portanto, alguns padrões foram definidos e este componente é responsável pela aplicação deles. Além destes, os atributos suporte a NFC, dual chip, LTE são agregados na base de dados com os valores *SIM* para caso positivo ao suporte e *NAO* caso contrário. Entretanto, esta informação não é puramente recuperada dos sites. Normalmente, esses três últimos atributos citados são apresentados nos sites em meio a outros atributos ou em um campo para características gerais. Portanto, o *Agregador* procura por palavras chaves na informação recuperada, para certificar se o suporte existe.

A segunda tarefa, denominada comparação, analisa cada atributo para comparar se os valores recuperados dos diferentes sites (*Kimovil* e *Phone Arena*) são iguais. Caso sejam idênticos, qualquer um pode ser utilizado para compor a base de dados. Caso contrário, o componente decide qual atributo é mais viável. Para essa decisão, o *Agregador* observa a discrepância entre os atributos numéricos, baseado em uma regra para cada atributo. Se um atributo possuir valores muito discrepantes para cada site, dentro desta regra, ele escolhe o atributo do *Kimovil*, por possuir uma base de dados mais robusta. Um exemplo para este caso seria a bateria. Se para um determinado aparelho a capacidade da bateria colhida a partir do *Phone Arena* possuir uma suposta diferença de 1000 *mAh* para a do *Kimovil*, o atributo escolhido será o do segundo site, pelo motivo citado anteriormente. Caso a diferença não seja relevante, dentro da regra para cada atributo, é escolhido, por convenção, o de menor valor. Para os atributos não numéricos que se diferem na comparação, são agregados a base de dados os do site *Kimovil*, justamente por possuir maior robustez. Essas decisões são repassadas ao componente *Logger* mostrando os valores para cada site e ressaltando qual foi escolhido.

A terceira tarefa do componente *Agregador* diz respeito a gravar os dados já formatados e selecionados na base de dados. Ou seja, ele é o responsável por agregar os 22 atributos recuperados pelo *crawler*, para cada linha, que refere a um aparelho. Esses atributos são adicionados a um arquivo resultante de formato CSV (*Comma-separated values*). Este formato se refere a um arquivo de texto que separa as informações por vírgula, tornando-o mais leve e de fácil manipulação.

3.4. Logger

Este último componente é responsável pela criação e atualização de um arquivo texto contendo todos os registros do processo. Cada decisão importante tomada pelo *CrawMobi* é registrada em um arquivo. Cada componente anteriormente descrito, repassa suas decisões para o componente *Logger*, como representado na Figura 1. A partir desses dados recebidos, o arquivo em questão é composto.

Ao iniciar a execução da aplicação, ele cria e preenche o cabeçalho do arquivo, denominado *logs.txt*. Com o decorrer do fluxo de dados, este componente vai agregando informações a este arquivo. Vindo dos componentes *Crawler de Informações* e *Crawler de Preço*, ele registra se o aparelho não foi encontrado nos sites pesquisados. Já o *Agregador* exige muito do *Logger*. Como explicado na seção 3.3, a cada divergência na comparação dos valores, o *CrawMobi* decide qual será agregado no arquivo resultante. Cada decisão desta é registrada no arquivo *logs.txt*, contendo o nome do aparelho, atributo e site envolvido. Com isso, o usuário poderá verificar se os valores escolhidos em

caso de conflito foram os mais apropriados ou não, alterando-os manualmente caso ache conveniente.

Depois da execução para todos os aparelhos listados no arquivo de entrada pelo cliente, o *CrawMobi* cria um arquivo compactado de extensão ZIP, denominado *crawMobi.zip*, contendo a base de dados (*handsets.csv*) e o arquivo log (*logs.txt*). Por fim, é retornado ao cliente o arquivo *crawMobi.zip*.

Com o intuito de medir a complexidade do *CrawMobi*, foi realizada uma análise assintótica. Assim foi percebido que a aplicação em questão possui complexidade linear, uma vez que o tempo de execução cresce constantemente de acordo com a entrada. Entretanto, como relatado anteriormente, o *CrawMobi* utiliza o *webdriver* do navegador *Google Chrome*. Portanto o tempo de execução, assim como o gasto de CPU e memória RAM advindos do uso deste navegador, devem ser acrescidos aos gastos da aplicação.

O tempo de execução do *CrawMobi* não depende somente da entrada mas principalmente do número de sites utilizados para a coleta dos dados. Cada um destes é acessado durante a execução do programa, para cada entrada. Assim, com mais sites teremos uma base de dados mais robusta porém, a complexidade do sistema pode se comportar de forma quadrática. Além deste aumento da complexidade, causado pelo navegador, o *webdiver* é totalmente dependente de rede. Uma conexão ruim faz com que o tempo de resposta do *CrawMobi* aumente indefinidamente.

4. Testes

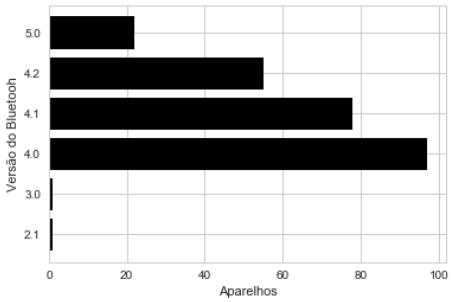
Com o intuito de testar o *CrawMobi*, foram listados 450 nomes de aparelhos que foram submetidos ao *Web Service*. Com a base de dados retornada, foram aplicadas algumas técnicas de análises de dados com o intuito de gerar informações válidas a partir dos smartphones, simulando possíveis situações para empresas de serviços móveis. As análises foram feitas utilizando a linguagem *Python* na versão 3.6, *Jupyter Notebook* (aplicação web para edição de documentos voltada para ciência de dados) e a *Pandas* (plataforma de ciência de dados).

A Tabela 2 e a Figura 2 mostram algumas informações básicas sobre os dados obtidos.

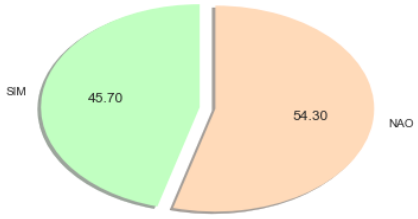
Tabela 2. Descrições e informações básicas sobre o conjunto de dados

	Média	**Desvio Padrão**	**Min**	**25%**	**50%**	**75%**	**Max**
Bateria (mAh)	2811.43	725.40	1030	2300	2840	3120	5300
RAM (GB)	2.40	1.37	0.5	1	2	3	8
Armazenamento (GB)	42.21	54.35	4	16	32	40	256
Câmera (Mpx)	12.09	4.59	2	8	13	13	23
Peso (g)	154.88	30.18	88	140	152	167	495
Tela (”)	5.19	0.56	3	5	5.2	5.5	9.6
Nota do Site	8.70	0.94	6	8.37	9	9.3	10
Nota dos Usuários	8.65	0.99	4.9	8.5	9	9.3	9.7
Preço (R$)	1211.53	898.96	242.00	603.93	884.50	1532.05	5691.92

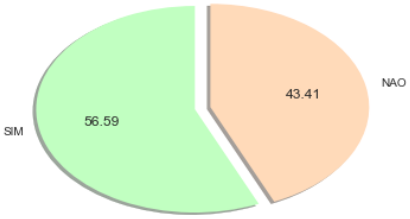
Na Tabela 2, na primeira linha, os atributos significam, respectivamente: Média, desvio padrão, valor mínimo, 25º percentil (1º quartil), 50º percentil (mediana), 75º percentil e valor máximo. Os atributos dos aparelhos avaliados foram: Capacidade da bateria (mAh), memória RAM (GB), memória de armazenamento(GB), resolução da câmera



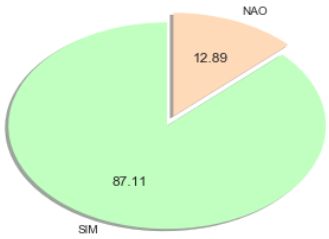
(a) Aparelhos por Versão de Bluetooh



(b) Aparelho com suporte a NFC



(c) Aparelho com suporte a Dual Chip



(d) Aparelho com suporte a LTE (4G)

Figura 2. Conhecendo os dados

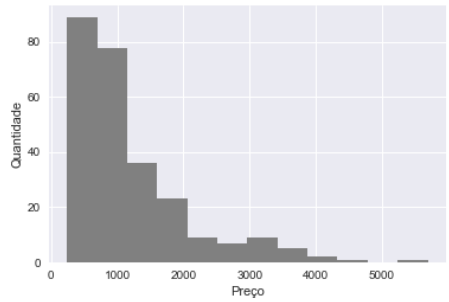
(Mpx), Peso (g), tamanho da tela (polegadas), avaliação do site, avaliação dos usuários e preço (R$).

Observando os dados da Tabela 2, fica fácil conhecer, por exemplo, os valores medianos dos atributos dos aparelhos desejados. Isso seria vantajoso para uma organização pois ela pode conhecer as preferências de seus clientes, sabendo especificamente os recursos que eles possuem. Analisando a Figura 2*(a)*, podemos observar que a maioria dos aparelhos desta coleção possuem o atributo *Bluetooh* nas versões *4*. Como mostram os gráficos de pizza, os aparelhos ficam bem divididos em relação ao suporte da tecnologia *NFC* e ao *Dual chip*, sendo que a maioria possui tecnologia *4G*.

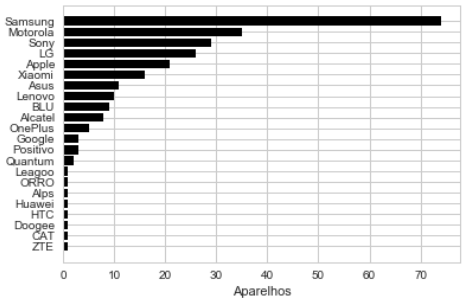
Além de fatores genéricos, é possível examinar detalhadamente alguns atributos, os correlacionando para extrair informações mais específicas. As Figuras 3 e 4 demonstram algumas dentre diferentes possibilidades para estas análises.

Analisando a Figura 3*(a)* pode-se perceber que a maioria dos aparelhos custam até aproximadamente *R$1000,00*. Assim, é possível inferir um perfil socio-econômico a partir deste conjunto de aparelhos. Na Figura 3*(b)* temos a quantidade de aparelhos por marca. Observa-se uma quantidade alta de aparelhos da *Samsung* devido a grande quantidade de versões para cada modelo. Com o gráfico da letra *(c)* podemos perceber que as marcas *Alps, Alcatel, Leagoo* e *Positivo* possuem aparelhos com médias de preços inferiores a *R$500,00*, sendo a última possuindo a média mais baixa. De forma oposta, a *Apple* e *Google* possuem as maiores médias de preços.

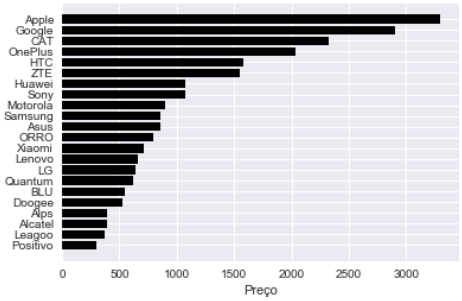
Por possuir uma média baixa de preço, podemos observar consequentemente na



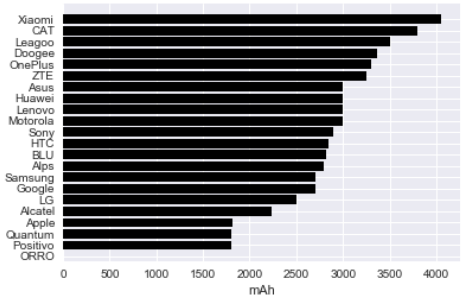
(a) Preço por Quantidade



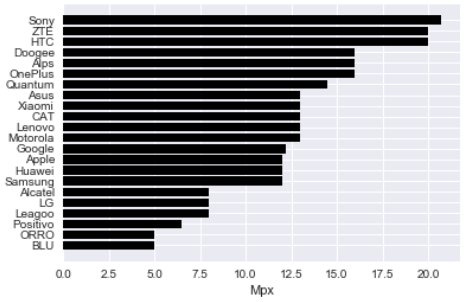
(b) Aparelhos por Marca



(c) Preço Médio por Marca



(d) Capacidade Média de Bateria por Marca

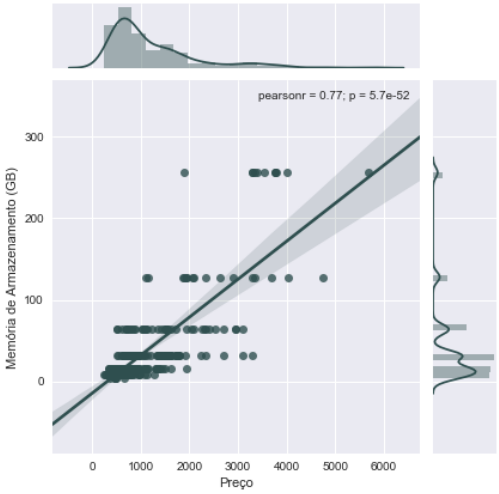


(e) Resolução Média da Câmera por Marca

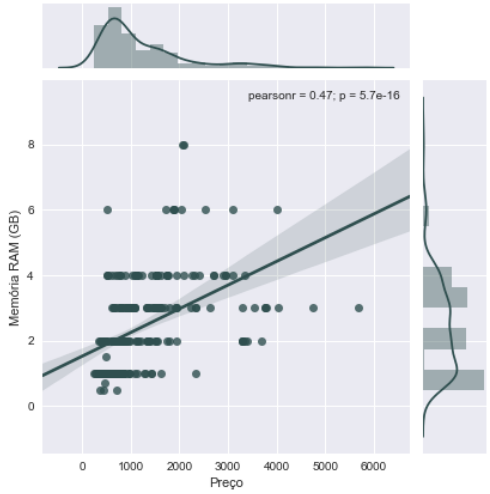
Figura 3. Análises dos dados

Figura 3*(d)* a baixa média de capacidade de bateria oferecida pela marca *Positivo* e outras de baixo valor. Já a marca *Xiaomi* se destaca neste quesito, mesmo possuindo uma média de preço relativamente baixa. Já na letra *(e)* observamos que a *Sony* possui alta resolução média de câmera, sendo um dos pontos fortes da marca. Novamente, algumas marcas de baixa média de preço podem ser observadas com os mais baixos recursos na câmera.

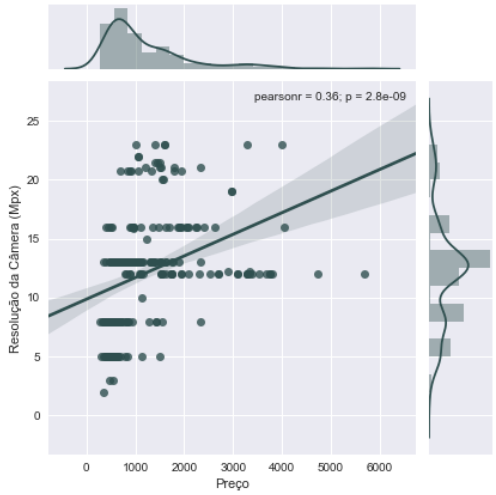
Na Figura 4 temos correlações de alguns atributos com o preço. A partir destas é possível observar a dependência do preço de um aparelho baseado em um destes atributos. Assim, é possível verificar o quão um atributo influencia em outro, com base no coeficiente angular da reta. Podemos observar que o atributo *Memória de Armazenamento*, dentre os mostrados, é o que mais impacta no preço. Aparelhos com maior memória de armazenamento tendem a ser mais caros. Por outro lado, o tamanho da tela não afeta muito significativamente o preço.



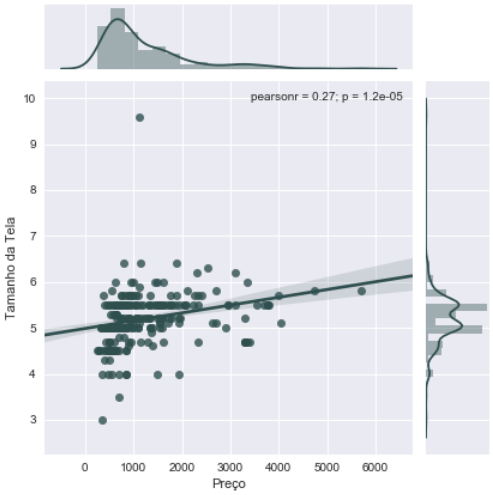
(a) Preço por Memória de Armazenamento



(b) Preço por Memória RAM



(c) Preço por Resolução da Câmera



(d) Preço por Tamanho da Tela

Figura 4. Correlações de Atributos com o Preço

Além destas análises mostradas, muitas outras podem ser elaboradas explorando a fundo todos os atributos oferecidos pelo *CrawMobi* para assim uma organização descrever precisamente o perfil de seus clientes com o intuito de oferecer melhores serviços.

5. Conclusão e Trabalhos Futuros

Neste trabalho, foi apresentado um *Web Service* para coleta de dados referentes a smartphones, com o intuito de coletar e organizar dados coesos para que uma organização conheça melhor seus clientes. A partir de *crawlers* para coleta de informações em sites confiáveis, foi possível a obtenção de informações importantes. Com a criação de um componente formatador e agregador de dados, foi possível a preparação e organização dos mesmos. Finalmente, somando estes serviços pode ser criada uma boa base de dados, de acordo com os aparelhos desejados.

Os testes efetuados com 450 aparelhos mostram que inúmeras análises dos dados podem ser efetuadas de acordo com os objetivos e hipóteses levantadas. Assim, pode-se traçar um perfil para o usuário móvel de acordo com seu smartphone.

Trabalhos futuros incluem a otimização do *crawler* para que o tempo de resposta da aplicação seja menor. Nesta versão, é utilizado o *webdriver* do *Google Chrome* e a ferramenta *selenium*, o que deixa a coleta de informações relativamente lenta e totalmente dependente de uma conexão com a rede. Os resultados obtidos pelo *CrawMobi* para os aparelhos, também serão atrelados a dados de geo-localização de seus usuários, para melhor inferência dos perfis.

Referências

Arena, P. (2018). Phone arena. Disponível em: `https://www.phonearena.com/`. Acesso em: 18 nov. 2018.

Elmasri, R. and Navathe, S. (2010). *Fundamentals of Database Systems*. Number 6 in Pearson Education, Inc. Addison-Wesley.

H. Falaki, R. Mahajan, S. K. D. L. R. G. and Estrin, D. (2010). Diversity in smartphone usage.

Kimovil (2018). Kimovil. Disponível em: `https://www.kimovil.com/pt/`. Acesso em: 18 nov. 2018.

Malmi, E. and Weber, I. (2016). You are what apps you use: Demographic prediction based on user's apps.

Oliveira, R. A. (2018). Security benchmarking for web service frameworks.

Rahmati, A. and Zhong, L. (2013). Studying smartphone usage: Lessons from a four-month field study.

Ronacher, A. (2010 - 2018). Flask. Disponível em: `http://flask.pocoo.org/`. Acesso em: 13 nov. 2018.

Rondon, T. (2010). Arquitetura rest e o serviço web 'restful'. Disponível em: `http://sao-paulo.pm.org/pub/arquitetura-rest-e-o-servico-web-restful-`. Acesso em: 13 nov. 2018.

Salton, G. (1968). *Automatic information organization and retrieval*. McGraw-Hill.

Sampaio, L. F. and Silva, K. A. (2017). Desenvolvimento de um web service baseado em rest para a interligação de dados entre uma aplicação mobile e um portal web.

Samú, P. H. (2018). Pacote de consulta a dados censitários do censo demográfico de 2010 do ibge utilizando linguagem python.

Santos, V. (2016). O que é análise de dados? como estruturar a sua? Disponível em: https://www.fm2s.com.br/analise-de-dados-como-estruturar/. Acesso em: 14 nov. 2018.