InMedia: Distribuição de conteúdo multimídia sensível à localização

Ruivaldo A. L. Neto e Manoel C. M. Neto
GSORT – Grupo de Pesquisa em Sistemas Distribuídos, Otimização, Redes e Tempo Real
IFBA – Instituto Federal de Educação, Ciência e Tecnologia da Bahia
Av. Araújo Pinho, nº 39 – Canela – CEP: 40.110-150 – Salvador-BA
Email: {ruivaldo, manoelnetom}@ifba.edu.br

***Resumo*—This work presents the development of InMedia, an open source web and mobile tool, whose aim is to facilitate the distribution of media based on context, with particular attention to the user location. The application provides an integrated environment for mapping and navigation using WiFi signal fingerprinting. To achieve better precision, the solution makes use of the Naive Bayes and Support Vector Machines classifiers.**

***Index Terms*—Location, Indoor Location, WiFi Networks, Naive Bayes, Support Vector Machines, Context Awareness.**

I. Introdução

A pervasividade dos *smartphones* e de outros dispositivos móveis evidenciou as vantagens das aplicações cientes de contexto. Dentre as informações contextuais utilizadas por estas aplicações, a localização configura-se como uma das mais importantes. O uso da localização como informação de contexto têm diversas aplicações, por exemplo: propaganda contextual [1], resposta a emergências e assistência à vida [2], robótica [3], e navegação em locais como aeroportos, shoppings e campis.

A maior disponibilidade e variedade de sensores nestes dispositivos permitem que sejam coletadas grandes quantidades de dados no processo de localização do usuário. Algoritmos de aprendizagem de máquina são uma solução natural para realizar a classificação destes dados.

Os algoritmos de aprendizagem de máquina melhoram suas performances através da experiência [4]. Estes algoritmos dividem-se em supervisionados e não supervisionados. Os supervisionados são treinados previamente com pares de atributos e saídas esperadas. Este treinamento permite que o algoritmo construa um modelo relacionando os atributos às saídas.

Esta monografia apresenta um sistema web e mobile, denominado InMedia, cujos objetivos são: simplificar o mapeamento de ambientes através da técnica de *fingerprinting* de sinais WiFi; auxiliar na distribuição de conteúdo multimídia de forma sensível à localização do usuário; e aplicar os algoritmos de aprendizagem de máquina supervisionados *Naive Bayes* e *Support Vector Machines* para a inferência da localização do usuário com maior precisão.

O trabalho está estruturado da seguinte forma:

- As seções II e III abordam os principais conceitos e características da computação ubíqua e da sensibilidade ao contexto;
- A seção IV trata da metodologia de mapeamento utilizando *fingerprinting*;
- A seção V apresenta os principais conceitos sobre aprendizagem de máquina e detalha o funcionamento dos classificadores utilizados neste projeto;
- A seção VI elenca os trabalhos relacionados;
- As seções VII, VIII e IX apresentam a arquitetura, a modelagem e as principais funcionalidades da solução;
- Nas seções seguintes são apresentados os resultados dos testes, as conclusões, os trabalhos futuros e as referências utilizadas.

II. Computação Ubíqua

Mark Weiser e demais colaboradores dos laboratórios Xerox Parc vislumbraram o que a computação deveria se tornar no futuro: algo invisível. A ideia foi formalizada em 1991 no artigo *The computer for the 21st century* [5], dando origem a uma linha de pesquisa até então inexplorada, a computação ubíqua (*Ubiquitous Computing*).

Neste novo contexto, os elementos computacionais estão imersos de forma imperceptível no cotidiano da sociedade. Este grau de imersão é obtido através da colaboração entre os dispositivos, que ocorre de forma transparente para o usuário e tem como objetivo prover o melhor serviço possível ao utilizador.

Weiser, considerado idealizador deste novo modelo computacional, utilizou a escrita como exemplo de tecnologia ubíqua [6]. Esta tecnologia, criada há milhares de anos, foi por muito tempo exclusiva dos mais privilegiados, economicamente e socialmente. Entretanto, passou a estar imersa no cotidiano, sendo consumida em larga escala e imperceptível como elemento tecnológico.

A ubiquidade da informática terá atingido sua plenitude quando a computação for aplicada em atividades do cotidiano de forma tão natural quanto a escrita. Ou seja, a era da computação ubíqua terá sido alcançada quando o computador deixar de ser uma peça única e de uso genérico, multiplicando-se em diversos elementos computacionais especializados.

No artigo entitulado *The World is not a Desktop* [7], Weiser define que o computador é uma tecnologia de comunicação e,

como todas as tecnologias de sucesso, tende a desaparecer. Os óculos, por exemplo, são uma tecnologia que auxilia na interação entre o ser humano e o mundo, mas o ser humano não concentra suas atenções nos óculos, mas no mundo. Assim como uma pessoa cega, ao utilizar sua bengala, percebe o mundo ao seu redor, e não a bengala.

Pesquisadores registraram, em suas publicações acadêmicas, características essenciais aos sistemas computacionais ubíquos:

Integração física: Espaços inteligentes (*smart spaces*) são definidos pela intensa colaboração entre elementos computacionais e elementos do mundo físico. Kindgberg e Fox [8] definem esta integração como ponto essencial para caracterização de sistemas ubíquos. O projeto MediaCups [9] é utilizado como exemplo de inserção de um objeto físico do mundo real em um cenário lógico computacional. Neste projeto, uma xícara de café - componente do mundo físico - além do seu papel natural de recipiente de café, também é dotada de sensores, recursos de processamento e de rede, que a permitem comunicar seu estado com os demais elementos, físicos ou lógicos, naquele espaço.

Invisibilidade: A computação ubíqua também é definida como computação invisível. Para Weiser [5], quanto maior a presença da tecnologia em nossas vidas, menos perceptível ela deverá ser. Weiser e John Brown [10] definiram a computação ubíqua com base nos conceitos de periferia e centro das atenções. Para os autores, a efetiva inserção das tecnologias no cotidiano se dá quando estas não exigem mais do que a atenção periférica do usuário. Portanto, os elementos computacionais de um sistema ubíquo devem primar pela discrição em seu funcionamento.

Pró-atividade: Os sistemas ubíquos devem ser capazes de se antecipar à intenção do usuário, a partir das informações de contexto e histórico de eventos armazenado.

Sensibilidade ao Contexto: Para que um sistema ubíquo tenha o mínimo de intrusão no mundo real, seus elementos de software devem ser sensíveis ao contexto [11]. Isto é, manter uma base extensa de informações a respeito das pessoas e do ambiente e, através destes dados, adaptar seu comportamento com o objetivo de atender plenamente às expectativas de seus usuários e preservar a característica de invisibilidade. Um contexto da computação ubíqua pode ser constituído por um conjunto de atributos físicos, fisiológicos, psicológicos, histórico de atividades, padrão de comportamento, entre outros [12].

Interoperabilidade espontânea: Conforme Kindgberg e Fox [8], sistemas ubíquos são divididos em componentes de infra-estrutura e componentes espontâneos. Os componentes de infra-estrutura são fixos em relação ao *smart space*, enquanto os espontâneos são entendidos como unidades de software que abstraem serviços, recursos ou dispositivos em constante deslocamento entre ambientes distintos. Logo, a interoperabilidade espontânea é uma característica necessária, para que seja possível a integração dinâmica entre componentes móveis e de infra-estrutura do sistema.

Interfaces Naturais: Em [7], Weiser já identificava a tendência da computação ubíqua de extrapolar a noção de *desktop* e seus elementos artificiais: modelo de janelas, monitor, mouse e teclado. E que para se ter uma computação realmente ubíqua, as interfaces utilizadas no dia-a-dia de uma sociedade - gestos, voz, entre outros - também deveriam ser aplicadas como meio de comunicação entre homem e máquina.

As características elencadas nesta seção não esgotam as virtudes requeridas à computação ubíqua, mas representam seus principais atributos.

III. Sensibilidade ao contexto

O termo sensibilidade ao contexto foi utilizado pela primeira vez por Schilit e Theimer em 1994 [13]. Em 2007, Malik [14] definiu sensibilidade ao contexto como a habilidade de um dispositivo ou aplicação de perceber e se adaptar à situação corrente do usuário. Esta adaptabilidade tem como objetivo entregar informações e serviços mais relevantes para o utilizador, diferenciando-se das aplicações estáticas tradicionais.

Diferentes definições são encontradas na literatura para o termo contexto:

- Contexto pode ser qualquer atributo que descreva o usuário, o dispositivo, a rede física ou o ambiente (e. g. o nível de bateria ou a temperatura do ambiente) [13];
- Contexto é qualquer informação que caracteriza a situação de uma entidade, que pode ser pessoa, lugar ou objeto, considerada relevante em uma interação usuário-aplicação, incluindo o próprio usuário e a aplicação [15].

A partir destas definições, compreende-se que a sensibilidade ao contexto é a capacidade da aplicação ser ciente ao que está em torno, atuando no ambiente conforme o contexto e suas mudanças ao longo do tempo. É importante ressaltar que a caracterização da situação é dependente do domínio e a relevância está relacionada com o que o usuário está fazendo e o objetivo desta atividade.

Para auxiliar na escolha das informações que podem ser utilizadas para construção do contexto foi definido [16] um conjunto com seis dimensões semânticas para especificação do contexto, conhecido como *5Ws + 1H*:

- **Quem?** (*Who ?*) identicar o usuário que está utilizando a aplicação;
- **Onde?** (*Where ?*) localizar o usuário;
- **Quando?** (*When ?*) o momento em que a interação está ocorrendo;
- **O quê?** (*What ?*) a atividade corrente do usuário;
- **Por quê?** (*Why ?*) entender as razões das ações do usuário; e
- **Como?** (*How ?*) de que maneira as informações contextuais serão capturadas.

Dentre as dimensões citadas, o principal domínio de interesse deste trabalho é o **onde**, com o objetivo de inferir a localização do usuário com alto grau de precisão, inclusive em ambientes fechados. Os dados para construção deste contexto são coletados através da técnica de *fingerprinting* de sinais *WiFi*.

IV. Técnica de Localização - Mapeamento RSSI (*Fingerprinting*)

O mapeamento de locais por *fingerprinting* pode ser implementado através de diferentes tecnologias baseadas em radiofrequência. Neste trabalho, os sinais das redes WiFi no entorno do usuário são utilizados como fonte de dados. Ainda que seja possível obter *fingerprints* a partir de qualquer propriedade do sinal WiFi analisado, o RSSI (*Received Signal Strength Indication*) é o parâmetro utilizado. O indicador de força de sinal é o parâmetro escolhido por ser disponibilizado na maioria dos controladores e por ser sensível às mudanças de local do receptor.

Esta metodologia de mapeamento é composta por duas fases: *off-line* e *online*. Na fase *off-line* ou fase de calibração, os sinais RSSI de diferentes *Access Points* são coletados por todo o ambiente e armazenados em um banco de dados. Os conjuntos de sinais coletados são vinculados a um local específico, produzindo o mapeamento. A Figura 1 apresenta, como exemplo, o mapeamento de uma sala através da coleta de pontos (em metros), relacionando-os às forças dos sinais de rede disponíveis.

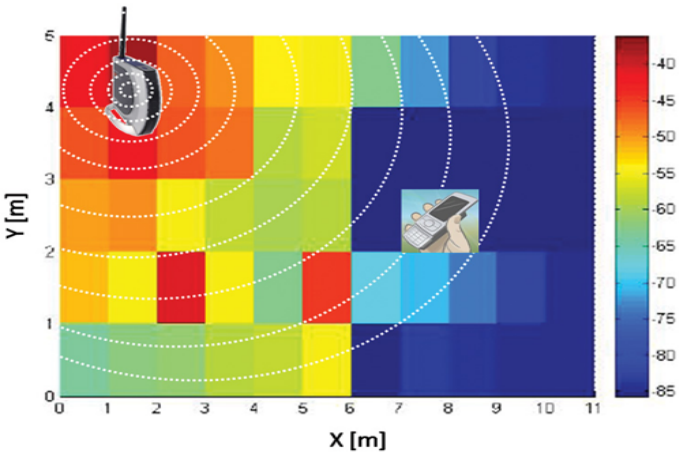

Figura 1. Gráfico exemplificando a base de dados constituída na fase *offline*.

Na fase *online*, um dispositivo faz a leitura dos sinais RSSI disponíveis e envia para um servidor, que os compara aos dados armazenados previamente, retornando o local inferido em tempo real. A aplicação desenvolvida neste trabalho utiliza algoritmos de aprendizagem de máquina para auxiliar nas comparações da fase *online* e obter melhores previsões.

V. Aprendizagem de Máquina

Simon [17] define que aprendizado de máquina é qualquer mudança em um sistema que melhore o seu desempenho de forma automática em uma posterior repetição da mesma tarefa ou em outra tarefa utilizando a mesma base.

O campo da inteligência artificial que estuda estes algoritmos é denominado de aprendizagem de máquina (*machine learning*). Tais algoritmos podem ser classificados conforme o seu tipo de aprendizado: supervisionado ou não-supervisionado.

No aprendizado supervisionado, uma massa de dados composta por uma coleção de atributos e as respectivas previsões esperadas é utilizado como treinamento inicial do algoritmo. Se os padrões de saída esperados possuírem valores discretos (e. g. positivo ou negativo), o problema se torna uma classificação. Mas, se os padrões forem compostos por valores contínuos (e. g. escala de números), o problema é definido como uma regressão.

A fase de treinamento inicial não está presente nos algoritmos não-supervisionados. Estes algoritmos baseiam seu aprendizado apenas nas relações presentes no conjunto de dados sob análise. Um exemplo de aplicação desta metodologia são os algoritmos de clusterização, cujo objetivo é agrupar dados similares entre si.

Classificadores são algoritmos de aprendizagem de máquina que determinam a qual classe, dentre um conjunto de classes previamente definido, um conjunto de dados pertence.

Ao elaborar um modelo utilizando esses algoritmos, deve-se observar como este generaliza. Isto é, como o algoritmo se comporta com novos dados - não contemplados na base de treino. A situação de *overfitting* ocorre quando o modelo funciona bem para a base de treino, mas não generaliza de forma satisfatória.

Outro aspecto a ser analisado ao classificar um conjunto de dados com estes modelos é a existência de pontos muito distantes dos demais da sua mesma classe. Esses pontos são denominados *outliers*.

A. Support Vector Machines

O *Support Vector Machines* é um algoritmo de aprendizagem de máquina supervisionado adequado para classificações. Seu funcionamento se baseia na construção de um hiperplano com a maior distância possível separando duas classes no espaço vetorial estudado [18]. Para utilizar este modelo de classificação, os dados devem ser representados em forma de vetores.

Para exemplificar o funcionamento dos *Support Vector Machines*, consideremos uma base de treino composta por círculos e quadrados - representados como vetores de duas dimensões - conforme Figura 2. O classicador irá analisar os dados e traçar uma reta separando as duas classes de dados. A figura apresenta, de forma simplificada, a separação realizada. Em casos reais, é comum que a quantidade de dimensões ultrapasse as centenas de milhares, impossibilitando uma representação gráfica.

Em alguns casos não é possível separar de forma linear a massa de treino. Nestes casos, um função Φ é aplicada para mapear os vetores para uma dimensão de ordem superior, permitindo a separação. Esta função é denominada *kernel* e o seu funcionamento é demonstrado na Figura 3.

Para evitar a situação de *overfitting*, a implementação do algoritmo permite a definição de pesos para pontos classificados de maneira errada, geralmente descrito como C (custo). Um C de valor alto aumenta a penalização por erro, aprimorando a precisão do modelo.

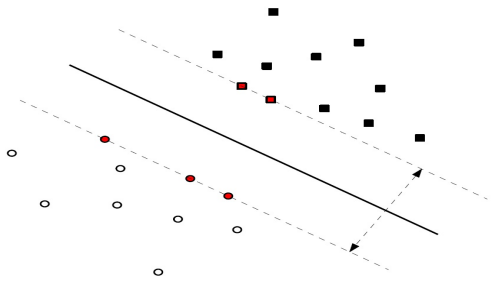

Figura 2. Hiperplano *Support Vector Machines*.

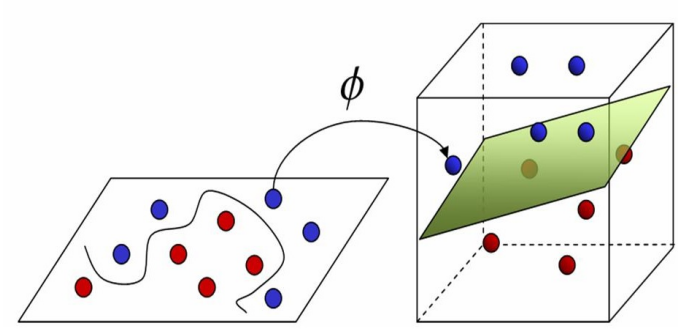

Figura 3. Função Φ mapeando os vetores para uma dimensão de ordem superior.

B. Naive Bayes

Naive Bayes é um algoritmo de aprendizagem de máquina supervisionado e probabilístico. Baseia-se no Teorema de Bayes, representado na Figura 4, sendo considerado um dos mais eficientes em tempo de processamento e precisão quando da classificação de novas amostras. Por ser de fácil implementação e apresentar bom desempenho, é um dos algoritmos mais utilizados em aprendizagem de máquina [19]. A facilidade de implementação e boa performance são justificadas pela abordagem simplória com que o algoritmo trata as características das amostras.

O algoritmo parte do princípio que dado um conjunto contendo grupos divididos por valores e atributos é possível predizer em qual grupo uma nova instância pertence. Este conceito é a base para o Teorema de Bayes, sendo definido como "a probabilidade de A dado B", ou seja, dado um conjunto de evidências B, qual a probabilidade da hipótese A estar em B.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Figura 4. Teorema de Bayes

O classificador é considerado *naive* (ingênuo) por assumir que os atributos são independentes. Isto é, a informação de um evento não tem efeito sobre nenhum outro (o que não é verdade para a maioria dos problemas práticos). Apesar da premissa simplista, o algoritmo apresenta desempenho satisfatório para diversas tarefas de classificação. A Figura 5 ilustra o modelo, indicando que os atributos *Ai* influenciam a classe *C*, mas não exercem efeito entre si.

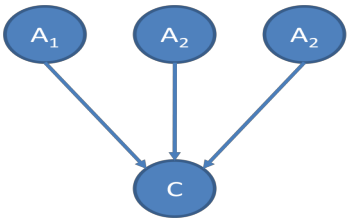

Figura 5. Modelo *Naive Bayes*

O algoritmo realiza a classificação de novos dados a partir da aplicação de fórmulas estatísticas e cálculos de probabilidades sobre os dados iniciais. Seu desempenho e acurácia são comparáveis às redes neurais e árvores de decisão [20].

Outra característica que torna o *Naive Bayes* eficiente é a realização em uma única etapa da leitura dos dados de treinamento e cálculo das probabilidades utilizadas durante a classificação. Além disso, o modelo pode ser usado de forma incremental, incluindo novos dados e efetuando a revisão das probabilidades.

Existem dois tipos de modelos estatísticos para os classificadores *Naive Bayes*: o modelo binário e o modelo multinomial. O modelo binário representa o objeto de análise através de um vetor binário, indicando a não-ocorrência de algum atributo com o valor 0 (zero) e o valor 1 (um) para ocorrências. Já o modelo multinomial assume que o objeto de análise é representado por um vetor de valores inteiros, caracterizando o número de vezes que cada atributo ocorre. Neste trabalho foi utilizado o modelo multinomial.

VI. Trabalhos Relacionados

Nas últimas duas décadas, diversas técnicas de localização foram propostas na literatura. Nesta seção serão apresentados os principais trabalhos correlatos, sendo analisadas as suas funcionalidades e características.

O mapeamento por *fingerprinting* permite a utilização de diferentes tipos de sinais. Pesquisadores exploraram e documentaram a utilização da técnica com variados tipos. Foram analisados cinco trabalhos sobre a utilização de diferentes tipos de sinais: WiFi [21], RFID [22], rádio FM [23], acústica [24] e magnetismo [25]. Após a análise dos trabalhos, constatou-se que as redes WiFi apresentam-se como a alternativa mais atrativa, graças a sua pervasividade, simplicidade e baixo custo.

Atualmente, os *smartphones* e outros dispositivos móveis dispõem de diversos sensores. Uma quantidade significativa dos trabalhos na área de localização têm utilizado estes sensores com o objetivo de reduzir ou minimizar o esforço de pesquisa *on-site* (fase *offline* do mapeamento). Esta técnica foi encontrada em três dos trabalhos analisados durante a elaboração deste referencial teórico: *LiFS* [26], *unloc* [27] e *Zee* [28].

Sofisticados modelos probabilísticos e algoritmos de aprendizagem de máquina têm sido aplicados [29] na fase *online* do mapeamento para obter maior precisão. O estudo [30] analisou diversos algoritmos em um ambiente realístico e verificou que, apesar de contraintuitivo, a utilização de algoritmos mais simples frequentemente apresenta precisão superior àqueles mais sofisticados.

No mapeamento através de *fingerprinting* de redes WiFi, a ocorrência de locais com assinaturas semelhantes é um problema frequente. Para diminuir este tipo de falha, os autores em [31] uniram dados da acústica do ambiente às assinaturas coletadas. Apesar de obter uma taxa menor de falhas, este método requer calibragem adicional ou a presença de um grande número de pessoas no ambiente.

Os sinais das redes WiFi tendem a sofrer influência de diversos elementos do ambiente, prejudicando a acurácia do processo de localização. Para minimizar essa instabilidade, alguns trabalhos utilizam informações da camada física da infraestrutura de rede (geralmente o parâmetro *Channel State Information - CSI*). A utilização destas informações traz ganhos significativos de precisão, porém apresenta alto custo para ubiquidade da solução, pois a maioria dos dispositivos móveis não dispõe destes dados.

Em relação aos trabalhos elencados, o InMedia diferencia-se por:

- Utilizar apenas a assinatura *RSSI* das redes WiFi do entorno do usuário;
- Aplicar algoritmos de aprendizagem de máquina simples e performáticos (*Naive Bayes* e *Support Vector Machines*);
- Permitir que usuários mapeiem ambientes e distribuam conteúdo digital conforme as localizações de forma trivial.

VII. Arquitetura

Um levantamento de requisitos bem feito é fundamental para o sucesso do projeto e para a produção de um software de qualidade. Conforme [32], o levantamento de requisitos de um projeto é a formalização do conhecimento coletado sobre o que a aplicação deve fazer e quais propriedades globais o sistema deve possuir.

Conforme [33], requisitos funcionais podem ser definidos como especificações de funcionalidades do sistema. Ou seja, como o sistema deve reagir a entradas específicas e como deve se comportar em determinadas situações. Para que não existam ambiguidades no projeto, é importante que a especificação destes requisitos seja completa e consistente. Os requisitos funcionais identificados para o InMedia são apresentados na Tabela I.

A definição dos requisitos não-funcionais está relacionada às propriedades globais (qualidades) que o sistema deve apresentar. Os requisitos não-funcionais tratam de características como escalabilidade, interoperabilidade, facilidade de manutenção, desempenho, portabilidade e segurança [33]. A análise dos requisitos não-funcionais do InMedia é demonstrada na Tabela II.

ID	Requisito Funcional	Ator
RF1	O sistema deve permitir o cadastro de novos usuários. Os *logins* utilizados devem ser únicos.	Usuário
RF2	O sistema deve permitir o download das aplicações de mapeamento e navegação em formato **apk** para a plataforma Android.	Usuário
RF3	O sistema deve permitir o mapeamento de locais através dos sinais das redes WiFi do entorno.	Usuário
RF4	O sistema deve permitir a apresentação de conteúdos, durante a navegação, sensíveis ao local em que o usuário se encontra.	Usuário
RF5	O sistema deve permitir a edição do conteúdo a ser exibido para os locais mapeados.	Usuário

Tabela I
Requisitos Funcionais do InMedia

ID	Requisito Não-Funcional	Categoria
RNF1	O sistema deverá ser acessado por um navegador de Internet moderno ou aplicativos para plataforma Android.	Usabilidade
RNF2	A interface deve ser de fácil utilização para o usuário.	Usabilidade
RNF3	O sistema deverá deverá utilizar folhas de estilo CSS no módulo Web e o padrão *Material Design* nas aplicações Android.	Padronização
RNF4	A conexão com o banco de dados deve ser feita através de um ORM, abstraindo as consultas SQL.	Portabilidade
RNF5	Usuários não devem ter acesso aos mapeamentos de outros usuários.	Segurança
RNF6	O sistema será desenvolvido utilizando as linguagens Ruby, Python e Java, além do banco de dados PostgreSQL.	Software

Tabela II
Requisitos não-funcionais do InMedia

A arquitetura do InMedia é dividida em quatro componentes principais: uma aplicação web, um subsistema de aprendizagem de máquina e dois aplicativos mobile.

O módulo web permite o gerenciamento dos usuários e dos conteúdos vinculados aos locais mapeados. O componente é organizado em três camadas, seguindo o padrão *Model-View-Controller* (MVC). O padrão MVC separa a representação da informação da interação do usuário com a mesma. A camada de modelo deve consistir de dados da aplicação, regras de negócio, lógica e funções. Uma visão pode ser qualquer saída que represente dados, como um gráfico, um diagrama ou uma página da web. O padrão possibilita a existência de múltiplas visões para um mesmo leque de dados. A camada de controle media a entrada, convertendo-a em comandos para camada de modelos ou de visão ou outros subsistemas. Este padrão tem como objetivos centrais a reutilização de código e separação de responsabilidades [34].

A camada de visão é representada pela interface gráfica e é responsável por fornecer as funções de gerenciamento de usuários e de conteúdos para os locais mapeados. As ações executadas na camada de visão geram requisições que são encaminhadas para a camada de controle. Esta camada é responsável por orquestrar os pedidos enviados a partir da camada de visão, que servem para acionar as funcionalidades da camada de modelo e do subsistema de aprendizagem de máquina. As informações retornadas são tratadas e repassadas

à camada de visão, onde são apresentadas para o usuário. A camada de modelo tem como função abstrair o acesso aos dados dos usuários e dos mapeamentos armazenados.

As camadas de visão, controle e modelo interagem através de conectores do tipo *Procedure Call*. Os conectores do tipo *Procedure Call* modelam o fluxo de controle utilizando técnicas de invocação e realizam a transferência de dados entre os componentes envolvidos através do uso de parâmetros. Exemplos de conectores *Procedure Call* incluem funções, procedimentos, métodos da orientação a objetos, chamadas de *callback* e *system calls* [35]. No componente web, os conectores são representados por métodos da orientação a objetos.

As aplicações móveis foram desenvolvidas para plataforma Android e são utilizadas para mapeamento e navegação. Estes aplicativos realizam a leitura periódica dos sinais WiFi do entorno e enviam para o módulo web através de uma API. A aplicação web realiza o armazenamento dos dados ou infere a localização do usuário com o apoio do subsistema de aprendizagem de máquina. A comunicação entre as aplicações e a API é realizada através de conectores *Remote Procedure Call*. Estes conectores também modelam o fluxo de controle, contudo os componentes envolvidos não se encontram no mesmo *host*.

Além dos componentes citados, um subsistema de aprendizagem de máquina completa o modelo arquitetural construído. Este subsistema é responsável por inferir a localização do usuário, aplicando os classificadores *Naive Bayes* e *Support Vector Machines* às informações contextuais passadas pela aplicação mobile de navegação. Toda a arquitetura do sistema pode ser visualizada na Figura 6.

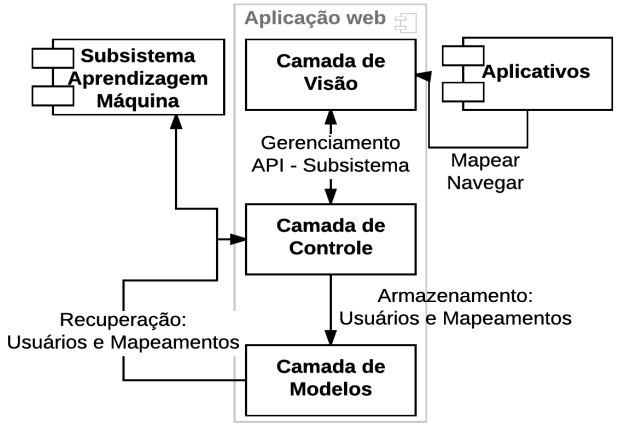

Figura 6. Arquitetura do InMedia

VIII. Modelagem do Sistema

A aplicação web é composta por seis classes principais, duas classes do tipo *controller* e quatro do tipo modelo. A classe *PagesController* orquestra as requisições oriundas da camada de visão para a visualização e alteração dos dados do gerenciamento de usuários e conteúdos vinculados aos mapeamentos. O controlador *ApiController* disponibiliza uma API às aplicações Android. Esta API permite a validação do *token* de segurança do usuário, o armazenamento de novos mapeamentos e rastrear (inferir) a localização do usuário. O diagrama de classes do módulo web é apresentado na Figura 7.

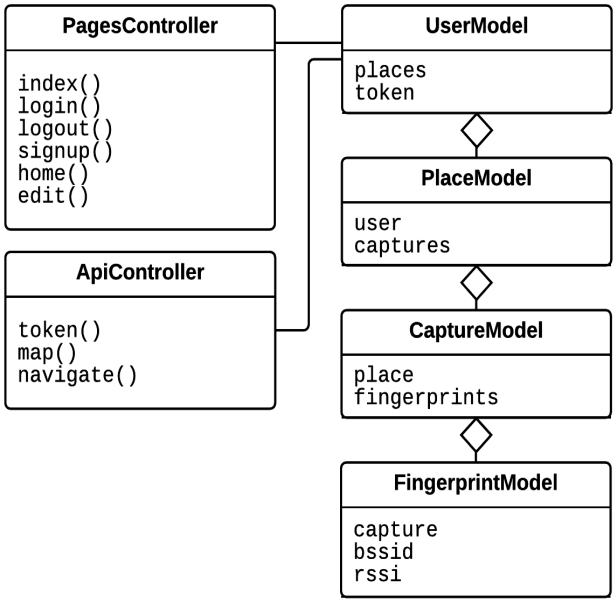

Figura 7. Diagrama de Classes - Módulo Web

Os modelos estão organizados de forma hierárquica, sendo que um usuário (*UserModel*) possui diversos locais (*PlaceModel*) mapeados através de diversas capturas (*CaptureModel*) de múltiplas assinaturas (*FingerprintModel*).

As aplicações móveis apresentam estruturas de classes semelhantes. Contudo, a apresentação visual e métodos da API utilizados são diferentes. Os aplicativos são compostos por três classes e uma interface. A classe principal *MainActivity* implementa a interface *WifiReceiverListener* através da concretização do método *scanResultsReceived()*. A lógica deste método é encapsulada em uma instância agregada da classe *WifiReceiver*. O objeto *WifiReceiver*, ao ser notificado de uma nova varredura realizada pelo controlador WiFi, realiza a inversão de controle para a atividade principal, que constrói o objeto *Fingerprint* e envia para a API do módulo web. A Figura 8 apresenta o diagrama de classes das aplicações móveis.

O processo de mapeamento é composto pela verificação do *token* de segurança do usuário e, após a confirmação, o armazenamento das assinaturas dos sinais das redes WiFi, relacionado-os a um local específico. A Figura 9 demonstra este processo através de um diagrama de sequência.

Outro processo importante para o InMedia é a inferência da localização, realizada a partir do aplicativo de navegação.

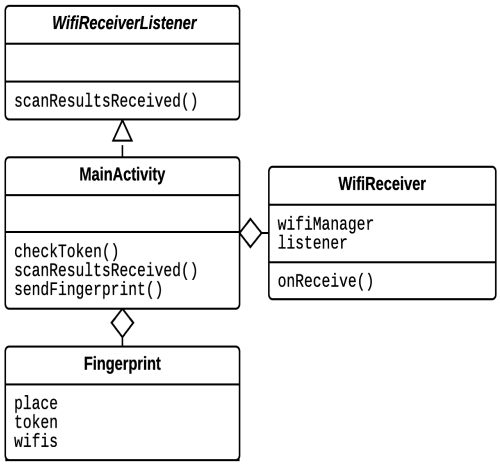

Figura 8. Diagrama de Classes - Aplicativos

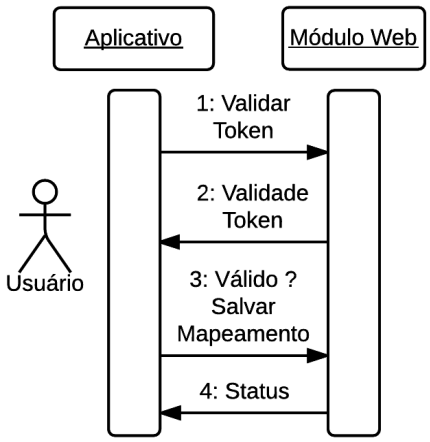

Figura 9. Diagrama de sequência - Mapeamento

Este procedimento também realiza a validação do *token* de segurança do usuário e, após a verificação, infere o local em que se encontra a partir das assinaturas das redes WiFi do entorno. Este procedimento é representado no diagrama de sequência 10.

O subsistema de aprendizagem de máquina foi desenvolvido utilizando a biblioteca *scikit-learn*. O classificador *Support Vector Machines* é implementado através da classe *LinearSVC*. Esta classe especifica um classificador de *kernel* linear, baseado na biblioteca *liblinear*. O classificador do tipo *Naive Bayes* foi feito utilizando a classe *MultinomialNB*. O subsistema cria instâncias das classes citadas e treina os modelos através do método *fit*. Para realizar inferências sobre os conjuntos de dados, o método *predict* é utilizado.

Ao receber um novo conjunto de dados, a aplicação realiza a classificação através dos dois algoritmos. O resultado do

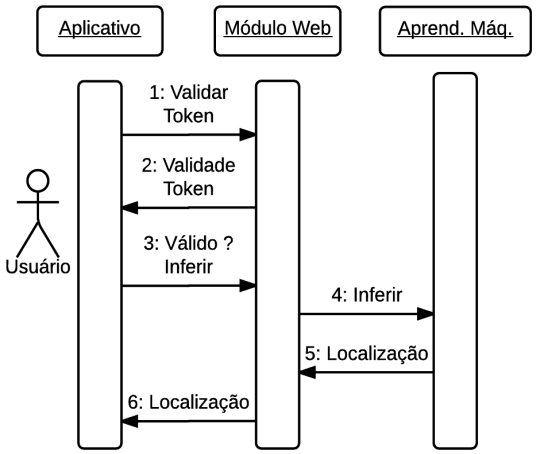

Figura 10. Diagrama de sequência - Inferência

algoritmo com maior índice de confiança é, então, utilizado.

IX. Principais Funcionalidades

O acesso ao InMedia é feito mediante um *login* tradicional com solicitação de usuário e senha. Assim, para um usuário acessar o sistema, ele deve ser previamente cadastrado. O cadastro requer apenas informações de *login* e *senha* e é público. Após a autenticação, o usuário é apresentado à tela inicial, representada na Figura 11.

Figura 11. Tela inicial InMedia

Na tela inicial é possível visualizar o *token* de segurança do usuário e realizar o download das aplicações móveis. No painel inferior da tela, são listados todos os locais mapeados pelo usuário. Através desta tabela é possível editar o conteúdo vinculado à localização.

Ao iniciar a aplicação de mapeamento, o *token* de segurança do usuário é solicitado (Figura 12). Após a validação do *token* junto à API, a tela inicial é apresentada.

A interface principal do aplicativo de mapeamento é composta por um botão *switcher*, um campo de texto - para inserção do nome do local a ser mapeado - e um *label* na

Figura 12. Aplicativo Mapeamento - Solicitação Token

parte inferior para apresentação das respostas da API (Figura 13).

Figura 13. Aplicativo Mapeamento - Tela Inicial

O mapeamento ocorre quando o nome da localidade é preenchido e o botão *switcher* é ativado. A Figura 14 representa a aplicação em execução.

A aplicação de navegação atua de forma similar à aplicação de mapeamento. Ao ser iniciada, também é solicitado o *token* de segurança do usuário. Diferentemente da aplicação de mapeamento, o aplicativo de navegação não apresenta opções de interação. Sendo composto apenas por um *label* na parte superior, que armazena o nome do local identificado,

Figura 14. Aplicativo Mapeamento - Em execução

e um componente *WebView* para apresentação do conteúdo vinculado ao local. A utilização deste componente permite que os conteúdos sejam formatados em linguagem HTML e incorporem elementos multimídia. A Figura 15 apresenta a tela principal do aplicativo de navegação.

Figura 15. Aplicativo Navegação

O aplicativo de navegação só apresenta o conteúdo para o local inferido após um processo de estabilização das previsões obtidas. Isto é, o aplicativo aguarda a ocorrência de pelo menos cinco (5) previsões iguais e sequenciais, para apresentar o conteúdo vinculado. Esta quantidade foi definida após testes da aplicação em ambientes variados.

Para testar o funcionamento da solução e a qualidade das previsões, foi criado um módulo de testes derivado da aplicação de navegação. Este módulo contém as mesmas funcionalidades que o aplicativo original, porém permite que o usuário marque as previsões obtidas como corretas ou incorretas. A aplicação é demonstrada na Figura 16.

Figura 16. Aplicativo Navegação - Testes

A. *Características Ubíquas*

Nesta subseção serão apresentadas as características ubíquas da solução em relação às funcionalidades detalhadas ao longo da seção.

- *Invisibilidade* - Ao iniciar a aplicação de navegação, o usuário passa a receber informações sobre o local onde se encontra, sem que o mesmo precise informar algum parâmetro para a aplicação. As tecnologias e sistemas integrados ao InMedia, tais como, API, subsistema de aprendizagem de máquina e o SGBD, são transparentes para o usuário.
- *Heterogeneidade* - O sistema proposto interliga tecnologias e plataformas distintas, dentre as quais pode-se destacar: dispositivos móveis, base de dados, *web service*, servidor de aplicação, entre outros.
- *Sensibilidade ao Contexto* - A aplicação adequa o conteúdo apresentado ao contexto - localização - do usuário. Esta adequação ocorre de modo transparente e tem como objetivo aumentar o grau de satisfação do utilizador.

X. Testes e Resultados

O processo de validação do InMedia foi realizado através de testes automatizados e resultados experimentais. As subseções seguintes apresentam as metodologias utilizadas e os resultados obtidos.

A. *Testes Automatizados*

Testes unitários, também chamados de testes de unidade, testes de módulos ou testes de componentes, avaliam a menor unidade lógica ou bloco de um programa. Na programação estruturada, por exemplo, esta unidade pode ser definida como um procedimento ou função [36].

Os testes unitários fazem parte de um conjunto de testes chamados de testes de caixa branca [37]. Os testes de caixa branca são testes que conhecem e utilizam a estrutura interna do sistema (funções, classes, etc.). Já os testes de caixa preta, enviam uma entrada e aguardam uma saída, de forma similar a como um usuário faria.

Testes automatizados têm como objetivo verificar se o software faz corretamente e adequadamente todos os detalhes especificados no projeto. Em sistemas grandes e complexos, compostos por camadas de lógica, dados e interface com o usuário, estes testes devem abranger todos os possíveis caminhos de código, avaliar telas, menus, mensagens, validação de campos e o tratamento de exceções.

Uma boa estratégia para aumentar a abrangência dos testes é aliar testes de caixa branca com testes de caixa preta. Segundo [36], testes de caixa branca como os testes unitários geralmente utilizam cerca de 60% do esforço e custo dos testes em um sistema, por outro lado, podem garantir um aumento de cerca de 40% na qualidade do software desenvolvido.

Para a criação dos testes automatizados do InMedia foram utilizados dois *frameworks*: JUnit e SeleniumHQ. O JUnit é um *framework* para codificação de testes unitários na linguagem Java. Através do JUnit é possível criar classes de teste que podem ser organizadas hierarquicamente, permitindo a realização de testes em partes separadas ou um teste completo do código [38]. Este *framework* é utilizado para realização de testes nas aplicações de mapeamento e navegação.

O Selenium HQ pode ser definido como um conjunto de ferramentas de código aberto para o desenvolvimento rápido de testes funcionais do tipo caixa preta. Este conjunto de ferramentas oferece diversas opções para a realização de testes em aplicações baseadas na web [39]. O InMedia utiliza uma extensão do JUnit que permite controlar as ferramentas do Selenium HQ através de comandos Java para testar o módulo web e a API disponibilizada às aplicações mobile.

B. *Resultados Experimentais*

O ambiente utilizado para a realização dos testes foi o campus do Instituto Federal de Educação, Ciência e Tecnologia da Bahia, localizado no bairro do Barbalho, na cidade de Salvador. Foram mapeadas seis edificações: o prédio administrativo, a cantina e outros quatro pavilhões de aulas. Dentre as edificações, o prédio administrativo foi o único a ter seus andares mapeados como locais diferentes.

Essas edificações estão representadas de forma simplificada na Figura 17. A Tabela III especifica as principais características das construções, tais como: descrição, andares mapeados e localizações vinculadas.

O mapeamento das localizações deu-se da seguinte forma:

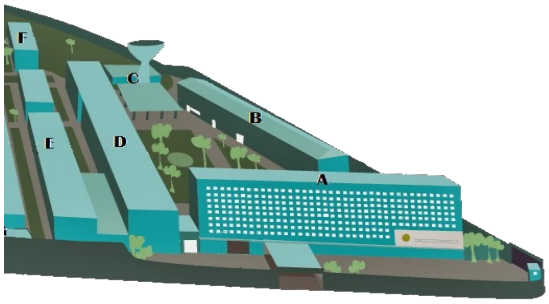

Figura 17. Edificações Mapeadas

Legenda	Descrição	Andares	Locais
A	Prédio administrativo.	1-3	A1, A2 e A3.
B	Pavilhão de aulas.	1	B
C	Cantina.	1	C
D	Pavilhão de aulas.	1	D
E	Pavilhão de aulas.	1	E
F	Pavilhão de aulas.	1	F

Tabela III
DESCRIÇÃO EDIFICAÇÕES MAPEADAS

1) Foram definidos cinco pontos de coleta para cada edificação, sendo um central e quatro nos extremos da edificação. Esta metodologia foi adotada com o objetivo de cobrir toda a região e capturar a maior variedade possível de assinaturas para aquele edifício;
2) Em cada ponto de coleta foram capturadas cinquenta (50) amostras dos sinais das redes disponíveis, totalizando duzentos e cinquenta (250) capturas por localização;
3) Durante a coleta, o dispositivo móvel sofreu translação em torno da pessoa que segurava o dispositivo. Assim, foram cobertos os mais diversos ângulos que um dispositivo pode ter dentro do espaço da edificação.

As amostras coletadas foram analisadas e tratadas para eliminar ruídos que atrapalham no processo de localização. Isto é, remover sinais oriundos de redes sem ponto de acesso (*AP*) fixo nas edificações. A Tabela IV apresenta a quantidade de redes WiFi detectadas e o número de assinaturas de sinal armazenadas para cada localização. É importante destacar que a quantidade de assinaturas coletadas não é um múltiplo exato do número de redes WiFi detectadas, pois as redes têm presença intermitente em relação ao ponto de coleta.

Local	Redes	Assinaturas
A1	4	911
A2	5	1.016
A3	7	1.740
B	4	930
C	3	691
D	6	1.476
E	5	1.057
F	5	1.188

Tabela IV
QUANTITATIVO DE REDES E ASSINATURAS COLETADAS

O módulo de testes foi utilizado para validar o mapeamento realizado. Para manter a integridade do experimento, a mesma metodologia foi aplicada. Para cada edificação foram realizadas duzentos e cinquenta (250) previsões e marcações, sendo cinquenta (50) em cada um dos cinco pontos de coleta definidos. A Tabela V apresenta os resultados desta análise.

Local	Acertos	Acertos (%)	Erros	Erros (%)
A1	145	58%	105	42%
A2	160	64%	90	36%
A3	177	71%	73	29%
B	190	76%	60	24%
C	195	78%	55	22%
D	197	79%	53	21%
E	190	76%	60	24%
F	200	80%	50	20%
Média	182	73%	68	27%

Tabela V
RESULTADOS - MÓDULO DE TESTES

Para este trabalho, considera-se acurácia como a razão entre o número de vezes que o local foi corretamente inferido e o número total de tentativas. O nível de acurácia obtido para cada local é representado na Figura 18.

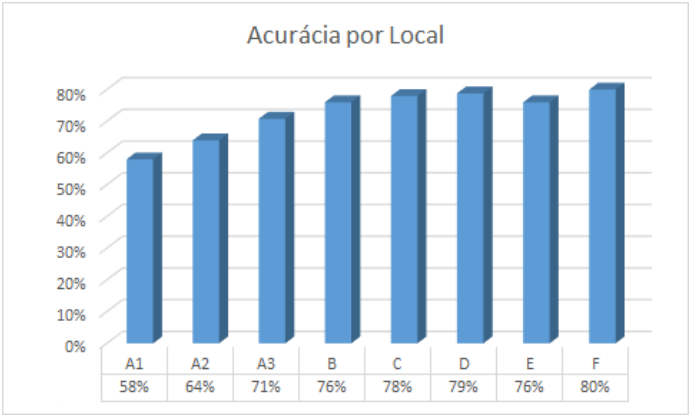

Figura 18. Acurácia x Local

A acurácia média encontrada depois de executar a aplicação para oito (8) locais, totalizando duas mil (2.000) previsões e análises, foi de 73%. Com acurácia por local mínima de 58% e máxima de 80%.

Um fator importante de se estimar é a sensibilidade do sistema à quantidade de redes WiFi detectadas. Para avaliar a importância deste fator, o nível de acurácia obtido e o número de redes encontradas por local foram relacionados. Esta relação está representada na Figura 19.

Verificou-se que o número de redes detectadas apresenta relação direta com o nível de acurácia obtido. Apresentando um efeito mais evidente para conjuntos de locais próximos. Esta relação é demonstrada ao analisar os níveis de acurácia obtidos para os andares mapeados no prédio administrativo (A1, A2 e A3).

Como forma de validar os classificadores utilizados neste trabalho (*Support Vector Machines* e *Naive Bayes*), o conjunto de dados produzido durante os testes também foi analisado

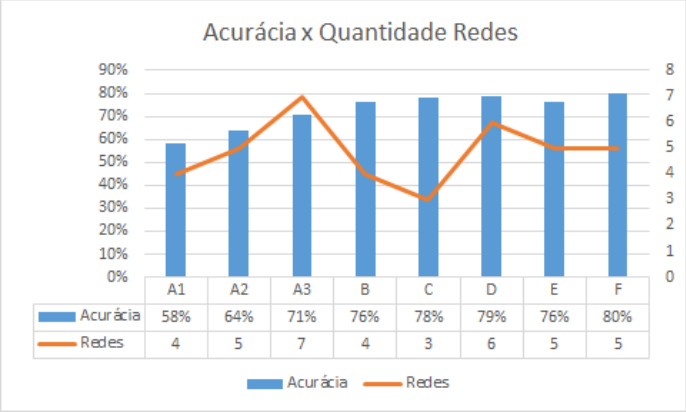

Figura 19. Acurácia x Redes

através dos algoritmos de aprendizagem de máquina: *K-Nearest Neighbor*, *K-Means* e *LinearRegression*. Estes algoritmos foram escolhidos pois, assim como o *Support Vector Machines* e *Naive Bayes*, são de fácil implementação e apresentam bons resultados em variadas tarefas e contextos de classificação.

O WEKA (*Waikato Environment for Knowledge Analysis*) [40] é uma ferramenta de código-aberto, desenvolvida na Universidade de Waikato, Nova Zelândia, que implementa diversos algoritmos de mineração de dados e aprendizagem de máquina. Esta ferramenta foi utilizada para testar os outros classificadores. Os resultados obtidos estão elencados na Tabela VI e representados graficamente na Figura 20.

Algoritmo	**Acurácia**
Naives Bayes e *Support Vector Machines*	73%
K-Nearest Neighbor	71%
K-Means	67%
LinearRegression	58%

Tabela VI
RESULTADOS - WEKA

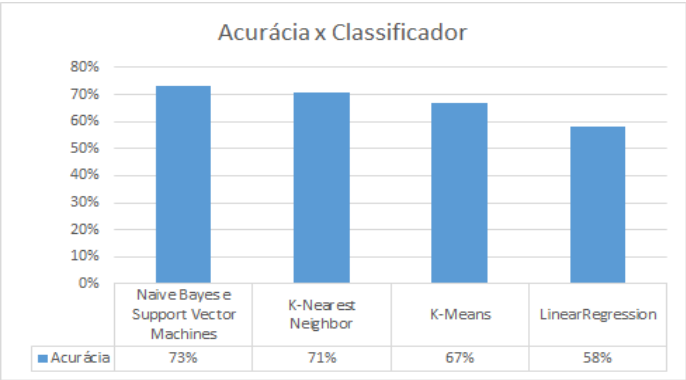

Figura 20. Acurácia x Classificador

A aplicação de outros classificadores permitiu confirmar que a utilização conjunta dos algoritmos *Naive Bayes* e *Support Vector Machines* é a configuração com maior acurácia. Apesar disso, o algoritmo *K-Nearest Neighbor* também obteve uma taxa satisfatória, devendo ter seu uso considerado em trabalhos futuros.

XI. Conclusão e Trabalhos Futuros

A ubiquidade dos smartphones e de outros dispositivos móveis realçou as vantagens das aplicações cientes de contexto. Dentre as dimensões de contexto existentes na literatura, a localização caracteriza-se como uma das mais importantes.

A pervasividade e o aumento da quantidade de sensores nestes dispositivos também permitiu que os sistemas de localização passassem a coletar grandes quantidades de dados. Para classificar com precisão e eficiência estes grandes conjuntos, os sistemas de localização têm aplicado algoritmos de aprendizagem de máquina.

Este trabalho apresentou o InMedia, um sistema web e mobile, criado para auxiliar na distribuição de mídias digitais de forma sensível ao contexto, principalmente à localização do usuário. Os algoritmos *Naive Bayes* e *Support Vector Machines* são aplicados em conjunto para obter maior precisão durante o processo de inferência da localização do usuário. A aplicação destes algoritmos em conjunto e a maior precisão obtida são os principais diferenciais deste trabalho.

A elaboração do projeto consistiu no desenvolvimento de uma aplicação web, dois aplicativos mobile para plataforma Android e um subsistema de aprendizagem de máquina. Sendo a aplicação web responsável pelo gerenciamento de usuários e conteúdos vinculados às localizações mapeadas; as aplicações mobile, pelos processos de mapeamento e navegação; e o subsistema de aprendizagem de máquina, pela inferência da localização do usuário.

A solução foi validada através de testes automatizados e resultados experimentais. Os testes automatizados foram utilizados na validação dos detalhes de implementação, confirmando se o que foi codificado está de acordo com o que foi definido durante a análise de requisitos.

Os resultados experimentais permitiram confirmar que a utilização dos classificadores *Naive Bayes* e *Support Vector Machines* em conjunto apresenta melhor acurácia em comparação a outros algoritmos de aprendizagem de máquina. Além disso, verificou-se que a quantidade de redes WiFi detectadas durante o mapeamento apresenta uma relação direta com a acurácia dos resultados. De forma mais evidente para conjuntos de locais próximos, e menos para conjuntos de locais distantes.

Apesar dos principais objetivos deste trabalho terem sido alcançados, a análise dos trabalhos relacionados e dos resultados obtidos demonstra que ainda existem trabalhos futuros a serem realizados. Como sugestões, pode-se elencar:

- Permitir a escolha dos algoritmos de aprendizagem de máquina a serem aplicados;
- Relatórios na aplicação web que permitam ao usuário avaliar a acurácia dos algoritmos em relação à base de mapeamentos constituída;
- Versão para plataforma iOS;
- Permitir a captura de outras informações sensoriais;

- Tornar as aplicações sensíveis a outras dimensões contextuais.

REFERÊNCIAS

[1] L. Aalto, N. Göthlin, J. Korhonen, and T. Ojala, "Bluetooth and wap push based location-aware mobile advertising system," in *Proceedings of the 2Nd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '04. New York, NY, USA: ACM, 2004, pp. 49–58. [Online]. Available: http://doi.acm.org/10.1145/990064.990073

[2] D. J. Cook, M. Huber, K. Gopalratnam, and M. Youngblood, "Learning to control a smart home environment," *In Innovative Applications of Artificial Intelligence*, vol. 2003, 2003.

[3] B. Sohn, J. Lee, H. Chae, and W. Yu, "Localization system for mobile robot using wireless communication with ir landmark," in *Proceedings of the 1st International Conference on Robot Communication and Coordination*, ser. RoboComm '07. Piscataway, NJ, USA: IEEE Press, 2007, pp. 6:1–6:6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1377868.1377876

[4] P. Norvig and S. Russell, *Inteligência artificial, 3a edição*. Elsevier Brasil, 2013.

[5] M. Weiser, "The computer for the 21st century," *Scientific American*, vol. 265, no. 3, pp. 66–75, January 1991. [Online]. Available: http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

[6] ——, "Hot topics—ubiquitous computing," *Computer*, vol. 26, no. 10, pp. 71–72, oct 1993.

[7] ——, "The world is not a desktop," *interactions*, vol. 1, no. 1, pp. 7–8, Jan. 1994. [Online]. Available: http://doi.acm.org/10.1145/174800.174801

[8] T. Kindberg and A. Fox, "System software for ubiquitous computing," *IEEE Pervasive Computing*, vol. 1, no. 1, pp. 70–81, Jan. 2002. [Online]. Available: http://dx.doi.org/10.1109/MPRV.2002.993146

[9] M. Beigl, H.-W. Gellersen, and A. Schmidt, "Mediacups: experience with design and use of computer-augmented everyday artefacts," *Computer Networks*, vol. 35, no. 4, pp. 401–409, 2001.

[10] M. Weiser and J. S. Brown, "Beyond calculation," P. J. Denning and R. M. Metcalfe, Eds. New York, NY, USA: Copernicus, 1997, ch. The Coming Age of Calm Technolgy, pp. 75–85. [Online]. Available: http://dl.acm.org/citation.cfm?id=504928.504934

[11] P. Dourish, "What we talk about when we talk about context," *Personal Ubiquitous Comput.*, vol. 8, no. 1, pp. 19–30, Feb. 2004. [Online]. Available: http://dx.doi.org/10.1007/s00779-003-0253-8

[12] M. Satyanarayanan, "Pervasive computing: Vision and challenges," *IEEE Personal Communications*, vol. 8, no. 4, pp. 10–17, aug 2001.

[13] B. N. Schilit and M. M. Theimer, "Disseminating active map information to mobile hosts," *Netwrk. Mag. of Global Internetwkg.*, vol. 8, no. 5, pp. 22–32, Sep. 1994. [Online]. Available: http://dx.doi.org/10.1109/65.313011

[14] N. Malik, U. Mahmud, and Y. Javed, "Future challenges in context-aware computing," in *proceedings of the IADIS International Conference WWW/Internet*, 2007, pp. 306–310.

[15] A. K. Dey, "Understanding and using context," *Personal Ubiquitous Comput.*, vol. 5, no. 1, pp. 4–7, Jan. 2001. [Online]. Available: http://dx.doi.org/10.1007/s007790170019

[16] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing*, ser. HUC '99. London, UK, UK: Springer-Verlag, 1999, pp. 304–307. [Online]. Available: http://dl.acm.org/citation.cfm?id=647985.743843

[17] H. A. Simon, *Why Should Machines Learn?* Berlin, Heidelberg: Springer Berlin Heidelberg, 1983, pp. 25–37. [Online]. Available: http://dx.doi.org/10.1007/978-3-662-12405-5_2

[18] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995. [Online]. Available: http://dx.doi.org/10.1023/A:1022627411411

[19] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 338–345. [Online]. Available: http://dl.acm.org/citation.cfm?id=2074158.2074196

[20] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[21] M. Youssef and A. Agrawala, "The horus wlan location determination system," in *MobiSys '05: Proceedings of the 3rd international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM Press, 2005, pp. 205–218. [Online]. Available: http://dx.doi.org/10.1145/1067170.1067193

[22] L. M. Ni, Y. Liu, Y. C. Lau, and A. P. Patil, "Landmarc: Indoor location sensing using active rfid," *Wirel. Netw.*, vol. 10, no. 6, pp. 701–710, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:WINE.0000044029.06344.dd

[23] S. Yoon, K. Lee, and I. Rhee, "Fm-based indoor localization via automatic fingerprint db construction and matching," in *Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '13. New York, NY, USA: ACM, 2013, pp. 207–220. [Online]. Available: http://doi.acm.org/10.1145/2462456.2464445

[24] S. P. Tarzia, P. A. Dinda, R. P. Dick, and G. Memik, "Indoor localization without infrastructure using the acoustic background spectrum," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 155–168. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000011

[25] J. Chung, M. Donahoe, C. Schmandt, I.-J. Kim, P. Razavai, and M. Wiseman, "Indoor location sensing using geo-magnetism," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '11. New York, NY, USA: ACM, 2011, pp. 141–154. [Online]. Available: http://doi.acm.org/10.1145/1999995.2000010

[26] Z. Yang, C. Wu, and Y. Liu, "Locating in fingerprint space: Wireless indoor localization with little human intervention," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 269–280. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348578

[27] H. Wang, S. Sen, A. Elgohary, M. Farid, M. Youssef, and R. R. Choudhury, "No need to war-drive: Unsupervised indoor localization," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '12. New York, NY, USA: ACM, 2012, pp. 197–210. [Online]. Available: http://doi.acm.org/10.1145/2307636.2307655

[28] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen, "Zee: Zero-effort crowdsourcing for indoor localization," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 293–304. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348580

[29] *WLAN location determination via clustering and probability distributions*, 2003. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1192736

[30] D. Turner, S. Savage, and A. C. Snoeren, "On the empirical performance of self-calibrating wifi location systems." in *LCN*, C. T. Chou, T. Pfeifer, and A. P. Jayasumana, Eds. IEEE Computer Society, 2011, pp. 76–84. [Online]. Available: http://dblp.uni-trier.de/db/conf/lcn/lcn2011.html#TurnerSS11

[31] H. Liu, Y. Gan, J. Yang, S. Sidhom, Y. Wang, Y. Chen, and F. Ye, "Push the limit of wifi based localization for smartphones," in *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, ser. Mobicom '12. New York, NY, USA: ACM, 2012, pp. 305–316. [Online]. Available: http://doi.acm.org/10.1145/2348543.2348581

[32] J. Conallen, "Modeling web application architectures with uml," *Commun. ACM*, vol. 42, no. 10, pp. 63–70, Oct. 1999. [Online]. Available: http://doi.acm.org/10.1145/317665.317677

[33] B. H. C. Cheng and J. M. Atlee, "Research directions in requirements engineering," in *2007 Future of Software Engineering*, ser. FOSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 285–303. [Online]. Available: http://dx.doi.org/10.1109/FOSE.2007.17

[34] E. Rozanski, N. e Woods, *Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives*. Pearson Education, 2011. [Online]. Available: http://books.google.com.br/books?id=nXRF77-gxRkC

[35] A. W. Kiwelekar, "Architectural connectors," http://www.cse.iitb.ac.in/~awk/aconnector.pdf, 2004, acessado em: 27/07/2013.

[36] E. Dustin, J. Rashka, and J. Paul, *Automated software testing: introduction, management, and performance*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[37] H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," *ACM Comput. Surv.*, vol. 29, no. 4, pp. 366–427, Dec. 1997. [Online]. Available: http://doi.acm.org/10.1145/267580.267590

[38] V. Massol and T. Husted, *JUnit in Action*. Greenwich, CT, USA: Manning Publications Co., 2003.

[39] B. David, *Selenium 2 Testing Tools: Beginner's Guide*. Packt Publishing, 2012.

[40] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278