Ranqueamento de Produtores de Dados na Internet das Coisas

Alternative Title: Data Producer Ranking in the Internet of Things

Emanoel C. G. F. Silva
Centro de Informática, UFPE
Jornalista Aníbal Fernandes
Recife, Brasil, 50740-560
ecgfs@cin.ufpe.br

Kiev Gama
Centro de Informática, UFPE
Jornalista Aníbal Fernandes
Recife, Brasil, 50740-560
kiev@cin.ufpe.br

Bernadete Farias Lóscio
Centro de Informática, UFPE
Jornalista Aníbal Fernandes
Recife, Brasil, 50740-560
bfl@cin.ufpe.br

RESUMO

Com a ascensão da Internet das Coisas (*Internet of Things - IoT*), bilhões de dispositivos estarão conectados à internet do futuro produzindo, consumindo e processando dados e se comunicando uns com os outros. Descobrir e selecionar de forma eficiente os dispositivos que melhor respondem a uma determinada necessidade se mostram como problemas relevantes a serem investigados no paradigma IoT. Face a este problema, o presente trabalho propôs: *(i)* o uso e o monitoramento dinâmico de métricas de qualidade na descrição dos produtores de dados, *(ii)* proposta de uma técnica de ranqueamento de produtores de dados que utilize atributos de qualidade diversos, *(iii)* a proposta do conceito de fila dinâmica de resultados no mecanismo de busca, visando economia de processamento e ganho de desempenho e *(iv)* uso do estilo arquitetural REST para a oferta dos produtores de dados como recursos. Por fim, foi considerado um cenário de uso do aplicativo móvel *Bike Cidadão* com o objetivo de avaliar a performance das contribuições propostas.

Palavras-Chave

Internet das Coisas, Qualidade de Contexto, Qualidade de Dados, Seleção de Dispositivos

ABSTRACT

With the rise of the Internet of Things (IoT) billion devices will be connected to the internet of the future producing, consuming and processing data and communicating with each other. Discover and select efficiently the devices best suited to a particular need, appear as relevant issues to be investigated in the IoT. In view of this problem, the present work proposes: *(i)* using and monitoring quality metrics in the description of the data producers, *(ii)* the propose of a ranking of data producers technique that makes use of quality attributes, *(iii)* the suggestion of dynamic queue results by the search engine and *(iv)* the use of REST style for providing data producers as resource. Finally, it was considered a use case scenario of the mobile application *Bike Cidadão* in order to evaluate the performance of the proposed contributions. The evaluation noted, above all, the response time required to perform queries to the catalog using, or not, the concept of dynamic queue results in different situations, varying the amount of data available to consumers and producers.

SBSI 2017 June 5^{th} – 8^{th}, 2017, Lavras, Minas Gerais, Brazil

CCS Concepts

•**General and reference** → *Metrics;* •**Computer systems organization** → *Architectures;* •**Software and its engineering** → *Interoperability;*

Keywords

Internet of Things, Quality of Context, Data Quality, Device Selection

1. INTRODUÇÃO

A Internet das Coisas *Internet of Things - IoT* permite que pessoas e coisas diversas possam ser conectadas entre si a qualquer momento e lugar, preferencialmente usando qualquer caminho/rede e qualquer serviço [10]. Percebe-se que sua aplicação é bastante ampla e diversos domínios de aplicação são beneficiados com o potencial de desenvolvimento de aplicações que antes não eram possíveis como, por exemplo, aquelas chamadas de *crowdsensing* [8].

Sistemas sensíveis a contexto [6] são diretamente beneficiados com o desenvolvimento da IoT. No entanto, dada a enorme quantidade de serviços que vem sendo ofertados pelos dispositivos utilizando a IoT, pode ser muito difícil identificar quais são aqueles que oferecem exatamente a informação que o consumidor deseja e, o mais importante, a um nível de qualidade adequado.

Vários trabalhos propõem formas de quantificar e considerar a qualidade do contexto, dos serviços e dos dados gerados pelos produtores de dados (sensores, serviços web, etc) visando selecionar aqueles resultados que melhor atendem às necessidades da aplicação. Na literatura já se encontra seleção de serviços considerando características de qualidade [29, 35, 33, 7] e um comparativo entre várias dessas técnicas, destacando sua aplicabilidade, seus prós e contras [28]. No entanto, considerando o cenário de IoT, onde não apenas humanos fazem requisições, mas também sistemas e dispositivos, é necessário que sejam satisfeitos requisitos básicos

como: a não exigência de interação explícita ente consumidor e mecanismo de busca e eficiência no tempo de resposta.

Este trabalho busca responder à questão de pesquisa *"É possível implementar uma técnica de ranqueamento que seja eficiente, do ponto de vista dos resultados recomendados e do custo computacional requerido no cálculo proposto?"*. Aqui apresentamos a proposta, implementação e validação de melhorias em algoritmos de ranqueamento existentes, de modo que a busca por fontes produtoras de dados em IoT possa levar em conta como parâmetros indicadores de qualidade sujeitos a variações ao longo do tempo. Foram levados em conta também requisitos não-funcionais como o custo computacional e o tempo de resposta. A proposta foi validada através da extensão de um catálogo de produtores de dados pré-existente *Waldo* [26]. Na implementação avaliamos também uma abordagem para economia de processamento e ganho de desempenho nas recomendações realizadas pelo mecanismo de busca. O restante deste artigo está organizado da seguinte forma: a seção 2 traz a fundamentação teórica; a seção 3 apresenta os trabalhos relacionados; a seção 4 traz a proposta; a seção 5 aborda a implementação; em seguida a avaliação e discussão de resultados na seção 6; e finalmente a conclusão na seção 7.

2. FUNDAMENTAÇÃO

2.1 Contexto computacional

Segundo Dey [6], contexto é: *[...] qualquer informação que possa ser usada para caracterizar a situação de entidades que são considerados relevantes para a interação entre o usuário e uma aplicação, incluindo o próprio usuário e a aplicação.*

Com o objetivo de facilitar o desenvolvimento de aplicações sensíveis a contexto, bem como a integração de diferentes tecnologias em IoT, diversas plataformas de mediação de dados são propostas. Entre suas possibilidades estão o gerenciamento de dispositivos, auxílio no armazenamento e na recuperação de dados gerados pelos dispositivos, processamento dos dados a fim de sumarizar estatísticas e disparar alertas entre outras características [32]. OpenIoT, GSN, Cosm, Fi-ware e Axeda são alguns exemplos.

Basicamente as plataformas de mediação de dados fazem uso de uma API para solicitar dispositivos específicos à plataforma. Esta, por sua vez, retira das aplicações, e implementa, as tarefas de representação, descoberta, busca, seleção dos dispositivos. No entanto, um problema evidente no âmbito de IoT é: dado que os dispositivos ofertados são autônomos e dinâmicos, *como selecionar os dispositivos que melhor atendem à necessidade da aplicação em determinado momento?*

2.2 QoC, QoS e QoD

Ao se trabalhar com contexto deve-se considerar o tratamento de incertezas, já que os elementos contextuais podem conter inconsistências, serem ambíguos ou incompletos [34, 22]. Por menor que seja, sempre haverá um erro inerente a cada elemento contextual capturado. Vários trabalhos propõem formas de considerar a qualidade do contexto. Da mesma forma, a qualidade do dado gerado pelos produtores de dados (sensores, bases de dados entre outros) e serviços também podem ter sua qualidade quantificada.

Três categorias de qualidade podem ser consideradas [2]: QoC (*Quality of Context*), QoS (*Quality of Service*) e QoD (*Quality of Device*). QoC é qualquer informação que descreve a qualidade daquilo que é usado como elemento contextual. QoS é qualquer informação que descreve quão bem um serviço é executado. QoD é qualquer informação sobre as propriedades técnicas do dispositivo e suas capacidades. Assim, os três tipos de qualidades podem intervir uns nos outros. De fato, "se informações de baixo nível tem um erro, as informações de alto nível naturalmente terão erro e mecanismos de raciocínio simplificados causarão ou propagarão erros" [17].

Assim, surgem métricas de qualidade como uma tentativa de qualificar, de forma objetiva, serviços, contextos, dados ou dispositivos. As principais métricas encontradas na literatura são: atualidade, probabilidade de corretude, confiabilidade, precisão, resolução, tempo de resposta e completude [2, 13, 35, 16, 15, 31, 9, 39, 17].

2.3 Algoritmos de ranqueamento

As consultas a produtores de dados devem ser atendidas conforme os diversos atributos escolhidos como elementos contextuais. É importante que a técnica de ranqueamento escolhida consiga lidar com a dinamicidade inerente aos produtores de dados, já que estes podem mudar seus níveis de QoS a todo momento. É importante que a técnica seja leve, do ponto de vista do custo computacional. Serão abordadas, a seguir, três técnicas apropriadas para este cenário, segundo a literatura disponível.

2.3.1 Pearson Correlation Coefficient

A métrica *Pearson Correlation* mede o grau de linearidade entre duas variáveis [3]. Como a similaridade entre dois itens pode ser dada pela correlação entre eles, a métrica *pearson correlation* vem sendo utilizada em diversos sistemas de recomendação, sobretudo aqueles baseados na estratégia de filtragem colaborativa, que pode ser: baseado em usuário ou baseado item [39]. Sendo $\sum(x, y)$ a covariância entre o conjunto de pontos x e y e σ é o desvio padrão [1], *Pearson Correlation* é dada pela fórmula $Pearson(x, y) = \frac{\sum(x,y)}{\sigma_x . \sigma_y}$.

O método baseado em usuário é utilizado para comparar a similaridade entre dois usuários tomando como base os serviços requisitados ou avaliados por ambos e pode ser calculada pela expressão [5, 39]:

$$Sim(a, u) = \frac{\sum_{i \in I}(w_{a,i} - \overline{w}_a)(w_{u,i} - \overline{w}_u)}{\sqrt{\sum_{i \in I}(w_{a,i} - \overline{w}_a)^2}\sqrt{\sum_{i \in I}(w_{u,i} - \overline{w}_u)^2}}$$

Sendo: $w_{a,i}$ é a avaliação dada pelo usuário a ao item i; $w_{u,i}$ é a avaliação dada pelo usuário u ao item i; $\overline{w}_a$ e $\overline{w}_u$ são as médias das avaliações dadas pelos usuários a e u.

Da mesma forma, o método baseado em item é utilizado para calcular a similaridade entre dois itens i e j e pode ser calculada pela expressão [5, 39]:

$$Sim(i, j) = \frac{\sum_{u \in U}(w_{u,i} - \overline{w}_u)(w_{u,j} - \overline{w}_j)}{\sqrt{\sum_{u \in U}(w_{u,i} - \overline{w}_i)^2}\sqrt{\sum_{u \in U}(w_{u,j} - \overline{w}_j)^2}}$$

2.3.2 Cosine Vector

O *Cosine Vector* é uma técnica bastante utilizada em cálculo de similaridade entre documentos. Sua lógica consiste em representar entidades em um espaço vetorial multidimensional. Considerando que uma entidade u tenha n atributos de qualidade, o espaço terá n dimensões e cada um destes atributos de qualidade será representado por um dos eixos coordenados. Ou seja, $u \in \mathbb{R}^n$, onde $u_i = w_i$ se a entidade já possui valor para o item i ou $u_i = 0$, caso contrário [5].

Após representar as entidades como vetores em um espaço multidimensional, a técnica utiliza a expressão a seguir para calcular o ângulo entre eles:

$$Sim(d_i, d_j) = \frac{\sum_k w_{ki} . w_{kj}}{\sqrt{\sum_k w_{ki}^2} . \sqrt{\sum_k w_{kj}^2}}.$$

Esta técnica considera que quanto menor o ângulo entre dois vetores, mais semelhantes eles são.

Apesar das técnicas *Pearson Correlation* e *Cosine Vector* serem bastante utilizadas em sistemas de recomendação [30, 1], sobretudo aqueles baseados em filtragem colaborativa, uma de suas principais implicações é a necessidade de avaliações explícitas dos consumidores sobre os resultados sugeridos. Como na IoT, não apenas humanos são consumidores, mas também serviços e até mesmo outros dispositivos, a exigência de avaliações explícitas dos resultados se torna inviável. Mesmo que os consumidores fossem apenas humanos, avaliar cada resultado não seria possível.

Pearson correlation e *cosine similarity* são consideradas ótimas escolhas para sistemas de recomendação, muito embora melhoramentos possam ser sugeridos para cada tipo de aplicação [1]. A técnica sugerida por este trabalho será melhor detalhada na Seção 4.1.

3. TRABALHOS RELACIONADOS

Três trabalhos principais mostraram afinidade com o tema atual, cada qual com suas peculiaridades, que serão mostradas nas seções a seguir.

O primeiro, RSDPP (*Real World Service Discovery and Provisioning Process*) [11], propõe um serviço de descoberta, busca e seleção de serviços do mundo real (sensores e atuadores). O processo inicia na etapa de *Types Query*, na qual o consumidor elabora sua consulta, definindo o serviço que ele busca, por meio de palavras chave. Serviços web são utilizados para expandir os termos informados pelo consumidor utilizando sinônimos. Então são descobertos quais serviços descrevem tal funcionalidade. Na etapa *Candidate Search* são localizados os dispositivos considerados candidatos a responderem de forma adequada à solicitação realizada. Além de considerar a estratégia de busca aumentada na etapa anterior, também é considerada informação de QoS destes dispositivos.

O segundo trabalho, proposto por Perera et al. [27], elabora uma estratégia de busca com atributos de qualidade. Foram considerados: disponibilidade, acurácia, tempo de resposta, frequência, precisão, resolução, bateria entre outros. Através de uma barra deslizante é possível distribuir diferentes pesos de importância aos atributos de qualidade. A busca é submetida e processada utilizando SPARQL, uma vez que os sensores são descritos através de ontologias. Por fim, estes candidatos são ranqueados após a aplicação da técnica de distância euclidiana ponderada através da representação dos sensores em um espaço multidimensional.

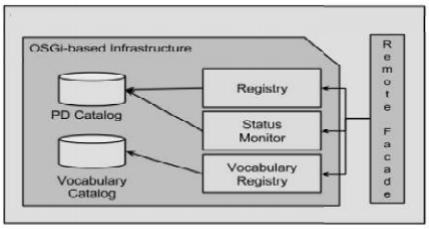
Já no COBASEN [21], a busca começa com a elaboração de uma query formada por palavras chave. Há uma tabela de índice invertido que ajuda a localizar os sensores que possuem um alto grau de similaridade com a *string* de busca submetida. Os resultados são listados e o consumidor pode selecionar as opções que mais se adequam à sua necessidade, considerando as características apresentadas.

Observa-se que das três propostas, apenas o trabalho de Perera et al. [27] possui uma estratégia de busca estruturada. Os outros dois [11, 21] aceitam como busca sequências de palavras chave que descrevem o fenômeno que se quer observar, detalhes de descrição dos dispositivos ou níveis de qualidade. Guinard et al. [11], como foi discutido, utiliza dicionários para expandir as palavras chave. No entanto, um problema desta abordagem é o retorno de falsos sinônimos, ou sinônimos que apontam para nenhum dispositivo. Por outro lado, observando apenas sob a perspectiva da técnica de ranqueamento utilizada, apenas o trabalho proposto por Perera et al. [27] utiliza algum cálculo explícito. [21] e [11] são altamente dependentes de similaridades entre palavras chave. Por fim, um aspecto negativo presente nos trabalhos de Lunardi et al. [21] e Perera et al. [27] é que eles exigem a interação explícita de um consumidor, humano, através de uma interface gráfica. Por outro lado, assim como [11], o presente trabalho pretende implementar uma interface REST, através da qual os dispositivos possam ser ofertados como recursos, acessados através de verbos HTML.

4. PROPOSTA

O Waldo [26] inicialmente trouxe uma proposta de catálogo de produtores de dados, cuja arquitetura é ilustrada na Figura 1. Seus componentes principais são: dois repositórios para armazenamento da descrição dos produtores de dados e vocabulários; e três módulos (Registro, Monitor e Registro de Vocabulário). Registro é o módulo responsável por adicionar, atualizar, remover e buscar produtores de dados. No Waldo as buscas são realizadas apenas considerando parâmetros como fenômeno de interesse, identificador entre outros atributos em seus metadados.

Figura 1: Arquitetura do Waldo



Fonte: Elaborado por [26]

Monitor é o módulo responsável por verificar se os produtores de dados registrados continuam ativos. O Módulo de Vocabulário se encarrega de administrar os termos utilizadas para a interoperabilidade na comunicação os componentes do próprio Waldo, ou com sistemas externos.

Apesar do trabalho de Oliveira et al. [26] ter demonstrado que sua proposta é viável para o contexto de IoT e cidades inteligentes, ainda carece de um mecanismo de busca mais refinado. Dessa forma, o presente trabalho lida com uma evolução do Waldo, cujos requisitos adicionais são discutidos a seguir.

4.1 Requisitos

Este trabalho propõe um mecanismo de ranqueamento que considere atributos de qualidade dos produtores de dados, endereçando algumas das lacunas encontradas na literatura. A implementação é construída como um mecanismo de busca que evolui o Waldo.

Ao propor alguma estratégia de busca deve-se, primeiramente, delinear a estratégia de seleção, ranqueamento e recomendação. Várias abordagens são propostas [24, 36, 19, 38, 18, 23, 39, 27, 37], inclusive algumas considerando atributos de qualidade de produtores de dados, no entanto, diversas delas são complexas demais e demandam muito processamento. Assim, deseja-se que o mecanismo de busca seja rápido e exija o mínimo, ou nenhuma interação explícita.

Os requisitos funcionais (RF) e não funcionais (RNF) do mecanismo de busca proposto serão discutidos a seguir:
RF001 – O mecanismo de busca deve considerar indicadores de qualidade como parâmetros. A estratégia adotada foi embasada na técnica de similaridade *Cosine Vector* [20]. No entanto, há dois pontos de melhoria desta técnica em nosso contexto de aplicação: *(i)* A técnica *cosine vector* não considera o tamanho dos vetores; *(ii)* A técnica dá a mesma prioridade para os atributos informados explicitamente na requisição e a aqueles monitorados pelo catálogo, mas não explicitados na estrategia de busca.

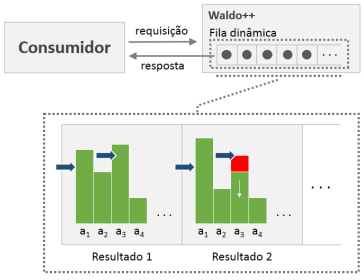
Assim, tendo como base a técnica *Cosine Vector* e assumindo as duas lacunas apresentadas acima, sugere-se a seguinte técnica de Cosine Vector melhorada:

$Score(a) = cos(I, a) * (\rho * \frac{\|a_e\|}{\|I_e\|} + (1 - \rho) * \frac{\|a_i\|}{\|I_i\|})$.

Sendo: I é o vetor ideal; a é o vetor que se deseja ranquear; $\|a_e\|$ é a norma do subvetor de a formado apenas pelos atributos de qualidade explícitos na requisição; $\|a_i\|$ é a norma do subvetor de a formado apenas pelos atributos de qualidade implícitos na requisição; $\|I_e\|$ é a norma do subvetor de I formado apenas pelos atributos de qualidade explícitos na requisição; $\|I_i\|$ é a norma do subvetor de I formado apenas pelos atributos de qualidade implícitos na requisição; ρ é um fator de ponderação.

[RF002] Estratégia de fila dinâmica. O mecanismo de busca pode implementar heurísticas para lidar com o trabalho do recálculo de ranqueamento a cada consulta submetida ao catálogo. Há duas formas de busca [21], no nosso caso, de produtores de dados: ou por consultas realizadas por iniciativa do consumidor, ou utilizando o padrão *publish/subscribe*. A estratégia implementada no Waldo, neste trabalho chamada de *Fila Dinâmica*, é um híbrido dessas duas opções.

Figura 2: Ilustração da Fila Dinâmica.



Fonte: Elaborado pelo Autor

Como ilustrado na Figura 2, uma vez que uma nova requisição é submetida, os resultados são entregues ao consumidor, uma fila é criada no catálogo e preenchida com os mesmos resultados. Quando o consumidor executa sua próxima requisição, são entregues os resultados presentes nesta fila. Se o resultado não satisfaz então é excluído da fila dinâmica. Se satisfaz, mesmo que os níveis de qualidade ou outras restrições tenham diminuído o resultado permanece na fila.

Na situação ilustrada pela Figura 2, as setas azuis representam os níveis de qualidade especificados na requisição do consumidor. O segundo resultado da fila deixou de atender o nível especificado para o atributo 3. Portanto, o resultado 2 é excluído da fila dinâmica e deixa de ser recomendado como resultado ao consumidor. Quando a fila deixar de atender satisfatoriamente a consulta inicial, a consulta é reenviada e a fila é novamente populada com novos resultados.

[RNF001] Desempenho. O mecanismo de busca deve lidar com a questão de performance, executando o cálculo de ranqueamento no menor tempo possível. Assim, é necessário que [RF001] tenha um tempo de resposta menor ou igual às técnicas já existentes *pearson correlation* e *cosine vector*.

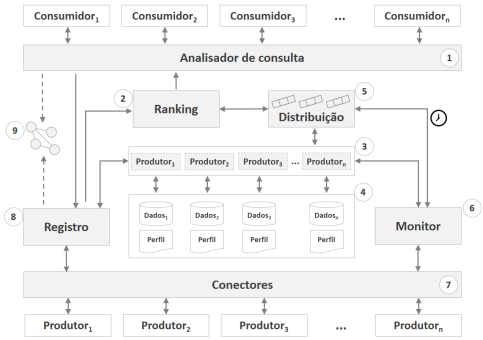
[RNF002] Armazenamento de dados flexível. O banco de dados utilizado deve ser capaz de armazenar e manipular dados com esquemas dinâmicos.

[RNF003] Esquema de dados flexível. Silva et al. [32] verificou que há um movimento que caminha em direção oposta à interoperabilidade, em que cada projeto propõe seu próprio modelo de dados, ou implementa um modelo de dados proprietário. Em 2014, a OGC [25] definiu o SensorML, adotado neste trabalho, como o padrão avançado para descrição de sensores, atuadores e processadores em internet das coisas. Vários trabalhos bem referenciados utilizaram o SensorML como padrão de modelo de dados [14, 4, 12], tornando-o uma opção cada vez mais consolidada para a descrição de objetos em IoT.

4.2 Arquitetura

A interação entre os componentes da arquitetura do Waldo se comporta como ilustrado na Figura 3. Os consumidores enviam suas requisições ao analisador de consulta (item 1 na Figura 3). Ele verifica se a requisição é válida, do ponto de vista do esquema de dados. Caso seja, ela é repassada ao módulo de registro (item 8). Sendo operação de busca, o módulo de registro busca os produtores de dados candidatos e repassa ao módulo de ranqueamento (item 2). Uma vez ranqueados, são inseridos nas filas dinâmicas e repassados ao consumidor. Sendo operações de registro, atualização e remoção o módulo de registro interage com a base de dados que armazena a descrição dos produtores de dados.

Figura 3: Arquitetura do Waldo++.



Fonte: Elaborado pelo Autor

O módulo monitor (item 6) envia requisições de tempos em tempos aos produtores e atualiza seus atributos de qualidade, se for o caso. Além disso, responde a requisições realizadas pelo módulo de distribuição (item 5). As requisições feitas pelo módulo de distribuição tem o objetivo de verificar se os resultados de ainda obedecem aos requisitos da requisição que sua respectiva fila dinâmica.

Como ilustrado no item 4 da Figura 3, os produtores de dados possuem uma representação "virtual", que é apresentada aos consumidores. Assim, quando se solicita o consumo de dados dos produtores, não se consome diretamente nos dispositivos, mas sim nos resultados armazenados localmente

no catálogo Waldo.

5. IMPLEMENTAÇÃO

O serviço REST de acesso ao Waldo foi implementado utilizando o *framework Jersey* 2.22.2. Através dele é possível enviar requisições de registro, atualização, busca e exclusão de produtores de dados através de requisições HTTP.

Para gerenciar o monitoramento permanente, realizado pelo módulo Monitor, foi utilizado o *RabbitMQ*. No Waldo, como visto anteriormente, de tempos em tempos o módulo Monitor verifica o status dos produtores de dados registrados com o objetivo de recalcular seus indicadores de qualidade. Para isso, um processo paralelo chamado `BasicCatalogMonitorScheduleQuality` agenda em uma fila do rabbitMQ chamada `TASK`, tarefas de verificação de status. Por outro lado, processos paralelos chamados `BasicCatalogMonitorQualityTask` ouvem permanentemente a fila `TASK` e uma vez que uma nova tarefa é agendada, algum processo `BasicCatalogMonitorQualityTask` retira o agendamento da fila e o processa, indo até o produtor de dados, verificando seu atual status e recalculando seus indicadores de qualidade.

Foi utilizada a especificação SensorML, descrita na Seção 4.1, como modelo de dados dos produtores de dados. Além disso, foi utilizada a especificação *Observations & Measurements* (O&M). Como as descrições dos produtores de dados são dados em formato de documentos JSON, o Waldo++ utilizou o MongoDB. O servidor de aplicação utilizado foi o *Glassfish* versão 4.1.1 e a linguagem de programação utilizada em toda a implementação foi o Java 1.8.0_91.

6. AVALIAÇÃO E ANÁLISE

Para executar todos os experimentos, foram utilizados os computadores descritos na Tabela 1. Os experimentos realizados serão detalhados nas seções a seguir.

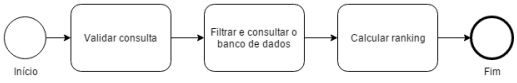
Tabela 1: Computadores utilizados

Nome	Configuração
Wi5.A	Intel Core i5 CPU @ 2.20GHz 6.00 RAM Win 8.1 64 bits
Wi7.A	Intel Core i7 CPU @ 3.40 GHz 8.00 RAM Win 8.1 64 bits
Wi7.B	Intel Core i7 CPU @ 3.40 GHz 8.00 RAM Win 8.1 64 bits

6.1 Técnica de ranqueamento

O primeiro aspecto a ser observado na técnica de ranqueamento sugerida é verificar se ela implica em uma perda de desempenho do mecanismo de busca do catálogo ao qual ela foi agregada. Assim, para este experimento foi mensurado o tempo gasto em diversas atividades de um processo de consulta (como ilustrado na Figura 4) com o objetivo de responder a seguinte pergunta: *a técnica de ranqueamento proposta é viável do ponto de vista computacional?*

Figura 4: Atividades de uma consulta.



Fonte: Elaborado pelo Autor

Foram parâmetros do experimento: quantidade de atributos explícitos na estratégia de busca e quantidade de produtores de dados candidatos retornados pela consulta. Quatro cenários foram simulados: 2, 4, 6 e 8 atributos de qualidade explícitos na estratégia de busca. Cada cenário variou a quantidade de produtores de dados candidatos (1000, 3000, 5000, 7000, 9000 e 10000). Os níveis dos atributos de qualidade obedeceram uma distribuição normal.

	#C	V	F	R	T	% T
Cenario 1	1000	0.1724	188.4828	58.7586	247.4138	23.7491
	3000	0.242	1011.212	141.303	1152.758	12.257
	5000	0.250	1499.964	236.857	1737.071	13.635
	7000	2.896	3159.241	338.344	3500.483	9.665
	9000	0.928	2656.857	474.892	3132.679	15.159
	10000	0.250	2615.429	579.642	3195.321	18.140
Cenario 2	1000	0.357	217.50	62.178	280.035	22.203
	3000	0.321	960.821	139.000	1100.143	12.634
	5000	0.434	1277.957	403.478	1681.870	23.989
	7000	0.259	3083.370	371.481	3455.111	10.751
	9000	0.310	2492.276	563.551	3056.138	18.440
	10000	0.172	2694.172	603.758	3298.103	18.306
Cenario 3	1000	0.344	232.655	78.793	311.7931	25.270
	3000	0.275	995.724	151.689	1147.690	13.216
	5000	0.413	1536.103	425.724	1962.241	21.695
	7000	0.379	3323.897	369.655	3693.931	10.007
	9000	0.241	2674.621	562.172	3237.034	17.366
	10000	0.333	2916.778	588.370	3505.481	16.784
Cenario 4	1000	0.344	234.103	71.413	305.861	23.348
	3000	0.448	1089.897	149.413	1239.759	12.058
	5000	0.428	1520.464	439.178	1960.071	22.406
	7000	2.833	3395.600	457.866	3856.300	11.873
	9000	0.300	2742.700	576.333	3319.333	17.362
	10000	0.533	3372.633	697.033	4070.200	17.125

Tabela 2: Tempos de resposta das etapas de busca em milissegundos. Sendo: $\#C$ o número de candidatos, V a fase de validação, F a fase de filtragem, R a fase de ranking, T o tempo total e % a porcentagem do tempo total.

Muito embora o cenário da IoT considere que milhões, ou até bilhões, de dispositivos estejam conectados à rede, deve-se considerar que ao submeter uma requisição de busca ao catálogo, o consumidor está interessado em resolver apenas um problema em específico. Ou seja, aqueles que já satisfazem todas as condições iniciais e participarão da recomendação.

O ambiente de simulação utilizou as duas máquinas descritas em na Tabela 1, sendo o catálogo Waldo++, com a técnica de ranqueamento implementada, executado em Wi5.A, recebendo requisições de Wi7.A. Além disso, cada cenário foi executado 32 vezes, mantendo as mesmas métricas, parâmetros e níveis. Dessas 32 medidas, a maior e a menor foram excluídas e as outras 30 tiveram a média calculada. A Tabela 2 ilustra a média do tempo de ranqueamento em todos os cenários simulados.

A Tabela 2 compara o tempo de ranqueamento com os tempos de cada etapa da busca. Para cada cenário é destacado o tempo das etapas de validação, filtragem e ranqueamento.

6.1.1 Análise da técnica de ranqueamento

Como visto na Tabela 2, o crescimento do tempo de ranqueamento se dá de forma praticamente linear. Para se ter uma visão objetiva, foi discriminado nesta tabela o tempo gasto em cada etapa do processo de busca. Percebe-se que o impacto que a técnica de ranqueamento incide sobre o processo de busca não aumenta à medida que cresce o número de produtores de dados candidatos. De fato, para o cenário 1 a técnica representou em média 15,43416% do tempo total de busca e nos cenários 2, 3 e 4 foram respectivamente de 17,7205%, 16,8896% e 17,362%.

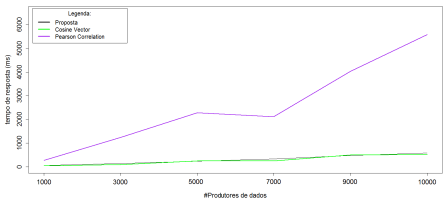
6.1.2 Comparativo das técnicas

Uma segunda verificação a ser feita é uma comparação do tempo de resposta da técnica proposta frente às técnicas Pearson Correlation [5, 39] e Cosine Vector [5].

Assim, para esta avaliação o tempo de resposta foi considerado como métrica. Os parâmetros do experimento foram a quantidade de atributos explícitos na estratégia de busca e

a quantidade de produtores de dados candidatos. A quantidade de produtores candidatos variou em 1000, 3000, 5000, 7000, 9000 e 10000 e o número de atributos de qualidade explícitos foi fixado em 2. Por fim, o ambiente de simulação implementou o catálogo na máquina Wi5.A, recebendo requisições da máquina Wi7.A (Tabela 1). O cenário foi executado 32 vezes, mantendo as métricas, os parâmetros e seus respectivos níveis. Após desconsiderar o maior e o menor valor, a média foi calculada com as 30 medidas restantes. Os dados coletados estão ilustrados a seguir.

Figura 5: Tempo de resposta das técnicas consideradas.



Fonte: Elaborado pelo Autor

Como ilustrado na Figura 5, bem como na Tabela 2, o tempo médio da técnica proposta para calcular o ranqueamento de 1000, 3000, 5000, 7000 e 10000 produtores de dados candidatos foi, respectivamente, 58.75862ms, 141.303ms, 236.8571ms, 338.3448ms, 474.8929ms e 579.6429ms. Para a técnica *cosine vector*, foi de, respectivamente, 48.066ms, 100.5ms, 253.366ms, 254.785ms, 505.555ms e 530.392ms, respectivamente. Já a técnica *pearson correlation* realizou as mesmas tarefas em 310.6ms, 1271.31ms, 2486.667ms, 2227.345ms, 4132.483ms e 5661.286ms.

A Figura 6 ilustra o nível de cada atributo de qualidade em cada uma das técnicas consideradas por este trabalho. Para cada técnica de ranqueamento foram executados 30 consultas. Em cada uma delas foram separados os dez melhores resultados e a média de seus atributos de qualidade foi calculada. Por fim, foi calculada a média dessas 30 médias anteriores (ver Figura 6). Deste gráfico pode-se extrair as diferenças de níveis dada por cada técnica de ranqueamento para cada um dos atributos.

6.1.3 *Análise da comparação das técnicas*

Sobre o desempenho quantitativo das técnicas, considerando o tempo de resposta como métrica, podemos observar que a técnica de recomendação proposta se mostra bastante semelhante à técnica *cosine vector*, enquanto que estas duas são dramaticamente mais eficientes que a técnica *pearson correlation*. À medida que a quantidade de produtores de dados a serem ranqueados aumenta, o tempo de resposta da técnica de ranqueamento proposta e do *cosine vector* cresce praticamente linear e com baixa inclinação. Por outro lado, o tempo de reposta da técnica *pearson correlation* cresce não linearmente, se acentuando à medida que o número de produtores de dados a serem ranqueados aumenta.

Apesar do tempo de resposta da técnica de ranqueamento proposta ser bastante semelhante ao *cosine vector*, a Figura 6 evidencia diferenças entre essas duas técnicas sob uma perspectiva qualitativa. Nela, percebe-se que a técnica proposta supera o *cosine vector* em todos os atributos de qualidade, exceto apenas em a_5 e a_8, onde a diferença foi de apenas 2.3% e 1.58%. Por outro lado, além de a técnica recomendada ser melhor que o *cosine vector* em todos os outros atributos de qualidade, a diferença entre elas chegou a níveis notáveis como 66.57% para a_{12} e 52.78% para a_{10}.

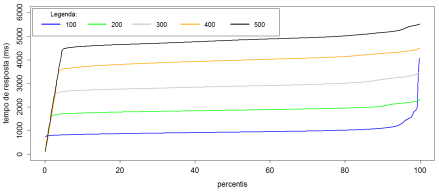
Comparando com *pearson correlation* a técnica recomendada também se mostrou melhor em todos os atributos, exceto em apenas dois (a_5 e a_8). Nos outros atributos, a técnica proposta superou o *pearson correlation* em níveis consideráveis, como 44.91% para a_7 e 36.56% para a_{12}.

6.2 Fila dinâmica

Foi modelado um cenário em que ciclistas pedalam por uma cidade enquanto que fornecem dados de sua rota ao catálogo, se comportando então como produtores de dados. Por outro lado, consumidores pesquisam quais rotas possuem ciclistas naquele exato momento. As rotas utilizadas neste experimento são reais, percorridas na cidade de Recife, extraídas da plataforma Strava[1].

O tempo de resposta foi considerado como métrica e a quantidade de consumidores simultâneos do catálogo e a quantidade de ciclistas simultâneos foram considerados como parâmetros. O ambiente de simulação utilizou três máquinas descritas na Tabela 1, sendo o catálogo Waldo executado em Wi5.A, com monitor da fila dinâmica executado em Wi7.B. As requisições foram submetidas ao catálogo por Wi7.A.

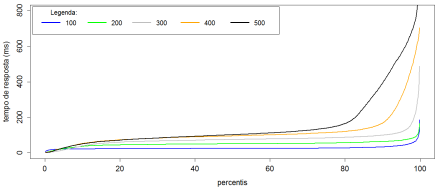
Figura 7: Mecanismo de busca sem a estratégia de fila dinâmica.



Fonte: Elaborado pelo Autor

A quantidade de consumidores requisitando produtores de dados ao catálogo variou em 100, 200, 300, 400 e 500, com requisições a cada 5 segundos. A quantidade de ciclistas simultâneos foi fixada em 100, gerando dados a cada 3 segundos. A Figura 7 ilustra o tempo de resposta do catálogo para buscas por ciclistas realizadas por diferentes quantidades de consumidores simultâneos. A Figura 8, por sua vez, ilustra o tempo de resposta das requisições de busca ao catálogo com a estratégia da fila dinâmica implementada.

Figura 8: Mecanismo de busca com a estratégia de fila dinâmica.



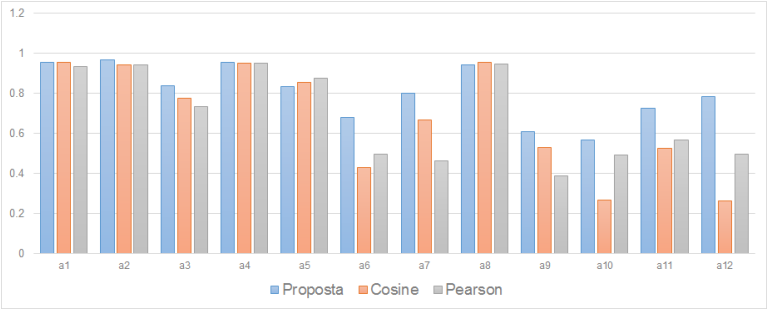
Fonte: Elaborado pelo Autor

6.2.1 *Análise da fila dinâmica*

Como evidenciado nas Figuras 7 e 8, a estratégia da fila dinâmica diminui consideravelmente o tempo de resposta de

[1]http://www.strava.com

Figura 6: Comparativo do nível de qualidade dos atributos.



Fonte: Elaborado pelo Autor.
Sendo: a_1, ..., a_{12} os atributos de qualidade considerados pelo trabalho: acurácia, disponibilidade, memória, largura de banda, bateria, custo, frequência, confiança, resolução, tempo de resposta, vazão e atualidade.

uma requisição de busca ao catálogo. Sem implementar a estratégia de fila dinâmica, o catálogo respondeu às requisições de 100, 200, 300, 400 e 500 consumidores simultâneos em uma média, respectivamente, de 987ms, 1860ms, 2869ms, 3909ms e 4739ms. Já com a estratégia de fila dinâmica implementada o tempos médios foram de, respectivamente, 26ms, 49ms, 74ms, 114ms e 163ms.

A grande vantagem da técnica é, além de diminuir o tempo de espera do consumidor, liberar processamento do catálogo para atender a novas requisições. Aumentando assim, a vazão de requisições processadas por unidade de tempo.

7. CONCLUSÃO

Este trabalho propôs uma técnica de ranqueamento que considera a característica dinâmica da qualidade dos produtores de dados em internet das coisas, sendo ela viável do ponto de vista computacional e facilmente acoplável em catálogos de produtores de dados. Além disso, foi proposta e analisada a estratégia de fila dinâmica, que tem por objetivo reduzir o custo do recalculo de ranqueamento a cada solicitação de busca enviada ao catálogo.

A técnica proposta por este trabalho não aumentou significativamente o tempo total de busca, representando em média apenas 16,85155% deste. Apesar do tempo de execução da técnica proposta ser bastante similar ao *cosine vector*, a técnica proposta chegou a superá-la em até 62.57%, em níveis de qualidade. Em relação ao *Pearson Correlation*, o tempo de execução foi consideravelmente menor, além de superá-la em até 44.91%, em níveis de qualidade.

Para uma continuidade deste trabalho, propõe-se: Criar alguma mecanismo de *intersecção de requisições*. Eventualmente algum consumidor pode submeter uma requisição de busca cujos requisitos sejam um subconjunto de uma requisição já submetida e que tem uma fila de resultados ativa. Além disso, entender o comportamento do MongoDB frente a outros bancos de dados NoSql baseados em memória.

8. AGRADECIMENTOS

Os autores agradecem ao suporte dado pelo CNPq, provido pelo projeto 485420/2013, à FACEPE, sob processo IBPG-0773-1.03/13 e ao INES, sob processos CNPq/465614/2014-0 e FACEPE/APQ/0388-1.03/14.

9. REFERÊNCIAS

[1] X. Amatriain, A. Jaimes, N. Oliver, and J. M. Pujol. Data mining methods for recommender systems. In *Recommender Systems Handbook*, pages 39–71. Springer, 2011.

[2] T. Buchholz, A. Küpper, and M. Schiffers. Quality of Context: What It Is And Why We Need It. *Proceedings of the workshop of the HP OpenView University Association*, pages 1–14, 2003.

[3] A. Chen. Context-aware collaborative filtering system: predicting the user's preferences in ubiquitous computing. *Proceedings of ACM CHI 2005 Conference on Human Factors in Computing Systems*, 2:1110–1111, 2005.

[4] M. Compton, P. Barnaghi, L. Bermudez, R. GarcíA-Castro, O. Corcho, S. Cox, J. Graybeal, M. Hauswirth, C. Henson, A. Herzog, et al. The ssn ontology of the w3c semantic sensor network incubator group. *Web Semantics: Science, Services and Agents on the World Wide Web*, 17:25–32, 2012.

[5] C. Desrosiers and G. Karypis. A comprehensive survey of neighborhood-based recommendation methods. In *Recommender systems handbook*, pages 107–144. Springer, 2011.

[6] A. K. Dey, G. D. Abowd, and D. Salber. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-computer interaction*, 16(2):97–166, 2001.

[7] Y. Evchina, J. Puttonen, A. Dvoryanchikova, and J. L. M. Lastra. Context-aware knowledge-based middleware for selective information delivery in data-intensive monitoring systems. *Engineering Applications of Artificial Intelligence*, 43:111–126, 2015.

[8] R. K. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[9] P. Gray and D. Salber. Modelling and using sensed context information in the design of interactive applications. *Engineering for Human-Computer Interaction*, (1):317–335, 2001.

[10] P. Guillemin and P. Friess. Internet of things strategic research roadmap, 2009. [Online. Acessado em 25 jul 2016].

[11] D. Guinard, V. Trifa, S. Karnouskos, P. Spiess, and D. Savio. Interacting with the SOA-based internet of things: Discovery, query, selection, and on-demand

provisioning of web services. *IEEE Transactions on Services Computing*, 3(3):223–235, 2010.

[12] C. A. Henson, J. K. Pschorr, A. P. Sheth, and K. Thirunarayan. Semsos: Semantic sensor observation service. In *Collaborative Technologies and Systems, 2009. CTS'09. International Symposium on*, pages 44–53. IEEE, 2009.

[13] N. Honle, U.-P. Kappeler, D. Nicklas, T. Schwarz, and M. Grossmann. Benefits of integrating meta data into a context model. In *Pervasive Computing and Communications Workshops, 2005. PerCom 2005 Workshops. Third IEEE International Conference on*, pages 25–29. IEEE, 2005.

[14] S. Jirka, A. Bröring, and C. Stasch. Discovery mechanisms for the sensor web. *Sensors*, 9(4):2661–2681, 2009.

[15] G. Judd and P. Steenkiste. Providing Contextual Information to Pervasive Computing Applications. *International Conference on Pervasive Computing and Communications*, page 133, 2003.

[16] B. K. Kahn, D. M. Strong, and R. Y. Wang. Information Quality Benchmarks: Product and Service Performance. *Commun. ACM*, 45(4):184–192, 2002.

[17] Y. Kim and K. Lee. A Quality Measurement Method of Context Information in Ubiquitous Environments. *2006 International Conference on Hybrid Information Technology*, 2:576–581, 2006.

[18] J. Lei, X. Yang, Y. Liu, Y. Qin, H. Tang, and Z. Zhao. Using physical–level context awareness to improve service ranking in wireless sensor network. *Journal of Networks*, 7(6):926–934, 2012.

[19] Z. Liu and X. Xu. S-abc-a service-oriented artificial bee colony algorithm for global optimal services selection in concurrent requests environment. In *Web Services (ICWS), 2014 IEEE International Conference on*, pages 503–509. IEEE, 2014.

[20] P. Lops, M. De Gemmis, and G. Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[21] W. T. Lunardi, E. De Matos, R. Tiburski, L. A. Amaral, S. Marczak, and F. Hessel. Context-based search engine for industrial IoT: Discovery, search, selection, and usage of devices. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2015-October(July 2016), 2015.

[22] J. McCarthy and S. Buvac. Formalizing context. 1997.

[23] W. Niu, J. Lei, E. Tong, G. Li, L. Chang, Z. Shi, and S. Ci. Context-aware service ranking in wireless sensor networks. *Journal of network and systems management*, 22(1):50–74, 2014.

[24] N. H. W. NWE, J.-m. BAO, and C. Gang. Flexible user-centric service selection algorithm for internet of things services. *The Journal of China Universities of Posts and Telecommunications*, 21:64–70, 2014.

[25] OGC. Open geospatial consortium, 2016. [Online. Acessado em 25 jul 2016].

[26] M. I. S. Oliveira, K. S. da Gama, and B. F. Lóscio. Waldo: Serviço para publicação e descoberta de produtores de dados para middleware de cidades inteligentes. *XI Simpósio Brasileiro de Sistemas de Informação*, 2015.

[27] C. Perera, A. Zaslavsky, P. Christen, M. Compton, C. Liu, and D. Georgakopoulos. Sensor search techniques for sensing as a service architecture for the internet of things. *IEEE 14th International Conference on Mobili Data Management*, 2013.

[28] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.

[29] A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3:62–70, 2004.

[30] S. Sawant. Collaborative Filtering using Weighted BiPartite Graph Projection - A Recommendation System for Yelp. 2013.

[31] K. Sheikh, M. Wegdam, and M. van Sinderen. Quality-of-context and its use for protecting privacy in context aware systems. *Journal of Software*, 3(3):83–93, 2008.

[32] E. C. G. F. Silva, M. I. S. Oliveira, E. Oliveira, K. Santos, and B. F. L. Um survey sobre plataformas de mediação de dados para internet das coisas. *SEMISH – Seminário Integrado de Software e Hardware*, 2015.

[33] B. Soediono. Managing context data for smart spaces. *Journal of Chemical Information and Modeling*, 53(October):160, 1989.

[34] V. Vieira, P. Tedesco, and A. C. Salgado. Modelos e processos para o desenvolvimento de sistemas sensíveis ao contexto. *André Ponce de Leon F. de Carvalho, Tomasz Kowaltowski.(Org.). Jornadas de Atualização em Informática*, pages 381–431, 2009.

[35] C. Xu and S.-C. Cheung. Inconsistency detection and resolution for context-aware middleware support. *ACM SIGSOFT Software Engineering Notes*, 30(5):336–345, 2005.

[36] Z. Yang and D. Li. Iot information service composition driven by user requirement. In *Computational Science and Engineering (CSE), 2014 IEEE 17th International Conference on*, pages 1509–1513. IEEE, 2014.

[37] K. K. F. Yuen and W. Wang. Towards a ranking approach for sensor services using primitive cognitive network process. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*, pages 344–348. IEEE, 2014.

[38] S. Zhao, Y. Zhang, B. Cheng, and J. Chen. A feedback–corrected collaborative filtering for personalized real–world service recommendation. 9(3):356–369, 2014.

[39] Z. Zheng, S. S. Member, H. Ma, M. R. Lyu, I. King, and S. S. Member. QoS-Aware Web Service Recommendation by Collaborative Filtering. *Ieee Transactions on Services Computing*, 4(2):140–152, 2011.