Francisco Augusto Cesar de Camargo Bellaz Guiraldelli

Identificação de Perfis de Permissões em Aplicativos Móveis Utilizando Agrupamento e Visualização

Sorocaba, SP

12 de Dezembro de 2019

Francisco Augusto Cesar de Camargo Bellaz Guiraldelli

Identificação de Perfis de Permissões em Aplicativos Móveis Utilizando Agrupamento e Visualização

Dissertação de mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação (PPGCC-So) da Universidade Federal de São Carlos como parte dos requisitos exigidos para a obtenção do título de Mestre em Ciência da Computação. Linha de pesquisa: Inteligência Artificial, Aprendizado de Máquina, Agrupamento.

Universidade Federal de São Carlos – UFSCar

Centro de Ciências em Gestão e Tecnologia – CCGT

Programa de Pós-Graduação em Ciência da Computação – PPGCC-So

Orientador(a): Profa. Dra. Katti Faceli

Sorocaba, SP

12 de Dezembro de 2019

de Camargo Bellaz Guiraldelli, Francisco Augusto Cesar

Identificação de Perfis de Permissões em Aplicativos Móveis Utilizando Agrupamento e Visualização / Francisco Augusto Cesar de Camargo Bellaz Guiraldelli. -- 2019.
106 f. : 30 cm.

Dissertação (mestrado)-Universidade Federal de São Carlos, campus Sorocaba, Sorocaba
Orientador: Katti Faceli
Banca examinadora: Katti Faceli, Alexandre Alvaro, Solange Oliveira Rezende
Bibliografia

1. Inteligência Artificial. 2. Análise de Agrupamentos. 3. Visualização. I. Orientador. II. Universidade Federal de São Carlos. III. Título.

Ficha catalográfica elaborada pelo Programa de Geração Automática da Secretaria Geral de Informática (SIn).
DADOS FORNECIDOS PELO(A) AUTOR(A)
Bibliotecário(a) Responsável: Maria Aparecida de Lourdes Mariano – CRB/8 6979

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências em Gestão e Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Francisco Augusto Cesar de Camargo Bellaz Guiraldelli, realizada em 12/12/2019:

Profa. Dra. Katti Faceli
UFSCar

Profa. Dra. Solange Oliveira Rezende
USP

Prof. Dr. Alexandre Alvaro
UFSCar

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Solange Oliveira Rezende e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

Profa. Dra. Katti Faceli

Dedico aqui esse trabalho às mulheres
mais importantes da minha vida:
minha adorada mãe Maria Helena;
minha amada esposa Katjy;
e minhas queridas filhas
Francesca e Fernanda.

Agradecimentos

Agradeço,

à Profa. Dra. Katti Faceli pelo longo tempo dedicado a mim como orientadora, sempre prestativa, mostrando-se sempre bom empenho e dedicação, demonstrando dessa forma a ótima profissional e pessoa de bom coração que é;

ao meu irmão Ricardo por sua dedicação em me ajudar na fórmula matemática com os grupos consenso;

ao meu amigo Arthur Fortes da Costa por compartilhar comigo sua implementação do algoritmo PAM e sua paciência e disposição em tirar minhas dúvidas sobre Jupyter Notebook, NumPy e suas variantes;

a todos os docentes, técnicos e pessoas na Universidade Federal de São Carlos campus Sorocaba que me ajudaram e me deram forças para eu concluir mais essa etapa na minha vida;

à minha família, pela paciência, compreensão, amor e força de vontade que me deram para eu nunca desistir;

"A imaginação é mais importante que o conhecimento.
O conhecimento é limitado, enquanto a imaginação
abraça o mundo inteiro, estimulando o progresso,
dando à luz à evolução."
(Albert Einstein)

Resumo

Na última década, aplicações móveis conquistaram ampla utilização com a popularização dos *smartphones*, transformando radicalmente a maneira como as pessoas acessam e utilizam os meios de informação, trazendo a internet e muitas tarefas do cotidiano na palma da mão. Tal facilidade porém, trouxe consigo novos desafios, como a necessidade de dispositivos com alta eficiência energética, boa capacidade de processamento, desempenho, boa forma ergonômica e, que sejam leves e práticos no seu manuseio. Apesar de grandes esforços despendidos por todas essas empresas, alguns problemas sobre classificação de aplicações móveis estão no topo das muitas preocupações desses desenvolvedores, pois defeitos de implementação, falta do conhecimento do domínio ou até mesmo na preparação para funcionalidades futuras, as aplicações móveis podem consumir recursos de *hardware* alocados incorretamente tanto para a necessidade do usuário da aplicação, quanto para o domínio para ao qual ela foi desenvolvida. Com foco na identificação de perfis de permissões, na correta utilização de recursos e na forma como são implementados os códigos desses aplicativos, esta pesquisa propõem um método que envolve coleta de dados da *Google Play Store*, técnicas de agrupamento e visualização, o que proporcionará aos fabricantes de dispositivos móveis, desenvolvedores de aplicativos móveis e pesquisadores uma forma simples de comparar e analisar aplicativos que utilizam recursos semelhantes. Para avaliar o método, foi realizado um estudo de caso relacionado com o consumo de energético dos aplicativos móveis que comprova sua eficácia nesse tipo de análise.

Palavras-chaves: Perfil de Permissão. Aplicativos móveis. Análise de agrupamento. Ensembles de agrupamento. Grupos consenso. Visualização.

Abstract

In the last decade, mobile applications have gained widespread use through smartphones, it has radically transformed the way people can access and use the media, bringing the internet and many everyday tasks in the palm of their hands. Such ease, however, brought with it new challenges, such as the need for devices with high energy efficiency, good processing capacity, performance, good ergonomic form, lightweight and easy handling. Despite the great efforts expended by all of these companies, some issues with mobile application classification are at the top of many developers' concerns, as implementation defects, lack of domain knowledge or even preparation for future functionalities can cause mobile applications to consume hardware resources incorrectly allocated to both the application user's need and the domain for which it was developed. Focusing on the identification of permissions' profiles, the correct use of resources and the way the application is implemented, this research proposes a method that involves *Google Play Store* data collection, to use techniques like clustering and visualization, which will provide to device manufacturers, mobile application developers, and researchers a simple way to compare and analyze mobile applications that use similar features. To evaluate the method, a case study related to the energy consumption of mobile applications was conducted, which proves its effectiveness in this type of analysis.

Key-words: Profile of permission. Mobile applications. Clustering. Cluster ensembles. Consensus groups. Visualization.

Lista de ilustrações

Figura 1 – A pilha de componentes do sistema operacional Android 20
Figura 2 – Interpretação geométrica da similaridade por cosseno e correlação de Pearson 25
Figura 3 – Método proposto para a análise de perfis de permissão 36
Figura 4 – Coleta de dados 38
Figura 5 – Identificação dos perfis de permissão 41
Figura 6 – Visualização 47
Figura 7 – Todos os grupos de consenso gerados 49
Figura 8 – Grupos de consenso com somente dois aplicativos 49
Figura 9 – Grupos de consenso com três e cinco aplicativos 50
Figura 10 – Gráfico de pesos das permissões 52
Figura 11 – Visualização dos pesos das permissões em sua forma completa . 53
Figura 12 – Visualização dos pesos das permissões com grupos de consenso com 2 aplicativos 54
Figura 13 – Visualização dos pesos das permissões com grupos de consenso com 3 e 5 aplicativos 54
Figura 14 – Visualização de peso das permissões com parâmetro de representatividade igual a 10% 55
Figura 15 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 25% 55
Figura 16 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 50% 55
Figura 17 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 75% 56
Figura 18 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 100% 56
Figura 19 – Distribuição dos aplicativos no arquivo de 4752 linhas 64
Figura 20 – Distribuição dos aplicativos selecionados para as categorias . . . 65
Figura 21 – Bloco H-41 do gráfico de permissões da categoria H 67

Figura 22 – Visualização dos pesos das permissões da categoria H. 68
Figura 23 – Gráfico conjunto dos perfis de permissão dos grupos A-20, E-32 e F-44 71
Figura 24 – Visualização conjunta dos pesos das permissões dos grupos A-20, E-32 e F-44 72
Figura 25 – Recorte do gráfico de perfis de permissão do Grupo A 81
Figura 26 – Recorte do gráfico de perfis de permissão do Grupo B 82
Figura 27 – Recorte do gráfico de perfis de permissão do Grupo C 83
Figura 28 – Recorte do gráfico de perfis de permissão do Grupo D 84
Figura 29 – Recorte do gráfico de perfis de permissão do Grupo E 85
Figura 30 – Recorte do gráfico de perfis de permissão do Grupo F 86
Figura 31 – Recorte do gráfico de perfis de permissão do Grupo G 87
Figura 32 – Recorte do gráfico de perfis de permissão do Grupo H 88
Figura 33 – Recorte do gráfico de perfis de permissão do Grupo I 89
Figura 34 – Recorte do gráfico de perfis de permissão do Grupo J 90
Figura 35 – Recorte do gráfico de perfis de permissão do Grupo K 91
Figura 36 – Recorte do gráfico de perfis de permissão do Grupo L 92
Figura 37 – Recorte do gráfico de perfis de permissão do Grupo M 93
Figura 38 – Recorte do gráfico de perfis de permissão do Grupo N 94
Figura 39 – Recorte do gráfico de perfis de permissão do Grupo O 95
Figura 40 – Visualização dos pesos das permissões do Grupo A 97
Figura 41 – Visualização dos pesos das permissões do Grupo B 98
Figura 42 – Visualização dos pesos das permissões do Grupo C 99
Figura 43 – Visualização dos pesos das permissões do Grupo D 100
Figura 44 – Visualização dos pesos das permissões do Grupo E 101
Figura 45 – Visualização dos pesos das permissões do Grupo F 102
Figura 46 – Visualização dos pesos das permissões do Grupo G 103
Figura 47 – Visualização dos pesos das permissões do Grupo H 104
Figura 48 – Visualização dos pesos das permissões do Grupo I 104
Figura 49 – Visualização dos pesos das permissões do Grupo J 105
Figura 50 – Visualização dos pesos das permissões do Grupo K 105
Figura 51 – Visualização dos pesos das permissões do Grupo L 106
Figura 52 – Visualização dos pesos das permissões do Grupo M 106

Figura 53 – Visualização dos pesos das permissões do Grupo N 106
Figura 54 – Visualização dos pesos das permissões do Grupo O 106

Lista de tabelas

Tabela 1 – Tabela inicial em formato csv de aplicativos e permissões 40
Tabela 2 – Tabela binária de aplicativos e permissões 42
Tabela 3 – Tabela binária de aplicativos e permissões com os aplicativos a2, a3, a7, a11, a13 e a14 43
Tabela 4 – Tabela distância cosseno dos aplicativos a2, a3, a7, a11, a13 e a14 44
Tabela 5 – Partições resultantes do algoritmo de agrupamento 45
Tabela 6 – Grupos consenso coloridos. 46
Tabela 7 – Grupos consenso organizados pelas cores. 46
Tabela 8 – Recorte das primeiras colunas da tabela 60
Tabela 9 – Recorte das últimas colunas da tabela 60
Tabela 10 – Recorte da matriz binária 61
Tabela 11 – Distribuição dos aplicativos em grupos de consenso 65
Tabela 12 – Permissões selecionadas que influenciam no consumo de baterias e seus pesos 69
Tabela 13 – Valores mínimos e máximos encontrados nas categorias 70

Sumário

1 INTRODUÇÃO 15

Introdução 15

2 CONCEITOS BÁSICOS 19
2.1 Dispositivos Móveis 19
2.1.1 Sistema Operacional 19
2.1.2 Permissões 21
2.1.3 Baterias de Lítio 21
2.2 Análise de Agrupamento 22
2.2.1 Algoritmos de Agrupamentos 27
2.2.2 *Ensembles* de Agrupamentos 28
2.2.3 Visualização de Agrupamentos 29
2.3 Trabalhos Relacionados 30

3 MÉTODO PROPOSTO 36
3.1 Coleta de dados 37
3.1.1 Seleção dos Aplicativos 37
3.1.2 Extração do nome do pacote *apk* 37
3.1.3 Extração das permissões 39
3.1.4 Resultado final da coleta de dados 40
3.2 Identificação dos perfis de permissão 41
3.2.1 Formatação dos dados coletados 42
3.2.2 Escolha da medida de similaridade ou dissimilaridade 43
3.2.3 Obtenção de um conjunto de partições base 44
3.2.4 Obtenção dos Perfis de Permissão 45
3.3 Visualização 47
3.3.1 Gráficos dos Perfis de Permissões 48
3.3.2 Seleção das permissões, ponderação e representatividade 50
3.3.3 Gráficos de Pesos das Permissões 51

4 APLICAÇÃO DO MÉTODO 57
4.1 Coleta dos dados 57
4.2 Identificação dos Perfis de Permissão 59
4.2.1 Formatação dos dados coletados 59
4.2.2 Cálculo da distância cosseno 62
4.2.3 Obtenção das partições base no algoritmo de agrupamento 62
4.2.4 Obtenção dos grupos consenso 63
4.3 Visualização 66

5 CONCLUSÃO 73

Conclusão 73

Referências 76

APÊNDICE A – GRÁFICOS DE PERFIS DE PERMISSÃO 80

APÊNDICE B – VISUALIZAÇÃO DOS PESOS DAS PERMISSÕES 96

1 Introdução

Na última década, aplicações móveis conquistaram ampla utilização através de *smartphones*, que, a partir de seu lançamento em 2007 pelo *IPhone* da *Apple* e logo em seguida em 2008, com o lançamento do sistema operacional *Android* desenvolvido pela *Google*, transformou radicalmente a maneira como as pessoas acessam e utilizam os meios de informação, trazendo a internet e muitas tarefas do cotidiano na palma da mãos. Entretanto isso traz novos desafios, como a necessidade de dispositivos com alta eficiência energética, boa capacidade de processamento, que tenham forma ergonômica e sejam leves e práticos no seu manuseio.

Assim, a eficiência energética de dispositivos móveis e de suas aplicações (CRUZ; ABREU, 2017) são fatores que norteiam as fabricantes de telefones celulares atualmente, pois, quanto maior a duração da bateria, maior será o tempo de utilização de aplicações e melhor será experiência dos usuários, o que consequentemente gerará maiores vendas e lucros.

Além disso, há diversos estudos feitos na área acadêmica, tanto em pesquisas relacionadas sobre o consumo desses dispositivos (CARROLL; HEISER, 2010; HE et al., 2017a; HE et al., 2017b; RAO et al., 2017), como no desenvolvimento de baterias para eles (PLACKE et al., 2017; KALLURI et al., 2017; LIN; LIU; CUI, 2017), demonstrando que é uma área fértil e promissora para pesquisas futuras.

Usualmente os aplicativos são classificados pelas suas características de negócios ou pela sua função ao facilitar atividades do dia-a-dia. Porém, podem haver aplicativos de uma mesma linha de negócios com comportamentos e funções distintas. Por exemplo, dois aplicativos muito utilizados na categoria transporte e mobilidade urbana são o *Waze* e *Uber*. Apesar de ambos pertencerem a uma mesma categoria, o primeiro tem a função de "navegador", mostrando as melhores rotas em tempo real, auxiliando o motorista a se locomover entre dois pontos da forma mais eficiente possível, evitando congestionamentos, sinalizando sobre radares, etc. Já o segundo, tem a função de oferecer um serviço alternativo aos táxis, localizando motoristas particulares que estejam próximos ao usuário, e da forma mais eficiente

possível, calcular uma rota pré-determinada e mostrar o custo aproximado do deslocamento desse usuário antes mesmo dele estar dentro do veículo que o levará.

Ambos os aplicativos utilizam sensores como GPS, redes de comunicação *WiFi* e 4G, além de recursos disponíveis do próprio *framework* do Android para triangular a posição de seus usuários de forma mais eficiente e precisa. Porém, pode-se observar que, enquanto o aplicativo *Waze* precisa ficar com a tela o tempo todo ligada para o motorista ver qual o melhor trajeto a seguir, o aplicativo *Uber*, após o usuário pedir o serviço de locomoção, não tem a necessidade de manter a tela ligada, porém, precisa manter os sensores de localização ligados.

Os dispositivos podem lidar com as situações citadas acima utilizando métodos de gerenciamento de energia disponibilizados pelo próprio *framework* do Android, denominados por *full wakelock* e *partial wakelock*. Quando um aplicativo utiliza o método de *full wakelock*, isso significa que o celular deve manter o seu processador e tela ativos, o que demanda uma grande quantidade de energia a ser consumida para seu funcionamento. Pesquisas mostram que a tela pode consumir entre 29% a 36% da energia total de um *smartphone* (YOON et al., 2012; CHEN et al., 2013; DUAN et al., 2013), portanto, caso um aplicativo utilize este método incorretamente, pode provocar um consumo exorbitante de energia desnecessariamente. Quando um aplicativo utiliza o método de *partial wakelock*, ele mantém disponíveis os mesmos recursos que o método anterior, com exceção da tela, que será desativada.

De acordo com Cruz (CRUZ; ABREU, 2017), aplicativos para dispositivos móveis são desenvolvidos com orientação a funcionalidades e não com base em seu consumo ou eficiência energética. Apesar de grandes esforços despendidos por todas essas empresas, implementações mal feitas, falta do conhecimento do domínio ou até mesmo falta de planejamento sobre funcionalidades futuras, tornam as aplicações móveis, vilãs no consumo de recursos de *hardware*, muitas vezes alocados incorretamente, tanto para a necessidade do usuário da aplicação, quanto para o domínio para ao qual ela foi desenvolvida.

O entendimento da relação entre os recursos usados pelos aplicativos e o consumo de bateria pode fornecer informações valiosas para fabricantes do

dispositivo, desenvolvedores de aplicativos e pesquisadores, que podem tomar medidas e conduzir pesquisas para melhoria de desempenho e consequentemente, proporcionar uma melhor experiência dos usuários com seus aparelhos.

Uma informação que pode ser útil para esse entendimento é a caracterização de perfis das permissões dos aplicativos. Analisadas de forma correta, as permissões podem ajudar a determinar diversas características, anomalias e funcionalidades que desenvolvedores, fabricantes e pesquisadores podem explorar a fim de melhorar seus produtos ou mesmo descobrir novas formas de analisar o comportamento desses aplicativos. Por exemplo, é possível identificar relações entre as permissões com o desempenho, funcionalidade, comportamento, recursos de *hardware* ou até inferir consumo de energia e gastos de baterias.

Atualmente, não há uma forma oficial de se classificar os aplicativos pelas permissões, assim é preciso criar um método que ajude nessa classificação e que dela possamos obter ferramentas que possam auxiliar na sua análise. Neste trabalho, está sendo proposto um método para identificação e comparação de perfis de permissões que permite classificar e comparar os aplicativos. Um perfil de permissão corresponde ao conjunto de permissões que o aplicativo compartilha com um conjunto significativo de outros aplicativos. Perfil de permissão ou grupo consenso é um conceito novo aqui proposto, baseado nas ideias de *ensembles* de agrupamento. Assim, a ideia do método proposto é integrar a utilização de métodos de agrupamentos, visualização e *ensembles* de agrupamento, para identificar semelhanças dos aplicativos e permitir uma análise comparativa visual que facilita a tarefa dos fabricantes, desenvolvedores e pesquisadores que trabalham com dispositivos móveis e/ou aplicativos.

Para ilustrar o método será apresentado um estudo de caso que analisa o consumo de energia dos aplicativos, passando por todas as etapas do método proposto. Da coleta e tratamento desses dados, passando pela identificação dos perfis de permissão e finalizando com a visualização desses perfis e da análise final.

Este documento é organizado em cinco capítulos principais, composto pela introdução, conceitos básicos, apresentação do método proposto, aplicação e validação do método e finalmente a conclusão e trabalhos futuros.

O Capítulo 1 introduz o contexto do trabalho, objetivos, motivações e a estrutura do texto. O Capítulo 2 refere-se à revisão da literatura dos conceitos básicos para a compreensão e execução do método proposto, falando dos dispositivos móveis, da análise de agrupamentos e finalmente dos trabalhos relacionados que ajudaram na idealização e aplicação deste trabalho. No Capítulo 3 é apresentado o método proposto de forma genérica, tentando assim, mostrar o funcionamento teórico do método e como ele pode ser aplicado. No Capítulo 4 é feita a aplicação e a validação do método proposto relacionando-o ao consumo de energia em relação aos aplicativos móveis. Finalmente, no Capítulo 5 são apresentadas as conclusões do trabalho, suas principais contribuições, limitações e trabalhos futuros.

2 Conceitos Básicos

Este capítulo é composto pela base teórica de conhecimento inicial necessária para o desenvolvimento das ideias e procedimentos necessários para a construção e o desenvolvimento do método proposto neste estudo. Aqui estão contidos os artefatos que dão base para o desenvolvimento deste estudo, iniciando com um introdução do sistema operacional Android e de seu projeto de código aberto AOSP (do inglês *Android Open Source Project*), passando por uma breve introdução às permissões do Android, logo após, uma breve introdução sobre as baterias de lítio e suas características, chegando então nos algoritmos de agrupamento e nos *ensembles* de agrupamento, para finalmente, apresentar os trabalhos relacionados que deram a motivação para este estudo e que foram chave para a aplicação do método proposto.

2.1 Dispositivos Móveis

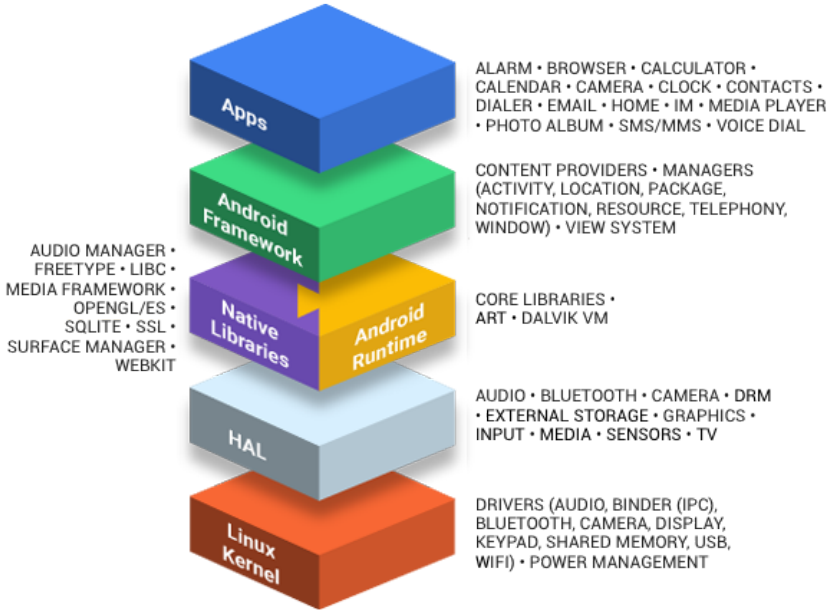
Os dispositivos móveis transformaram radicalmente a maneira como as pessoas acessam e utilizam os meios de informação, acessam internet, resolvem tarefas do cotidiano, trabalham e socializam. Segundo o relatório de 2019 do sítio *We are social*, da população mundial, 67% possuem dispositivo móvel, 52% acessam a internet e 42% usam as redes sociais somente utilizando esses dispositivos móveis.

2.1.1 Sistema Operacional

Android é um sistema operacional baseado em Linux desenvolvido inicialmente para dispositivos com telas de toque como celulares e *tablets*. Atualmente é o sistema operacional de dispositivos móveis mais popular, correspondendo a 76,67% do total desses dispositivos segundo o sítio *StatCounter Global Stats*. Sua versão 1.0 foi lançada no dia 23 de Setembro de 2008.

O sistema operacional pode ser compreendido como uma pilha de componentes descritos e enumerados da camada mais alta para a mais baixa enumerado abaixo e ilustradas na Figura 1.

1. Aplicativos do Sistema Operacional;
2. Java API Framework;
3. Bibliotecas nativas C/C++ e *Android Runtime*;
4. Camada de abstração de *hardware*;
5. Kernel do Linux.



Fonte: ***<https://source.android.com/setup>***

Figura 1 – A pilha de componentes do sistema operacional Android

O projeto AOSP (do inglês *Android Open Source Project*) é o projeto oficial de desenvolvimento do sistema operacional, a fonte de informações e o repositório oficial liderado pela *Google* e empresas que vão desde operadoras de rede celular, a fabricantes de semicondutores, fabricantes de dispositivos móveis e empresas de *software*.

Atualmente o projeto AOSP se encontra na sua versão 29[1] e nela estão contidas as descrições e implementações das permissões que são objetos de estudo dessa pesquisa.

[1] <https://source.android.com/setup/start/build-numbers>

2.1.2 Permissões

As permissões fazem parte do Android e servem para liberar a utilização de recursos que são providos pelo sistema operacional. Esses recursos fazem parte tanto do *hardware* tanto quanto de funcionalidades que o *framework* dispõem. Dessa forma, quando o aplicativo precisa acessar um recurso de rede como o *Wifi* pode-se utilizar a permissão `ACCESS_WIFI_STATE` ou a permissão `ACCESS_NETWORK_STATE` que dá acesso ao *Wifi*, *WiMax* e modems.

Permissões que acessam tela, serviços de rede, áudio e processador, consomem a maior parte dos recursos energéticos – *hardware* e bateria, e portanto, devem ser usados com parcimônia.

Para esse estudo foram vericados a existência de mais de duas mil permissões, porém muitas delas eram criadas por fabricantes de celulares ou relacionadas às *push notifications*, que são mensagens enviadas ao celular como forma de notificação ou publicidade, utilizadas somente pelos aplicativos individualmente e portanto, excluídas da análise desse estudo, focando-se então em 456[2] permissões principais utilizadas no projeto AOSP.

2.1.3 Baterias de Lítio

Baterias de lítio são baterias recarregáveis, com alta densidade de energia, alta tensão, baixo nível de auto-descarregamento e trabalha em uma ampla faixa de temperatura (FRASER; HACKER; KORDESCH, 2009). Essas características a tornam a bateria mais utilizada comercialmente, principalmente em dispositivos eletrônicos como notebooks e celulares.

Como a bateria de lítio é uma bateria recarregável, ela tem uma vida útil composta por ciclos de recarga. Portanto, há uma relação inversamente proporcional entre ciclos de recarga e perda de capacidade da bateria, onde esta perda pode ser classificada em reversível e irreversível. A primeira é referente a sua auto-descarga e pode ser recuperada recarregando-a novamente. Já a segunda está intimamente ligada com a degradação da bateria, o que, para produtos eletrônicos é aceitável

[2] <https://github.com/aosp-mirror/platform_frameworks_base/blob/master/core/res/AndroidManifest.xml>

até uma perda aproximada de 20%, acima desse valor a bateria chega ao fim de sua vida útil. (SPOTNITZ, 2003)

2.2 Análise de Agrupamento

Problemas de agrupamento são extensivamente estudados em mineração de dados e aprendizado de máquina, graças a suas inúmeras aplicações em sumarização, aprendizado, segmentação, marketing, mídias sociais entre tantas outras áreas. Na ausência de informações rotuladas, o agrupamento pode ser considerado um modelo conciso de dados que podem ser interpretados no sentido de um modelo resumido ou generativo. Dessa forma, podemos definir um problema de agrupamento, numa tradução livre, da seguinte forma (TAN et al., 2013):

> A análise de agrupamentos, agrupa objetos de dados com base somente nas informações encontradas nos dados que descrevem os objetos e seus relacionamentos. O objetivo é que os objetos dentro de um grupo sejam semelhantes (ou relacionados) entre si e diferentes de (ou não relacionados com) os objetos em outros grupos. Quanto maior a semelhança (ou homogeneidade) dentro de um grupo e maior a diferença entre os grupos, melhor ou mais distinto é o agrupamento.

Apesar da definição dada acima, existem vários tipos de grupos descritos e utilizados em diferentes algoritmos na literatura, o que implica que não há uma definição formal e única para este conceito. Porém, para uma melhor compreensão, podemos citar algumas definições (FACELI et al., 2011a; BARBARA, 2000):

Grupo bem separado um conjunto de pontos tal que o ponto contido num certo agrupamento, está mais próximo de outros pontos desse agrupamento, do que outros ao qual não pertence.

Grupo baseado em centro um conjunto de pontos tal que o ponto contido num certo agrupamento, está mais próximo do centro desse agrupamento, do que outros ao qual não pertence.

Grupo contínuo ou encadeado um conjunto de pontos tal que o ponto contido num certo agrupamento, está mais próximo dos pontos desse agrupamento, do que outros ao qual não pertence.

Grupo baseado em densidade um agrupamento é uma região densa de pontos que está separado de outros agrupamentos por uma região de baixa densidade.

Grupo baseado em similidade um agrupamento é uma região de pontos semelhantes, enquanto pontos que não pertencem ao agrupamento são diferentes.

Cada possível definição de agrupamento resulta em um critério de agrupamento, que além de ser a representação de um algoritmo de agrupamento, está ligado a avaliação dos resultados desse algoritmo, podendo ser agrupados em 3 grandes categorias (FACELI et al., 2011a): (a) compactação; (b) encadeamento ou ligação e; (c) separação espacial.

Além disso, devemos ter em mente que há um grande número de algoritmos de agrupamento e para utilizá-los é necessários que se tenha conhecimento sobre a análise de agrupamento e suas etapas,tais como preparação dos dados, proximidade, agrupamento, validação e interpretação, descritas a seguir (FACELI et al., 2011a):

Preparação dos dados é composta por aspectos relacionados ao pré-processamento que pode incluir normalização, conversão de tipos e redução de atributos, extração de características e além disso, é preciso preocupar-se com a representação apropriada a ser utilizada por um algoritmo de agrupamento.

Proximidade nessa etapa, é onde se define as medidas de proximidade apropriadas ao domínio da aplicação e a informação que deseja extrair dos dados.

Medidas de similaridade e dissimilaridade (distância) são bases para os algoritmos de agrupamento. Para dados quantitativos a distância é normalmente utilizada para reconhecer o relacionamento entre os dados, enquanto a similaridade é normalmente utilizada para lidar com dados qualitativos. (XU; TIAN, 2015)

Existem medidas apropriadas para diferentes tipos de atributos. Para esse trabalho as medidas mais relevantes são os atributos quantitativos. Apesar

de existirem outros tipos de atributos como os qualitativos e o heterogêneos, estes não fazer parte deste estudo.

Em relação ao atributos quantitavos, há medidas de distâncias para atributos contínuos e racionais como: (i) baseadas na métrica de Minkowski, (ii) distância euclidiana, (iii) distância de Manhattan e (iv) distância *supremum*. Assim como para atributos binários como: (i) distância de Hamming, (ii) coeficiente de casamento simples e (iii) coeficiente de Jaccard. É importante frisar aqui duas formas muito utilizadas para a avaliar a similaridade entre pares de objetos que são dadas pelo valor absoluto das medidas de separação angular (cosseno) e a correlação de Pearson. A utilização delas a grosso modo, servem para entendermos se as relações entre os objetos são diretamente ou inversamente proporcionais. A correlação Pearson, descrita frequentemente como uma medida forma, por ser insensível na diferença de magnitude dos atributos, é muito usando para determinar a similaridade, em que apenas o padrão de variação dos atributos dos objetos é importante, como por exemplo, nas áreas de Bioinformática, dada pela Equação 2.1.

$$pearson(uv) = \frac{\sum_{i=1}^{n}(u_i - \overline{u})(v_i - \overline{v})}{\sqrt{\sum_{i=1}^{n}(u_i - \overline{u})^2}\sqrt{\sum_{i=1}^{n}(v_i - \overline{v})^2}} \tag{2.1}$$

Para esta pesquisa vale frisarmos a utilização da distância por cosseno, onde será utilizada mais a frente como base para o algoritmo de agrupamento, e formalmente, podemos defini-la como:

$$dist\hat{a}ncia\ cosseno = 1 - \cos\Theta \tag{2.2}$$

onde:

$$\cos\Theta = \frac{u \cdot v}{\|u\|\,\|v\|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}} \tag{2.3}$$

Considerando u_i e v_i dois vetores no espaço d-dimensional, a separação angular por cosseno e a correlação de Pearson podem ser interpretadas como o cosseno

dos ângulos dos vetores. A Figura 2 ilustra num espaço bidimensional, onde o cosseno de α se refere a similaridade por cosseno e o cosseno de β corresponde a correlação de Pearson.

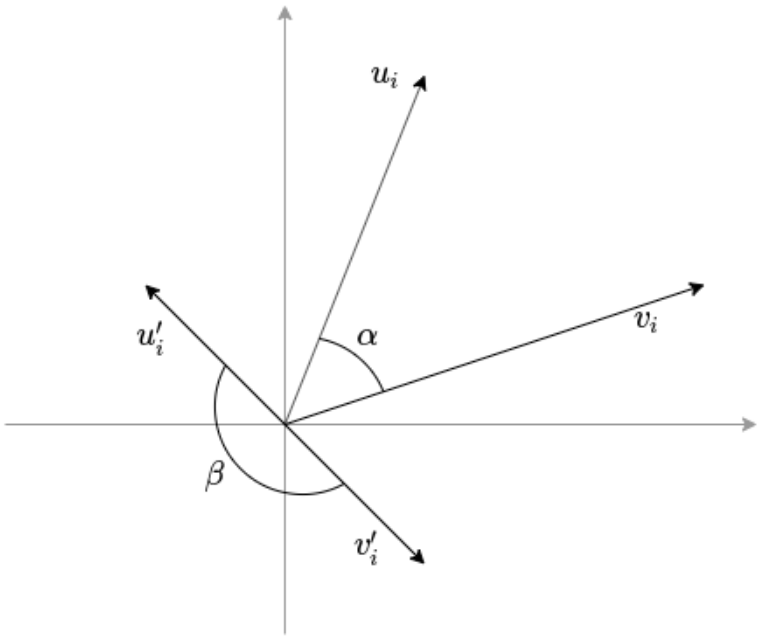


Figura 2 – Interpretação geométrica da similaridade por cosseno e correlação de Pearson

Agrupamento é a etapa central na análise de agrupamento, onde diferentes estruturas de agrupamento podem ser encontradas nos dados, tais como, partições, hierarquias de partições e partições *fuzzy*. Essa etapa é a que consiste da aplicação de um ou mais algoritmos de agrupamento dos dados. Tais algoritmos podem ser categorizados de diversas maneiras, tais como em relação ao tipo de estruturas que podem ser encontradas, como por exemplo, agrupamento exclusivo × não exclusivo e agrupamento hierárquico × particional. Neste trabalho, será considerado algoritmos que identificam partições nos dados. Formalmente, a definição de agrupamento é dado conforme descrito abaixo:

Dado um conjunto de objetos $X = \{x_1, x_2, \ldots, x_n\}$ onde x_i é o i-ésimo objeto e n é o número total de objetos de X, uma partição de X em K^i grupos ou

clusters pode ser definida por:

$$\pi^i = \{C_1^i, C_2^i, \ldots, C_{K^i}^i\} \quad \text{com} \quad K^i < n \tag{2.4}$$

onde C_j^i é o j-ésimo grupo ou *cluster* e K^i é o número total de grupos ou *cluster* em π^i, tal que:

(1) $C_j^i \neq \emptyset,\ j = 1, \ldots, K^i$,

(2) $\bigcup_{j=1}^{K^i} C_j^i = X$ e

(3) $C_j^i \cap C_\ell^i \neq \emptyset,\ \ell = 1, \ldots, K^i \quad \text{e} \quad j \neq \ell$.

Validação é a etapa que avalia o resultado de um agrupamento e determina se resultado é ou não significativo. Levando em conta a complexidade e importância desta etapa, há que se ater, de forma sintetizada, nas seguintes proposições: (i) a maior parte das abordagens de agrupamento, tem como principal problema, a geração de diferentes agrupamentos partindo de um único conjunto de dados; (ii) a maioria dos problemas que envolvem agrupamento são NP(não determinístico polinomial); (iii) não há um algoritmo de agrupamento universal e nem tampouco podemos dizer que um agrupamento é melhor; (iv) a análise e a comparação de algoritmos de agrupamento são tarefas complexas e que dependem de muito conhecimento; (v) o agrupamento é uma tarefa não supervisionada, portanto, não há classificação conhecida para os objetos; (vi) estruturas e o grau de sobreposição do agrupamento, *outliers* e a escolha da medida de similaridade, tem grande influência no desempenho das técnicas de agrupamento; (vii) algoritmos de agrupamento são mais caracterizados com os modelos do que com os critérios de agrupamento Finalmente, podemos considerar de forma prática, a validação de um agrupamento baseados em índices estatísticos, que julgam de uma maneira quantitativa e objetiva as estruturas encontradas, podendo citar 3 critérios que empregam esses índices, são eles, os critérios relativos, internos e externos.

Interpretação é o processo de examinar cada agrupamento com relação aos seus objetos, rotulando-os e descrevendo a natureza do agrupamento. Para reconhecer e identificar os significados do agrupamento e suas possíveis relações, nesta fase, é de extrema importância o apoio de um especialista

do domínio, podendo assim permitir avaliações subjetivas que tenham um significado prático. Ferramentas para a visualização dos grupos obtidos são um apoio importante para os especialistas nessa tarefa. O *CVis* (FACELI; SAKATA; HANDL, 2017) é uma dessas ferramentas.

2.2.1 Algoritmos de Agrupamentos

Na literatura, há uma grande quantidade e variações de algoritmos de agrupamento, cada um empregando um critério de agrupamento diferente e impondo uma estrutura aos dados. Pode-se classificar os algoritmos de diferentes maneira. Uma das mais comuns é de acordo com o método adotado para definir o agrupamento, em que tem-se as categorias hierárquicos, particionais, baseados em grid e baseados em densidade (JAIN; MURTY; FLYNN, 1999; HALKIDI; BATISTAKIS; VAZIRGIANNIS, 2001).

No contexto deste trabalho, o foco será voltado para os algoritmos de agrupamento particionais. O mais famoso deles é o *k-means*, amplamente utilizado em muitas aplicações. A ideia desse algoritmo é que dado um agrupamento, deve-se realocar cada ponto para o seu centro mais próximo, atualizar os centroides dos agrupamentos, calculando a média dos pontos membros, repetindo o processo de realocação e atualização até atingir um critério de convergência (JIN; HAN, 2017a). Porém, o algoritmo *k-means* é sensível a *outliers*, pelo simples fato da média ser influenciada por valores extremos. Um algoritmo semelhante ao *k-means* e que lida melhor com *outliers* é o *k-medoids*. Assim, o algoritmo PAM (do inglês *Partitioning Around Medoids*) (KAUFMAN; ROUSSEEUW, 1990), representante do *k-medoids*, foi escolhido para utilização neste trabalho. Ele é mais robusto e resistente a ruídos e *outliers* do que o *k-means* (HAN; PEI; KAMBER, 2011).

No algoritmo PAM, um medóide é definido como o ponto no grupo, cujas diferenças são mínimas com todos os outros pontos do desse grupo. A distância do medoid C_i e o objeto P_i é calculado por $|P_i - C_i|$, desse modo, o custo do algoritmo *k-medoids* é dado pela Equação 2.5 e sua complexidade é dada por $O(k(n-k)^2)$.

O Algoritmo 1 ilustra o funcionamento do algoritmo PAM.

$$c = \sum_{C_i} \sum_{P_i \in C_i} |P_i - C_i| \tag{2.5}$$

Algoritmo 1: Algoritmo de agrupamento *k-medoids* (PAM)

Entrada: k, número de grupos e X um conjunto de dados de n pontos
Saída: Uma partição de X em K agrupamentos
início
  Selecione k pontos aleatórios dentre os n pontos de dados como os medóides;
  Associe cada ponto de dados ao medóide mais próximo usando qualquer método métrico de distância comum;
  enquanto *custo diminui* **faça**
    para cada *ponto p em X* **faça**
      Encontre o ponto mais próximo e atribua a p o seu agrupamento correspondente
    aleatoriamente selecione um ponto p_{aleat};
    calcular o custo total c ao trocar um ponto p_i por p_{aleat};
    se $c < 0$ **então**
      troque p_j por p_{aleat} para formar um novo conjunto de K pontos

2.2.2 *Ensembles* de Agrupamentos

Ensembles de Agrupamentos é a técnica que utiliza os resultados de vários algoritmos de agrupamentos ou de um algoritmo de agrupamento com diferentes inicializações dos parâmetros para compor um conjunto de partições base, a fim de obter uma partição consenso de melhor qualidade, ou seja, uma partição que melhor represente essas partições base. Inspirada nas técnicas utilizadas para combinação de estimadores independentes em aprendizado supervisionado, apesar de haver a necessidade de novas técnicas ou adaptações, podemos definir de forma simplificada os *ensembles* de agrupamento conforme (TOPCHY; JAIN; PUNCH, 2003):

> “ Dado um conjunto de n^I partições base, formalizado como:
>
> $$\prod = \{\pi^1, \pi^2, \ldots, \pi^{n^I}\} \tag{2.6}$$
>
> produzido a partir de um conjunto de dados X, pela aplicação de um ou mais algoritmos de agrupamento, encontra-se-á uma partição final π^F denominada **partição consenso** de melhor qualidade que as partições iniciais, chamadas também de **partições base**”.

Além disso, segundo (FACELI et al., 2011b) essa “melhor qualidade” pode estar relacionado a robustez, novidade, estabilidade, computação distribuída, reuso de conhecimento, consistência e desempenho e custo.

Para a geração de agrupamentos iniciais podemos considerar o *ensemble* homogêneo: onde apenas um algoritmo de agrupamento é utilizado para gerar as partições, porém, deve ser executado várias vezes com inúmeras inicializações distintas; e também podemos executar o *ensemble* heterogêneo: onde diversos algoritmos de agrupamentos são executados e para este caso, devem ser considerados algoritmos com característica bastante distintas gerando uma grande diversidade de partições.

Por fim, deve se determinar uma função consenso, utilizada para encontrar uma partição consenso, sendo heurísticas propostas para a resolução do problema formal de obtenção de uma partição consenso. Existem diversos tipos de funções consenso segundo (TOPCHY; JAIN; PUNCH, 2004), tais como:

1. Funções baseadas em coassociação;
2. Funções baseadas em grafo/hipergrafo;
3. Funções baseadas em informação mútua e;
4. Funções baseadas em votação.

2.2.3 Visualização de Agrupamentos

O objetivo de técnicas de visualização de agrupamentos é tornar a interpretação dos dados proveniente dos algoritmos de agrupamento muito mais fácil. Porém,

a análise de agrupamentos além de obter múltiplos aspectos, tem uma série de dificuldades e limitações. Ferramentas de visualização como *PVis*(FACELI et al., 2014) e *CVis*(FACELI; SAKATA; HANDL, 2017) que tem como função representar graficamente e interativamente diferentes tipos e coleções de agrupamento, facilitando a identificação de grupos que ocorrem em diferentes agrupamentos e que responde a questões sobre a frequência e a natureza da ocorrência desses grupos. Para este trabalho foram utilizadas como ferramentas de visualização o *Jupyter-Notebook*[3] e bibliotecas de visualização de gráfico como *matplotlib*[4] e *seaborn*[5].

2.3 Trabalhos Relacionados

Há vários trabalhos que relacionam a eficiência energética e o consumo dos *smartphones*, como os estudos já citados (CRUZ; ABREU, 2017; CARROLL; HEISER, 2010; HE et al., 2017a; HE et al., 2017b; RAO et al., 2017; PLACKE et al., 2017; KALLURI et al., 2017; LIN; LIU; CUI, 2017; CHEN et al., 2013; DUAN et al., 2013; AHMAD et al., 2017; GORLA et al., 2014). Aqui será dado ênfase em 3 trabalhos que foram bases para a idealização do método proposto, sendo o primeiro e segundo mais relacionados com a ideia da aplicação do método sobre consumo energético dos aplicativos móveis, e o terceiro está relacionado diretamente com a utilização de técnicas de agrupamento e com a utilização de permissões como fonte de dados para determinar o comportamento dos aplicativos móveis.

No primeiro trabalho (CRUZ; ABREU, 2017), é dito que aplicativos são desenvolvidos orientados a funcionalidades, portanto, seus desenvolvedores normalmente pensam em primeiro lugar na praticidade que um aplicativo pode prover a um usuário e, posteriormente, no decorrer da implementação, ele pense nos recursos de *hardware* necessário para o perfeito funcionamento desse aplicativo.

Nesta linha de pensamento o autor tenta responder as seguintes indagações — a) Se boas práticas de programação podem ser cegamente aplicadas a fim de melhorar a eficiência energética da aplicações; b) Se práticas relacionadas a melhorias na perfomance também melhoram a eficiência energética; c) Se essas

[3] <https://jupyter.org/>
[4] <https://matplotlib.org/>
[5] <https://seaborn.pydata.org/>

práticas realmente impactam em aplicações maduras e bem consolidadas. — ou seja, corrigindo *anti-patterns* e seguindo guias de programação, isso leva a criar aplicações mais eficientes, economizando assim o consumo da bateria?

Para responder a isso, o autor utiliza um dispositivo chamado *ODROID-XU*, um *bare-board computer* similar a outros como *Raspberry-PI*, que roda uma versão 4.4.2 de Android - *API level* 17. O experimento foi conduzido em 6 etapas: 1. Seleção das aplicações Android; 2. Análise estática e refatoração; 3. Testes automáticos de UI; 4. Ajuste das ferramentas de medição de energia; 5. Execução dos experimentos e; 6. Análise dos dados obtidos.

Durante essas etapas, foram corrigidos erros na alocação de objetos durante o carregamento de *layouts*, no uso incorreto de *wakelocks*, na correção de telas redesenhadas por várias vezes, na deleção de elementos de interface e em outros recursos não utilizados.

Finalmente, como resultado do experimento, em alguns casos como o padrão *ViewHolder* houve um ganho energético de 5%, equivalendo a aproximadamente 65 minutos a mais de bateria, considerando um dia de uso normal de um *smartphone*. Outros casos com a alocação de objetos no carregamento de *layouts* e correção de *wakelocks* renderam por volta de 1% de ganho energético.

Casos como a deleção de recursos e hierarquias de definição de *layout* não utilizados não surtiram qualquer efeito no ganho energético e surpreendentemente, ao corrigir sobreposições e redesenhos de telas, houve um aumento de 2.2% no consumo de energia.

O segundo trabalho (AHMAD et al., 2017) faz um estudo de várias bibliografias relacionadas com esquemas de modelagem e estimativas do consumo de energia em *smartphones*, tentando elucidar suas vantagens e desvantagens. Além disso, classifica-as em duas categorias, análise do código e estimativa baseada em modelos de energia de componentes devido aos seus projetos arquitetônicos. Os esquemas de modelagem e estimativa que utilizam análise de código são separados ainda em categorias baseadas em simulação e em perfil. Esses modelos são comparados com base em um conjunto de parâmetros comuns na maioria dos casos, destacando dessa forma, as semelhanças e diferenças entre a literatura relatada. Além disso, afirmam

os autores que os esquemas de estimativa de energia de aplicativos existentes são pouco precisos, caros e não são escaláveis, pois consideram sensores de tensão e corrente de baterias "inteligentes", além de ferramentas de captura de energia de baixa frequência e ambiente de configuração laboratorial para propor modelos de energia para estimativa de energia da aplicação de *smartphones*.

Durante o curso do trabalho, são mostrados taxonomias, gráficos e tabelas que comparam os esquemas de modelagem, artigos relacionados e aplicações que estimam o consumo de energia de *smartphones*, detalhando os conceitos e exemplificando tanto a análise de código, quanto os modelos de componentes de potência e energia. Finalmente, nas últimas seções são citadas tecnologias que podem auxiliar no menor consumo da bateria como a computação em nuvem e também tecnologias que como a internet das coisas (IoT) e de identificação por radiofrequência como RFID, necessitam de fontes de tamanhos reduzidos e com grande eficiência energética. Fala dos problemas atuais em pesquisa e dos desafios enfrentados na imprecisão da medição do estado de carga da bateria versus o seu tempo de envelhecimento, da crescente demanda dos *smartphones*, das dificuldades de se mapear *bugs* relacionados com a falha e danos das baterias, por cartões de memória infectados e cartões SIM danificados. Em suas duas últimas seções, discute sobre a tendência de se obter melhor eficiência energética de aplicações legadas e de quanto desenvolvedores podem se beneficiar em considerar o design de suas aplicações voltados a um uso efetivo dos recursos de bateria, além da utilização de novas técnicas como a inclusão de múltiplos níveis de cache, aumentando assim, as estimativas de consumo de energia e identificando os locais de armazenamento do código dentro da aplicação. Ao fim, os autores fazem conclusões sobre o estudo e citam trabalhos futuros utilizando um possível *framework* de baixo nível de consumo e que faz análise do código estático das aplicações para predizer com mais precisão os caminhos de execução e o local de armazenamento de dados.

No terceiro estudo (GORLA et al., 2014) os autores verificam a veracidade das descrições dos aplicativos dadas por seus desenvolvedores, comparando-os com o código de classes e com arquivos como o `manifest.xml` no Android. Nesses artefatos, estão contidas as permissões de acessos de recursos, como por exemplo, acesso a internet, lista de contatos, galeria de fotos, arquivos e outras dezenas de

permissões necessárias para o devido funcionamento desse aplicativo.

Existem desenvolvedores e empresas que publicam aplicativos na *Google Play Store*, utilizam de permissões e funcionalidades que não são divulgadas, omitindo dessa forma para o usuário final as suas reais intenções. Esses casos em geral, tem a finalidade de coletar dados para fins de publicidade, práticas de *malware*, sequestro de dados, *phishing* entre outras práticas duvidosas e mal intencionada. Para reconhecer esse tipo de prática é apresentado em (GORLA et al., 2014) um método chamado `CHABADA`[6], que utilizando das etapas descritas abaixo, procura encontrar *outliers* que representam esses aplicativos suspeitos que omitem suas reais funcionalidades e finalidades:

1. Primeiramente, utilizando um *script* automatizado, rodando em intervalos regulares e durante o período do inverno e primavera de 2013, foram coletados 150 aplicativos mais populares e gratuitos de 30 categorias diferentes na loja da *Google Play*. Uma simples rodada desse tal *script*, coletava por volta de 4500 aplicativos, todos eles, aparentemente selecionados por serem considerados aplicativos "bons", ou seja, aplicativos que supostamente não apresentariam nenhum tipo de risco à privacidade, dados ou outro comportamento suspeito como *malwares*, vírus, etc. No total, foram coletados um total de 32.136 aplicativos e adicionado a isso, também foram coletados metadados referentes a nome, descrição, data de publicação, notas de usuários e *screenshots*.

2. Logo após, considerando o inglês como a língua padrão das descrições e excluindo as outras línguas, foram utilizadas técnicas de processamento de linguagem natural, removendo palavras como "*the*", "*is*", "*at*" aplicando a técnica de "*stemming*", que identifica a raiz de um grupo de palavras. Isso reduzia as descrições dos aplicativos a não mais que 10 palavras. Finalmente, retiraram-se *tags* HTML, numerais, *links*, endereços de email. Foram retiradas aplicações que não possuíam APIs sensíveis, resultando então num total de 22.521 aplicações ao final dessa etapa;

3. Numa próxima etapa, a fim de identificar um conjunto de tópicos — onde cada tópico se refere a um *cluster* de palavras que frequentemente ocorrem juntas

[6] do inglês CHecking App Behavior Against Descriptions of Apps

— foi utilizado uma modelagem estatística utilizada em textos não "tageados" chamada "Alocação Latente de Dirichlet", onde são agrupadas palavras como por exemplo "*map*", "*traffic*", "*position*" e "*route*" em um *cluster*. Dessa forma, foram separados 30 possíveis tópicos, onde cada aplicativo poderia pertencer a no máximo 4 tópicos considerando uma probabilidade mínima de 5% para ser considerado pertencente a um tópico específico;

4. Como próximo passo, foi utilizado o *K-means* como algoritmo de agrupamento para categorizar os aplicativos, no intuito de agrupá-los visando a suas similaridades nas descrições. Como um dos desafios do *K-means* é estimar a quantidade ideal de *clusters* a serem usados, eles optaram rodá-lo diversas vezes, considerando os extremos indo de $K = 2$ até $K = número\ aplicativos$, por fim, para auxiliar na escolha ideal do número de *clusters*, utilizou o método da silhueta. No total foram obtidos 32 *clusters*, onde cada um contém no máximo 4 tópicos que o representam. Por exemplo foi obtido o *cluster* denominado "*music*", formado pelos tópicos "*music*", "*share*" e "*settings and utils*".

5. Na penúltima etapa do processo, para cada *cluster*, o método `CHABADA` identifica as *APIs* que cada aplicativo acessa estaticamente, considerando somente *APIs* "sensíveis", ou seja, aquelas que necessitam de permissões de usuário, como mostrado na *Listing* 2.1[7].

6. Na última etapa, foi aplicado o método de "*One-Class SVM*", que classifica e faz uma lista onde os primeiros listados são os aplicativos com contém indícios de anomalia em relação as *APIs* que eles utilizam, sendo forte candidato a aplicativos maliciosos ou que não condiz com as funcionalidades descritas na sua página da *Google Play Store*.

[7] Adicionando permissões no manifesto — <https://developer.android.com/training/permissions/declaring.html#perm-add>

Listing 2.1 – AndroidManifest.xml

```
1 <manifest xmlns:android="http://schemas.android.com/apk/
    res/android"  package="com.example.snazzyapp">
2
3     <uses-permission android:name="android.permission.
        SEND_SMS"/>
4
5     <application ...>
6         ...
7     </application>
8
9 </manifest>
```

3 Método Proposto

Neste capítulo será apresentado o método proposto que utiliza as permissões dos aplicativos para identificar perfis de permissões dos aplicativos e analisá-los para extração de conhecimento.

O método pode ser dividido nas etapas: (i) Coleta de dados, (ii) Identificação dos perfis de permissão e (iii) Visualização, ilustradas na Figura 3

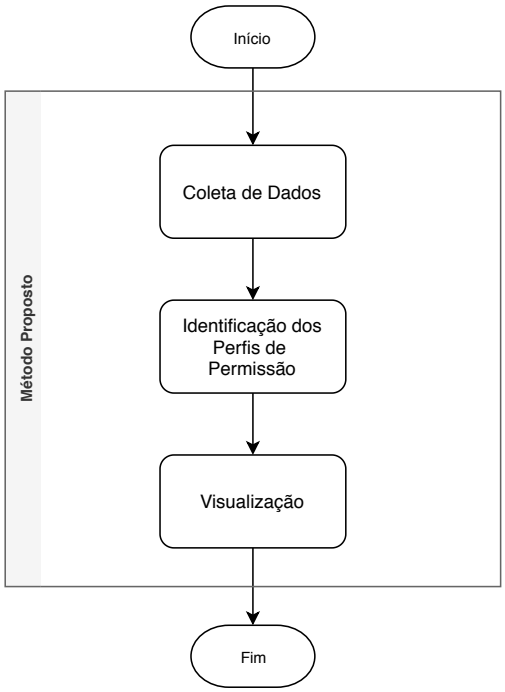


Figura 3 – Método proposto para a análise de perfis de permissão

Na etapa da **coleta de dados**, detalhada na Seção 3.1, devem ser selecionados os aplicativos de interesse até a extração das permissões desses aplicativos.

Em **identificação dos perfis de permissão**, detalhada na Seção 3.2, encontra-se o núcleo principal do método proposto, onde são feitos a formatação dos dados, o cálculo de similaridade, a aplicação do algoritmo de agrupamento e

finalmente a obtenção dos **grupos consenso**, onde estão identificados os perfis de permissões.

Finalmente, na etapa de **visualização**, detalhada na Seção 3.3, são produzidas visualizações dos perfis de permissão que permitem as análises e extração de conhecimento.

Durante todas as etapas, será introduzido um exemplo para ilustrar e facilitar a melhor compreensão da utilização do método proposto.

3.1 Coleta de dados

A coleta de dados é a etapa onde são coletados e selecionados os aplicativos e extraída as permissões para a futura etapa de identificação de perfis de permissão. A Figura 4 ilustra os componentes e processos dessa etapa.

3.1.1 Seleção dos Aplicativos

Nesta etapa inicial devem ser selecionados os aplicativos que serão utilizados na aplicação do método, podendo a técnica de coleta ser escolhida livremente pelo usuário, desenvolvedor ou pesquisador desses aplicativos. Podem ser feitas diversas formas distintas de coletas, de forma individual ou conjunta, como por exemplo, a utilização da técnica de *web crawler*, a utilização de um robô coletor de dados, uma lista disponível na internet, repositórios de dados ou até uma lista criada pelo próprio indivíduo que fará a análise.

3.1.2 Extração do nome do pacote *apk*

Com a lista de aplicativos definida, é necessária a obtenção de pelo menos o nome dos pacotes desses aplicativos. Para obtê-los é necessário acessar a página do *Google Play Store*, selecionar cada aplicativo dessa lista e na composição do *link* da página de cada aplicativo, capturar o nome do seu pacote. Num exemplo, o aplicativo *snapchat*, no *link* para a página que contém suas informações será da seguinte forma:

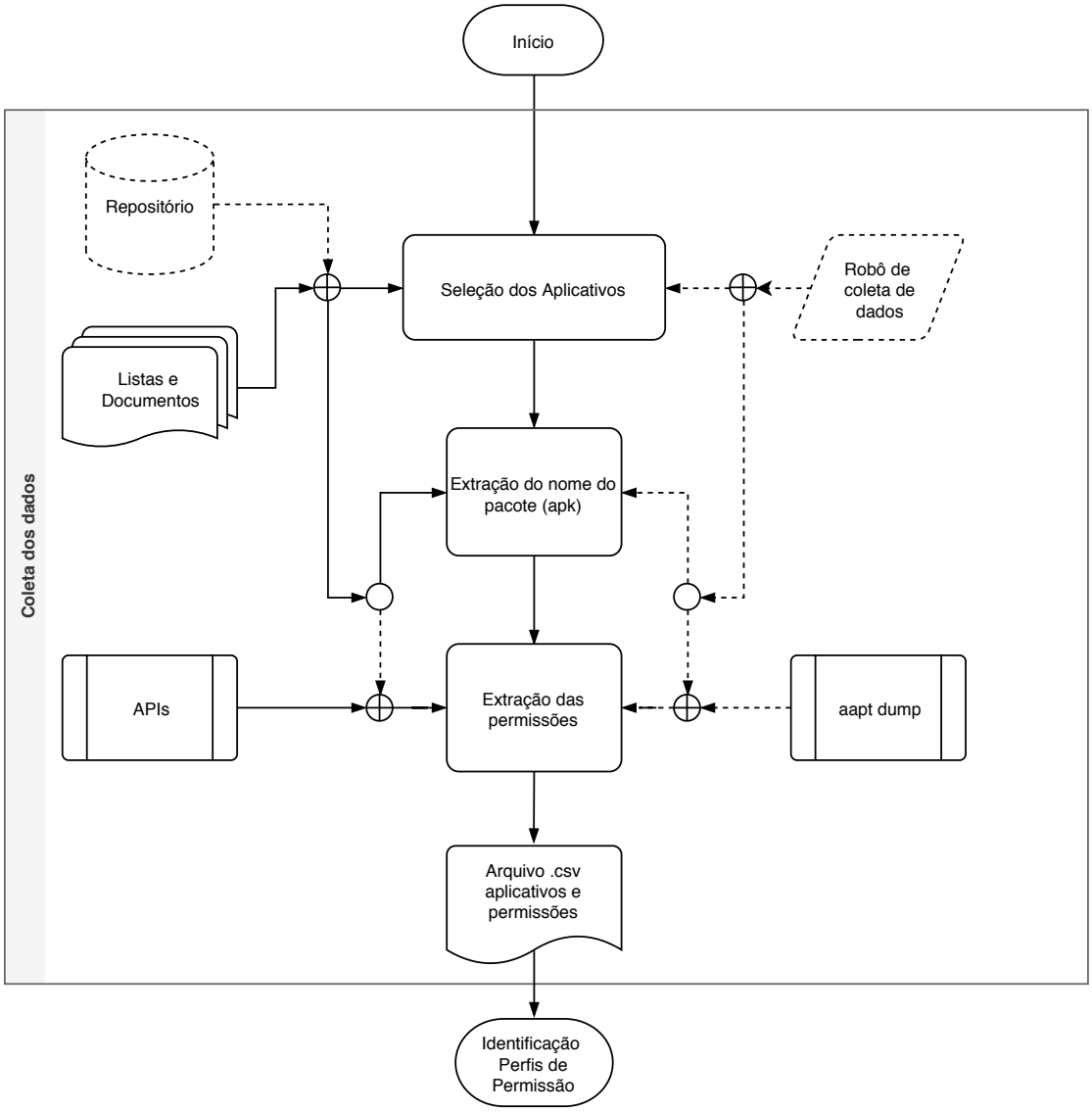


Figura 4 – Coleta de dados

<https://play.google.com/store/apps/details?id=com.snapchat.android>

onde o que está após "*?id=*" é o nome do pacote desse aplicativo, que neste caso será
"*com.snapchat.android*".

Com o nome dos pacotes dos aplicativo já é possível prosseguir para o próximo passo onde é extraída a permissões desses aplicativos.

3.1.3 Extração das permissões

Na coleta de dados dos aplicativos pode-se escolher diversas técnicas já implementadas para colher essas informações ou mesmo desenvolver sua própria técnica ou algoritmo para essa captura. Existem muitos *APIs* que podem ser utilizados para a coleta das permissões, alguns exemplos são:

- Python Android Market Library (<https://github.com/liato/android-market-api-py>)
- Google Play Unofficial Python API (<https://github.com/egirault/googleplay-api>)
- Google Play Unofficial Python 3 API Library (<https://github.com/alessandrodd/googleplay_api>)
- Google play python API (<https://github.com/NoMore201/googleplay-api>)

Outro método que pode ser utilizado seria o download de cada "apk", ou seja, o arquivo de instalação de cada aplicativo seguida da extração do arquivo `AndroidManifest.xml` utilizando os comandos `aapt dump badging <apk file>` ou `aapt dump xmltree <apk file> AndroidManifest.xml` para coletar as permissões necessárias de cada aplicativo. Apesar de ser um método eficaz para a extração dos dados de permissões, essa abordagem é extremamente trabalhosa por ser necessário baixar esses arquivos, além de requerer recursos adicionais de espaço em disco para armazenar os arquivos `apk` de instalação do aplicativo para somente após isso extrair via aapt as permissões.

A filtragem inicial das permissões pode ser feita baseando-se no projeto AOSP, onde na sua versão atual apresenta um total de 456 permissões. Porém, como já mencionado na Seção 2.1.2, muitas permissões podem ser relacionadas às *push notifications* ou ainda a permissões específicas do fabricantes dos dispositivos. Fica então a critério aplicar-se outras permissões que não sejam do domínio do projeto AOSP, pois fabricantes de dispositivos ou desenvolvedores de aplicativos podem incluir suas próprias permissões.

3.1.4 Resultado final da coleta de dados

O resultado final da coleta de dados deve conter todos os aplicativos coletados e suas respectivas permissões. Apesar de haver inúmeras possibilidades de armazenamento desses dados, a mais simples e provavelmente mais eficaz é a utilização de um arquivo do tipo *csv* que contém na sua primeira coluna o nome do aplicativo e nas próximas colunas as permissões dele extraídas.

A partir desse ponto será iniciado um exemplo em que seus dados iniciais são mostrados na Tabela 1 e ao longo da descrição do método será utilizado a fim de ilustrar todas as suas etapas, visando assim, a devida compreensão de seu funcionamento.

a1	p1	p2								
a2	p5	p7	p9							
a3	p2	p3	p4							
a4	p7	p5	p8	p3						
a5	p4	p1	p6	p9	p7	p3				
a6	p7	p3	p5	p2						
a7	p1	p5	p9							
a8	p6	p3								
a9	p1	p3	p5	p6	p7	p8	p9			
a10	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10
a11	p5	p8	p10							
a12	p1	p9	p4	p7	p3					
a13	p1	p2	p3							
a14	p4	p5	p6							
a15	p7	p8	p9	p10						
a16	p1	p3	p5	p7						
a17	p2	p4	p6	p8						
a18	p10	p9	p6	p2	p1					
a19	p1	p2	p5	p6	p8	p9	p10			
a20	p3									

Tabela 1 – Tabela inicial em formato csv de aplicativos e permissões

3.2 Identificação dos perfis de permissão

A identificação dos perfis de permissão é a etapa em que aplicam-se as técnicas de agrupamento para identificação dos grupos consenso. Para isso, é necessário a escolha de algoritmos de agrupamento apropriados e a preparação dos dados para a utilização com esses algoritmos. Assim, é feita uma formatação dos dados coletados, o cálculo de proximidade, a aplicação do agrupamento em si e a identificação dos grupos consenso. O final dessa etapa resultará nos perfis de permissão que serão visualizados. A Figura 5 ilustra os processos dessa etapa.

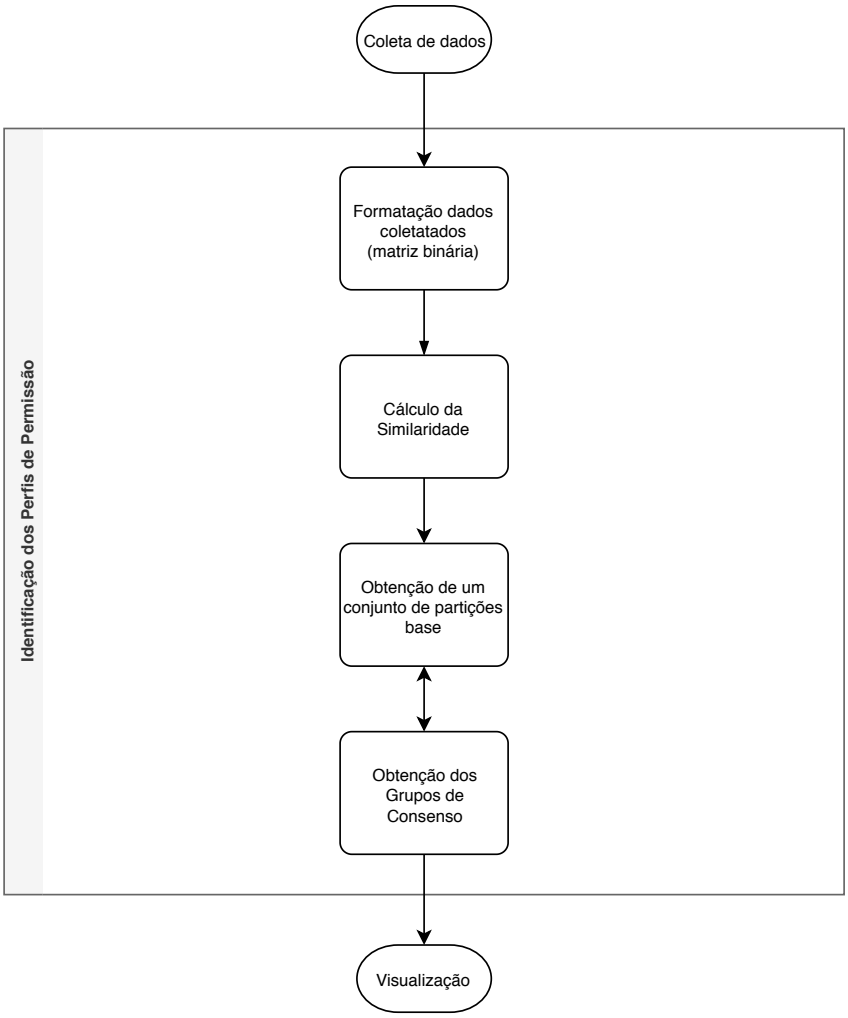


Figura 5 – Identificação dos perfis de permissão

3.2.1 Formatação dos dados coletados

Após a coleta de dados e com o arquivo do tipo csv preparado, deve-se representar os dados apropriadamente. Dependendo dos algoritmos de agrupamento que serão utilizados, pode-se escolher uma forma de representação diferente. Uma forma apropriada para a maioria dos casos é uma matriz binária, onde os atributos indicam a presença ou ausência da permissão. Na Tabela 2 são convertidos os valores da Tabela 1 para uma matriz onde as linha representam os aplicativos e os atributos são representados pelas permissões.

	p1	p10	p2	p3	p4	p5	p6	p7	p8	p9
a1	1	0	1	0	0	0	0	0	0	0
a2	0	0	0	0	0	1	0	1	0	1
a3	0	0	1	1	1	0	0	0	0	0
a4	0	0	0	1	0	1	0	1	1	0
a5	1	0	0	1	1	0	1	1	0	1
a6	0	0	1	1	0	1	0	1	0	0
a7	1	0	0	0	0	1	0	0	0	1
a8	0	0	0	1	0	0	1	0	0	0
a9	1	0	0	1	0	1	1	1	1	1
a10	1	1	1	1	1	1	1	1	1	1
a11	0	1	0	0	0	1	0	0	1	0
a12	1	0	0	1	1	0	0	1	0	1
a13	1	0	1	1	0	0	0	0	0	0
a14	0	0	0	0	1	1	1	0	0	0
a15	0	1	0	0	0	0	0	1	1	1
a16	1	0	0	1	0	1	0	1	0	0
a17	0	0	1	0	1	0	1	0	1	0
a18	1	1	1	0	0	0	1	0	0	1
a19	1	1	1	0	0	1	1	0	1	1
a20	0	0	0	1	0	0	0	0	0	0

Tabela 2 – Tabela binária de aplicativos e permissões

3.2.2 Escolha da medida de similaridade ou dissimilaridade

O método proposto utiliza-se de dados binários como apresentados na Tabela 2 e, portanto, esses dados são classificados como quantitativos e binários, o que os torna passíveis da utilização de técnicas de proximidade que utilizam a distância (dissimilaridade) para calcular a similaridade entre os objetos.

Apesar de no último século aproximadamente 76 medidas de similaridade e distância serem utilizadas para valores binários (CHOI; CHA; TAPPERT, 2010), é preciso atentar-se ao fato de que um aplicativo pode ter uma quantidade grande de atributos e que a posição da coluna onde os valores binários estão situados fazem toda a diferença. Para melhor compreensão dessa situação, pode-se filtrar a Tabela 2, mostrando somente os aplicativos `a2`, `a3`, `a7`, `a11`, `a13` e `a14`, como na Tabela 3. Percebe-se então, que esses aplicativos contém três permissões cada e, apesar de alguns até compartilharem a mesma permissão, ao reparar para a linha de cada aplicativo, torna-se perceptível a enorme distinção entre essas linhas.

	p1	p10	p2	p3	p4	p5	p6	p7	p8	p9
a2	0	0	0	0	0	1	0	1	0	1
a3	0	0	1	1	1	0	0	0	0	0
a7	1	0	0	0	0	1	0	0	0	1
a11	0	1	0	0	0	1	0	0	1	0
a13	1	0	1	1	0	0	0	0	0	0
a14	0	0	0	0	1	1	1	0	0	0

Tabela 3 – Tabela binária de aplicativos e permissões com os aplicativos a2, a3, a7, a11, a13 e a14

O problema é como conseguir distinguir os aplicativos se por exemplo, se tentarmos somarmos suas permissões, o resultado será sempre três? Haveria alguma forma, de poder identificar essas permissões e os aplicativos distintamente? Para solucionar esse problema, é necessário que se enxergue cada linha como um documento e cada atributo como uma palavra, assim, pode-se comparar as linhas como vetores e utilizar então a similaridade por cosseno, que é uma técnica muito empregada para comparar a diferença entre dois vetores em relação ao ângulo entre eles. Sua aplicação na análise de textos é muita utilizada, demonstrando

assim bons resultados como mostrados em (HUANG, 2008; AMINE; ELBERRICHI; SIMONET, 2010).

Dessa forma, ao enxergar as linhas dos aplicativos como documentos e seus atributos como palavras, a similaridade por cosseno seria uma ótima técnica para se aplicar a esse propósito, e melhor ainda seria, se houvesse uma forma de obter uma medida que quanto menor ela fosse, maior similaridade haveria entre os objetos. Para isso, utilizou-se a distância cosseno, apresentada anteriormente na Seção 2.2. Para exemplificar a Tabela de distância a ser utilizada pelo algoritmo de agrupamento, ilustrada parcialmente na Tabela 4.

	a2	a3	a7	a11	a13	a14
a2	0.000000	0.452277	0.367544	0.163340	0.683772	0.452277
a3	0.452277	0.000000	0.422650	0.345346	1.000000	1.000000
a7	0.367544	0.422650	0.000000	0.433053	1.000000	1.000000
a11	0.163340	0.345346	0.433053	0.000000	1.000000	0.781782
a13	0.683772	1.000000	1.000000	1.000000	0.000000	0.422650
a14	0.452277	1.000000	1.000000	0.781782	0.422650	0.000000

Tabela 4 – Tabela distância cosseno dos aplicativos a2, a3, a7, a11, a13 e a14

3.2.3 Obtenção de um conjunto de partições base

Com a distância escolhida ou matriz de distância calculada, aplica-se um ou mais algoritmos de agrupamento para agrupar os aplicativos por permissões. A escolha do algoritmo de agrupamento também fica a critério do aplicador do procedimento, porém, há algoritmos que podem ter melhores resultados, principalmente aqueles que não forem sensíveis a *outliers*, como é o caso do *k-medoids* (ARORA; VARSHNEY et al., 2016; HAN; PEI; KAMBER, 2011; JIN; HAN, 2017b). Além disso, também fica a critério do individuo aplicador do método a quantidade de partições e de grupos nessas partições. Nessa etapa são definidos todos esses elementos e são executados os algoritmos produzindo um conjunto de partições. Para ilustrar, utilizando como base os dados data Tabela na Tabela 2, aplicando a distância de cosseno e aplicando o algoritmo *k-medoids*, determinando o número de partições em cinco, ou seja, $k \in [2, 6]$ e com o valores iniciais do medóide randômico, foi obtido como resultado o conjunto de partições base representados na Tabela 5

	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18	a19
π_1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
π_2	0	0	0	0	0	1	0	0	1	0	0	2	0	1	0	0	0	2	0	0
π_3	2	1	0	1	2	1	1	3	1	1	1	3	2	2	3	1	1	3	3	3
π_4	0	0	1	2	0	3	0	2	4	0	0	0	2	4	0	2	0	0	0	0
π_5	3	0	1	2	3	4	0	2	0	0	0	5	3	3	0	2	0	5	0	0

Tabela 5 – Partições resultantes do algoritmo de agrupamento

Empiricamente, durante a execução da aplicação deste método, foi verificado que a quantidade de partições ideal para dados reais devem estar entre $k \in [2, 20]$, evitando-se assim uma excessiva dispersão ao identificar os perfis de permissão.

3.2.4 Obtenção dos Perfis de Permissão

A obtenção dos perfis de permissão dá-se através da utilização de uma técnica muito similar aos *ensembles* de agrupamentos, relacionada na Seção 2.2.2, porém, ao invés de utilizar uma função consenso para obter uma partição consenso, obtém-se **grupos consenso**.

Define-se **grupos consenso** como um subconjunto dos elementos de um conjunto de dados X que se agrupam em todas as partições base, geradas por um algoritmo de agrupamento. Formalizando, temos que, dado um conjunto de partições base, um grupo consenso C^* é definido como:

$$\begin{aligned} C^* = \Big\{ & C_j^i \cap C_m^\ell \mid i \neq \ell \ \wedge \ C_j^i \cap C_m^\ell \neq \varnothing \ \wedge \ |C_j^i \cap C_m^\ell| > 1, \\ & C_j^i \in \pi^i, C_m^\ell \in \pi^\ell, \\ & \pi^i, \pi^\ell \in \prod, \\ & i, \ell \in \{1, \ldots, n^I\} \Big\} \end{aligned} \tag{3.1}$$

Para encontrar grupos consenso, deve-se identificar os aplicativos que estão sempre juntos em um mesmo grupo, independentemente da partição que eles se encontram. Dessa forma, baseando-se na Tabela 5, que representa o conjunto de partições base, podemos encontrar o grupo consenso C_1^* representado pelos aplicativos $a0$ e $a4$, o grupo consenso C_2^* representado pelo aplicativos $a1$, $a6$, $a9$,

$a10$ e $a16$ e assim por diante. Ao identificar todos os grupos consenso contidos na Tabela 5, temos os seguintes grupos:

- $C_1^* = \{a0, a4\}$
- $C_2^* = \{a1, a6, a9, a10, a16\}$
- $C_3^* = \{a3, a15\}$
- $C_4^* = \{a11, a17\}$
- $C_5^* = \{a14, a18, a19\}$

Por outro lado, pode-se perceber que os aplicativos a2, a5, a7, a8, a12 e a13 são aplicativos não agrupados.

Uma forma melhor de visualizar esses grupos consenso é colorir a Tabela 5 e colori-la de acordo com os grupos consenso. Isso é exemplificado na Tabela 6 em que o grupo consenso C_1^* foi colorido na cor amarela, o grupo consenso C_2^* na cor azul e assim por diante. Para facilitar ainda mais a observação dos grupos, podemos ainda reorganizá-los pelas cores, obtendo então a Tabela 7.

	a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14	a15	a16	a17	a18	a19
π_1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
π_2	0	0	0	0	0	1	0	0	1	0	0	2	0	1	0	0	0	2	0	0
π_3	2	1	0	1	2	1	1	3	1	1	1	3	2	2	3	1	1	3	3	3
π_4	0	0	1	2	0	3	0	2	4	0	0	0	2	4	0	2	0	0	0	0
π_5	3	0	1	2	3	4	0	2	0	0	0	5	3	3	0	2	0	5	0	0

Tabela 6 – Grupos consenso coloridos.

	a0	a4	a1	a6	a9	a10	a16	a3	a15	a11	a17	a14	a18	a19	a2	a5	a7	a8	a12	a13
π_1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
π_2	0	0	0	0	0	0	0	0	0	2	2	0	0	0	0	1	0	1	0	1
π_3	2	2	1	1	1	1	1	1	1	3	3	3	3	3	0	1	3	1	2	2
π_4	0	0	0	0	0	0	0	2	2	0	0	0	0	0	1	3	2	4	2	4
π_5	3	3	0	0	0	0	0	2	2	5	5	0	0	0	1	4	2	0	3	3

Tabela 7 – Grupos consenso organizados pelas cores.

3.3 Visualização

Nessa etapa, são obtidos os gráficos que permitem a visualização dos grupos consenso e é onde se extrai o conhecimento necessário para obter os resultados desejados, analisando as permissões, ponderando-as e filtrando-as. A Figura 6 ilustra os processos dessa etapa.

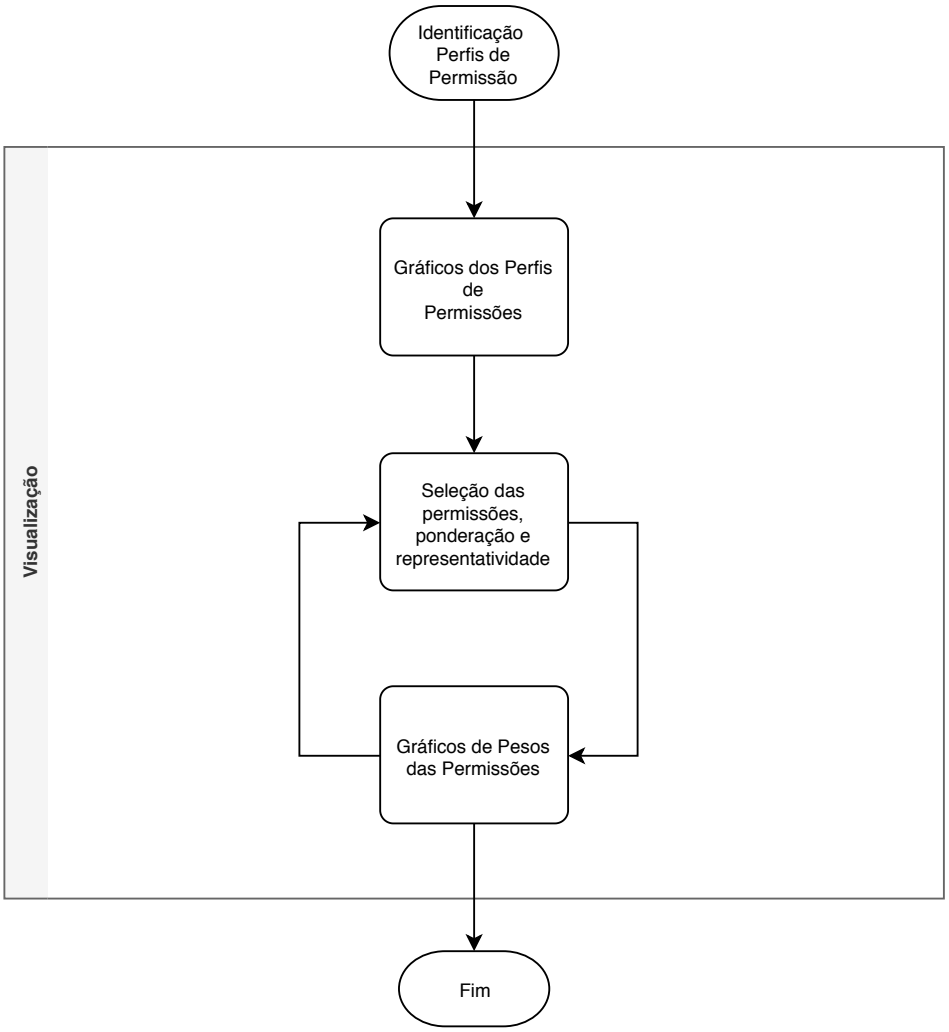


Figura 6 – Visualização

3.3.1 Gráficos dos Perfis de Permissões

Um gráfico de permissão é a representação gráfica dos grupos consensos. Neles estão contidos as informações visuais necessárias que darão uma base de conhecimento para a seleção e a ponderação das permissões.

O gráfico de perfis de permissão é montado a partir de cada grupo consenso. Cada linha dentro de um bloco representa um aplicativo e cada coluna do gráfico representa uma permissão. O gráfico é representado na forma de mapa de calor, colorido de forma binária, proporcionando assim, grande clareza ao ver a presença da permissão em azul escuro e a ausência da permissão em amarelo claro. Assim, ao observar uma única linha, visualiza-se o aplicativo e todas as suas permissões que ele têm e que fazem correspondência às permissões empregadas.

Ao observar um bloco vê-se os aplicativos que tem a maioria de suas permissões em comum. Além disso, é possível visualizar todas as permissões que os aplicativos compartilham entre si e quais permissões estão entre a maioria dos aplicativos ou em todos os aplicativos do grupo consenso.

Quando se compara blocos desse gráfico, ou seja, grupos consenso distintos, nota-se a diferença entre esses grupos observando-se a distribuição das permissões entre eles. Pode-se visualmente, com grande facilidade, perceber a distribuição distinta das permissões entre esses diferentes grupos consenso e pode-se analisar, neste momento, tipos de aplicativos que são categorizados diferentemente, mas que fazem parte de um mesmo grupo consenso por compartilhar permissões semelhantes.

Pode-se plotar todos os grupos consenso conforme a Figura 7, bem como refiná-la pela quantidade de aplicativos. Por exemplo, plotar somente os grupos consenso com 2 aplicativos conforme a Figura 8 ou os grupos consenso com três e cinco aplicativos conforme a Figura 9.

Na Figura 7, pode-se reparar que a permissão `p1` aparece frequentemente em pelo menos 50% dos aplicativos de todos os grupos de consenso de C_0^* a C_3^*, porém em C_4^* esta permissão aparece em 33% dos aplicativos; assim, implica-se que para o grupo consenso C_4^*, a permissão `p1` tem menor frequência para esse grupo consenso em relação aos demais. Da mesma forma, pode-se ver que a permissão `p4` está presente em todos os aplicativos do grupo consenso C_2^*, assim como a permissão

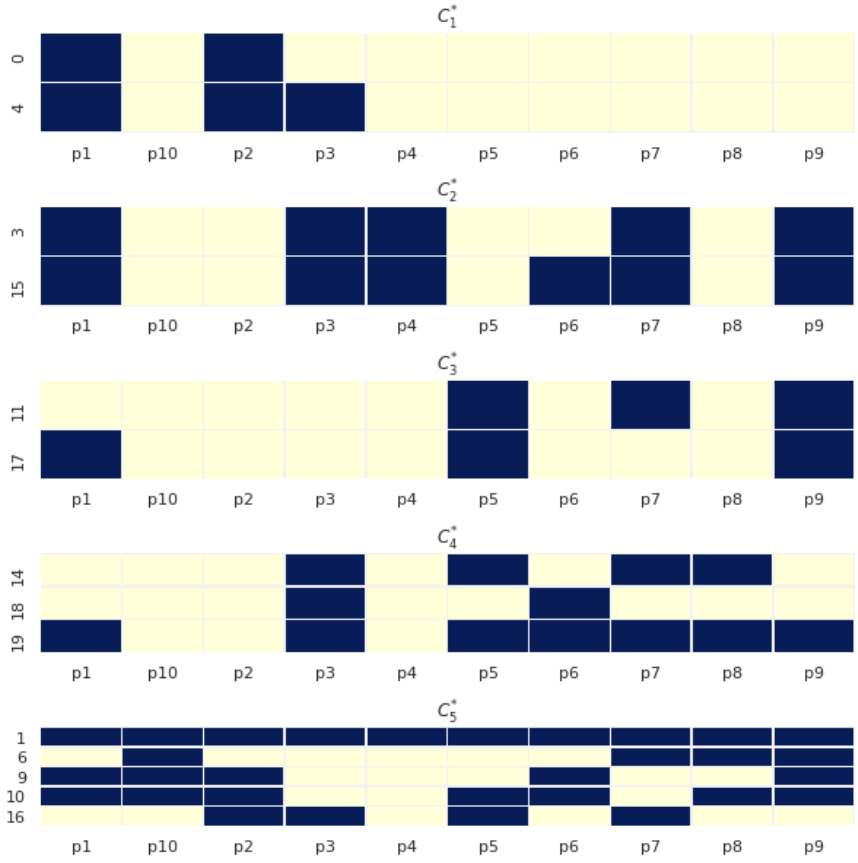


Figura 7 – Todos os grupos de consenso gerados

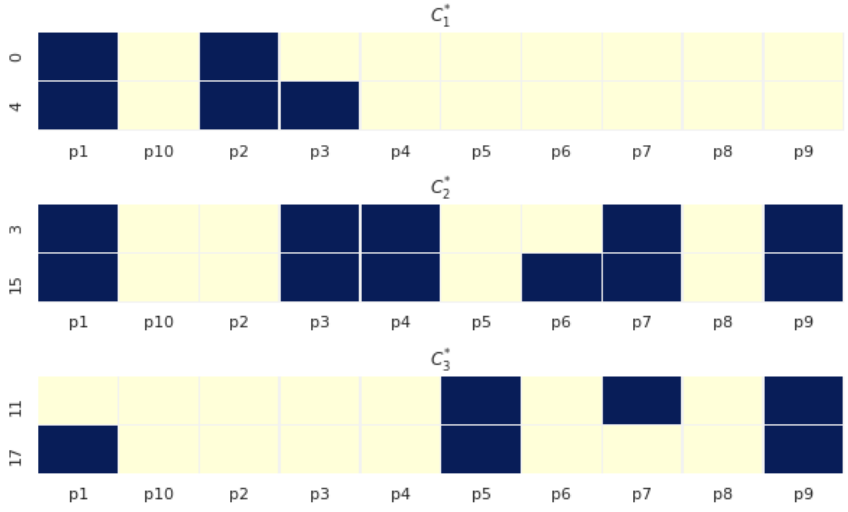


Figura 8 – Grupos de consenso com somente dois aplicativos

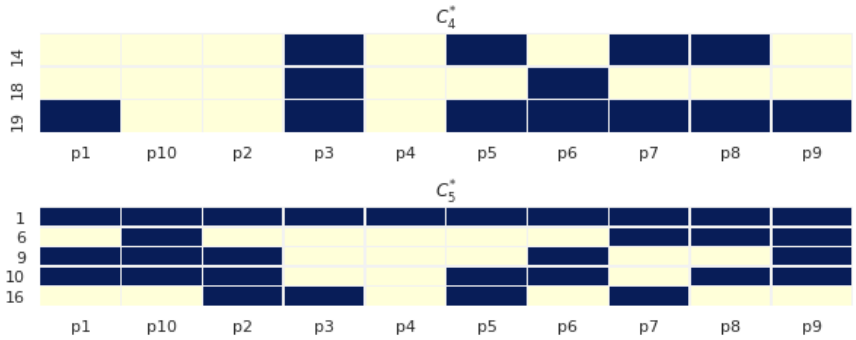


Figura 9 – Grupos de consenso com três e cinco aplicativos

p9 em está presente na maioria dos aplicativos dos grupo consenso de C_2^*, C_3^* e C_5^* e assim por diante.

Todas essas possibilidades em visualizar os grupos consenso, plotar os gráficos com diferentes quantidades de aplicativos e olhar as frequências das permissões — ajustar a representatividade das permissões — permitem analisar e visualizar os grupos consenso e utilizá-los para análise, podendo assim o aplicador do método decidir quais as permissões que poderão ser relevantes para seu estudo para depois selecioná-las e ponderá-las gerando posteriormente o gráfico de pesos das permissões.

3.3.2 Seleção das permissões, ponderação e representatividade

Analisado os gráficos de permissões, a próxima etapa é relacionada com a seleção, ponderação e representatividade das permissões.

Esta etapa é considerada como uma preparação para a construção do gráfico de pesos das permissões e que será revisitada para refinar este gráfico por diversas vezes. Além disso, é nesta fase que começa a se pôr em prática a análise dos dados, estudando cuidadosamente os gráficos de permissões. Nesse ponto, deve-se usar conhecimento prévio, por exemplo da literatura e de dados empíricos, para selecionar as permissões que mais convém para o estudo e, em seguida dá-lhe pesos.

Feita a seleção e a ponderação das permissões, deve o aplicador olhar para os grupos de consenso e analisar a frequência com que essas permissões aparecem, tentando assim, precisar um valor de corte ou um limiar para que essas permissões

sejam consideradas para o próximo gráfico de pesos das permissões. Essa frequência com que as permissões aparecem nos grupos de consenso, também é chamada neste estudo como a *representatividade de uma permissão*, sendo essa medida representada pela frequência em porcentagem que uma permissão aparece num grupo consenso.

Continuando o exemplo desde o início do método e baseando na Figura 7, poderia-se supor que as permissões `p1`, `p2`, `p4`, `p7` e `p8` são relevantes ao estudo em questão e que chegou-se a conclusão que os pesos dessas permissões seriam representados pelos valores de `10`, `30`, `40`, `70` e `90` respectivamente, para uma escala de 0 a 100, onde o zero seria um fator nulo de relevância e 100 seria o fator de maior relevância. Dessa forma, já se tem a base necessária para a construção de um gráfico de pesos das permissões. Porém, ainda está faltando mais um dado que é um limiar de representatividade a partir do qual as permissões serão selecionadas consideradas para a construção do gráfico de pesos. Para este caso específico, poderia-se supor que a representatividade será considerada com um valor de corte de 40%, o que implica que se uma permissão não tiver frequência em pelo menos 40% dos aplicativos num grupo consenso, essa permissão não será considerada para a construção do gráfico de peso das permissões.

Todos os dados de seleção, ponderação e representatividade, podem ser refeitos ao longo da aplicação do método, podendo assim o aplicador simular diversos cenários que fizerem sentido para a análise e produção de conhecimento para seu estudo. Portanto, esta fase que é anterior ao gráfico de pesos das permissões pode ser constantemente mudada, revisitada e reanalisada pelo aplicador do método quantas vezes for necessário para aperfeiçoar e refinar as análises.

3.3.3 Gráficos de Pesos das Permissões

O gráfico de pesos das permissões é a última etapa desse método e nele se encontra o resultado de todo conhecimento utilizado pelo aplicador do método na análise dos gráficos de permissões e da seleção, ponderação e representatividade das permissões.

Cada linha desse gráfico representa um grupo consenso distinto, diferente-

mente do gráfico de permissões em que cada linha representa um aplicativo. Porém, ambos os gráficos têm em comum que cada coluna é uma permissão diferente; no entanto, o gráfico de pesos das permissões têm menos colunas devido a seleção das permissões que foi feita na etapa anterior.

A coloração do gráfico, que varia da cor verde escura, que representa o menor valor no peso das permissões, para a cor vermelho escuro, que representa o maior valor no peso das permissões, está diretamente ligada ao impacto maior que aquela permissão representa ao grupo consenso, havendo ainda outras variações de cores que representam valores intermediários que vão do verde claro, amarelo, laranja, vermelho claro e outras cores que representam o espectro de variação entre as cores verde, amarela e vermelha.

Caso a permissão não esteja presente no grupo consenso ou ela foi descartada devido a sua baixa representatividade ela aparecerá com o valor zero e será representada pela cor verde escura, ou seja, ela não tem impacto referente à questão sendo investigada, naquele grupo consenso. Uma ilustração inicial do gráfico de pesos das permissões é mostrado na Figura 10, onde suas linhas fazem correspondência com os blocos da Figura 7, os números contidos dentro de cada área que corresponde às cores são os valores dados aos pesos e ao lado da visualização desse pesos é mostrado uma barra com o espectro de cores e os valores correspondentes.

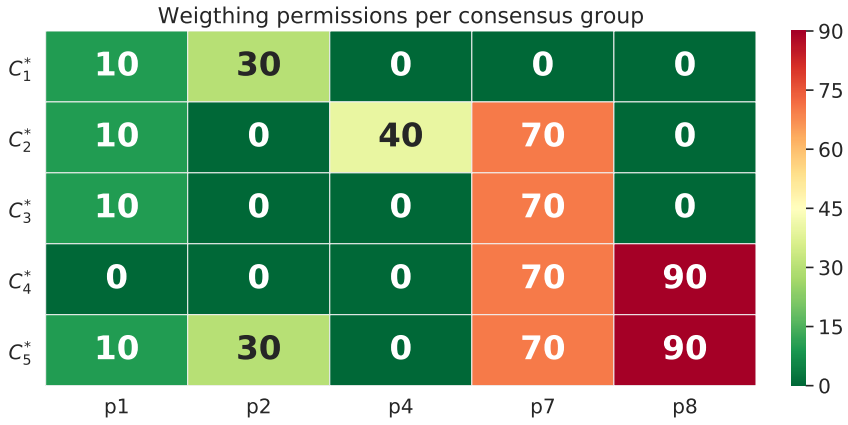


Figura 10 – Gráfico de pesos das permissões

Muitas vezes porém, somente o gráfico da Figura 10 não expõem todas as informações necessárias para decidir qual grupo consenso tem o somatório das

permissões próximo do valor mínimo, máximo ou intermediário. Para expor todas as informações necessárias propõem-se que a visualização de pesos das permissões seja composta por três gráficos mostrados conjuntamente: o primeiro, já mostrado na Figura 10, mostra cada permissão selecionada e o peso dado a cada uma delas; um segundo gráfico, de apenas uma coluna, mostra o somatório dos pesos daqueles grupos de consenso do primeiro gráfico; e, um terceiro gráfico de um coluna também que compara o somatório dos pesos com o total de grupos de consenso caso haja outros que no momento não estejam sendo mostrados. Finalmente, a Figura 11 ilustra a visualização de pesos das permissões em sua forma completa com os 3 gráficos mostrados conjuntamente.

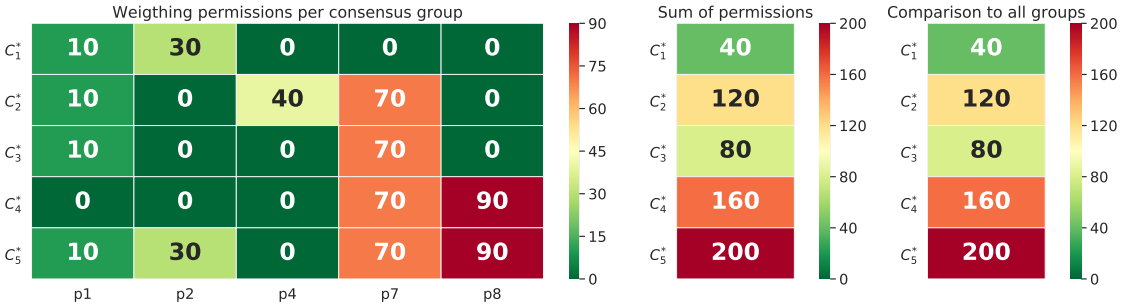


Figura 11 – Visualização dos pesos das permissões em sua forma completa

Na Figura 11 ao olhar para o primeiro gráfico da visualização dos pesos, pode-se ver os valores dos pesos sendo mostrados ao longo do grupo consenso. Por exemplo, o grupo consenso C_1^* tem os seguintes valores dos pesos: $p1 = 10$, $p2 = 30$, $p4 = 0$, $p7 = 0$ e $p8 = 0$; o segundo gráfico, mostra o somatório dos pesos das permissões, no caso, $p1 + p2 + p4 + p7 + p8 = 40$ e utilizando esse valor, sua cor será definida em relação ao grupos de C_1^* a C_5^* que estão sendo mostrados nessa visualização. O terceiro e último gráfico é opcional, sendo necessário somente quando há outros grupos consenso categorizados em outras visualizações e devido a grande quantidade desses grupos consenso, fica inviável mostrá-los todos juntos. Dessa forma, após a visualização dos pesos de todos os grupos consenso, o aplicador pode ajustar o valor máximo do terceiro gráfico como sendo o valor do maior somatório encontrado entre todos os grupos consenso. Portanto, a comparação desses grupos que no momento estão sendo analisados, com outros grupos consenso é possível graças a esses terceiro gráfico.

A quantidade de linhas do gráfico de peso das permissões é igual a quantidade de blocos que aparecem nos gráficos de permissão, pois ambos são a representação de um grupo consenso e portanto, estão intimamente ligados com referência a essa variável, ou seja, caso o aplicador varie a quantidade de aplicativos, automaticamente os blocos que representam os grupos de consenso no gráfico de permissões será alterado e consequentemente as linhas do gráfico de pesos das permissões também mudará. Para mostrar esse comportamento, podemos fazer a ligação entre a Figura 7 com a Figura 11, a Figura 8 com a Figura 12 e a Figura 9 com a Figura 13.

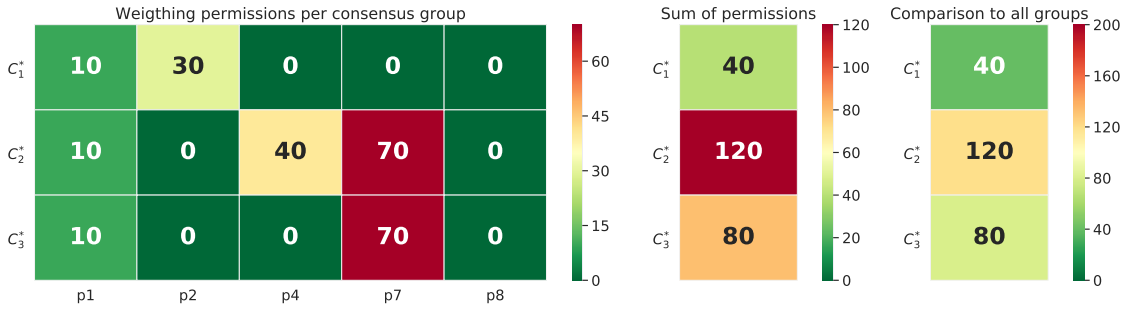


Figura 12 – Visualização dos pesos das permissões com grupos de consenso com 2 aplicativos

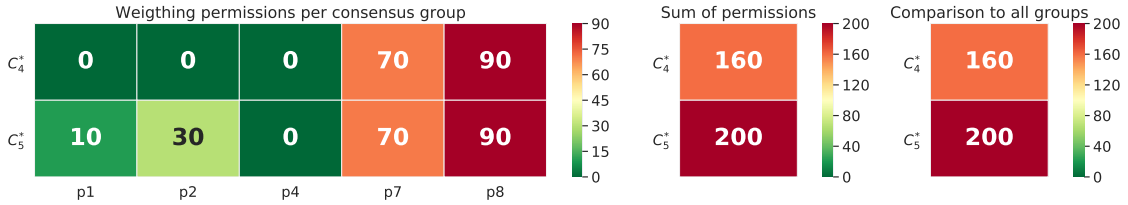


Figura 13 – Visualização dos pesos das permissões com grupos de consenso com 3 e 5 aplicativos

Uma outra variável que pode mudar a visualização dos pesos das permissões é a representatividade ou a frequência que com que uma permissão aparece em um grupo consenso. Se utilizarmos a Figura 11 como referência, onde o limiar da representatividade das permissões foi de 40%, pode-se perceber a mudança deste ao alterar esse valor como são mostrados nas Figuras 14, 15, 16, 17 e 18 com os parâmetros de representatividade ajustados com os valores de 10%, 25%, 50%, 75% e 100% respectivamente.

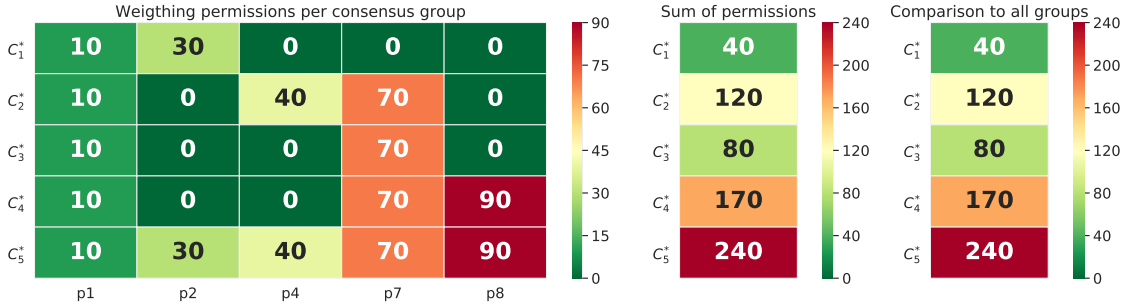


Figura 14 – Visualização de peso das permissões com parâmetro de representatividade igual a 10%

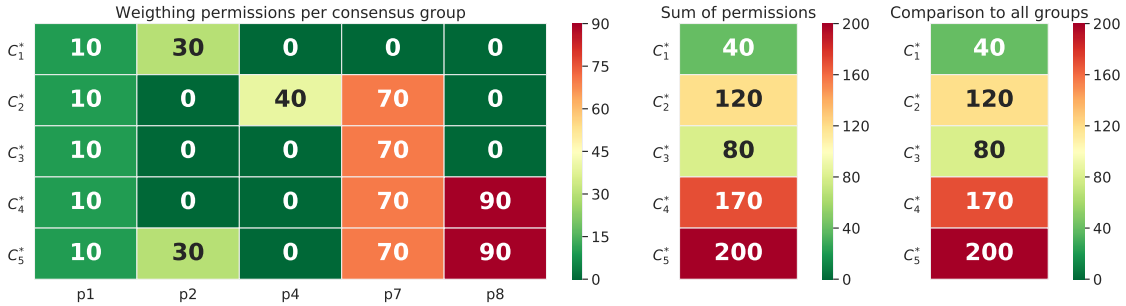


Figura 15 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 25%

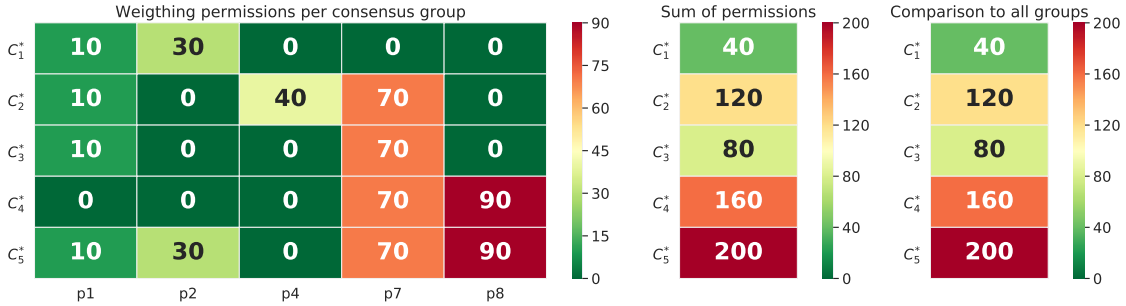


Figura 16 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 50%

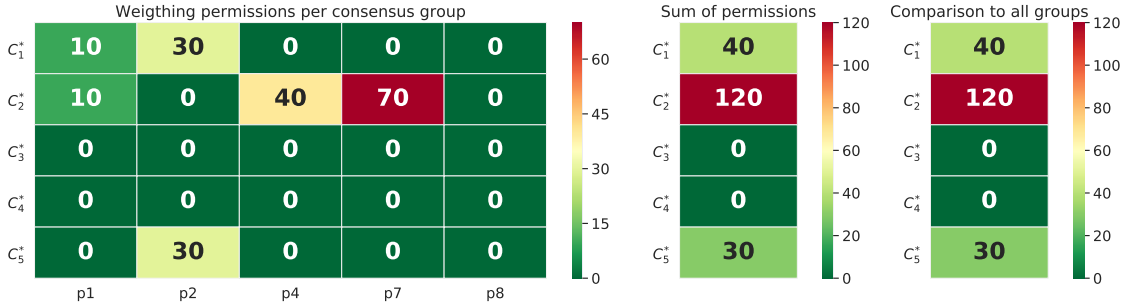


Figura 17 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 75%

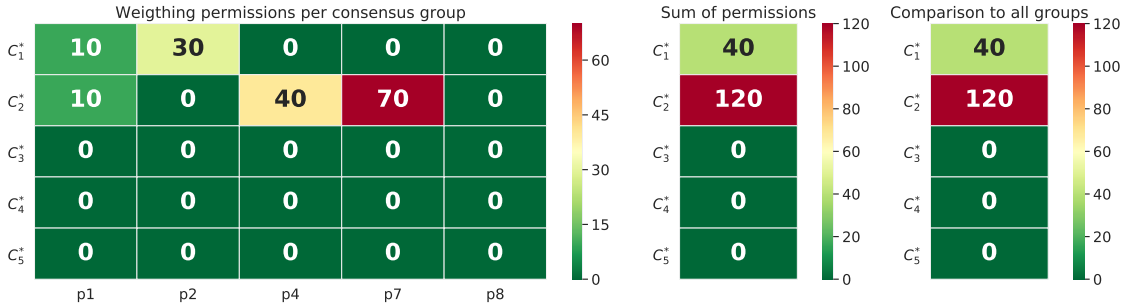


Figura 18 – Visualização dos pesos das permissões com parâmetro de representatividade igual a 100%

4 Aplicação do método

A aplicação do método baseou-se no estudo das permissões para fazer a análise energética dos aplicativos, ou seja, o quanto um aplicativo pode ser eficiente gastando menos bateria.

O entendimento da relação entre os recursos usado pelos aplicativos e o consumo de bateria pode fornecer informações valiosas para fabricantes do dispositivo, desenvolvedores de aplicativos e pesquisadores, que podem tomar medidas e conduzir pesquisas para melhoria de desempenho e consequentemente, proporcionar uma melhor experiência dos usuários com seus aparelhos.

Baseando-se no estudo de eficiência energética, foi utilizado uma base de dados de mais de 400.000 aplicativos dos quais foram obtidos a permissão de aproximadamente 95.000 aplicativos. As permissões empregadas para este estudo são as do projeto AOSP, já discutidas na Seção 2.1.2.

Na Seção 4.1, que fala sobre a coleta de dados, é apresentada a base de dados, sua estrutura e como ela foi coletada. Em identificação dos perfis de permissão na Seção 4.2, são demonstrados como os dados foram formatados, a aplicação do cálculo de distância e da obtenção do grupos consenso. Finalmente, na Seção 4.3 são mostradas as visualizações compostas pelos gráficos de perfis de permissões e pela visualização dos pesos de permissões.

Todo este estudo, tem como objetivo final, encontrar relações das permissões com o consumo energético dos aplicativos que estão diretamente relacionados com o consumo de bateria.

4.1 Coleta dos dados

A base de dados utilizada para a aplicação do método proposto foi retirada do sítio *Kaggle*[1] e contém um total de 400.000 aplicativos obtidos da *Google Play*

[1] <https://www.kaggle.com/orgesleka/android-apps>

Store e cada aplicativo contém as seguintes informações:

- nome;
- data de publicação;
- número de *downloads*;
- tamanho do arquivo;
- nome do pacote;
- classificação agregada;
- versão do *software*;
- contagem de classificação;
- data da coleta dos dados;
- url.

Obtida a base de dados, foi executado um *script*[2] para coletar as permissões referentes a cada aplicativo.

Para coletar essas permissões utilizou-se como referência o nome do pacote do aplicativo no comando `python permissions.py <nome do pacote>`, obtendo assim as permissões, sem a necessidade de fazer o *download* do aplicativo. Porém, nem todos os aplicativos pesquisados retornaram suas respectivas permissões, resultando em aproximadamente 95.000 amostras, o equivalente a 23,75% do total de aplicativos da lista original.

Finalizado a coleta das permissões, foi necessário desenvolver um algoritmo para transformar a saída desse *script* que coletava as permissões, num arquivo do tipo csv que apresentasse a estrutura da forma [nome_aplicativo][permissão 1 $\cdots$ permissão n] e foi removido as permissões do tipo `<permission android:name="<package-name>.permission.C2D_MESSAGE"` que são utilizadas para executar *push notifications* nos aplicativos e que atualmente

[2] <https://github.com/egirault/googleplay-api>

são obsoletas no desenvolvimento para Android para depois finalmente convertê-lo numa matriz binária.

4.2 Identificação dos Perfis de Permissão

É nesta etapa que é feito todo o processamento necessário para identificar os perfis de permissão, da formatação dos dados coletados, ao cálculo de similaridade, da definição do número de grupos que pretende-se utilizar no algoritmo de agrupamento e na execução deste e, finalmente a obtenção e identificação dos grupos consenso. Toda esta etapa foi executada utilizando a ferramenta *Jupyter Notebook* e as várias bibliotecas que podemos utilizar nesta ferramenta, como *Numpy*, *Scipy*, entre muitas outras.

4.2.1 Formatação dos dados coletados

A formatação do dados foi feita convertendo o arquivo csv da etapa final da coleta de dados, primeiro, verifica-se se a permissão existe no projeto AOSP, se sim, ela é incluída na lista de permissões — caso contrário, a permissão é eliminada — após, procura-se a palavra `token`, que segundo a documentação oficial[3] deve ser utilizada quando é necessário acesso a mensagens SMS para receber códigos de autenticação que protejam o usuário contra fraude, o que não é objeto deste estudo pois não possui o padrão `android.permission` e não impacta a análise de eficiência energética para esta aplicação do método. Após isso, é feita a leitura linha a linha até contabilizar o aplicativo que contém o maior número de permissões (126 no caso da base de dados da aplicação) e transformando-o num *pandas.Dataframe*[4], dessa forma, obtém-se uma matriz de 95.000 linhas por 200 atributos correspondentes às permissões que estão contidas no projeto AOSP. Porém, apesar de todos os aplicativos terem agora 200 atributos, a maioria deles contém o valor `NaN` que depois ao serem convertidos para a matriz binária, apresentarão o valor 0 (zero). Um exemplo ilustrativo é mostrados nas Tabelas 8 e 9.

[3] <https://developer.android.com/training/permissions/usage-notes>
[4] <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>

	0	1	2	3
0	idteam.app.crazypianofree	android.permission.ACCESS_NETWORK_STATE	android.permission.INTERNET	android.permission.MODIFY_AUDIO_SETTINGS
1	com.extremeenjoy.deutschnews	android.permission.ACCESS_NETWORK_STATE	android.permission.INTERNET	android.permission.RECEIVE_BOOT_COMPLETED
2	com.extremeenjoy.news.kannadacinema	android.permission.ACCESS_NETWORK_STATE	android.permission.INTERNET	android.permission.RECEIVE_BOOT_COMPLETED
3	com.vivek.batteryfloater	android.permission.READ_EXTERNAL_STORAGE	android.permission.READ_PHONE_STATE	android.permission.SYSTEM_ALERT_WINDOW
4	coolcherrytrees.games.reactor4	android.permission.INTERNET	android.permission.WAKE_LOCK	NaN
5	com.mobikolik.tumgazeteler	android.permission.ACCESS_NETWORK_STATE	android.permission.INTERNET	android.permission.WAKE_LOCK
6	com.miniclip.gravityguy	android.permission.ACCESS_COARSE_LOCATION	android.permission.ACCESS_NETWORK_STATE	android.permission.INTERNET
7	com.digitalnatives.taxilike	android.permission.ACCESS_COARSE_LOCATION	android.permission.ACCESS_FINE_LOCATION	android.permission.ACCESS_NETWORK_STATE
8	com.unbound.android.ubmmi	android.permission.ACCESS_NETWORK_STATE	android.permission.GET_ACCOUNTS	android.permission.GET_TASKS
9	com.gtomato.talkbox	android.permission.ACCESS_COARSE_LOCATION	android.permission.ACCESS_FINE_LOCATION	android.permission.ACCESS_NETWORK_STATE

10 rows × 126 columns

Tabela 8 – Recorte das primeiras colunas da tabela

8	9	...	116	117	118	119	120	121	122	123	124	125
com.android.launcher.permission.INSTALL_SHORTCUT	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
com.android.vending.BILLING	com.google.android.c2dm.permission.RECEIVE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
android.permission.READ_EXTERNAL_STORAGE	android.permission.READ_PHONE_STATE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
android.permission.WAKE_LOCK	android.permission.WRITE_EXTERNAL_STORAGE	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
android.permission.READ_CALL_LOG	android.permission.READ_CONTACTS	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Tabela 9 – Recorte das últimas colunas da tabela

Após a criação da tabela mostrada nas Tabelas 8 e 9, é feita sua transformação de forma a representar uma matriz binária em que as linhas são todos os aplicativos e as colunas todas as permissões, tendo essas colunas valores 1 (um) representado a presença da permissão e 0 (zero) representado a falta dessa permissão. Para isso foram utilizadas as ferramentas *pandas.Dataframe*[5] e *pandas.crosstab*[6] Em termos de tamanho de arquivo, a matriz binária tem tamanho aproximado a 2 GB em formato csv[7,8], porém, para efeitos de economia de espaço e pela necessidade de maior desempenho na leitura do arquivo, ao invés de utilizar o arquivo em formato csv, foi decidido utilizar o formato npy[9,10]. Na Tabela 10 podemos ver um recorte da matriz binária.

5 <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>
6 <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.crosstab.html>
7 <https://tools.ietf.org/html/rfc4180>
8 <https://en.wikipedia.org/wiki/Comma-separated_values>
9 <https://numpy.org/devdocs/reference/generated/numpy.lib.format.html>
10 <https://github.com/numpy/numpy/blob/master/numpy/lib/format.py>

1 0	access_cache_filesystem	access_checkin_properties	access_coarse_location	access_content_providers_externally
ADG.Bank	0	0	0	0
AKnght.Studios.Kids123Lite	0	0	0	0
AKnght.Studios.MW3Calc	0	0	0	0
AR.Qibla.Finder	0	0	1	0
Abbanza.Arrived.Android	0	0	0	0
Abbanza.Bixpe.DispositivosPCI.Android	0	0	0	0
Air.Density	0	0	0	0
Air.Lite	0	0	0	0
Android.ComicData	0	0	0	0
Android.Page	0	0	0	0

10 rows × 200 columns

Tabela 10 – Recorte da matriz binária

4.2.2 Cálculo da distância cosseno

O cálculo da similidade é a etapa que utilizou-se da distância cosseno para construir uma matriz quadrada de 95.000×95.000 com a distância entre os aplicativos. Nesta matriz é calculado os aplicativos que têm permissões parecidas, ou seja, àqueles que tendem a 0 (zero) são os aplicativos que mais assemelham-se em relação às permissões que utilizam; e os que tendem a 1(um) são àqueles aplicativos que não compartilham poucas ou nenhuma permissão entre eles. Dessa forma, gerou-se um arquivo csv de 100 GB de tamanho que posteriormente foi convertido num arquivo `npy` com 36 GB de tamanho.

A utilização da similaridade por cosseno (HAN; PEI; KAMBER, 2011), tem referência à sua utilização na verificação de semelhança entre textos (CROFT et al., 2013) e melhor que outras medidas como *jaccard* e *dice* (THADA; JAGLAN, 2013) e na pesquisa de documentos. Pelas razões citadas anteriormente, essa medida de similaridade foi escolhida por este estudo que também apresentou ótimo comportamento referente à comparação dos atributos binários contido na matriz de aplicativos por permissões.

Para a aplicação do método, o cálculo de similaridade é de extrema importância e a etapa que demanda maior recurso computacional. Devido ao tamanho do arquivo, gerado como resultado do seu cálculo, foi necessária a utilização de uma máquina de 768 GB de RAM e um tempo de processamento aproximado de 2 horas para a execução do cálculo.

Apesar da necessidade da utilização de um recurso computacional alto, o cálculo de similaridade é executado apenas uma única vez, tornando-se base para a execução do algoritmo de agrupamento que, demandando menor custo computacional para este estudo de caso, pôde ser executado diversas vezes, facilitando assim a determinação empírica da quantidade de grupos a serem parametrizados.

4.2.3 Obtenção das partições base no algoritmo de agrupamento

A definição de quantidades de grupos para este estudo de caso foi determinado empiricamente, iniciando em $k = 2$ e aumentando unitariamente até a quantidade de $k = 18$, obtendo-se assim, 16 partições que visualmente satisfizessem

na etapa posterior ao algoritmo de agrupamento na obtenção e visualização dos grupos de consenso.

O algoritmo de agrupamento utilizado para este estudo de caso foi o do *k-medoids*[11], mais precisamente sua variação chamada PAM(*Partitioning Around Medoids*) (KAUFMAN; ROUSSEEUW, 1990). Sua escolha neste estudo de caso foi devido a sua menor sensibilidade a *outliers* e por ser um algoritmo relativamente rápido na sua execução, tanto que o seu tempo de execução foram de alguns poucos minutos utilizando a matriz binária no formato `npy`, onde anteriormente demorou aproximadamente 2 horas para ser calculada.

Como mencionado na seção anterior, foram obtidos 16 partições com grupos variando unitariamente de $k = 2$ a $k = 18$, gerando 16 arquivos representando um conjunto de partições base e que posteriormente foram utilizados para obtenção dos grupos de consenso.

4.2.4 Obtenção dos grupos consenso

Os grupos consenso representam a novidade desse estudo e são extremamente importantes para a geração dos gráficos de perfis de permissão e da visualização dos pesos das permissões, pois sem a obtenção desses grupos de consenso não há a possibilidade de chegar a uma conclusão do estudo das permissões e por fim não há como tirar outras conclusões em relação ao comportamento, as funcionalidades, o desempenho ou a eficiência energéticas desses aplicativos.

Neste trabalho, isso foi feito utilizando de um *script* que faz parte da ferramenta de visualização *CVis* (FACELI; SAKATA; HANDL, 2017), e que produz como saída um arquivo único em que cada linha deste arquivo representa os elementos(nesse caso aplicativos) de um grupo de consenso ou aplicativos não agrupados. O arquivo resultante continha 4752 linhas, das quais 2361 eram grupos consenso e 2391 eram aplicativos não agrupados. Do total de 2361 grupos consenso, foram considerados para análise um total de 530, essa escolha foi baseada na análise da distribuição dos aplicativos. Para isso foram utilizadas as Figuras 19 e 20. A

[11] <https://github.com/arthurfortes/ClusteringAlgorithms/blob/master/clustering/kmedoids.py>

Figura 19 mostra que a distribuição de quantidade de aplicativos por quantidade de grupos de consenso está mais concentradas no canto inferior esquerdo do gráfico, ampliando esta parte do gráfico sucessivas vezes até obtermos uma distribuição satisfatória, obtemos então a Figura 20 onde a quantidade mais significativa estava distribuída entre os grupos consenso que variava de 14 a 341 aplicativos.

Dessa forma, os grupos de consenso foram categorizados pelo tamanho para uma melhor visualização futura. Essas categorias são mostrados na Tabela 11. A conveniência da categorização dos grupos de consenso deve-se a enorme quantidade de dados gerados, o que causaria posteriormente muita dificuldade na visualização dos gráficos e na produção de conhecimento.

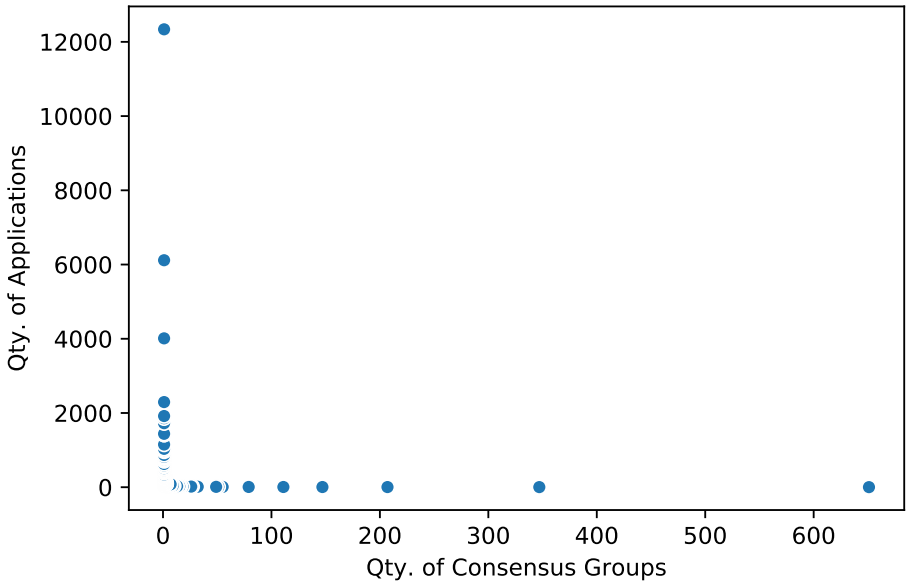


Figura 19 – Distribuição dos aplicativos no arquivo de 4752 linhas

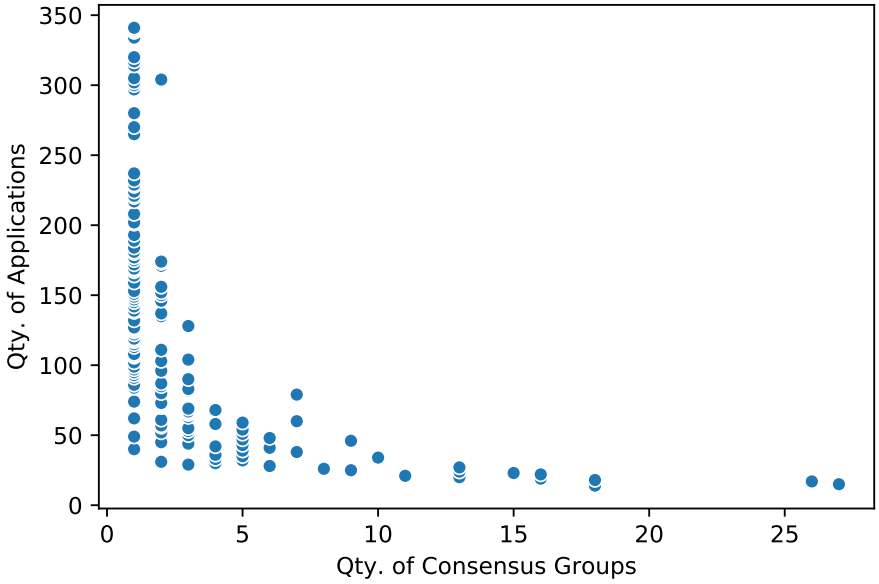


Figura 20 – Distribuição dos aplicativos selecionados para as categorias

Descrição dos Grupos Consenso		
Categoria	**Tamanho dos grupos consenso**	**Quantidade de grupos consenso**
A	14–15	45
B	16–17	52
C	18–20	47
D	21–24	55
E	25–33	54
F	34–43	54
G	44–58	55
H	59–80	52
I	81–100	22
J	101–140	33
K	141–180	29
L	181–220	09
M	221–300	11
N	301–334	08
O	335–341	04
Total		**530**

Tabela 11 – Distribuição dos aplicativos em grupos de consenso

4.3 Visualização

A visualização é a etapa do método que é composta pelos gráficos de permissões e da visualização dos pesos das permissões. Ambos se baseiam nos grupos consenso encontrados e se complementam para a produção do conhecimento do objeto ou do assunto em estudo.

No total foram gerados 15 gráficos de permissões de perfis, onde seus recortes (por serem gráficos muito extensos) podem ser visualizados no Apêndice A, e 15 visualizações de pesos das permissões, mostrados no Apêndice B, além disso, eles podem ser vistos na íntegra em formato `png` ou `pdf` no sítio <https://github.com/fguiraldelli/master_thesis/tree/msc_dissertation/googleplay-api/Final_Graphs/pictures>.

Devido a grande quantidade de dados tratados e analisados, os grupo consensos foram categorizados pelo tamanho, conforme já mostrado na Tabela 11. Portanto, cada gráfico gerado, corresponde a uma categoria que por sua vez, cada categoria tem um tamanho limitado de aplicativos e uma quantidade de blocos limitada. Todo esse trabalho de separação foi realizado exatamente para otimizar a visualização, pois se blocos de tamanhos distintos fossem gerados dentro de um mesmo gráfico de permissão, poderia-se gerar uma distorção gráfica na apresentação dos blocos, onde os maiores blocos seriam achatados no gráfico, impactando assim a visualização. Dessa forma, as categorias foram criadas tanto para melhorar a estética da visualização, quanto para modular a grande quantidade de informações, facilitando assim o seu entendimento.

Em um bloco do gráfico de perfis de permissões, como mostrado na Figura 21, é possível ver, de uma só vez, os aplicativos de um grupo consenso e as 200 permissões (selecionadas aplicando a intersecção das 456 permissões do projeto AOSP com as permissões que foram encontradas em todos os aplicativos). Esse bloco permite a fácil visualização das permissões, onde é possível ver um padrão conciso na distribuição das permissões e também identificar permissões consideradas *outliers* naquele grupo consenso. Por sua vez, os gráficos de permissões tem a característica de serem extensos, sendo compostos neste estudo de caso, por até 55 blocos em sua composição.

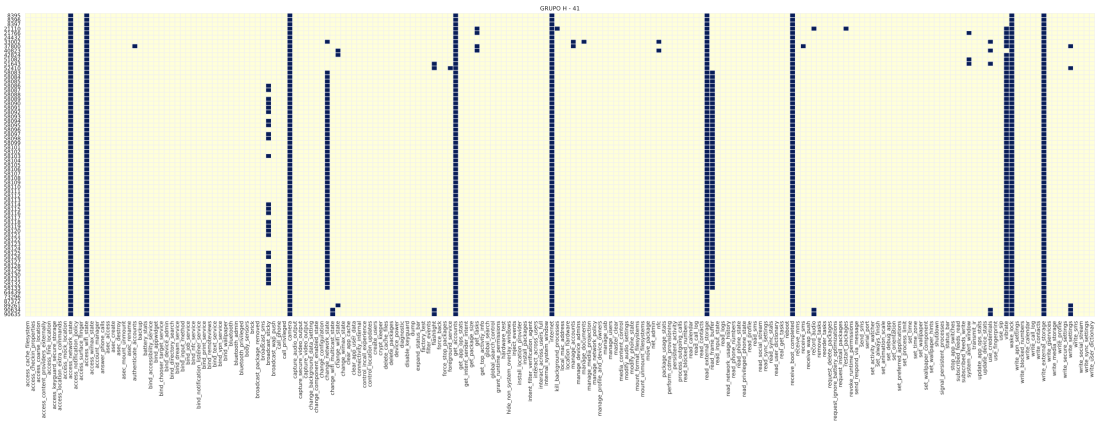


Figura 21 – Bloco H-41 do gráfico de permissões da categoria H

Na visualização de pesos das permissões, gerados a partir do peso dado às permissões e ao somatório delas, conforme a Figura 22, está representada toda uma categoria, pois suas linhas representam cada bloco do gráfico de permissões, o que o torna um gráfico bem menos extenso e mais otimizado na visualização.

Para a determinação das permissões que seriam selecionadas para a construção da visualização de pesos das permissões, houve o embasamento na literatura, considerando como maiores consumidores de baterias aquelas permissões diretamente ligadas com os recursos de tela, cpu, *wi-fi*, *modem* $3G/4G$, gps, camera, vibrador, luz de flash, saída de som, microfone, leitura e gravação nas memórias não-voláteis, *NFC*, *bluetooth*, transmissor de infravermelho (DICKT, 2010; YOON et al., 2012; CHEN et al., 2013; DUAN et al., 2013). Dessa forma, foram definidas 20 permissões que estão diretamente ligadas a esses recursos. Além disso, foram definidas suas respectivas ponderações baseando-se nos mesmos princípios pelos quais essas permissões foram escolhidas e de forma a quantificar esses pesos foram definidos valores de 10 a 100, onde os valores são diretamente proporcionais ao impacto no consumo de bateria, mostrados na Tabela 12.

Para a sua construção, durante a análise dos gráficos de perfis de permissão, foi tomada a decisão que permissões com representatividade menor que 40% seriam excluídas da análise. A determinação deste número foi determinada pela forma que muitas permissões estavam distribuídas ao longo dos gráficos de permissões, dando assim a certeza que muitas permissões consideradas *outliers* seriam eliminadas e de

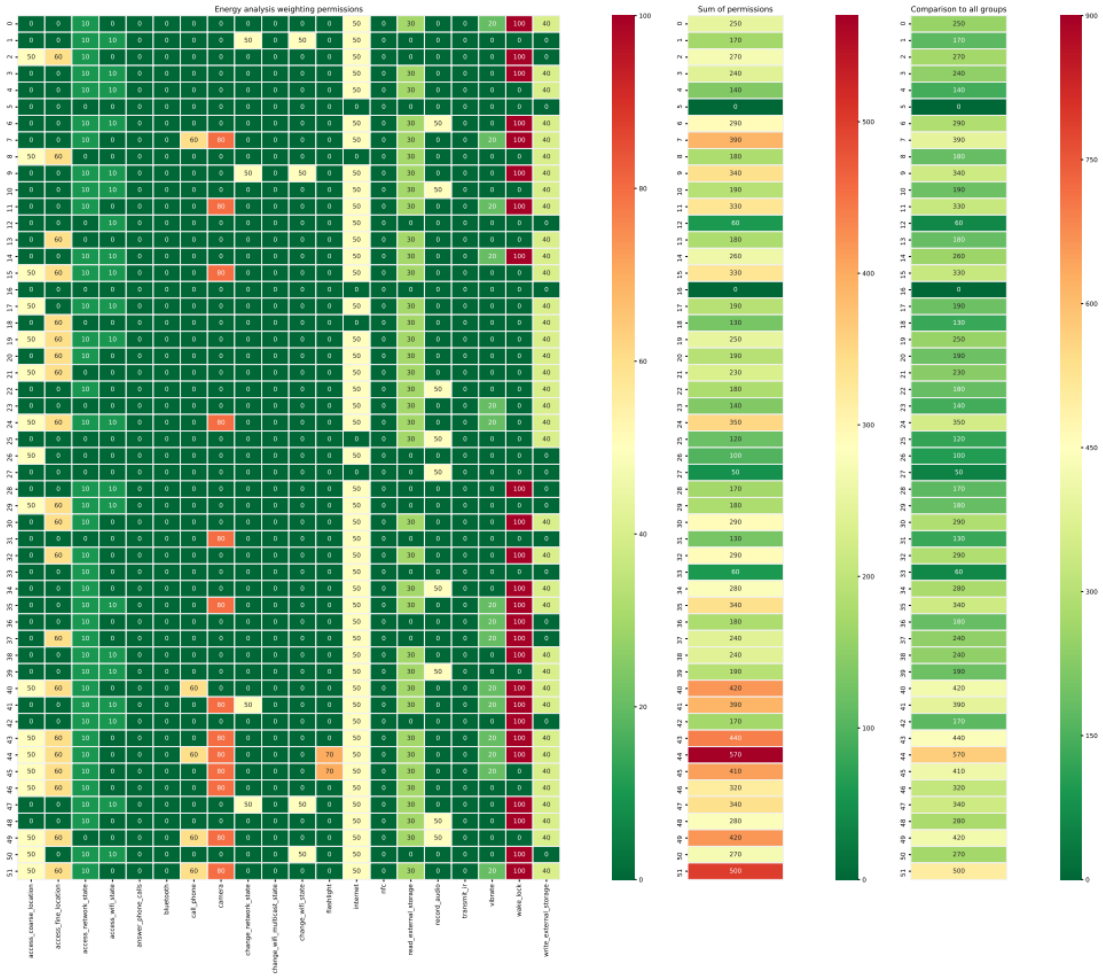


Figura 22 – Visualização dos pesos das permissões da categoria H.

forma empírica, outras porcentagens de 25%, 30% e 50% foram também testadas ao longo da construção do gráficos, porém, visualmente a representatividade de 40% apresentava-se como um limiar ideal para o gráfico de pesos.

A análise desses gráficos, num contexto de um fabricante de dispositivos móveis, poderia dar-lhe informações pertinentes sobre os grupos que consomem mais energia e nesse sentido, este fabricante poderia encontrar soluções ou otimizar seus dispositivos de forma que na execução desses aplicativos, haja um menor consumo energético possível. Por exemplo, aprofundando-se na análise das visualizações, percebe-se que a maior somatória dos pesos tem valor de 760, relacionados a

Permissões	**Valor dos Pesos**
access_coarse_location	50
access_fine_location	60
access_network_state	10
access_wifi_state	10
answer_phone_calls	60
bluetooth	10
call_phone	60
camera	80
change_network_state	50
change_wifi_multicast_state	50
change_wifi_state	50
flashlight	70
internet	50
nfc	20
read_external_storage	30
record_audio	50
transmit_ir	10
vibrate	20
wake_lock	100
write_external_storage	40

Tabela 12 – Permissões selecionadas que influenciam no consumo de baterias e seus pesos

categoria A, mais precisamente o grupo A-20, vindo em segundo lugar com valor dos pesos em 670, as categorias E e F, com os grupos E-32 e F-44. As outras categorias apresentaram valores menores, mesmo contendo uma grande quantidade de aplicativos, pode-se visualizar os valores máximos e mínimos relacionados às categorias na Tabela 13

Ao aprofundar-se nos grupos citados do parágrafo anterior o grupo A-20 tem 50% dos aplicativos relacionados com gravadores de voz e um aplicativos da chama atenção por ser o *WeSync*, um aplicativos da empresa *Tencent*, que é a criadora do aplicativo *WeChat*, a versão chinesa do *WhatsApp*. O grupo E-32 tem 57,7% dos seu aplicativos com o nome do pacote finalizado com as letras **im**, do inglês *instant message*, ou seja, de aplicativos de mensagens. E finalmente, no grupo F-44, pelo menos 41% dos aplicativos são relacionados a segurança como os antivírus da

Valores de Pesos das Categorias		
Categoria	**Peso mínimo**	**Peso Máximo**
A	0	760
B	0	550
C	0	480
D	50	500
E	0	670
F	0	670
G	0	450
H	0	570
I	10	450
J	80	570
K	0	570
L	90	350
M	60	390
N	70	460
O	100	240

Tabela 13 – Valores mínimos e máximos encontrados nas categorias

Norton e Avira e também um aplicativo anti-roubo da empresa Avast.

O fabricante pode gerar um gráfico de perfis de permissão e uma visualização de pesos especificadamente para esses 3 grupos, podendo então focar melhor nas permissões que esses aplicativos são compostos, como mostrado nas Figuras 23 e 24, tendo assim, uma grande quantidade de informações interessantes para refletir e produzir conhecimento a partir de sua análise, criando assim, planos de ação a fim de lidar essas permissões de forma a otimizar seus processamentos, diminuindo dessa forma o custo no consumo da bateria.

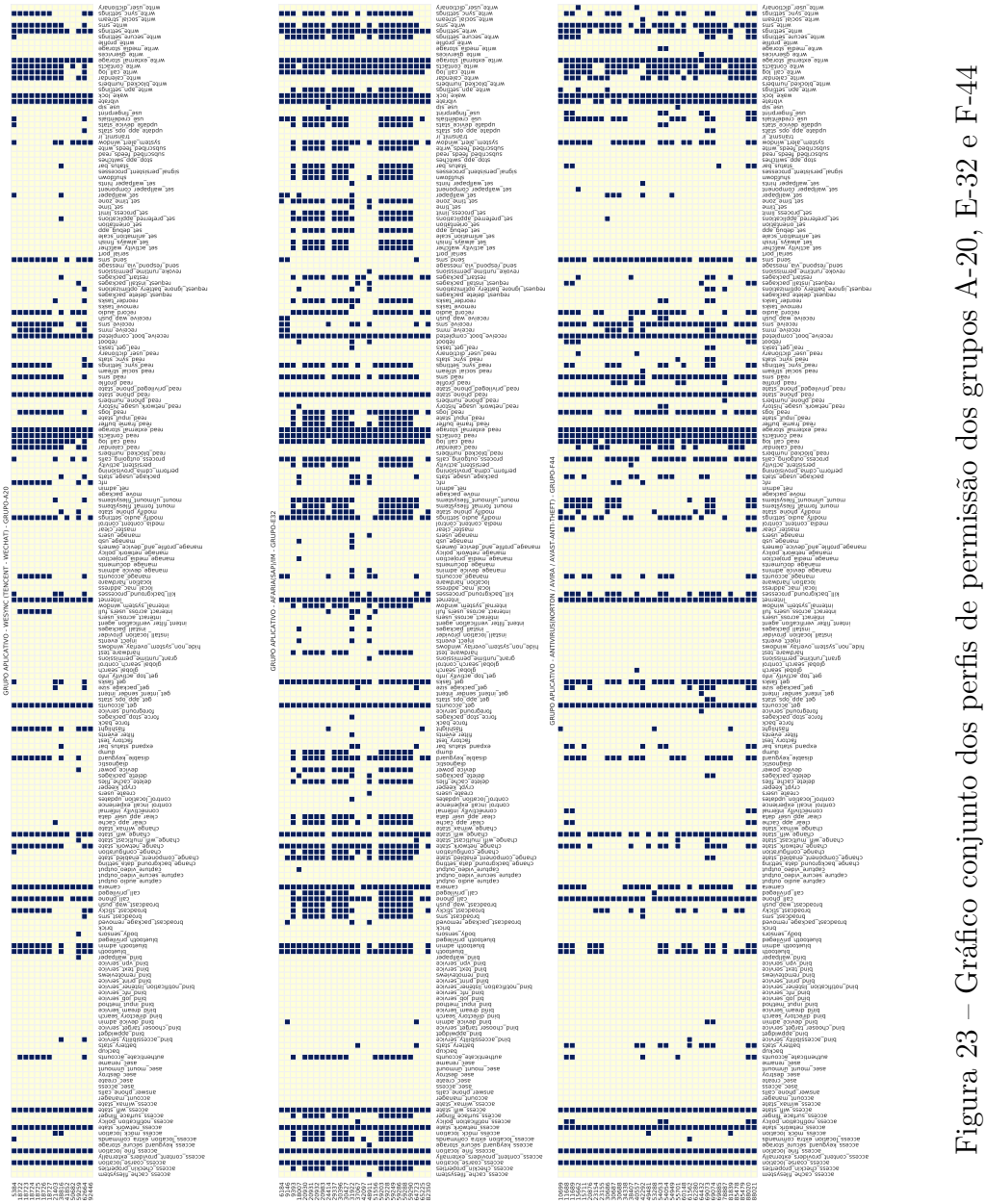


Figura 23 – Gráfico conjunto dos perfis de permissão dos grupos A-20, E-32 e F-44

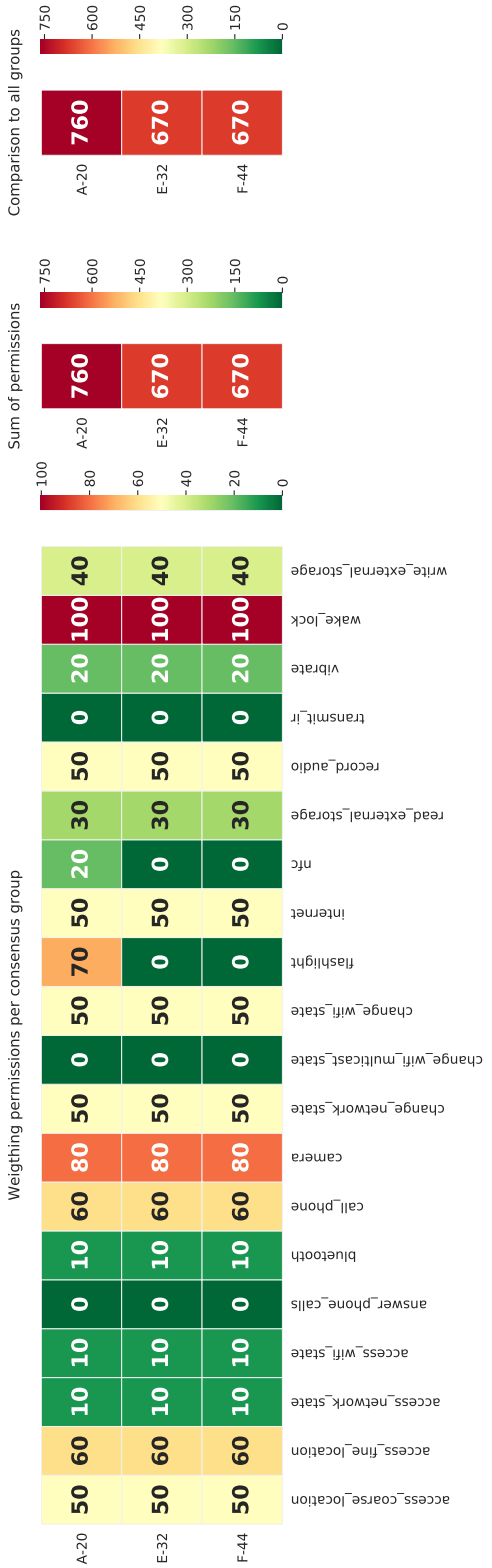


Figura 24 – Visualização conjunta dos pesos das permissões dos grupos A-20, E-32 e F-44

5 Conclusão

Este trabalho teve como objetivo propor um método para identificação e comparação de perfis de permissões que permite classificar e comparar os aplicativos, sendo possível assim, utilizar as permissões para ajudar a determinar diversas características, anomalias e funcionalidades que desenvolvedores, fabricantes e pesquisadores podem explorar a fim de melhorar seus produtos ou mesmo descobrir novas formas de analisar o comportamento desses aplicativos.

Nesse método é possível identificar relações entre as permissões com o desempenho, funcionalidade, comportamento, recursos de *hardware* e inferir consumo de energia e gastos de baterias. Ao classificar os aplicativos pelas permissões, criou-se um método que identifica e compara os perfis de permissões.

Além disso, foi proposto aqui um novo conceito chamado de grupo consenso, baseado nas ideias de *ensembles* de agrupamento, integrando a utilização de métodos de agrupamentos, visualização e o próprio *ensembles* de agrupamento, para identificar semelhanças dos aplicativos e permitindo uma análise comparativa visual.

O método foi avaliado por um estudo de caso que analisou teoricamente o consumo de energia dos aplicativos, passando por todas as etapas da coleta, tratamento desses dados, identificação dos perfis de permissão, finalizando com a visualização desses perfis e da análise final. Dessa forma, foi mostrada a sua eficácia na classificação das permissões, mostrando grupos que apresentam elevado consumo energético, como os aplicativos de mensagens instantâneas, aplicativos ligados a segurança e antivírus e até aplicativos ligados a grandes empresas como a Tencent, criadora do aplicativo *WeChat*, a versão chinesa do *WhatsApp*.

A parte de visualização dos gráficos de permissões apesar de serem extensas, apresentaram melhor visualização a respeito das permissões ao invés da utilização de tabelas ou planilhas utilizadas para esse mesmo fim, além de facilitar a detecção visual da quantidade de permissões utilizadas pelos aplicativos e a representatividade delas em relação ao grupo consenso, representado pelos blocos do gráficos.

Na visualização dos pesos das permissões ficou claro o impacto das permissões nos grupos de consenso, o que provê todo o arcabouço necessário para uma análise e produção do conhecimento necessários para atingir metas e objetivos relacionados com o uso das permissões. A relação delas com a categoria que está sendo analisada e sua comparação com outras categorias.

Com a utilização dos grupos consenso para a identificação de perfis de permissão, podemos ainda propor outros tipos de aplicações para este método, como por exemplo, detecção de *malwares*, comparação com outros aplicativos com fins semelhantes, verificação da qualidade das escolhas de permissões que estão diretamente ligadas a implementação de aplicativos móveis, classificação de aplicativos com base nas permissões e muitas outras aplicações, que podem ajudar pesquisadores, desenvolvedores de aplicativos e fabricantes de dispositivos móveis a melhorar seus produtos finais, contribuindo assim para uma melhora na qualidade de seus trabalhos.

Como trabalhos futuros seria interessante a comprovação da teoria levantada neste método em relação ao consumo de energia, verificando por meio de medições em laboratório a efetividade da visualização do pesos das permissões, suas características apresentadas e o impacto que elas causam ao consumo energético dos aplicativos, além de outras formas de aplicação citadas no parágrafo anterior.

Por fim, diferente tecnologias, sistemas operacionais e aplicativos, sempre estão em constante mudanças e evoluções. O que implica em sempre ficar atento com o dinamismo que é trazido com as novas versões do Android, onde são depreciadas antigas permissões e traz-se à tona novas permissões e procedimentos, tornando-se assim, um constante desafio por parte dos desenvolvedores, pesquisadores e fabricantes de dispositivos móveis sempre estarem atualizados e bem preparados para futuras pesquisas e análises. Para isso, propõem-se criar meios de coletar, atualizar e armazenar dinamicamente e de forma constante os dados referentes às permissões e aos aplicativos, evitando assim que o método proposto neste trabalho seja preterido.

Finalmente, todas as contribuições relativas a este estudo estão disponibilizadas integralmente em <https://github.com/fguiraldelli/master_thesis/tree/msc_

dissertation>, sendo seu conteúdo livre para alterações e atualizações. Além disso, será produzido um artigo relatando esta pesquisa, artigo este que será submetido à iSys - Revista Brasileira de Sistemas de Informação[1].

[1] <http://www.seer.unirio.br/index.php/isys>

Referências

AHMAD, R. W. et al. A survey on energy estimation and power modeling schemes for smartphone applications. *International Journal of Communication Systems*, v. 30, n. 11, p. 1–22, 2017. ISSN 10991131. Citado 2 vezes nas páginas 30 e 31.

AMINE, A.; ELBERRICHI, Z.; SIMONET, M. Evaluation of text clustering methods using wordnet. *Int. Arab J. Inf. Technol.*, v. 7, n. 4, p. 349–357, 2010. Citado na página 44.

ARORA, P.; VARSHNEY, S. et al. Analysis of k-means and k-medoids algorithm for big data. *Procedia Computer Science*, Elsevier, v. 78, p. 507–512, 2016. Citado na página 44.

BARBARA, D. An introduction to cluster analysis for data mining. *Retrieved March*, v. 13, p. 2006, 2000. Disponível em: <https://www-users.cs.umn.edu/~hanxx023/dmclass/cluster_survey_10_02_00.pdf>. Citado na página 22.

CARROLL, A.; HEISER, G. An analysis of power consumption in a smartphone. *Proceedings of the 2010 USENIX conference on USENIX annual technical conference*, p. 21–21, 2010. Disponível em: <http://portal.acm.org/citation.cfm?id=1855861>. Citado 2 vezes nas páginas 15 e 30.

CHEN, X. et al. How is Energy Consumed in Smartphone Display Applications? *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications*, v. 1, n. 412, p. 3:1—-3:6, 2013. Disponível em: <http://doi.acm.org/10.1145/2444776.2444781>. Citado 3 vezes nas páginas 16, 30 e 67.

CHOI, S.-S.; CHA, S.-H.; TAPPERT, C. C. A survey of binary similarity and distance measures. *Journal of Systemics, Cybernetics and Informatics*, Citeseer, v. 8, n. 1, p. 43–48, 2010. Citado na página 43.

CROFT, D. et al. A fast and efficient semantic short text similarity metric. In: IEEE. *2013 13th UK Workshop on Computational Intelligence (UKCI)*. [S.l.], 2013. p. 221–227. Citado na página 62.

CRUZ, L.; ABREU, R. Performance-based Guidelines for Energy Efficient Mobile Applications. *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, p. 46–57, 2017. Disponível em:

<https://doi.org/10.1109/MOBILESoft.2017.19>. Citado 3 vezes nas páginas 15, 16 e 30.

DICKT, R. P. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. 2010. Citado na página 67.

DUAN, L. T. et al. Energy analysis and prediction for applications on smartphones. *Journal of Systems Architecture*, Elsevier B.V., v. 59, n. 10 PART D, p. 1375–1382, 2013. ISSN 13837621. Disponível em: <http://dx.doi.org/10.1016/j.sysarc.2013.08.011>. Citado 3 vezes nas páginas 16, 30 e 67.

FACELI, K. et al. *Inteligência artificial: uma abordagem de aprendizado de máquina*. [S.l.]: LTC, 2011. 396 p. Citado 2 vezes nas páginas 22 e 23.

FACELI, K. et al. Inteligência artificial: Uma abordagem de aprendizado de máquina. 2011. Citado na página 29.

FACELI, K. et al. Pvis—partitions' visualizer: Extracting knowledge by visualizing a collection of partitions. In: IEEE. *2014 International Joint Conference on Neural Networks (IJCNN)*. [S.l.], 2014. p. 3056–3061. Citado na página 30.

FACELI, K.; SAKATA, T. C.; HANDL, J. CVis - towards a novel visualization tool to explore the relationship between input and output partitions in multi-objective clustering ensembles. 2017. Citado 3 vezes nas páginas 27, 30 e 63.

FRASER, S. D.; HACKER, V.; KORDESCH, K. Fuel Cells – Alkaline Fuel Cells | Cells and Stacks. *Encyclopedia of Electrochemical Power Sources*, p. 344–352, 2009. Disponível em: <http://www.sciencedirect.com/science/article/pii/B9780444527455002847>. Citado na página 21.

GORLA, A. et al. Checking app behavior against app descriptions. *Proceedings of the 36th International Conference on Software Engineering - ICSE 2014*, p. 1025–1035, 2014. ISSN 02705257. Disponível em: <http://dl.acm.org/citation.cfm?doid=2568225.2568276>. Citado 3 vezes nas páginas 30, 32 e 33.

HALKIDI, M.; BATISTAKIS, Y.; VAZIRGIANNIS, M. On clustering validation techniques. *Journal of intelligent information systems*, Springer, v. 17, n. 2-3, p. 107–145, 2001. Citado na página 27.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011. Citado 3 vezes nas páginas 27, 44 e 62.

HE, L. et al. Battery state-of-health estimation for mobile devices. In: IEEE. *2017 ACM/IEEE 8th International Conference on Cyber-Physical Systems (ICCPS)*. [S.l.], 2017. p. 51–60. Citado 2 vezes nas páginas 15 e 30.

HE, L. et al. Battery-Aware Mobile Data Service. *IEEE Transactions on Mobile Computing*, v. 16, n. 6, p. 1544–1558, 2017. ISSN 1536-1233. Disponível em: <http://ieeexplore.ieee.org/document/7530866/>. Citado 2 vezes nas páginas 15 e 30.

HUANG, A. Similarity measures for text document clustering. In: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*. [S.l.: s.n.], 2008. v. 4, p. 9–56. Citado na página 44.

JAIN, A. K.; MURTY, M. N.; FLYNN, P. J. Data clustering: a review. *ACM computing surveys (CSUR)*, Acm, v. 31, n. 3, p. 264–323, 1999. Citado na página 27.

JIN, X.; HAN, J. K-means clustering. *Encyclopedia of Machine Learning and Data Mining*, Springer, p. 695–697, 2017. Citado na página 27.

JIN, X.; HAN, J. K-medoids clustering. In: _____. *Encyclopedia of Machine Learning and Data Mining*. Boston, MA: Springer US, 2017. p. 697–700. ISBN 978-1-4899-7687-1. Disponível em: <https://doi.org/10.1007/978-1-4899-7687-1_432>. Citado na página 44.

KALLURI, S. et al. Feasibility of Cathode Surface Coating Technology for High-Energy Lithium-ion and Beyond-Lithium-ion Batteries. *Advanced Materials*, 2017. ISSN 15214095. Citado 2 vezes nas páginas 15 e 30.

KAUFMAN, L.; ROUSSEEUW, P. J. Finding groups in data: An introduction to cluster analysis–john wiley & sons. *Inc., New York*, 1990. Citado 2 vezes nas páginas 27 e 63.

LIN, D.; LIU, Y.; CUI, Y. Reviving the lithium metal anode for high-energy batteries. *Nature Nanotechnology*, Nature Publishing Group, v. 12, n. 3, p. 194–206, 2017. ISSN 1748-3387. Disponível em: <http://www.nature.com/doifinder/10.1038/nnano.2017.16>. Citado 2 vezes nas páginas 15 e 30.

PLACKE, T. et al. Lithium ion, lithium metal, and alternative rechargeable battery technologies: the odyssey for high energy density. *Journal of Solid State Electrochemistry*, Journal of Solid State Electrochemistry, v. 21, n. 7, p. 1939–1964, 2017. ISSN 14328488. Citado 2 vezes nas páginas 15 e 30.

RAO, K. et al. Application-Specific Performance-Aware Energy Optimization on Android Mobile Devices. *Proceedings - International Symposium on High-Performance Computer Architecture*, p. 169–180, 2017. ISSN 15300897. Citado 2 vezes nas páginas 15 e 30.

SPOTNITZ, R. Simulation of capacity fade in lithium-ion batteries. v. 113, n. September 2002, p. 72–80, 2003. Citado na página 22.

TAN, P. et al. *Introduction to Data Mining*. Pearson Education, 2013. (What's New in Computer Science Series). ISBN 9780133128901. Disponível em: <https://books.google.com.br/books?id=_ZQ4MQEACAAJ>. Citado na página 22.

THADA, V.; JAGLAN, V. Comparison of jaccard, dice, cosine similarity coefficient to find best fitness value for web retrieved documents using genetic algorithm. *International Journal of Innovations in Engineering and Technology*, v. 2, n. 4, p. 202–205, 2013. Citado na página 62.

TOPCHY, A.; JAIN, A. K.; PUNCH, W. Combining multiple weak clusterings. In: IEEE. *Third IEEE International Conference on Data Mining*. [S.l.], 2003. p. 331–338. Citado na página 28.

TOPCHY, A.; JAIN, A. K.; PUNCH, W. A mixture model for clustering ensembles. In: SIAM. *Proceedings of the 2004 SIAM international conference on data mining*. [S.l.], 2004. p. 379–390. Citado na página 29.

XU, D.; TIAN, Y. A comprehensive survey of clustering algorithms. *Annals of Data Science*, Springer, v. 2, n. 2, p. 165–193, 2015. Citado na página 23.

YOON, C. et al. Appscope: Application energy metering framework for android smartphone using kernel activity monitoring. In: *Presented as part of the 2012 {USENIX} Annual Technical Conference ({USENIX}{ATC} 12)*. [S.l.: s.n.], 2012. p. 387–400. Citado 2 vezes nas páginas 16 e 67.

APÊNDICE A – Gráficos de Perfis de Permissão

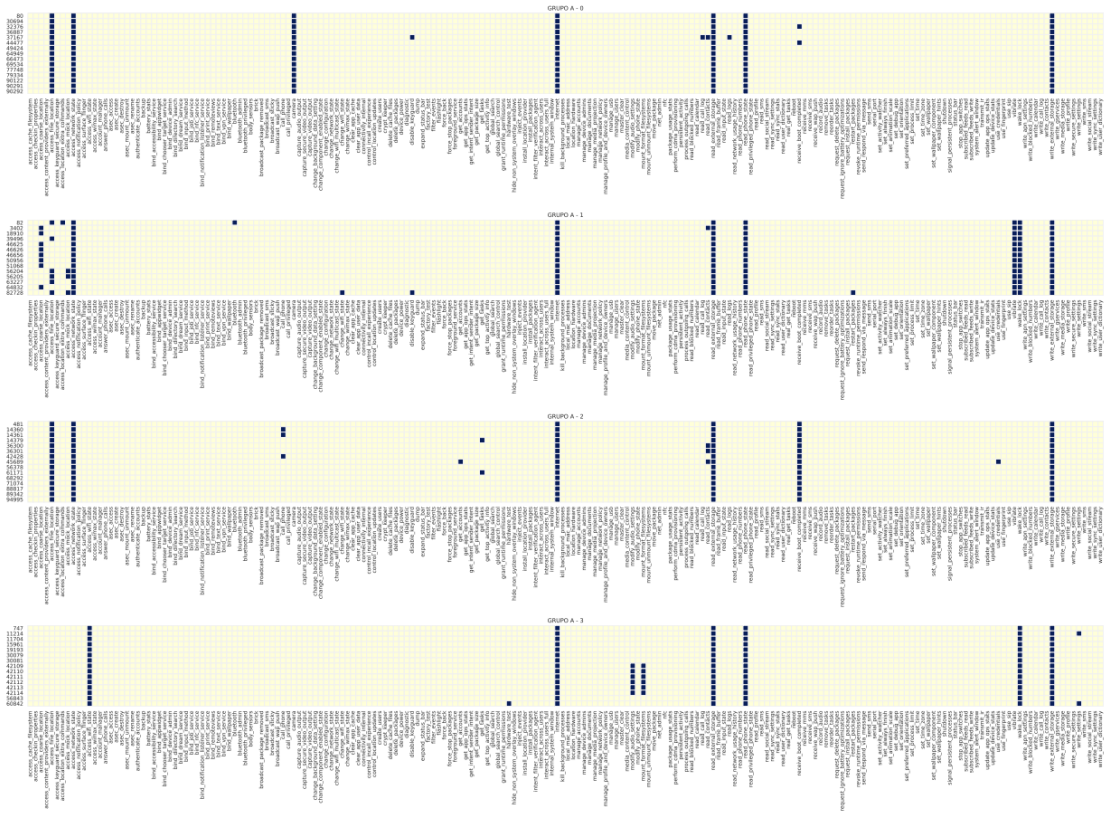


Figura 25 – Recorte do gráfico de perfis de permissão do Grupo A

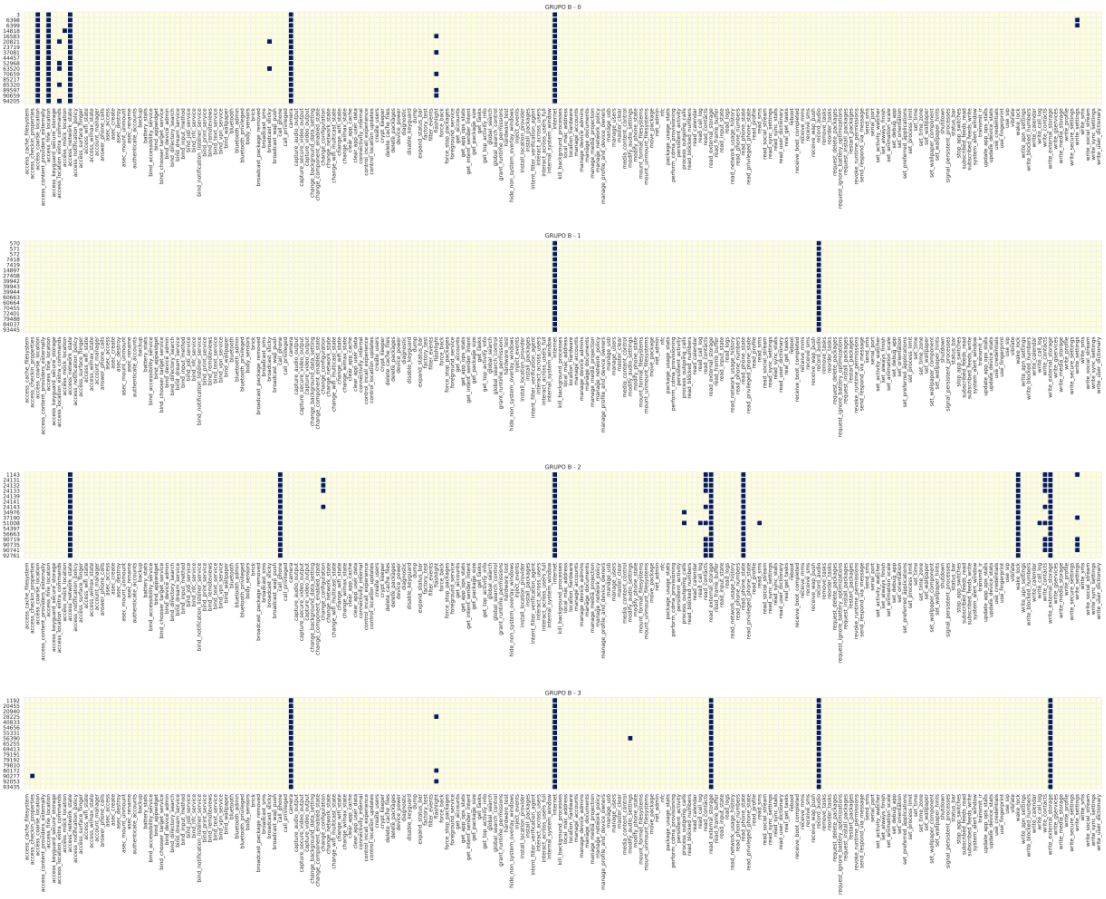


Figura 26 – Recorte do gráfico de perfis de permissão do Grupo B

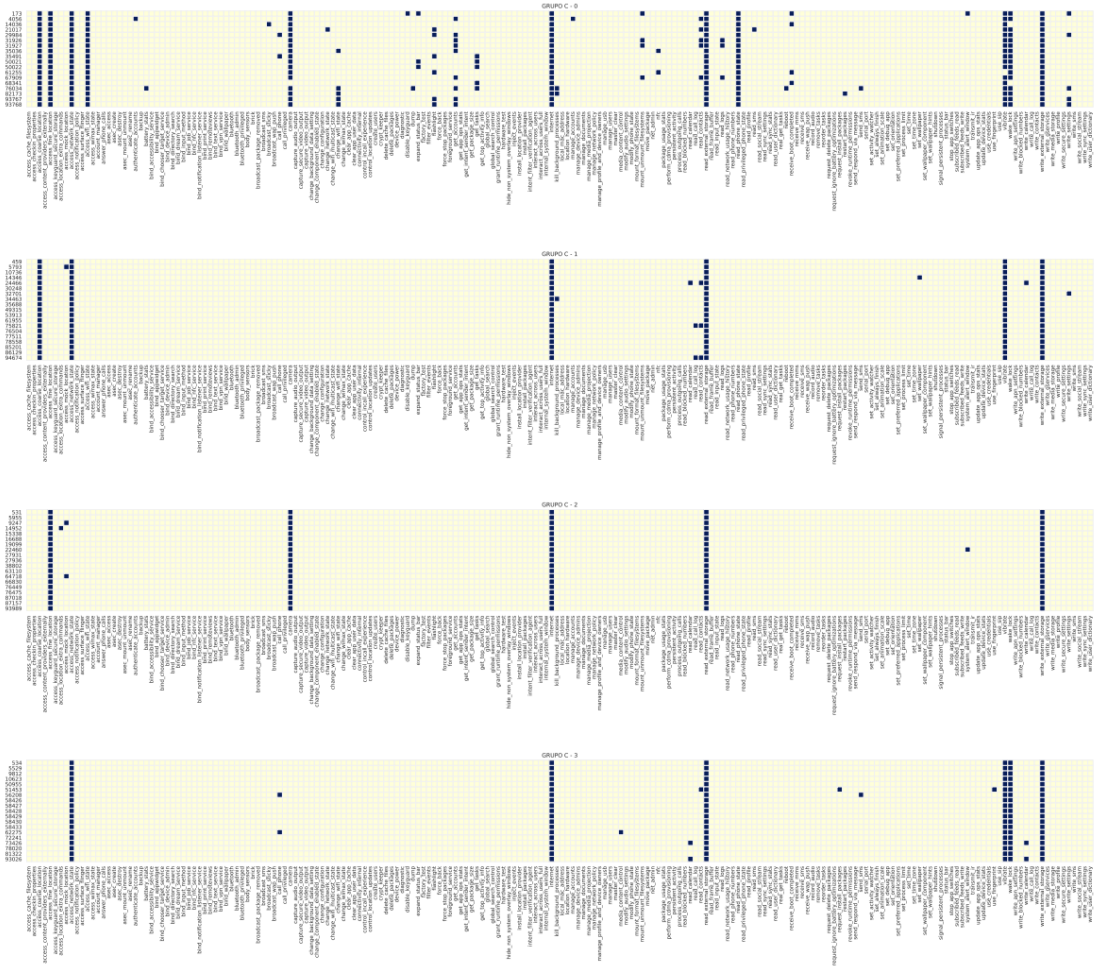


Figura 27 – Recorte do gráfico de perfis de permissão do Grupo C

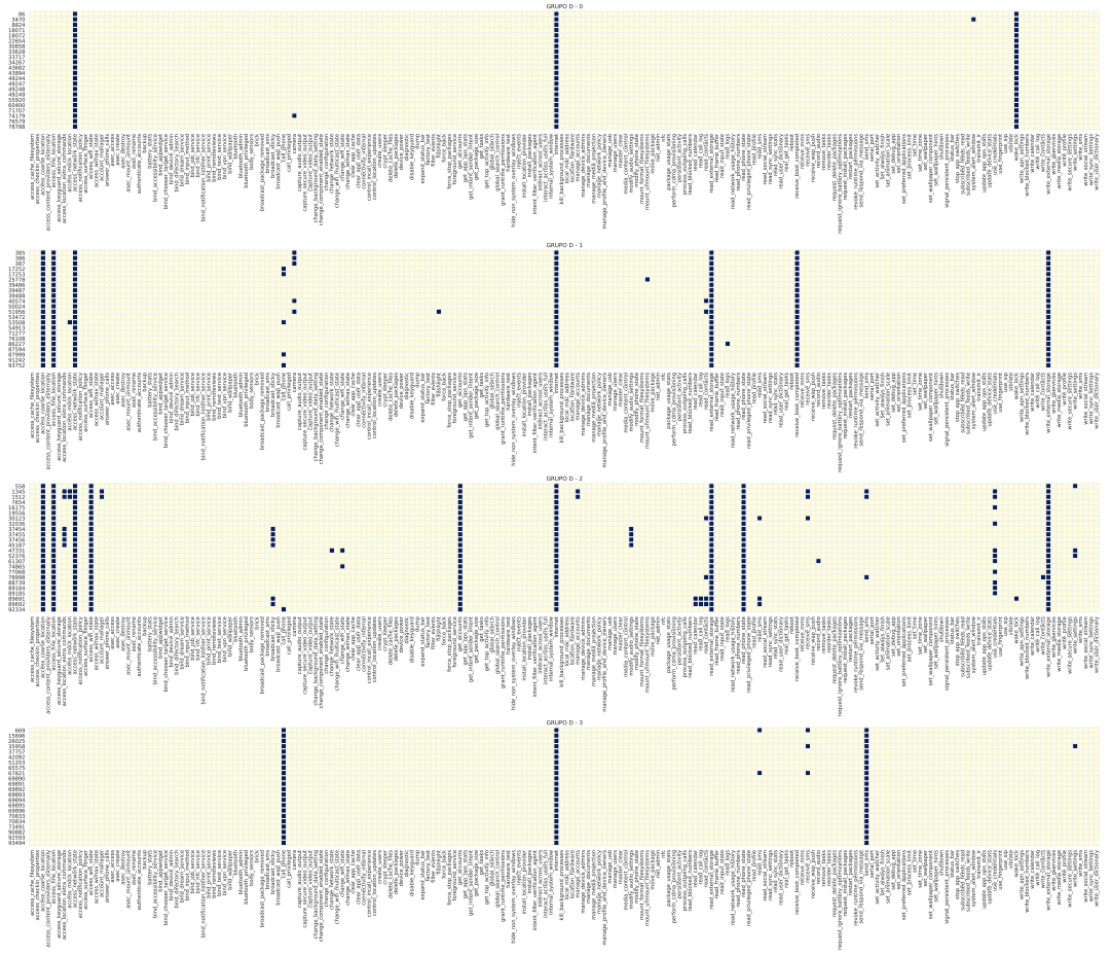


Figura 28 – Recorte do gráfico de perfis de permissão do Grupo D

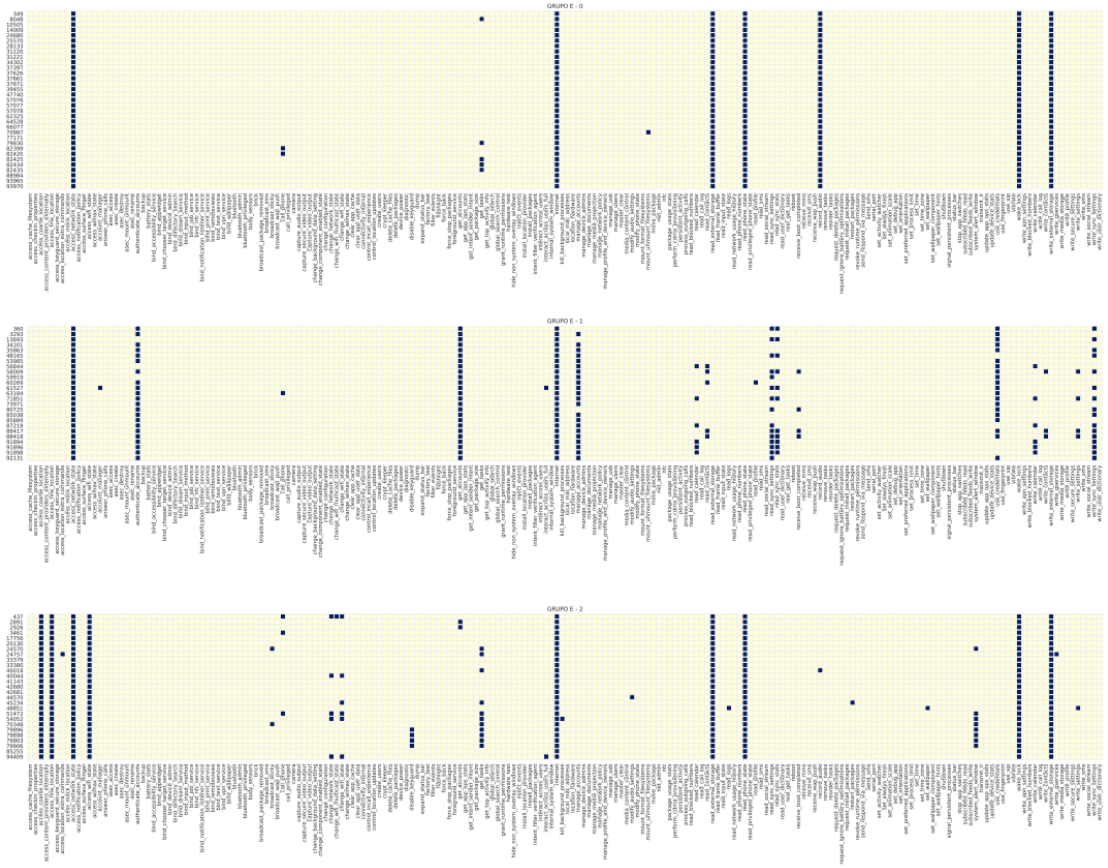


Figura 29 – Recorte do gráfico de perfis de permissão do Grupo E

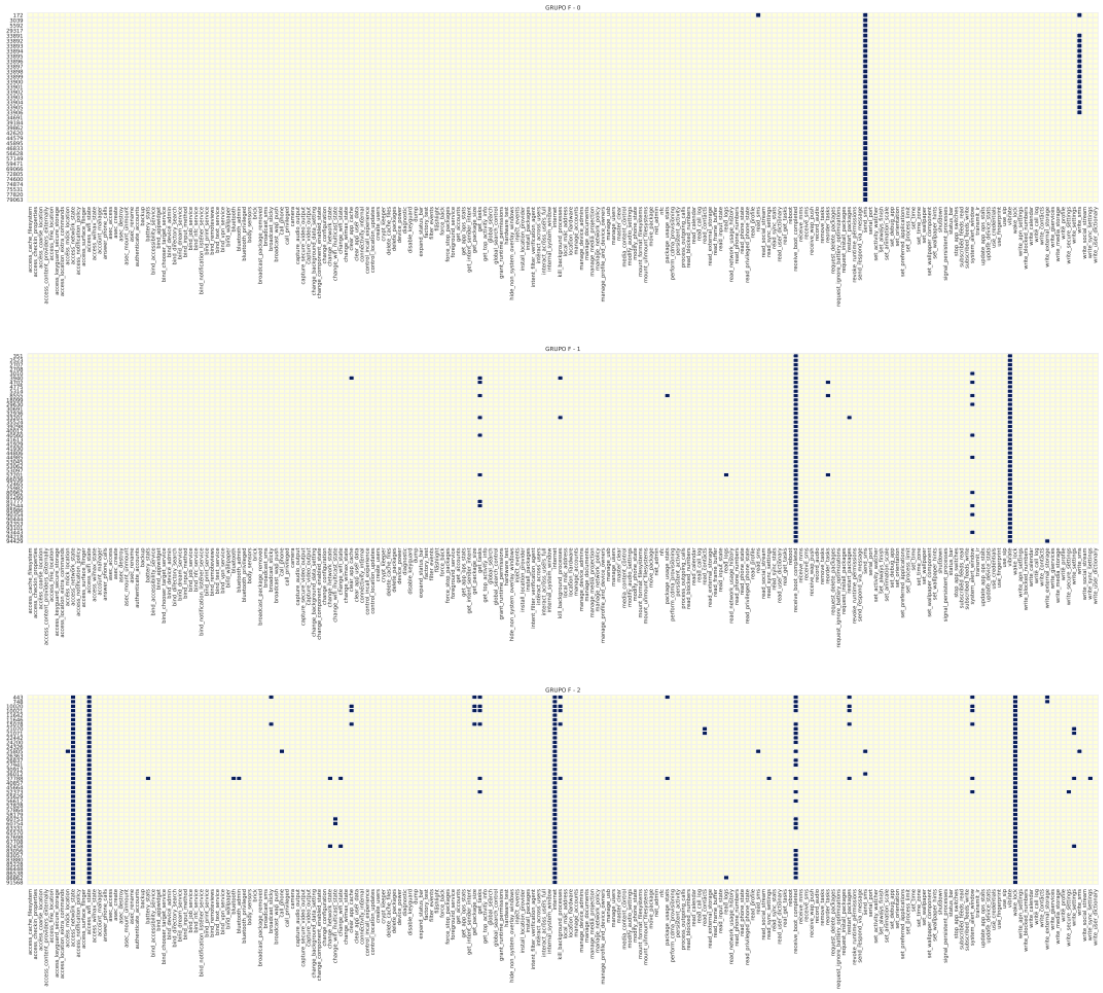


Figura 30 – Recorte do gráfico de perfis de permissão do Grupo F

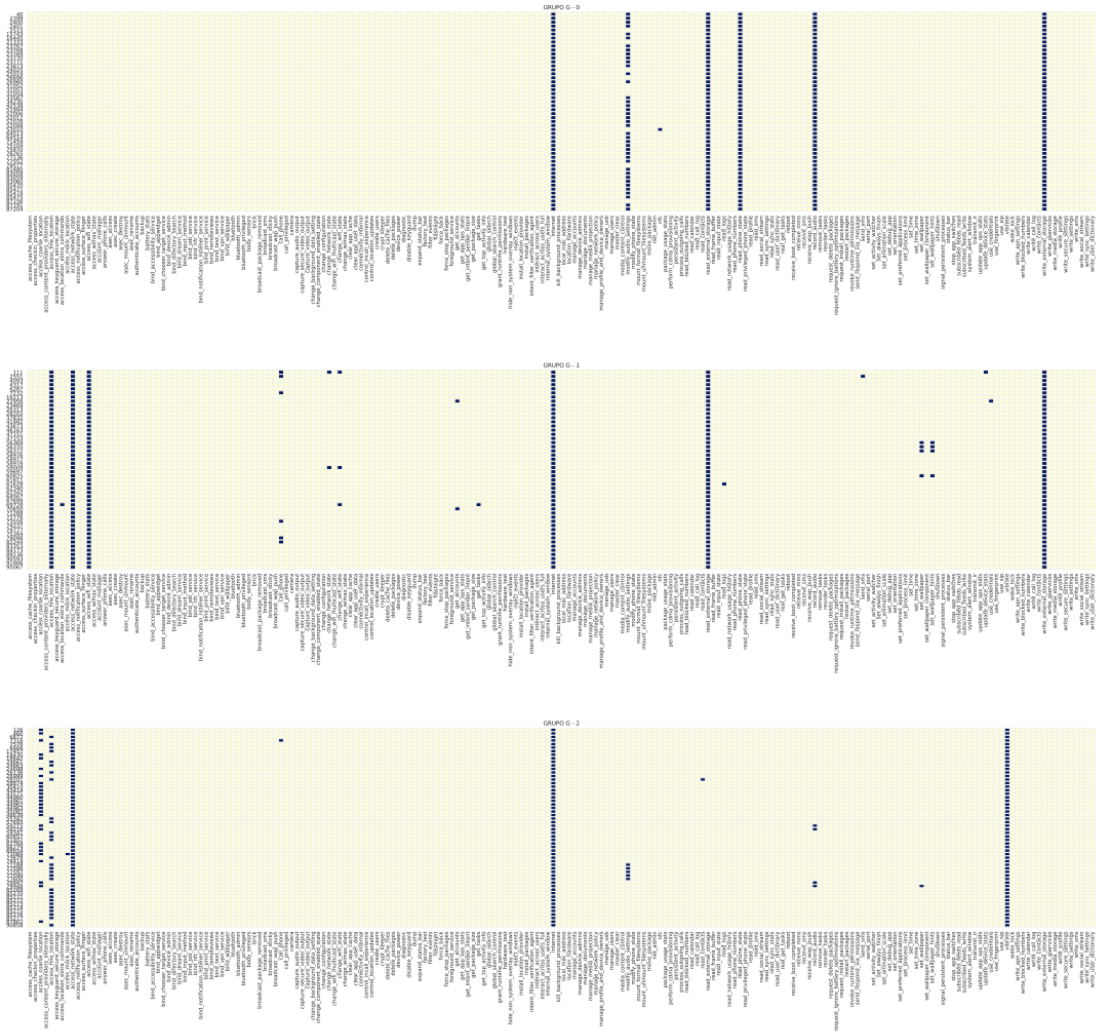


Figura 31 – Recorte do gráfico de perfis de permissão do Grupo G

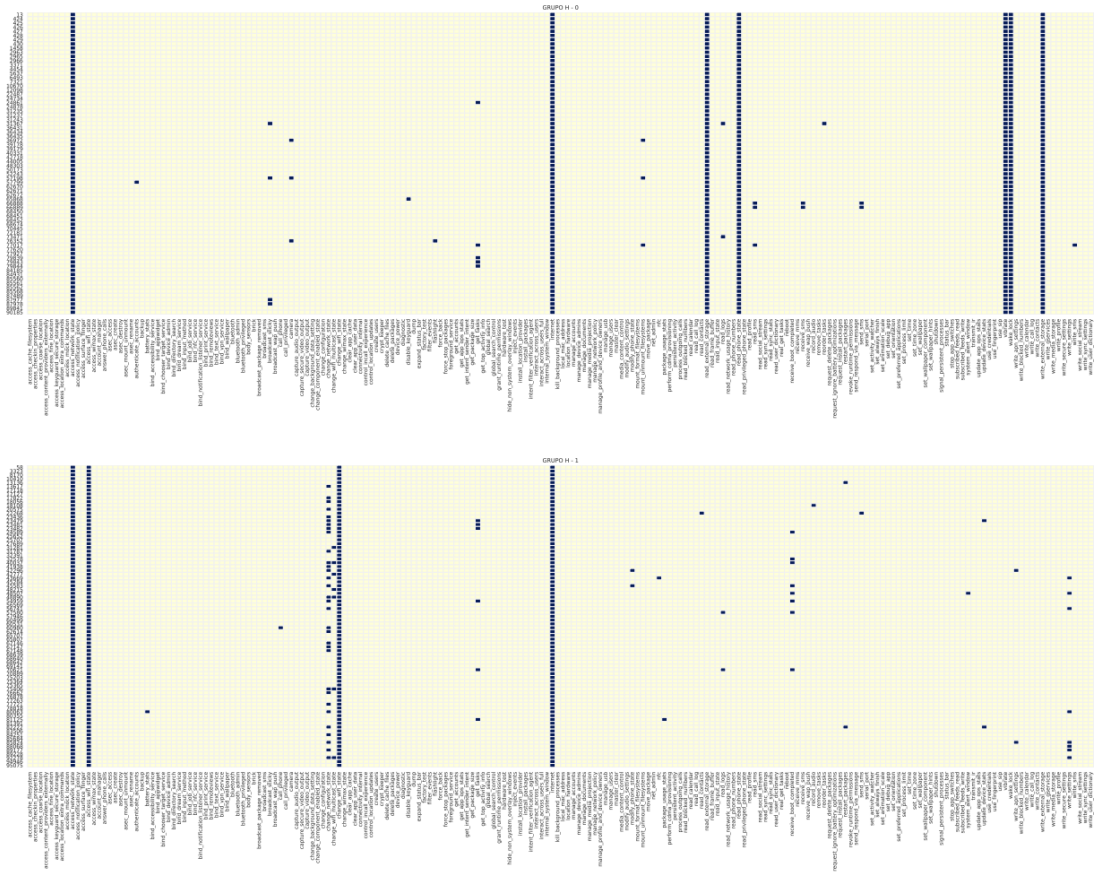


Figura 32 – Recorte do gráfico de perfis de permissão do Grupo H

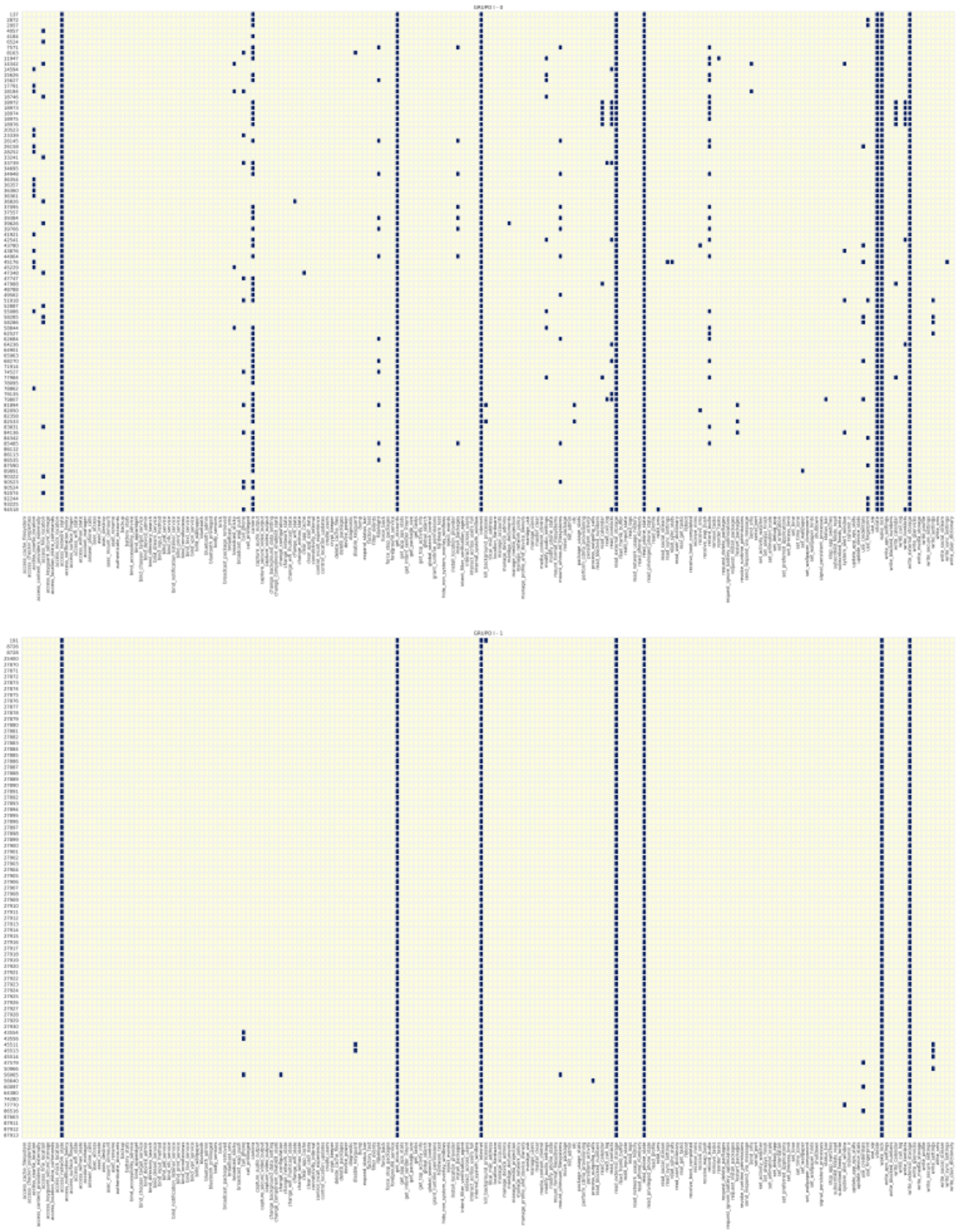


Figura 33 – Recorte do gráfico de perfis de permissão do Grupo I

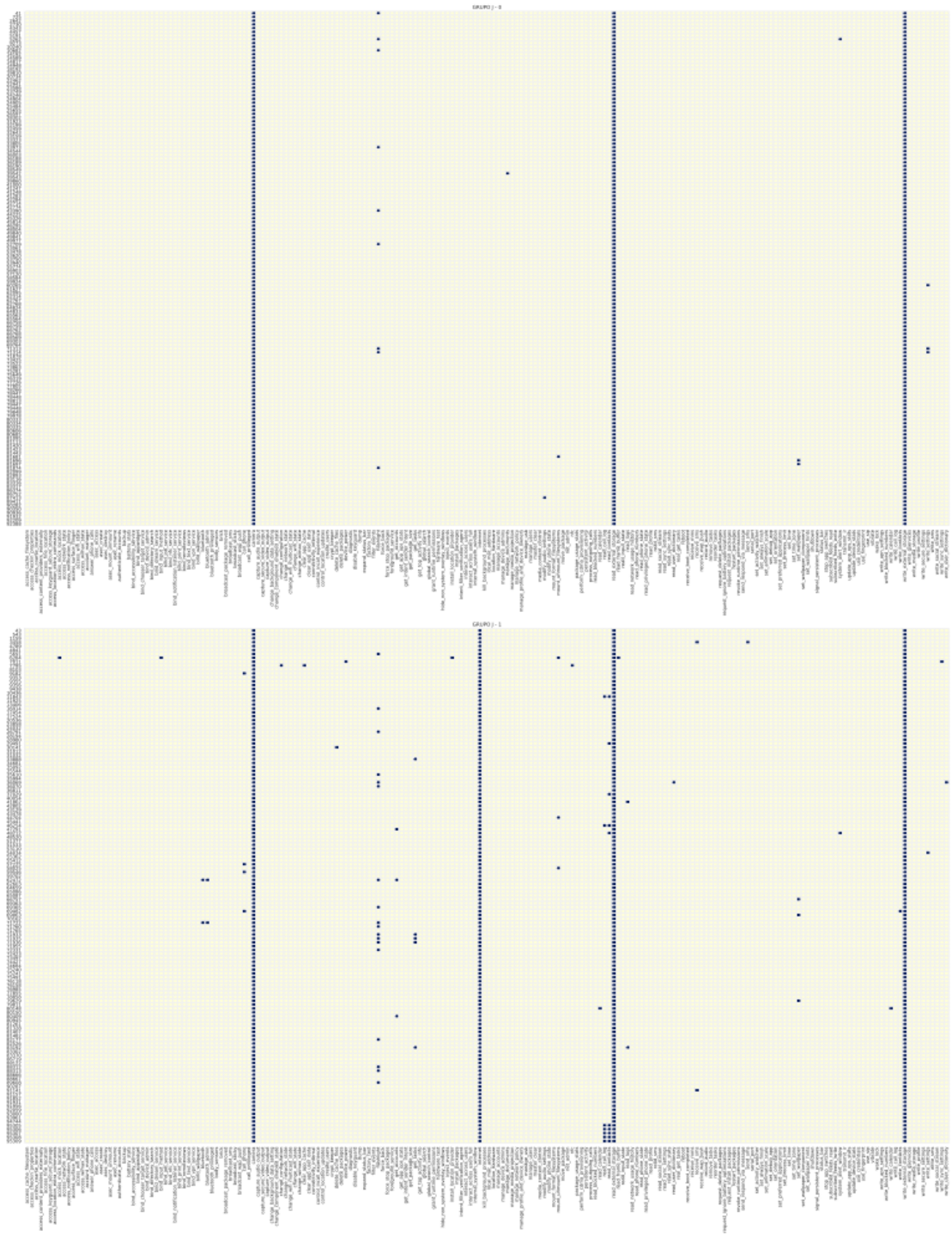


Figura 34 – Recorte do gráfico de perfis de permissão do Grupo J

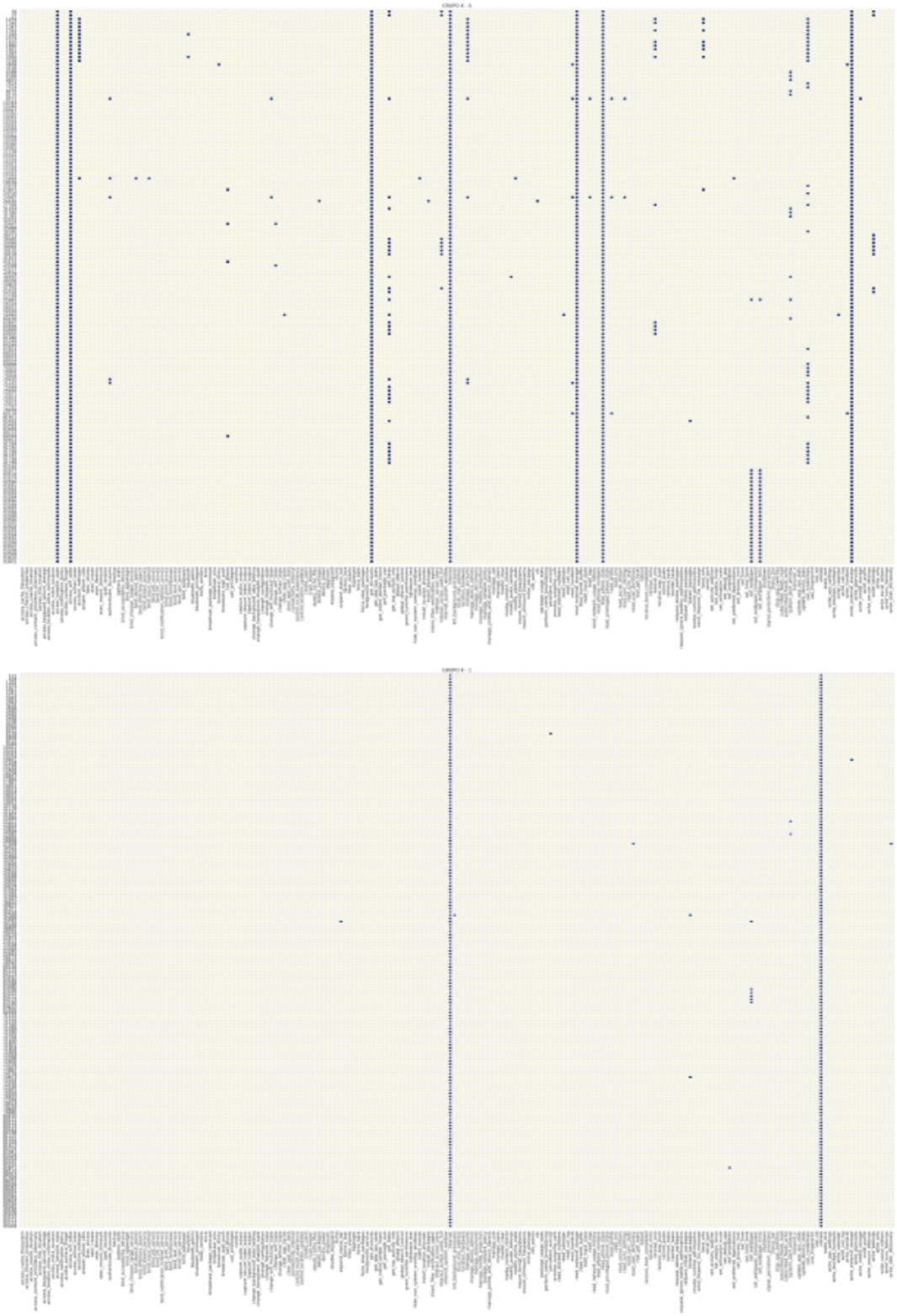


Figura 35 – Recorte do gráfico de perfis de permissão do Grupo K

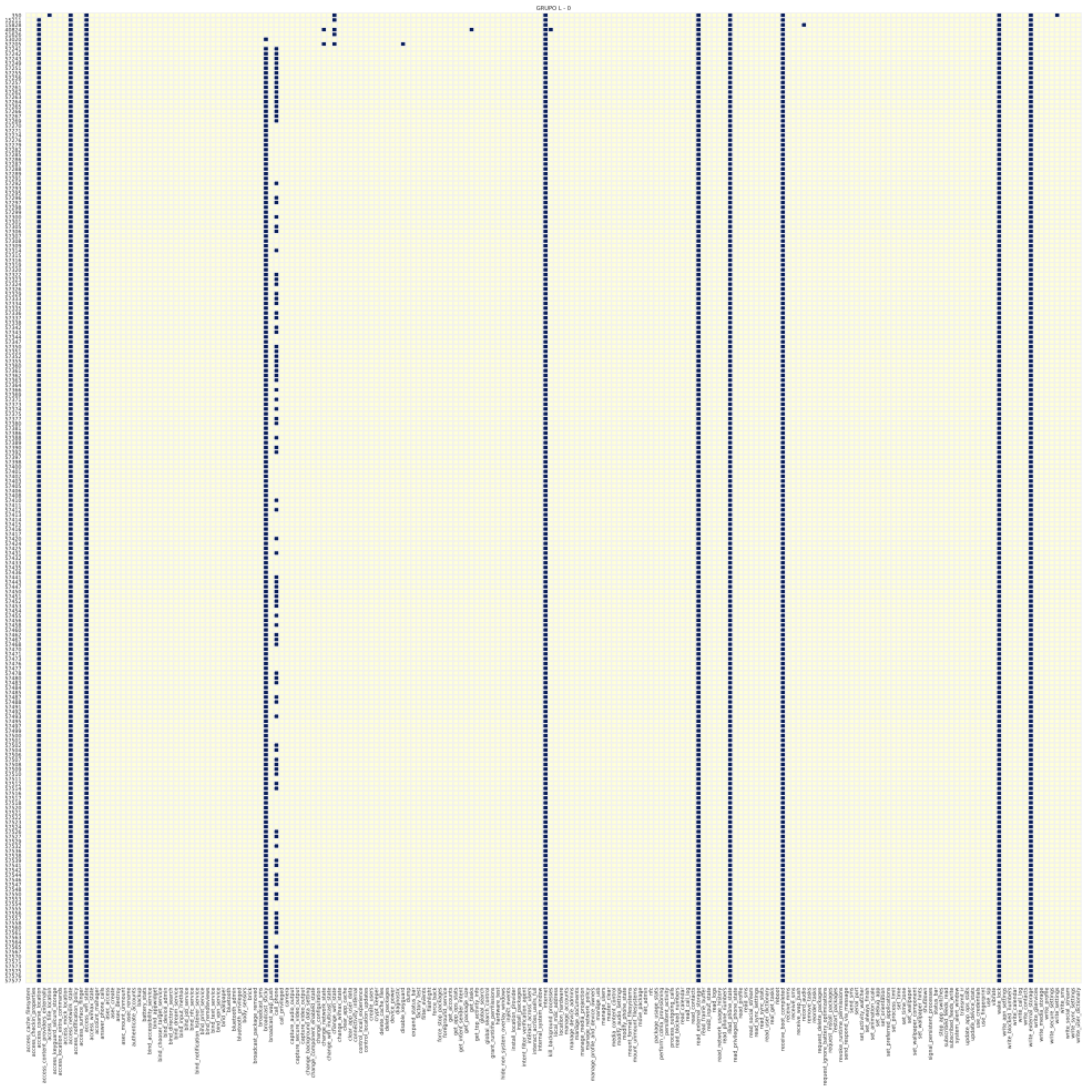


Figura 36 – Recorte do gráfico de perfis de permissão do Grupo L

Figura 37 – Recorte do gráfico de perfis de permissão do Grupo M

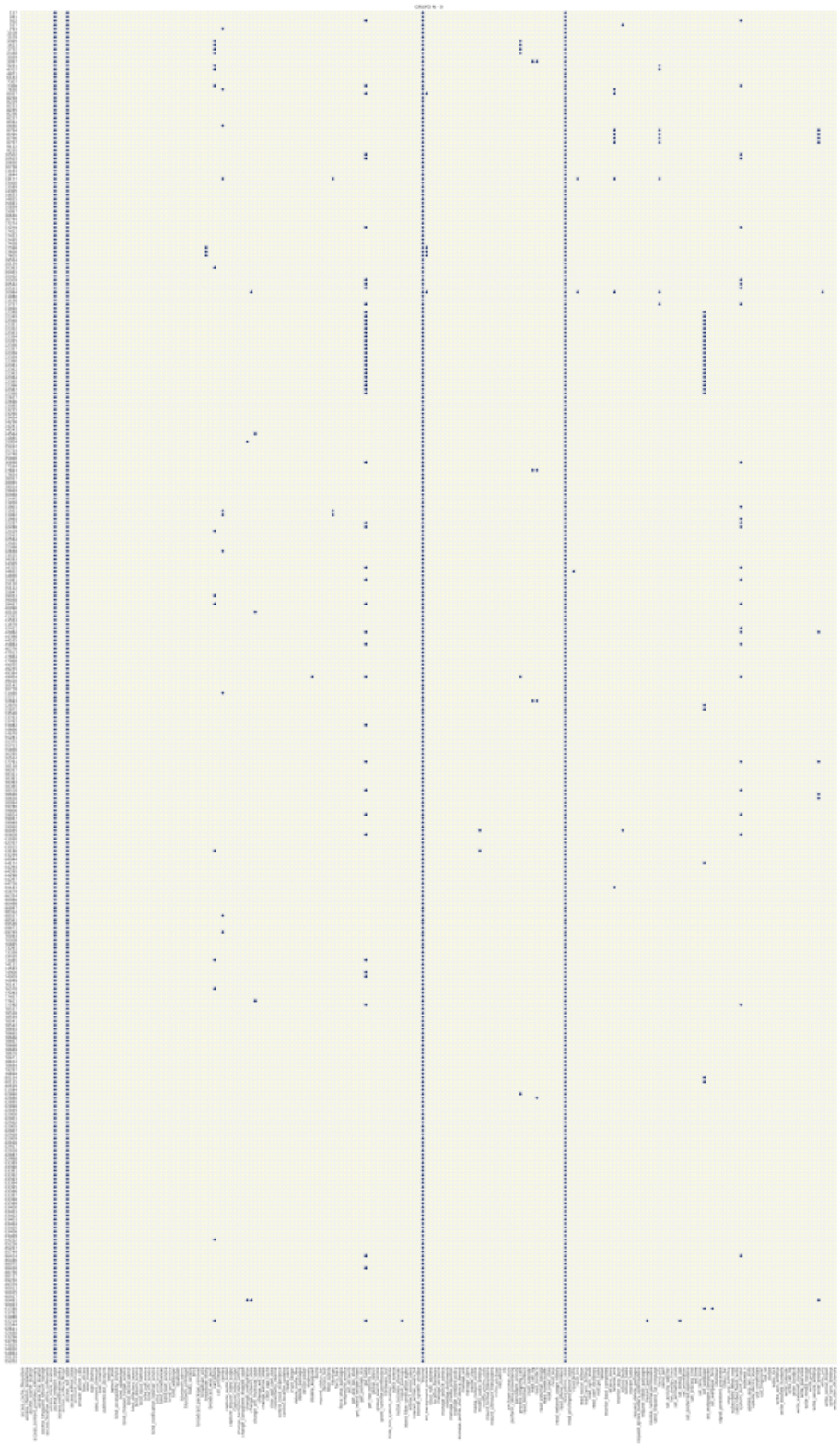


Figura 38 – Recorte do gráfico de perfis de permissão do Grupo N

Figura 39 – Recorte do gráfico de perfis de permissão do Grupo O

APÊNDICE B – Visualização dos Pesos das Permissões

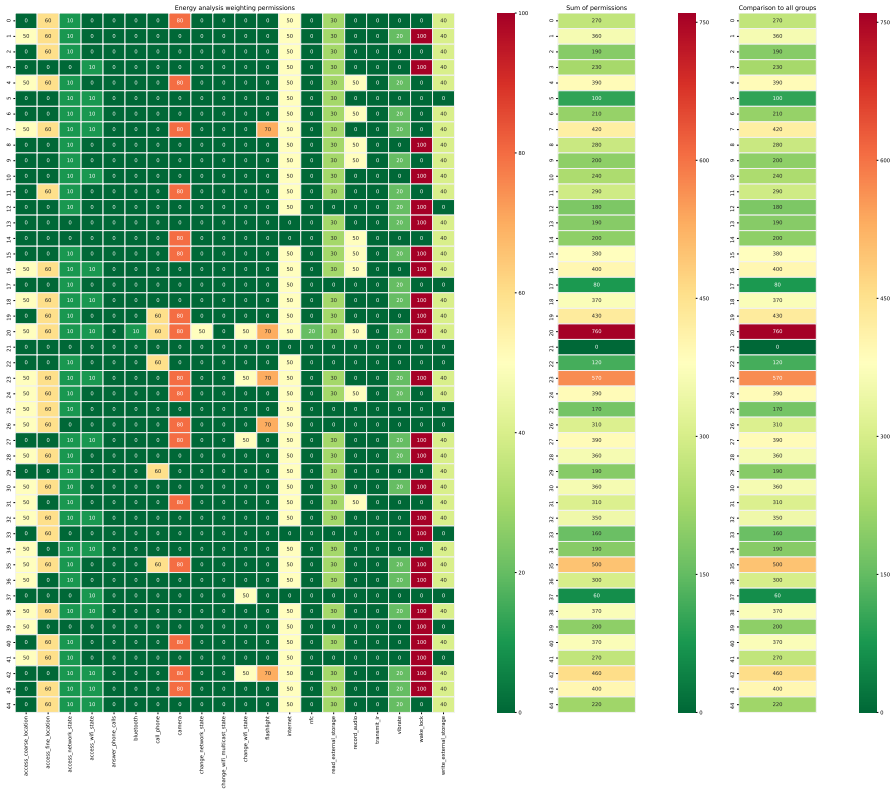


Figura 40 – Visualização dos pesos das permissões do Grupo A

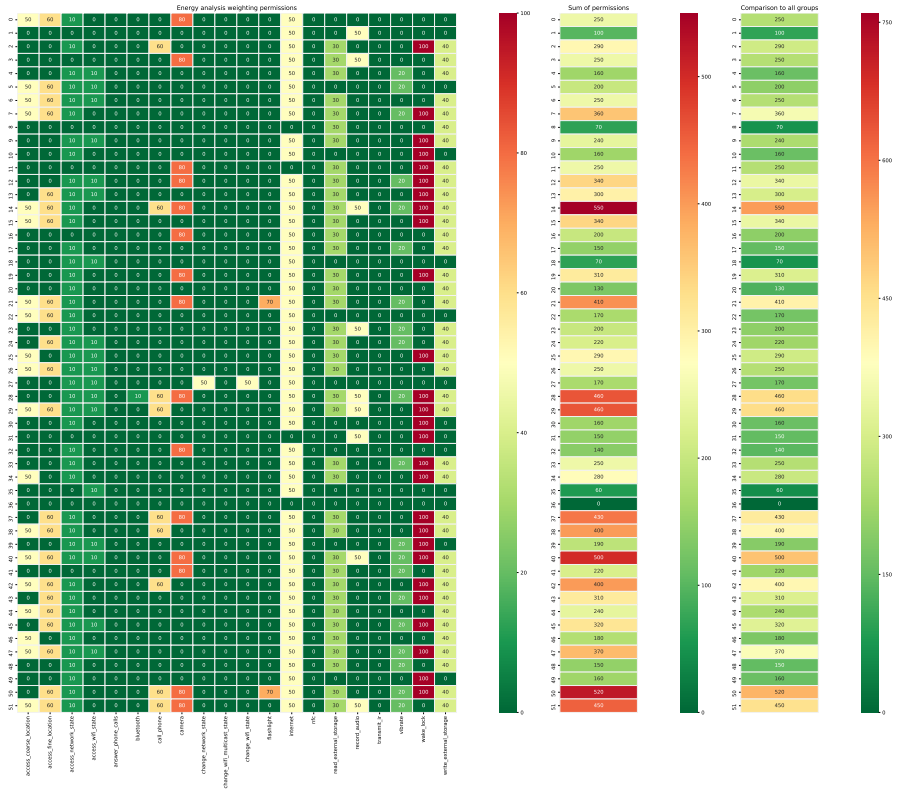


Figura 41 – Visualização dos pesos das permissões do Grupo B

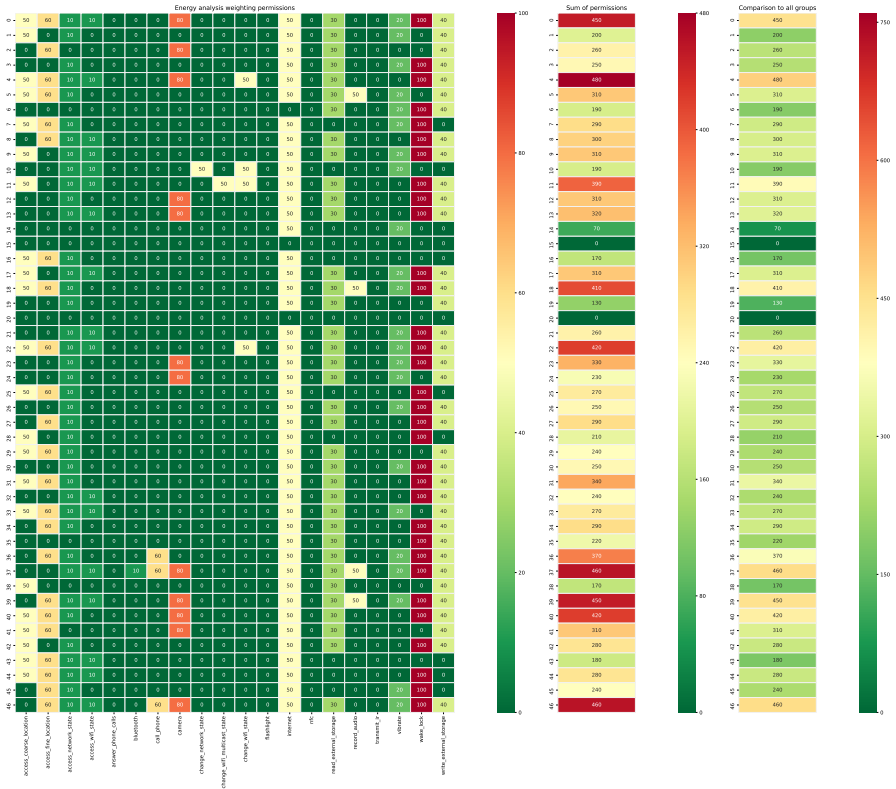


Figura 42 – Visualização dos pesos das permissões do Grupo C

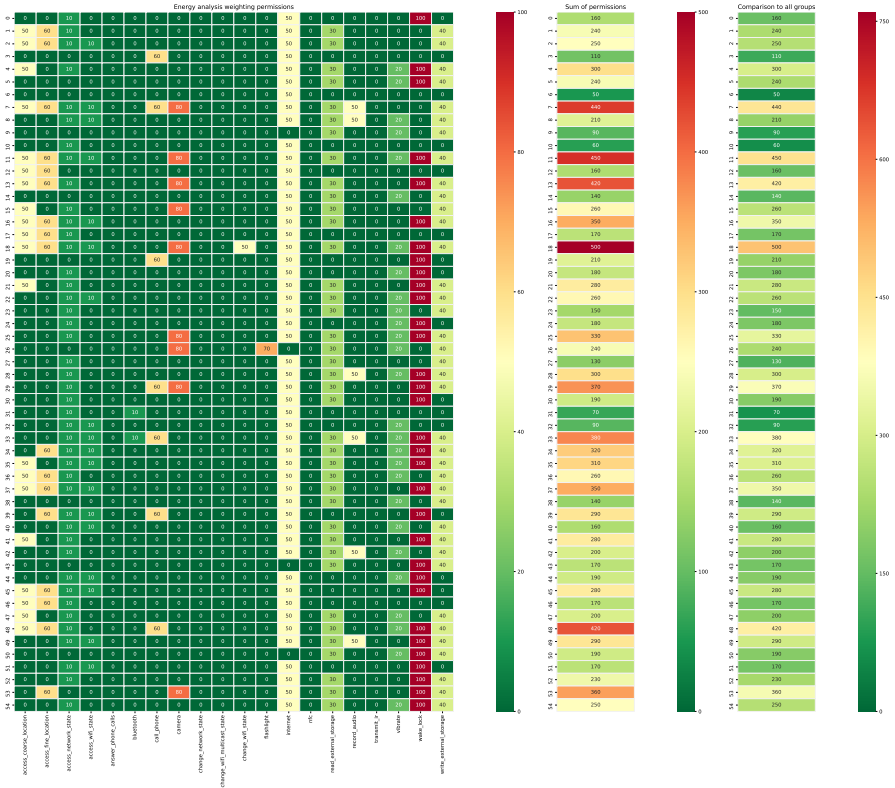


Figura 43 – Visualização dos pesos das permissões do Grupo D

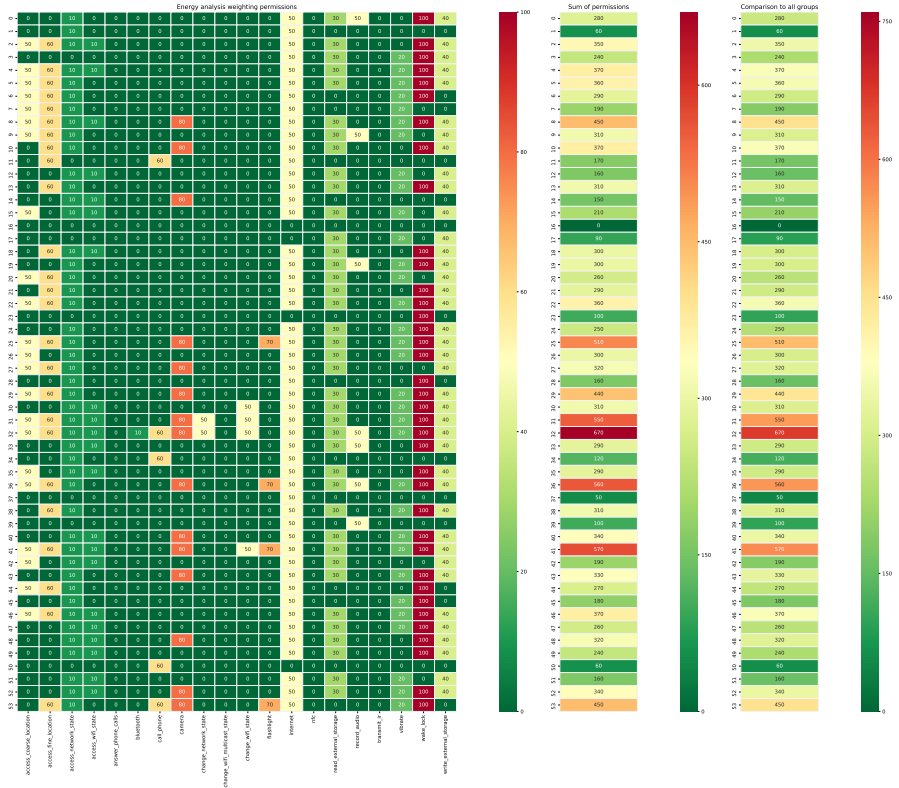


Figura 44 – Visualização dos pesos das permissões do Grupo E

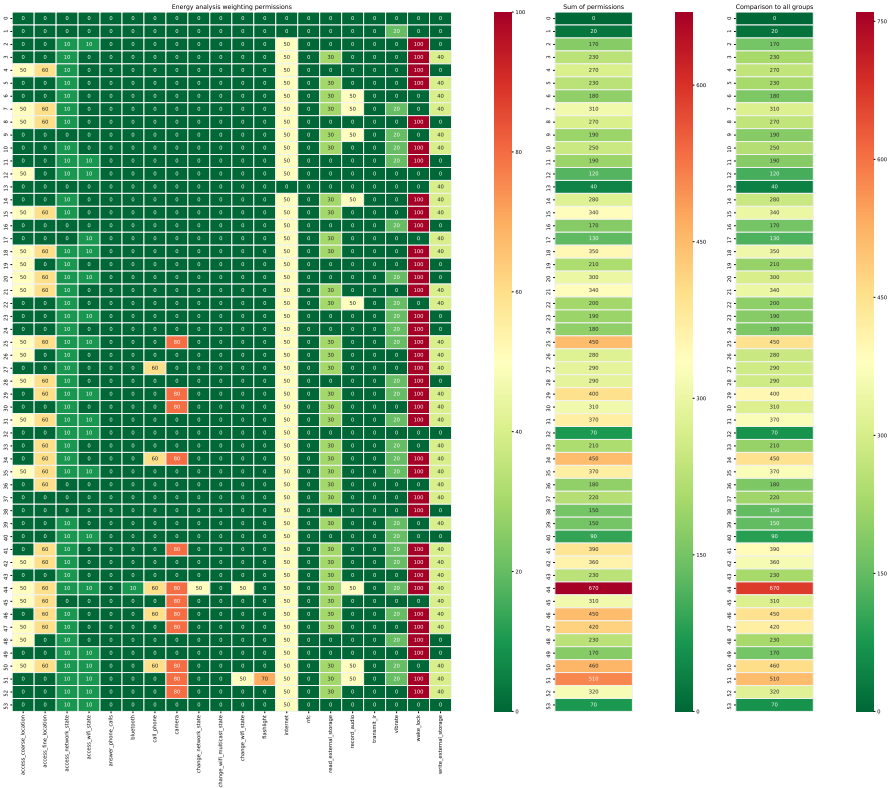


Figura 45 – Visualização dos pesos das permissões do Grupo F

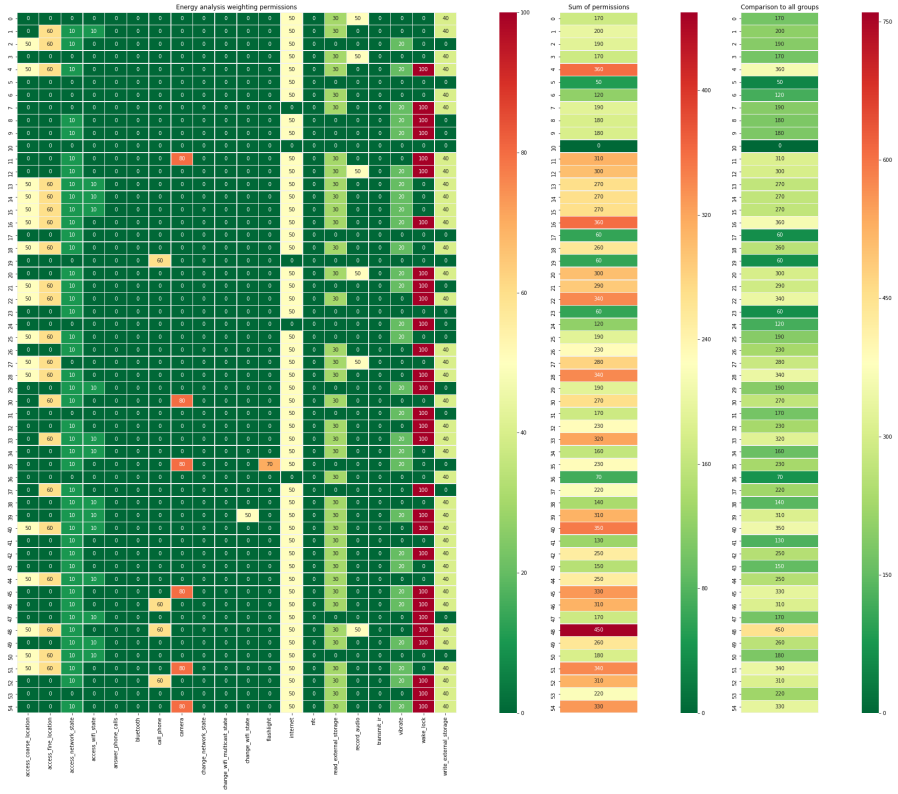


Figura 46 – Visualização dos pesos das permissões do Grupo G

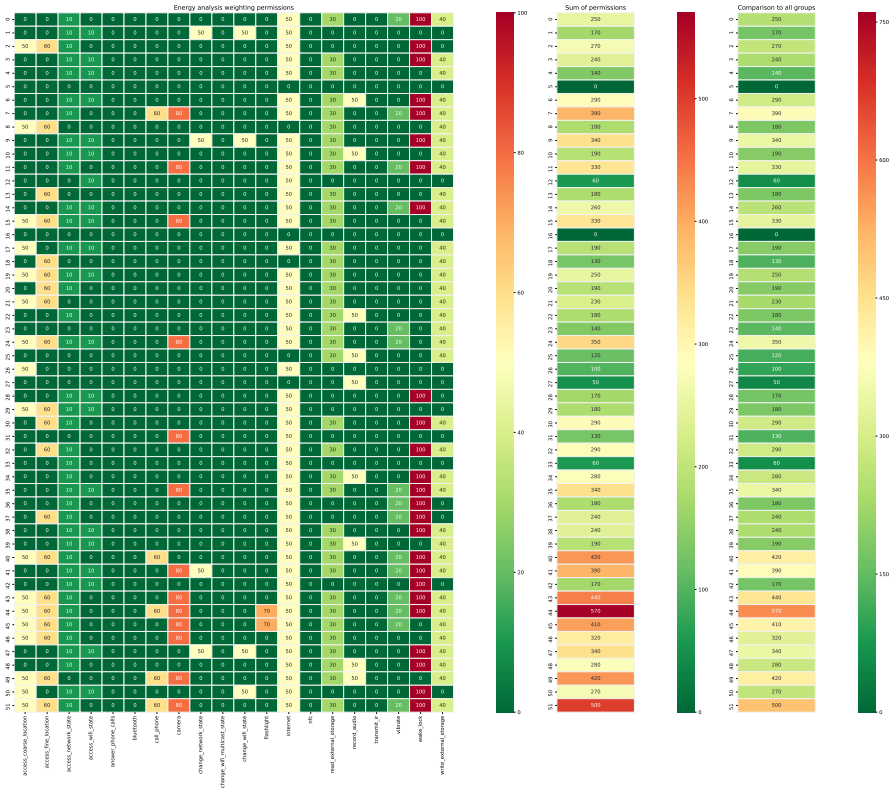


Figura 47 – Visualização dos pesos das permissões do Grupo H

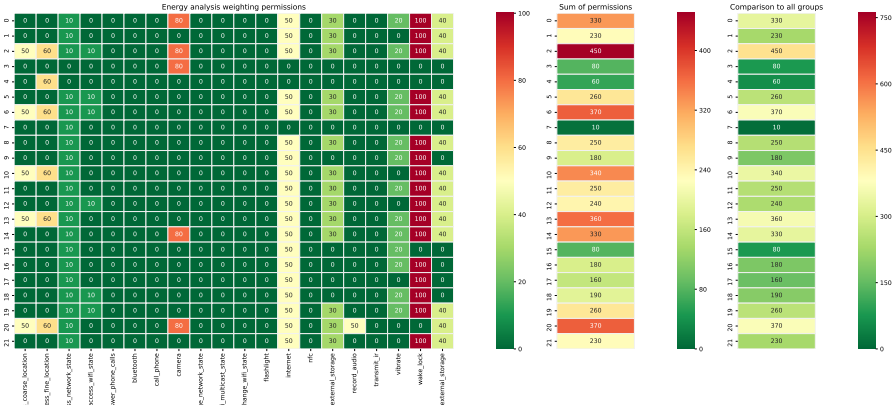


Figura 48 – Visualização dos pesos das permissões do Grupo I

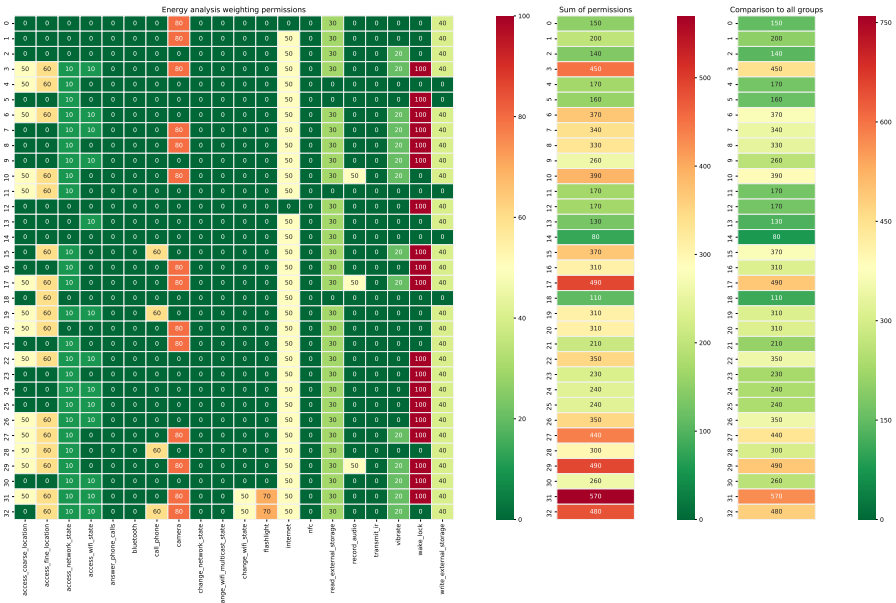


Figura 49 – Visualização dos pesos das permissões do Grupo J

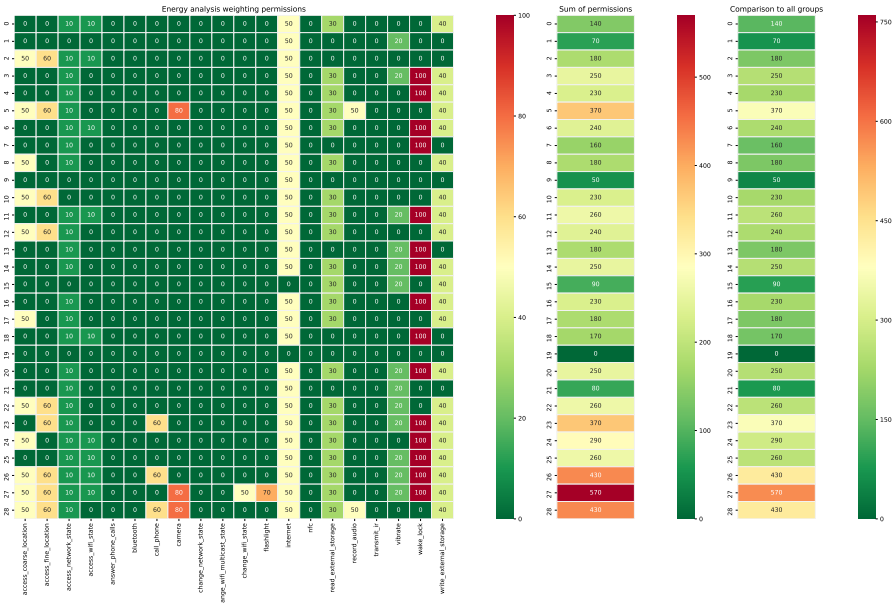


Figura 50 – Visualização dos pesos das permissões do Grupo K

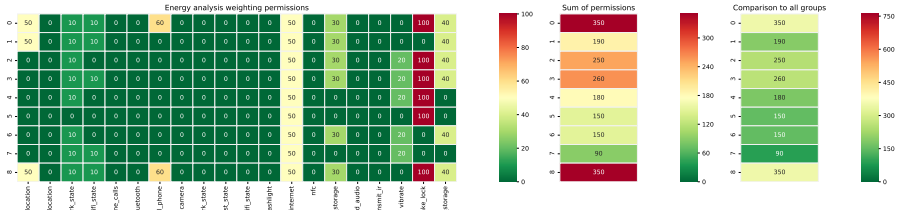


Figura 51 – Visualização dos pesos das permissões do Grupo L

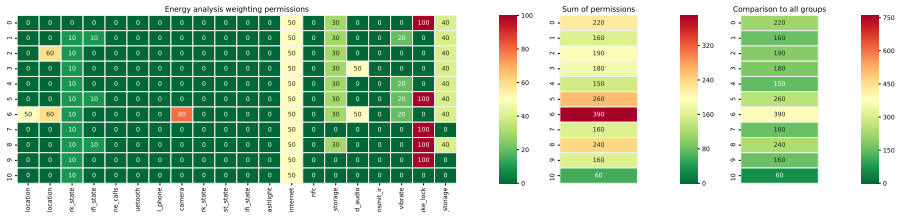


Figura 52 – Visualização dos pesos das permissões do Grupo M

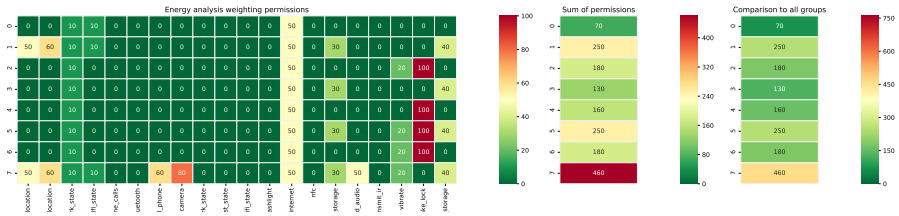


Figura 53 – Visualização dos pesos das permissões do Grupo N

Figura 54 – Visualização dos pesos das permissões do Grupo O