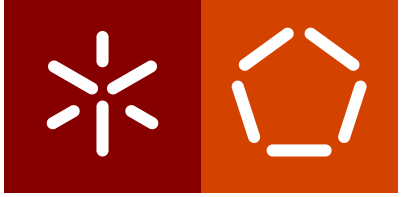
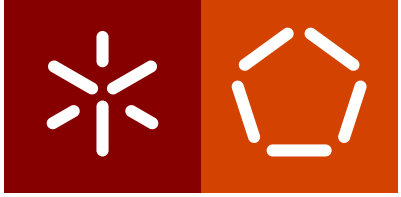


Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ricardo Jorge Barroso Certo

Utilização de Algoritmos de *Sketching* na Análise de *Streams* de Dados – Um Caso de Estudo

February 2021



Universidade do Minho
Escola de Engenharia
Departamento de Informática

Ricardo Jorge Barroso Certo

Utilização de Algoritmos de *Sketching* na Análise de *Streams* de Dados – Um Caso de Estudo

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Professor Doutor Orlando Belo

February 2021

DIREITOS DE AUTOR E CONDIÇÕES DE UTILIZAÇÃO DO TRABALHO POR TERCEIROS

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

LICENÇA CONCEDIDA AOS UTILIZADORES DESTA DISSERTAÇÃO:

Atribuição
CC BY
https://creativecommons.org/licenses/by/4.0/

AGRADECIMENTOS

Como em qualquer etapa, esta não foi realizada sozinha. Por isso, aproveito esta oportunidade para agradecer às pessoas que me marcaram, motivaram ou me ajudaram direta ou indiretamente na concretização desta dissertação.

Em primeiro lugar, gostaria de agradecer à minha família especialmente aos meus pais pela educação e por tornarem possível o meu ingresso na universidade.

Aos meus amigos e namorada, por estarem sempre presentes quer nos bons e nos maus momentos e por me proporcionarem as mais diversas aventuras.

Ao CeSIUM, que me proporcionou momentos incríveis enquanto membro da direção. Permitiu que evoluísse muito enquanto pessoa, e que tivesse o primeiro contacto com o mundo pós-universitário.

A todos os funcionários do Departamento de Informática, um obrigado.

A todos os docentes da UM com quem tive a oportunidade de privar e que de uma forma ou de outra contribuíram para o meu sucesso nesta casa.

Reservo um agradecimento especial, para o professor Orlando Belo por me ter orientado ao longo da minha dissertação, por estar sempre disponível a ajudar os seus alunos. Os seus conselhos ao longo da realização da dissertação foram muito importantes e contribuíram para o cumprimento de todos os objetivos traçados numa primeira fase da mesma.

A todas as pessoas que de certo modo contribuíram para a minha evolução, o meu muito obrigado.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Utilização de Algoritmos de *Sketching* na Análise de Streams de Dados – Um Caso de Estudo

Hoje é vulgar encontrarmos inúmeras aplicações que envolvem nos seus serviços tarefas de processamento e análise de *streams* de dados. Estas tarefas não são de fácil implementação nem de execução, pelo menos em situações nas quais pretendemos fazer a análise do conteúdo de uma *stream* no tempo próximo do tempo real. Um dos principais problemas que enfrentamos na realização dessas tarefas é saber como identificar um ou mais elementos de dados que possam estar a circular na *stream* em tempo útil. A rapidez com que os dados são transportados por uma *stream* é enorme o que complica imenso o seu processo de análise, dificultando, obviamente, a identificação dos elementos de dados que se pretendem analisar. Uma das técnicas que podem ser utilizadas para implementar a análise de streams de dados é o *sketching*, que nos fornece mecanismos muito interessantes para descobrir padrões de dados de forma bastante expedita. Isso é possível, porque o *sketching* permite armazenar pequenos elementos de informação sobre os dados que circula numa *stream*, usando para isso uma combinação de técnicas de *hashing*, de contagem e de filtragem, mantendo um bom equilíbrio entre o uso de memória e a precisão da identificação dos elementos de dados em análise. Nesta dissertação estudámos e analisámos um conjunto de algoritmos de *sketching*, com o objetivo de verificar a forma como cada um deles atua no processamento e análise de *streams* de dados em tempo muito próximo do real. Para isso, desenvolvemos um sistema específico capaz de fazer a identificação de eventuais pontos de estrangulamento de uma rede de distribuição de água, aplicando cada um dos diferentes algoritmos que estudámos. O sistema desenvolvido permitiu correlacionar os diferentes parâmetros que podem influenciar o funcionamento de uma rede de distribuição de água, que utilizámos como caso de estudo, e, consequentemente, avaliar o desempenho de cada um dos algoritmos aplicados.

PALAVRAS-CHAVE Análise de *Streams* de Dados, Algoritmos de *Sketching*, Processamento de Dados, Sketching de Dados.

ABSTRACT

Using Sketching Algorithms in Data Streams Analysis - A Case Study

Nowadays it is usual to find several applications that engage, in its services, processing tasks and data stream analysis. In this context, given certain situations, for example, when the purpose is to analyse the content of a stream in time near the real time, these tasks are neither easy to implement nor to execute. One of the biggest problems faced during the implementation of these tasks is to understand how to identify one or more data elements that may be circulating in the stream in useful time. Moreover, the quickness of the data transportation throughout a stream is huge and this stunts its analysis process, making the identification of the data elements intended to be analysed more difficult. Therefore, one of the techniques that can be applied to implement the data stream analysis is the sketching that provides interesting mechanisms used to find data patterns in a very spontaneous way. Obviously, this is possible hence the sketching allows to store small elements of information about the data that circulates in a stream through a combination techniques of hashing, counting and filtration, while keeping a good balance between the memory use and the precision about the identification of the elements in the data in analysis. Throughout this dissertation it was studied and analysed a range of sketching algorithms with the aim to verify the way as each of them proceed in the processing and analysis of data stream in a time very near the real time. To do that, it was developed a specified system capable to do the identification of possible strangulation points in a water distribution grid applying each one of the algorithms that were studied. The developed system allowed to correlate the different parameters that can disturb the normal function of a water distribution grid, the case used as the study case, and, consequently, evaluate the performance of each one of the applied logarithms.

KEYWORDS Data Processing, Data Sketching, Data Streams Analysis, Sketching Algorithms.

CONTEÚDO

LISTA DE FIGURAS

LISTA DE TABELAS

1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Nos dias de hoje, o número de serviços nos quais é necessário processar grandes quantidades de informação, quase em tempo real, tem vindo a aumentar exponencialmente. De uma forma geral, podemos caracterizar um sistema de processamento de dados quase em tempo real, como sendo um sistema com grande disponibilidade para manipular *streams* de dados, com baixa latência de intervalos entre transações (exemplo de algumas aplicações de *Internet of Things*) e com um elevado volume de dados, que por vezes pode, obviamente, originar sobrecargas de trabalho a um qualquer sistema.

Processos como o descrito podem ser realizados por sistemas específicos de extração, transformação e carregamento de dados. As sequências destas três operações de processamento de dados definem as três principais fases de trabalho de um sistema de *Extract Transform Load* (ETL) (Sabtu and Ismail, 2017). Na primeira fase, estes sistemas fazem a coleta de dados, de forma escalonada, usualmente a partir de diversos locais. Depois, na fase seguinte, procedem às transformações de dados necessárias, de acordo com os modelos de análise e de processamento que foram estabelecidos. Por fim, na terceira fase, inserem os dados que foram coletados e preparados nas fases anteriores num dado repositório. Estes sistemas podem envolver inúmeras ações de trabalho sobre dados, desempenhando um papel de relevo em organizações nas quais é necessário estabelecer processos de migração de dados especializados para povoarem os seus sistemas operacionais e de suporte à decisão. Porém, o trabalho destes sistemas complica-se quando estes têm que lidar com fontes de informação que fornecem dados de uma forma contínua (*stream*) e em tempo real.

As *streams* de dados caracterizam-se pelo seu elevado volume, originando por vezes dificuldade no processamento das mesmas. Os dados são obtidos em tempo real e devem suportar uma consulta aos mesmos em qualquer instante, com um tempo de resposta na ordem dos milissegundos. A nível aplicacional estas são normalmente usadas na monotorização sensorial. Geralmente, o processamento de *streams* de dados em tempo real alia-se ao termo *Big Data*, que pode ser caracterizado como sendo um conjunto de dados (*dataset*) grande que não pode ser processado pelos *Sistemas de Gestão de Base de Dados* (SGBD) tradicionais, devido ao seu elevado volume.

O *Big Data* (Galdino, 2016) é um conceito recente, que tem gerado bastante interesse e discussão em tudo aquilo que diz respeito às suas possíveis aplicações e potenciais benefícios. Ao abordarmos a questão do *Big Data* pressupomos que estamos perante cenários aplicacionais que dada a sua natureza ou operacionalidade envolvem o tratamento de volumes de dados de grande dimensão, vulgarmente provenientes de variadas fontes e que exigem grandes capacidades de processamento.

O desenvolvimento de sistemas que tem por base o conceito de *Big Data*, tornou-se numa prática bastante frequente por parte das empresas, uma vez que a informação que está presente nos dados é extremamente valiosa para suporte a processos de análise e tomada de decisão. As características dos dados gerados neste âmbito são cada vez mais diversas e desafiantes, o que origina uma dificuldade crescente no processamento destes por parte das tecnologias comuns. Pela necessidade e oportunidade identificadas, começou um grande desenvolvimento de um conjunto de tecnologias com a capacidade de darem resposta a estes desafios de forma eficiente. No seu livro sobre *Big Data*, Cezar Taurian (Taurion, 2013a) faz a seguinte analogia, as ferramentas de *Big Data* representam para as empresas e para a sociedade o que o microscópio representou para a medicina. Desta forma, através de uma ferramenta de análise genérica podemos extrair informações, prevenir acidentes e melhorar linhas de produção. No entanto, temos de ter em conta o princípio de que não existe melhor ferramenta, mas sim a melhor que se adequa para o caso em estudo. Dado que estamos num processo de crescimento do conceito de *Big Data* não sabemos precisar com exatidão o impacto que este poderá causar na tecnologia. E com todo este volume de dados surge a necessidade de conseguirmos processar e armazená-los em tempo real, da maneira mais eficiente possível.

Uma vertente aplicacional do *Big Data*, com grande emergência em vários contextos aplicacionais, está relacionada com o processamento de *streams* de dados (Galdino, 2016). As *streams* de dados caracterizam-se pela contínua produção de dados em tempo real e por possuírem um elevado volume que dificulta o seu processamento por parte dos sistemas tradicionais. Estas são utilizadas em diferentes tipos de aplicações, como por exemplo, na monotorização de dados em sensores, no tráfego de operações na *internet* e na distribuição de dados em mercados financeiros.

Um dos maiores desafios que as *streams* de dados nos colocam, em particular quando pretendemos analisar os dados que transportam, é saber como identificar um ou mais elementos de dados que possam estar a circular na *stream* em tempo útil. A rapidez com que os dados transportados por uma *stream* nos chegam às mãos é enorme. Isso complica imenso o processo da sua análise, dificultando bastante a identificação dos elementos de dados que, de facto, queremos obter e analisar.

Para conseguirmos identificar elementos específicos numa *stream* de dados aquando o seu processamento precisamos de estabelecer uma relação entre os dados. Normalmente, usamos essas relações quando pretendemos descobrir padrões, aumentar a eficiência de uma linha de produção, prevenir doenças, identificar o caminho com menor transito, entre outros dependendo do caso em estudo. O principal desafio passa por

arranjar forma de filtrar essa elevada quantidade de dados de modo a que estes consigam ser analisados pelos sistemas presentes nas empresas. As vantagens que estes tipos de análises trazem fazem-nos afirmar que a não análise dos mesmos remete-nos para um desperdício de recursos o que é algo insustentável neste momento. Perante a possibilidade de se obterem estes benefícios as empresas estão a começar a utilizar tecnologias que lhes permitam ter a capacidade de processar e analisar o elevado número de dados por eles produzidos. Um exemplo disso é o facto de um médico poder monitorizar dados dos seus pacientes para prever possíveis doenças com maior exatidão.

Com o intuito de processar este elevado fluxo de dados de forma eficiente, o uso de algoritmos de *Sketching* tornou-se numa solução muito importante para este problema. Estes algoritmos caracterizam-se pela sua capacidade de armazenar informações reduzidas sobre os dados, usando para isso uma combinação de técnicas de *hashing*, de contagem, de filtragem e por conseguirem manter um bom equilíbrio entre o uso de memória e precisão.

As abordagens baseadas em algoritmos de *Sketching* tem vindo a desempenhar um papel cada vez mais importante, devido à produção contínua de dados que está a acontecer no panorama mundial. Assim sendo, em conjunto com ferramentas de análise podemos alcançar uma redução de custos numa empresa, determinar falhas, entre outros, tudo isto sempre num tempo muito próximo do real.

1.2 MOTIVAÇÃO E OBJETIVOS

Essencialmente, a motivação para a realização deste trabalho de dissertação foi provocada por um desafio bastante simples, mas concreto: estudar e conceber um processo, um sistema, que possibilitasse a análise de *streams* de dados em tempo real ou próximo de tempo real, utilizando técnicas de *sketching*. Basicamente, precisamos de encontrar soluções para a análise dos diferentes tipos de dados que uma *stream* possa transportar, aplicando diferentes algoritmos de *sketching* de modo a verificarmos qual o melhor caso a que cada um se pode aplicar.

Como tal, focámos a nossa atenção no estudo e na análise dos principais algoritmos de *sketching* com o objetivo de simplificar o processo de tratamento e de análise de *streams* de dados em tempo muito próximo do real. Para isso, primeiramente precisámos de realizar um estudo detalhado, abordando os aspetos mais essenciais desse tipo de algoritmos, de forma a compreendermos o seu funcionamento e avaliarmos a sua aplicação em sistemas que tenham que lidar com *streams* de dados. Com base no conhecimento adquirido, desenvolvemos um sistema especificamente orientado para a identificação dos pontos de estrangulamento de uma rede de distribuição de água, aplicando os diferentes algoritmos estudados. O sistema idealizado teve como base um conjunto de dados de grande dimensão, multifacetados, que nos permitiu correlacionar os diferentes parâmetros

que podem influenciar o funcionamento da rede de distribuição de água que utilizámos como plataforma de teste para a aplicação dos diversos algoritmos de *sketching*.

O sistema implementado tem a capacidade de analisar o fluxo de dados que é captado por cada sensor usando para isso os algoritmos adotados. Estes vão ser responsáveis por efetuarem a análise da *stream* de dados produzindo assim um resultado final que irá ser preponderante no processo de tomada de decisão. Com isto conseguimos tornar o sistema eficiente a nível do desempenho através da aplicação de técnicas de *sketching* de modo a identificarmos quais podem ser os pontos de estrangulamento do mesmo, permitindo-nos assim tomar medidas atempadamente para evitar situações anómalas no sistema.

A aplicação de técnicas de *sketching* a este caso concreto de aplicação permitiu:

- Simplificar o processo de análise de dados. Este é o grande objetivo a atingir com a elaboração desta dissertação que é a simplificação do processo de análise de dados que usualmente um sistema ETL realiza quase em tempo real. A melhoria na eficiência deste processo é algo muito importante no seio empresarial pois caminhamos para um futuro em que a produção de dados tem vindo a aumentar exponencialmente e a análise dos mesmos beneficia essa entidade. Na atualidade, este processo é bastante dispendioso e complexo por isso, é que a simplificação do mesmo é bastante importante.

- Prevenir ou antecipar determinados comportamentos. Este objetivo consiste em obter o benefício do processamento de *streams* de dados e respetiva análise dos resultados em tempo real, de forma a conseguirmos prevenir ou antecipar determinados comportamentos. Este tipo de capacidade de tomada de decisão tem uma elevada importância no mercado de trabalho, uma vez que em certos sistemas precisamos de processar dados rapidamente para podermos tomar decisões em função dos resultados obtidos, ajudando, por exemplo, a prevenir comportamentos de risco.

1.3 ESTRUTURA DO DOCUMENTO

Para além do presente capítulo, esta dissertação está organizada da seguinte maneira:

- **Capítulo 2 – *Big Data*, Fundamentos e Aplicações**

  Neste capítulo apresentamos o conceito de *Big Data* e as suas principais características, bem como as suas aplicações e vantagens.

- **Capítulo 3 – Algoritmos de *Sketching***

  Aqui apresentamos e explicamos os diversos algoritmos de *sketching* que encontrámos na literatura, apresentando e ilustrando a forma como cada um deles atua e comparando as suas principais características.

- **Capítulo 4 – Caso de Estudo**

  Neste capítulo apresentamos o nosso caso de estudo e descrevemos a sua implementação, dando particular atenção à metodologia adotada, ao modo de funcionamento do laboratório e às tecnologias usadas.

- **Capítulo 5 – Teste e Análise dos Algoritmos**

  Apresenta uma descrição de todos os testes realizados ao laboratório ao nível da sua performance tendo em conta os diferentes parâmetros em estudo. Estes resultados são apresentados em ambiente gráfico de forma a podermos realizar uma discussão de resultados mais intuitiva.

- **Capítulo 6 – Conclusões e Trabalho Futuro**

  Neste capítulo fazemos uma breve apreciação ao trabalho que foi realizado nesta dissertação, analisando os principais aspetos do trabalho e problemas com que nos deparámos. Terminamos este capítulo, apresentando e explicando algumas linhas de trabalho que podem ser desenvolvidas a curto e médio prazo.

2

BIG DATA, FUNDAMENTOS E APLICAÇÕES

2.1 EMERGÊNCIA E APLICAÇÃO

Ao longo do tempo podemos verificar que os dados produzidos e manipulados pelas pessoas têm vindo a aumentar exponencialmente. Segundo uma pesquisa da IBM (Taurion, 2013b), realizada em 2013, verificou-se que, no ano de 2000, apenas 25% dos dados eram digitalizados, mas, já em 2007, esse número subiu para 93% e poucos anos depois, em 2013, voltou a subir para 98%. Este crescimento, deve-se principalmente a fatores como o aumento do acesso a dispositivos eletrónicos (telemóvel, computador, máquinas industriais, etc.) e ao crescente uso da *internet* por parte de todas as gerações, o que está a gerar uma grande revolução na forma de como processamos os dados.

Com base neste aumento do volume de dados, surgiu a necessidade de as empresas implementarem o uso do *Big Data* nos seus sistemas de modo a conseguirem melhorar as suas operações, fornecer um melhor apoio ao cliente, criar campanhas de *marketing* com base nas preferências de um cliente, e por fim, rentabilizar ao máximo os seus lucros.

As empresas que implementaram este conceito nos seus sistemas, possuem uma vantagem sobre aquelas que ainda não o fizeram, pois podem tomar decisões de uma forma mais rápida e precisa, usando para isso os dados gerados de forma eficaz. Este tipo de implementação geralmente envolve *Terabytes* (TB), *Petabytes* (PB) e em alguns casos pode usar até *Exabytes* (EB) de dados capturados de forma contínua.

O *Big Data* veio criar uma nova era na sociedade moderna. Como consequência, o processo de análise de dados mudou drasticamente, bem como os paradigmas da ciência e da economia, por exemplo. O conceito de *Big Data* baseia-se no modelo dos 5Vs (Sheriff, 2019), nomeadamente:

- **Volume**

  Esta característica representa todas as atividades que realizamos no quotidiano; no nosso dia a dia, existe um grande volume de dados, desde a troca de *emails*, redes sociais, transações bancárias; é este o "v" que caracteriza o *Big Data*.

- **Variedade**

  Ao possuirmos um elevado número de dados, por consequência, obtemos uma grande variedade de elementos; podemos encontrar dados produzidos nos mais diversos tipos, como, por exemplo, CSV, JSON, imagem, entre outros; esta variedade de tipos de dados é vista como um dos grandes desafios do *Big Data*.

- **Velocidade**

  Esta característica é extremamente importante, pois o movimento dos dados é praticamente efetuado em tempo real; cada vez mais queremos esperar menos pelas conclusões que resultem de *queries* sobre esses elementos de dados.

- **Veracidade**

  Para que se possam tirar conclusões e serem tomadas decisões adequadas precisamos que a informação seja fidedigna.

- **Valor**

  Quanto maior for a riqueza dos dados, maior é a importância do processo de seleção e de análise dos mesmos, proporcionando, assim, benefícios para o problema com que estamos a lidar.

O volume é a característica deste tipo de sistemas que mais vezes é mencionada. Um sistema de *Big Data* não precisa de ter necessariamente uma grande quantidade de dados, mas um grande número dos mesmos permite tirar maior partido deste conceito. Estes são recolhidos através de *logs* de sistema e de sistemas de processamento de *streams* de dados que possuem a capacidade de produzir volumes de dados massivos de forma contínua e em tempo real. Os dados podem ser obtidos através de inúmeras fontes com ou sem um elevado grau de diversidade entre eles numa escala proporcional com o volume dos mesmos.

Hoje em dia, podemos afirmar que a principal fonte de dados é a *Internet* e em particular as redes sociais (e.g. número de pesquisas, *blogs*, *posts*, consultas ou *feed* de notícias), os dados provenientes de transações (e.g. compras *online* ou novos registos), os dados de biometria (e.g. reconhecimento facial, DNA ou impressões digitais), os dados gerados pelos utilizadores (e.g. documentos, exames ou o número de cliques realizados sobre uma página) ou os dados gerados pelas mais diversas máquinas (e.g. sensores ou GPS).

Com o uso de diferentes tipos de dados, o *Big Data* fornece às empresas vários *insights* valiosos sobre os seus clientes, que podem ser usados para aprimorar campanhas e redefinir estratégias de *marketing* com o objetivo de ter uma interação mais personalizada com cada cliente.

Em *Big Data* podemos classificar os dados em três categorias diferentes, nomeadamente (Halper and Krishnan, 2013):

- **Dados estruturados**

  São aqueles que apresentam uma estrutura rígida, previamente definida, como é o caso das bases de dados relacionais.

- **Dados semiestruturados**

  Não se encontram de acordo com uma estrutura formal uma vez que possuem uma estrutura flexível, o que implica que seja necessário efetuar, previamente, uma breve análise para identificarmos uma maneira de extrairmos a informação que pretendemos. Veja-se, por exemplo, os casos dos ficheiros JSON, XML ou RDF.

- **Dados não estruturados**

  Nos quais é necessário efetuar um pré-processamento ou realizar algumas ações de *parsing* para obtermos a informação necessária para o estudo que quisermos realizar. Este tipo de dados caracteriza-se por não possuir uma estrutura definida. Mais de 80% dos dados gerados utilizam este tipo de estrutura, em particular aqueles que podemos encontrar em textos, arquivos, imagens, vídeos e áudios.

Esta diversidade de tipos de dados representa um desafio em *Big Data*, principalmente naquilo que diz respeito à manipulação de dados semiestruturados e não estruturados com a finalidade de os processar de forma eficiente para que estes possam ser analisados o mais rapidamente possível. O processamento dos dados é realizado com a ajuda de algoritmos, previamente implementados, o que nos permite fundamentar aquilo que podemos concluir com base na análise dos resultados obtidos. Existem vários tipos de algoritmos, cada um deles com o seu próprio objetivo. Assim, não podemos afirmar qual deles é o melhor, mas sim dizer qual é aquele que melhor se adequa ao nosso caso em estudo. Por exemplo, se quisermos determinar o comportamento de uma linha de produção ou saber o número de visualizações de um vídeo num dado intervalo de tempo temos que usar diferentes algoritmos, dado possuírem objetivos distintos. Os algoritmos que atuam sobre dados que vão sendo obtidos em tempo real, para conseguirem produzir um resultado com algum sucesso, têm que lidar com um fator determinante: o erro. Como tal, é necessário estabelecer um equilíbrio entre o fator de erro e a eficiência do processamento do algoritmo, de modo a ser possível obter um resultado fiável, obtido muito próximo do tempo real.

Os sistemas de base de dados relacionais, com grande aplicação na generalidade das empresas, hoje em dia não têm a capacidade necessária para lidar com este tipo de dados e fornecer resultados em tempos muito próximos do real. Assim, para se adaptarem a esta nova situação, as empresas tiveram que implementar outros tipos de sistemas que tivessem a capacidade de trabalhar grandes volumes de dados em tempo real

e lidar com fluxos de angariação de dados praticamente contínuos. A implementação destes novos sistemas trouxe soluções tecnológicas mais rápidas (e simples), tal como hoje podemos constatar quando utilizamos, por exemplo, uma aplicação que nos permita escolher o melhor caminho para chegar a um dado sítio, tendo sempre em conta o congestionamento do trânsito.

Atualmente, temos já acesso a soluções *Big Data* bastante interessantes em diversas áreas de trabalho, que permitem obter informação bastante pertinente, não só para as tradicionais ações operacionais, como também para suporte à tomada de decisão, gerando interessantes retornos financeiros e, obviamente, operacionais. No entanto, também temos de ter em conta que essas soluções são bastante recentes e, assim, necessitam de serem melhoradas, especialmente em termos dos algoritmos que utilizam, do conhecimento possuído pelo meio empresarial acerca dessas soluções, e, por fim, das pessoas que necessitam de conhecimento especializado para poderem trabalhar com essas soluções de forma prática e eficiente.

2.2 ÁREAS DE APLICAÇÃO

Podemos aplicar as técnicas e soluções de *Big Data* nos mais diversos ramos de atividade, de forma a trazer melhorias significativas ao nível operacional e de tomada de decisão. De seguida, vamos analisar algumas das aplicações *Big Data* que podemos encontrar nesses ramos, nos quais, acreditamos, que poderão revolucionar grande parte dos processos de análise de dados.

2.2.1 *O Setor da Saúde*

Nos dias que correm, *Digital Health* é uma das palavras do momento no setor da saúde. Essencialmente, refere o uso de sistemas de informação com elevada capacidade para recolher, agregar e trabalhar dados estruturados, semiestruturados e não estruturados, que usualmente são gerados e tratados no setor da saúde, em particular em sistemas de gestão de informação clínica.

Neste domínio podemos utilizar vários algoritmos de previsão para se conseguir lidar, de forma eficiente, com o elevado volume de informação que todos os dias enfrentamos na área da saúde. Com esses algoritmos conseguimos analisar e combinar os vários elementos de dados que possam estar contidos nessa informação, tais como sintomas de doenças, medições de instrumentos médicos, ou relatórios produzidos por médicos acerca deste ou daquele processo médico. Toda essa informação pode ser correlacionada de forma a que se possa obter nova informação, novos elementos de dados, como, por exemplo, prever a idade ou o sexo de uma pessoa em que a probabilidade de contrair uma dada doença é mais elevada.

No setor da saúde podemos verificar claramente a emergência de aspetos relacionados com *Big Data* quando um profissional de saúde realiza, por exemplo, um *Eletronic Health Registers* (EHR) (Hecht, 2019). Na Figura 1 podemos ver alguns elementos de dados de um EHR.

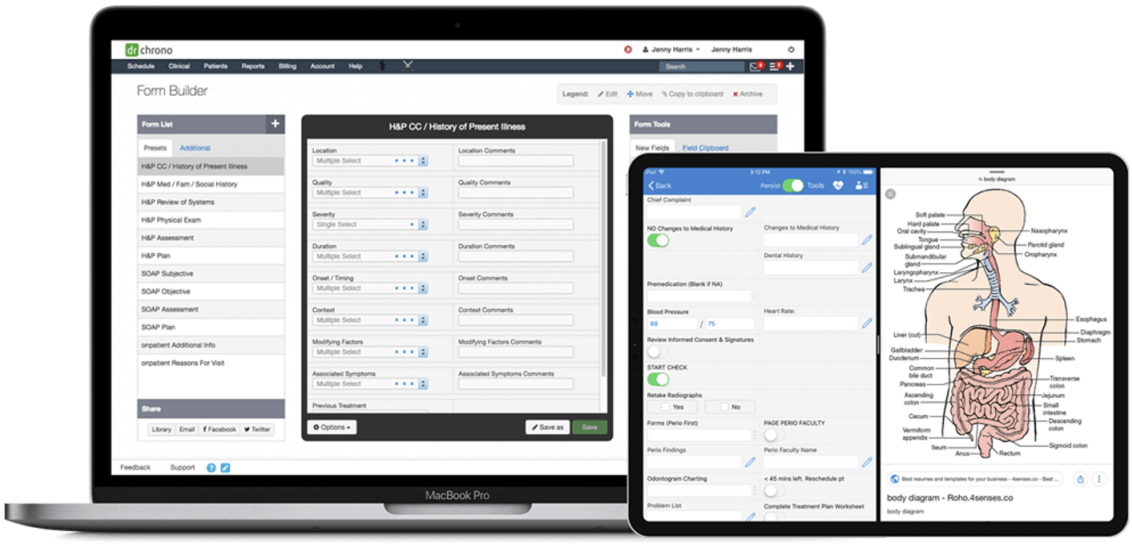


Figura 1: Exemplo de um sistema EHR - Figura extraída de (DrChrono)

Num sistema EHR cada paciente possui o seu próprio registo digital, o qual inclui um conjunto de informação bastante diversificado, como, por exemplo, a caracterização do próprio paciente, o seu histórico médico, uma relação das suas alergias, o indicador do seu tipo de sangue, ou os resultados de exames que já tenha realizado. Neste sistema, por uma questão de segurança e de garantia de privacidade, todos os registos são partilhados através de sistemas de informação encriptados, tanto por unidades de saúde públicas como de privadas. Esses registos são facilmente modificáveis de modo a permitir um uso facilitado do serviço, quer por parte do médico quer por parte do paciente. O EHR também possui outro tipo de funcionalidades como o adicionar avisos, a marcação de consultas ou a disponibilização de prescrições médicas.

Embora os sistemas EHR tenham revolucionado este setor, ainda existem poucos países nos quais tenham sido implementados. Os EUA são o país que mais apostou nesta tecnologia. Cerca de 85% dos seus hospitais já introduziram sistemas EHR nos seus serviços. Por sua vez, em Portugal o recurso a esta tecnologia só agora é que começou a ser uma maior aposta, sendo que para alguns hospitais já é possível efetuar marcações de consultas e visualizar receitas.

2.2.2 *O Setor da Aviação*

Os maiores gastos das companhias aéreas são realizados em combustível. Estes gastos representam cerca de 30% a 35% do seu orçamento mensal (Galdino, 2016). Para tentar reduzir estes valores, uma empresa, com sede nos Estados Unidos, decidiu utilizar *Big Data* em todos os seus processos. Através da informação recolhida por sensores colocados nos aviões, verificou que num voo transatlântico são gerados 640 TB de dados, que ao serem processados e analisados permitem prever, com mais exatidão, o tempo que falta para a próxima revisão, o momento para efetuar o próximo abastecimento, a rota mais económica para um dado voo, entre outras informações. Tudo isto ajuda, obviamente, o processo de tomada de decisões. Supondo que este tipo de análise irá trazer benefícios à empresa em questão na ordem dos 1%, ao fim de 15 anos poderemos estar a falar de um lucro de cerca de 30 biliões de dólares. Assim sendo, com esta pequena melhoria, é possível verificar o impacto que o uso desta tecnologia pode vir a ter neste tipo de empresas.

No setor da aviação nenhum mecanismo consegue garantir uma segurança completa nos voos. Porém, com a ajuda do *Big Data* estão a ser desenvolvidas tecnologias que tem como objetivo principal reduzir o número de sinistros. A aplicação de *Big Data* neste domínio evidencia-se, principalmente, em duas situações específicas. A primeira está relacionada com o trajeto que um avião realiza. Através da utilização de *Big Data* conseguimos identificar, analisar e processar, em tempo real, potenciais situações de risco e, ao mesmo, tempo acompanhar todo o percurso realizado pelo avião. Por exemplo, caso ocorram problemas com uma das asas do avião, podemos obter em tempo real essa informação e, assim, atuarmos de forma imediata de modo a minimizar potenciais danos. A segunda situação está relacionada com as inúmeras falhas que existem nas caixas negras de um avião. Apesar de hoje já possuirmos sistemas de armazenamento de dados bastante sofisticados, as caixas negras continuam idênticas às de há 30 anos atrás. Por isso um dos próximos passos a dar neste domínio, seria aplicar o processo de *Big Data* nestas caixas de forma a que estas possam comunicar com a entidade controladora em tempo real, permitindo, assim, identificar situações anómalas mais facilmente e de forma mais eficaz.

2.2.3 *O Setor dos Sistemas de Recomendação*

Os sistemas de recomendação (Mira, 2019) têm como objetivo principal fazer a recomendação de produtos ou serviços que sejam do interesse das pessoas, com base nas ações que vão realizando em vários domínios de atividade. Para isso, estes sistemas usam dados dos clientes, que quando submetidos a critérios específicos de seleção permitem identificar os diferentes padrões de comportamento desses clientes quando em contacto ou utilizando um determinado produto ou serviço. Conhecendo-se esses padrões de comportamento, é possível desenvolver medidas que contribuam para o aumento das vendas ou do grau de satisfação dos clientes. Um exemplo de aplicação deste processo pode ser visto na Figura 2, que mostra um quadro de sugestões usado pela Netflix. Este quadro foi definido a partir das visualizações que um utilizador realizou. Ao analisar essas

visualizações, a Netflix consegue sugerir ao utilizador filmes ou séries que tenham um conteúdo idêntico ao das visualizações já realizadas.

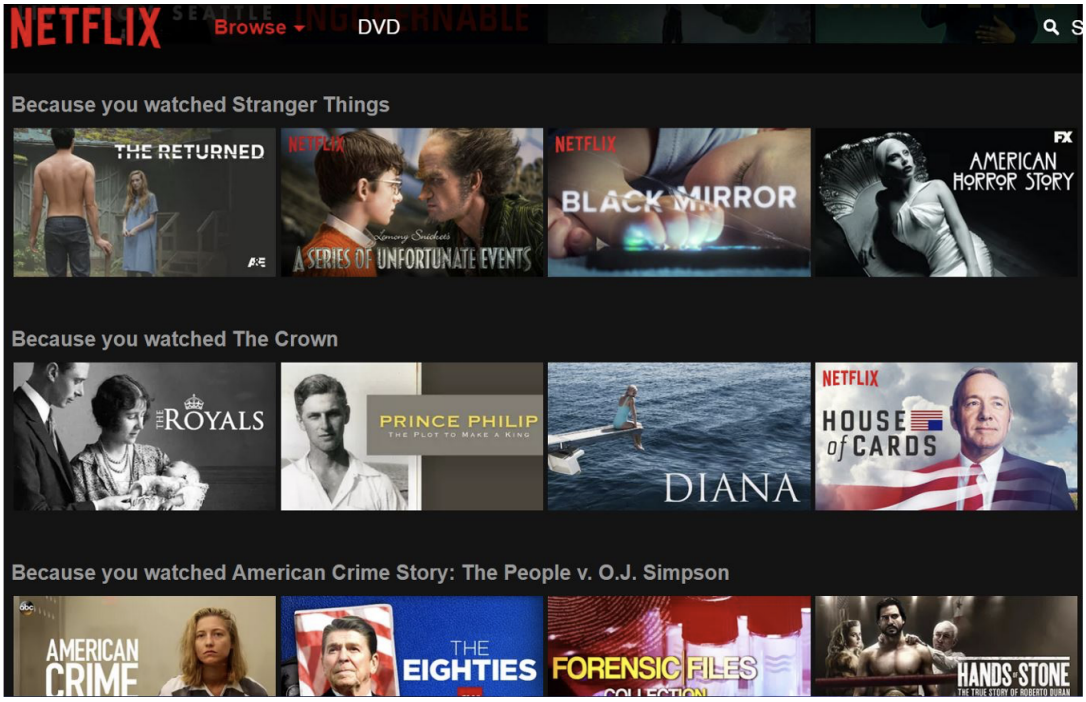


Figura 2: Exemplo de um Sistema de Recomendação - Figura extraída de (Natalie, 2017)

Os sistemas de recomendação podem fazer as suas sugestões utilizando várias técnicas e métodos. De referir (Marcelino, 2014):

- **Recomendações com base em vizinhanças**

  Este método utiliza um sistema de grafos para acolher os relacionamentos entre os itens e os utilizadores para construir o modelo de sugestão. Através do grafo gerado é possível observar os vários relacionamentos entre itens parecidos. Isso facilita a geração das sugestões, uma vez que assim é possível identificar os itens com maior interesse para os utilizadores. Por outro lado, também é possível ver como os utilizadores estão relacionados entre si através dos seus gostos comuns. Isso reduz muito o esforço computacional usado para a sugestão de um novo item.

- **Recomendações baseadas em conteúdo**

  Estas recomendações baseiam-se no conteúdo que é usualmente selecionado e visto pelo utilizador. Com base na análise dos conteúdos selecionados pelo utilizador é possível estabelecer relações com outros conteúdos, através da aplicação de análises de similaridade, e sugerir aqueles que possam agradar ao utilizador.

- **Filtragem com colaboração**

  Na filtragem colaborativa, por omissão, usamos uma estrutura com base numa matriz que tem como objetivo relacionar um dado conteúdo com o utilizador. Os relacionamentos são estabelecidos com base no histórico das pesquisas do utilizador. Com esses relacionamentos conseguimos recomendar um conteúdo adequado ao utilizador, ajustando as sugestões às mudanças nos padrões que possam vir a acontecer.

- **Recomendações híbridas**

  Os sistemas de recomendação híbridos são os mais eficazes de todos os aqueles que apresentámos até ao momento, já que utilizam combinações dos diversos métodos e técnicas anteriores. Nestes sistemas potenciam-se os pontos fortes e diminuem-se os pontos fracos de cada técnica, o que permite criar sistemas de sugestão mais eficazes e capazes de mostrar os conteúdos mais relevantes para os utilizadores.

Um excelente exemplo para combinar recomendações personalizadas é a filtragem coletiva. Por sua vez, a abordagem híbrida é usada para sugestões mais precisas de modo a cumprirem todas as expectativas do utilizador. A abordagem da Netflix, por exemplo, é baseada na criação de sugestões para os seus utilizadores. Essas sugestões utilizam as avaliações realizadas pelos utilizadores, o conteúdo visto recentemente e a combinação entre utilizadores com gostos similares.

Ao implementarmos um sistema de recomendação conseguimos obter os seguintes benefícios (Techlabs, 2017):

- **Satisfação do cliente**

  Com a ajuda destes sistemas, a experiência dos clientes torna-se muito mais agradável. Por exemplo, os clientes tendem a olhar para recomendações de produtos baseados na sua última navegação. Principalmente porque acham de que daí vão surgir melhores oportunidades para produtos do seu interesse. Este tipo de abordagem para com o cliente leva à retenção do mesmo.

- **Receita**

  Os algoritmos de recomendação tem sido explorados e executados cada vez mais. Foi comprovado que a sua aplicação consegue gerar mais lucros à empresa e um melhor grau de satisfação por parte do utilizador. Com o objetivo de aumentar os rendimentos, é necessária uma maior aproximação aos clientes, através do conhecimento das suas preferências proporcionando uma navegação personalizada para cada um.

- **Personalização**

  De uma forma frequente, aceitamos recomendações de amigos e familiares porque confiamos na sua opinião. Eles sabem o que nos poderá interessar com maior certeza. Esta é a razão principal pela qual eles são bons na recomendação de conteúdo e é o que os sistemas de recomendação tentam modelar.

- **Relatórios e Estatísticas**

  A capacidade de gerar relatórios faz parte do processo de personalização. Quando um sistema tem a capacidade de dar ao utilizador recomendações relevantes tem por base esses relatórios que permitem dar uma sugestão mais adequada. Com o auxílio desses relatórios, por vezes, o sistema gera recomendações de produtos com baixa procura, com o objetivo de aumentar o número de vendas dos mesmos.

Os sistemas de recomendação vieram revolucionar o setor das vendas *online*. Hoje, podemos afirmar que os sistemas de recomendação vão-se tornar cada vez mais essenciais em ambientes de *e-commerce*. Quando devidamente configurados, estes sistemas podem aumentar o número de vendas consideravelmente, gerando mais receitas e proporcionando ao utilizador experiências personalizadas na utilização de tais ambientes.

2.2.4 *O Setor Empresarial*

No setor empresarial encontramos inúmeras situações nas quais os dados são gerados por sensores, que são usados, por exemplo, na monotorização dos transportes ou no controlo dos semáforos, por máquinas com capacidade de comunicar, em particular em domínios de atividade agrícolas, e máquinas com capacidade de gerar *logs* de dados, como são as máquinas de um sistema produtivo ou de um sistema computacional. Nestes tipos de casos, o *Big Data* é utilizado para melhorar a eficiência e ao mesmo tempo para economizar recursos.

Por exemplo, a NIKE utiliza o *Big Data* para conseguir monitorizar os hábitos e os comportamentos desportivos do seu público alvo utilizando vários tipos de aplicações (FIA, 2018). Essas aplicações possuem a capacidade de gerar informação diversa, relacionada, por exemplo, com a distância percorrida, velocidade média, ou os locais mais frequentes para o treino, visualizar peças de roupa e calçado, que podem ser configurados na aplicação com o objetivo de criar o *outfit* dos utilizadores com os produtos disponíveis para venda, entre outras coisas mais. Com este tipo de sistema a empresa tem a capacidade de sugerir produtos cada vez mais alinhados com as expectativas do público alvo, fazendo com que estes se fidelizem à marca, gradualmente, sem se aperceberem. Na Figura 3, podemos observar alguns dos parâmetros utilizados pelo sistema de recomendação da NIKE.

Figura 3: Exemplo de aplicação no setor empresarial - Figura extraída de (Bower, 2017)

2.2.5 *O Setor Público*

Diariamente, os gestores públicos enfrentam escolhas difíceis quando tem de decidir qual o orçamento a aplicar ou quando tem de estudar a melhor opção para investir. Logicamente, que gerir e analisar todos esses dados manualmente se torna inviável, o que pode causar, eventualmente, algum tipo de colapso administrativo. Devido a questões como essas, os gestores de grandes empresas optaram por utilizar *Big Data* em tarefas de análise de dados de forma a ultrapassarem esses problemas. Atualmente, já existem várias ferramentas neste domínio orientadas especificamente para a área governamental, por exemplo, a Agenda Digital Europeia, o DOME (que é a implementação de um sistema de supercomputação com a capacidade de lidar com um conjunto de dados superior a um *Exabyte* por dia) e em diferentes aplicações da Segurança Nacional de cada país (Kin, 2014).

Com a utilização do *Big Data* os gestores públicos adquirem ferramentas que lhe podem ajudar a (Munné, 2016):

- **Combater a corrupção e desvio de receitas**

  A generalidade dos governos costuma cruzar a informação de milhares de contribuintes com o intuito de combater, por exemplo, a lavagem de dinheiro e outras fraudes fiscais. Podemos ver o sucesso desta iniciativa com o exemplo dos *Panana Paper* (Nietner), um caso que aconteceu recentemente.

- **Implementar "cidades inteligentes"**

  As iniciativas relacionadas com as cidades ditas inteligentes têm como objetivo melhorar as condições de vida das pessoas no seu quotidiano citadino, monitorizando em tempo real vários dos serviços desenvolvidos no seio das cidades. Esses serviços têm como objetivo providenciar às pessoas melhores condições de vida, através, por exemplo, de uma regulação mais efetiva do trânsito ou na ajuda na identificação dos locais da cidade com maior taxa de poluição. Na Figura 4 podemos ver uma ilustração bastante interessante das diferentes áreas de aplicação de iniciativas de cidades inteligentes.

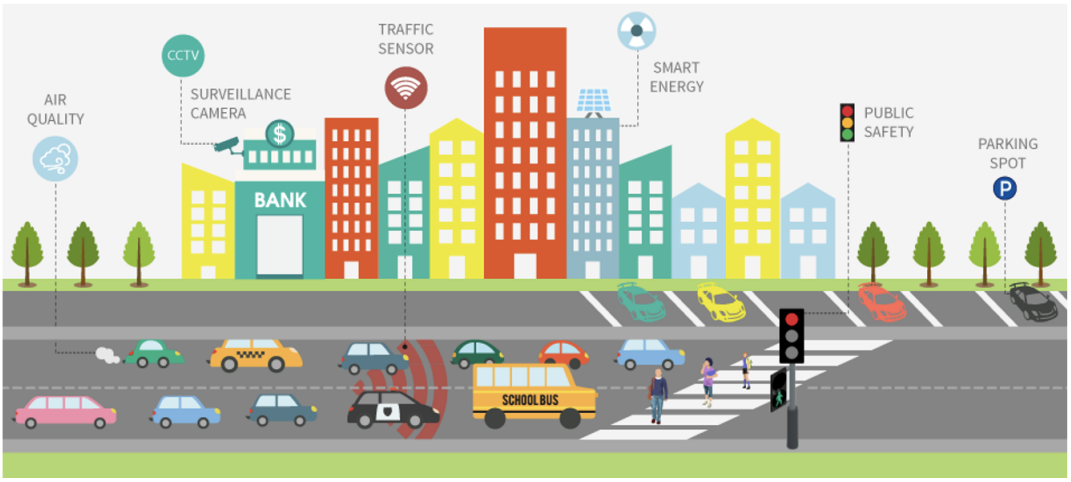


Figura 4: Exemplo de uma *Smart City* - Figura extraída de (Saikali, 2016)

- **Monitorizar o nível de satisfação da população**

  Quando uma empresa lança um produto novo no mercado, é vulgar fazer-se uma recolha de informação pertinente acerca desse produto através da realização de inquéritos às pessoas para se saber se o produto vai ser uma boa aposta ou não. Essa recolha de informação também pode ser feita através de *reviews* realizadas pelos clientes sobre os produtos ou serviços, tendo em consideração mensagens como aquelas que podemos ver na Figura 5. A análise destas *reviews* permite às empresas ficarem com uma noção mais concreta sobre a qualidade dos seus produtos ou serviços prestados e, consequentemente, identificar possíveis aspetos a melhorar.

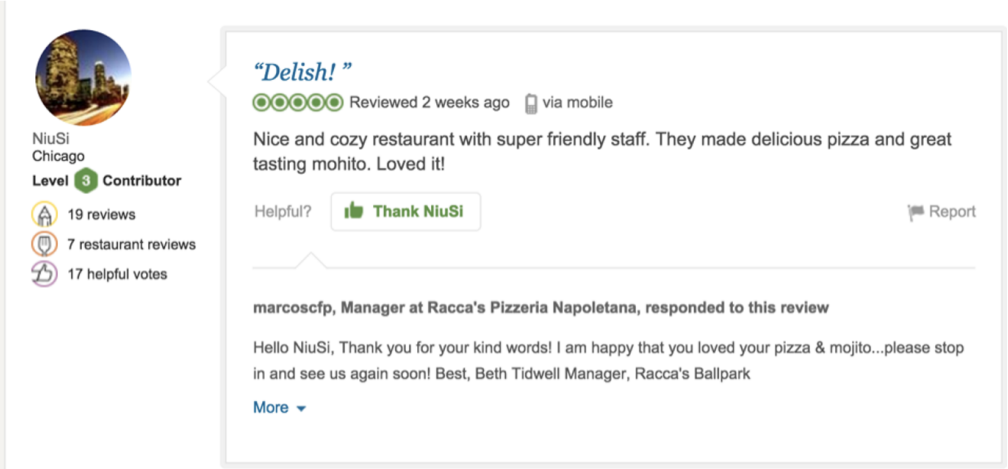


Figura 5: Exemplo do grau de satisfação do cliente - Figura extraída de (NiuSi)

2.3 SISTEMAS TRADICIONAIS E SISTEMAS DE *big data*

Hoje em dia, a gestão de dados tem-se tornado cada vez mais importante, tendo sido adotada por um grande número de empresas com o objetivo de aperfeiçoar os seus processos de trabalho. Em 2013, Krishnan (Krishnan, 2013) definiu o processamento tradicional de dados como algo que se desenvolve ao longo de cinco etapas, nomeadamente: o armazenamento, a extração, a transformação dos dados, o carregamento e os resultados obtidos (Figura 6). Este tipo de dados caracteriza-se por possuir um volume médio de forma estruturada, garantindo assim uma menor preocupação com a correção de irregularidades.

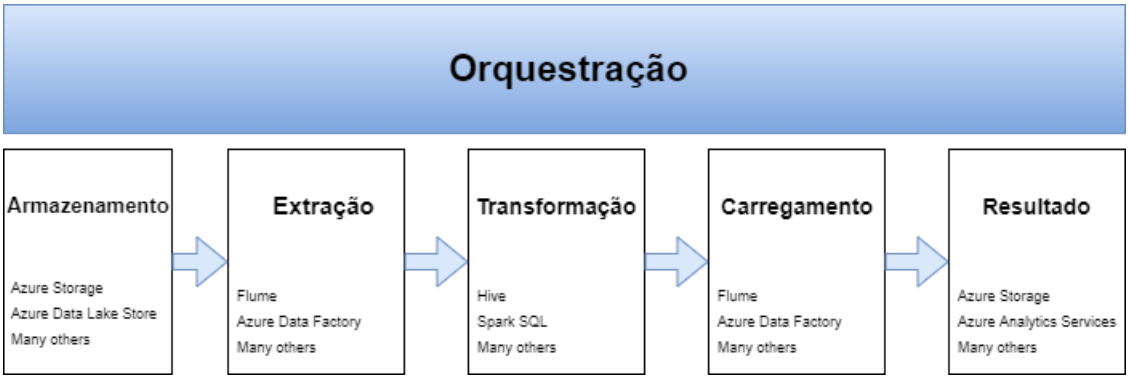


Figura 6: Exemplo do grau de satisfação do cliente - Figura adaptada de (Howell, 2020)

A diferença mais evidente no processamento de dados em sistemas tradicionais e nos sistemas de *Big Data* é a escalabilidade, quer esta seja vertical ou horizontal. A escalabilidade horizontal consiste no uso de computação paralela, que é usada para processar uma elevada quantidade de dados que apenas uma máquina é incapaz

de processar, assim, conseguimos uma maior eficiência e também reduzimos custos ao evitar comprar uma máquina que possua a capacidade de processar todo este fluxo. A escalabilidade vertical é, maioritariamente, usada em sistemas *Structured Query Language* (SQL), para se conseguir aumentar o poder de processamento, através de investimentos em máquinas com tecnologias mais avançadas e dispendiosas. Em alguns casos, a escalabilidade consegue-se alcançar através da simples criação de mais processos, que são responsáveis por fazer a distribuição dos dados. O investimento é basicamente em *hardware*, nomeadamente em processadores, memória central e armazenamento de dados, sempre com o objetivo de aumentar a capacidade da máquina.

Por incrível que pareça a escalabilidade horizontal pode ser mais barata que a vertical, pelo menos naquilo que diz respeito ao custo de aquisição da infraestrutura. Mas, o custo da sua gestão e desenvolvimento pode influenciar o custo total. No entanto, a escalabilidade horizontal tem a vantagem de ser mais tolerante a falhas, no sentido de ser mais fácil reverter uma operação em caso de erro. Contudo, são raras as aplicações que precisem de escalabilidade horizontal. Ela é útil quando pretendemos dar mais confiabilidade a um sistema e não quando precisamos de envolver mais recursos.

Nos sistemas tradicionais, aplicamos o conceito de *Business Intelligence* (BI), que consiste, em aplicar uma técnica que permite gerir negócios com base na análise de informação, com o objetivo de determinar, por exemplo, aspetos que influenciam negativamente ou positivamente o negócio, o que ajuda, obviamente, nos processos de tomada de decisão. Por outro lado, nos sistemas de *Big Data*, não existe uma preocupação com a exatidão que é fornecida num sistema de BI, mas sim no processamento de dados em busca de correlações e descobertas. Desta forma, a grande diferença nestes sistemas está aliada à possibilidade e à oportunidade em cruzar estes dados oriundos de diversas fontes para obtermos *insights* valiosos.

A enorme diversidade dos dados presente nos dispositivos conectados em rede, comprovou as limitações dos sistemas relacionais que serviam bem para um fluxo reduzido e para dados estruturados, o que na atualidade é algo muito específico devido á quantidade e diversidade dos dados produzidos. Esse problema originou a busca de ferramentas *Not Only SQL* (NoSQL), pois estas possuem a capacidade de lidar com todo o tipo de dados e com um fluxo maior, mas por outro lado estas requerem um processamento mais complexo do que os sistemas tradicionais. Em ambos os sistemas, os resultados obtidos podem ser visualizados através de gráficos, *dashboards*, entre outras ferramentas usadas para a análise.

Quando pretendemos implementar um sistema que utilize *Big Data* temos de ter sempre em conta dois tipos de requisitos: os de infraestrutura e os de processamento de dados. Os primeiros consideram-se os requisitos relacionados com o uso da aplicação em termos de desempenho, usabilidade, confiabilidade, escabilidade e segurança. Já por sua vez, os segundos referem-se a todo o tipo de operações que envolvem os dados.

Em termos gerais, podemos ter que analisar os seguintes requisitos de infraestrutura (Cross, 2018):

- **Escabilidade Linear**

  Este requisito envolve a adição de servidores para permitir uma computação 100% paralela com o objetivo de reduzir os custos. Geralmente, esta é uma solução implementada a longo prazo, pois passar de um sistema monolítico para um deste tipo é um processo complexo, embora extremamente eficaz.

- ***Throughput* Elevado**

  O *Throughput* é definido através do rácio do total de dados processados e do tempo de processamento. Para manter este valor elevado, é necessário possuir uma infraestrutura com velocidades elevadas de processamento e armazenamento.

- **Tolerância a falhas**

  É um requisito que garante que os sistemas não colapsem devido à ocorrência de uma falha ou de um erro. Para sistemas que necessitam de alta disponibilidade e escabilidade é necessário ter em consideração a implementação de um bom sistema de tolerância a falhas, por exemplo, uma das soluções mais usadas é a utilização de recursos redundantes.

- ***Auto Recovery***

  Podemos caracterizar o *Big Data* como sendo a recolha em tempo real de dados em alta velocidade. Quando acontece algum imprevisto, a capacidade que este sistema tem para recuperar assume uma grande importância. Pois se perdemos uma hora de dados o impacto é mínimo, mas se perdemos um dia ou mais, o impacto pode ser muito grande no negócio. Assim sendo, a recuperação automática deste tipo de acontecimentos fornece garantias sobre interrupções inesperadas.

- **Interfaces para Linguagens de Programação**

  As próprias linguagens de programação desenvolveram interfaces de modo a facilitar o manuseamento destes dados, tornando assim o processo de *Big Data* mais intuitivo.

- **Alto nível de paralelismo**

  Devido ao aumento exponencial do tamanho dos dados gerados, o uso do paralelismo neste tipo de sistemas tem vindo a melhorar a velocidade de processamento dos mesmos.

- **Processamento de dados distribuídos**

  Este tipo de processamento permite que um problema seja subdividido em enumeras partes mais pequenas que possuem um melhor tempo de processamento. Os resultados dessas partes, são todos combinados para produzirem o resultado final numa etapa de agregação.

De seguida, vamos apresentar os requisitos que devemos ter em conta ao nível do processamento de dados:

- **Arquitetura Modelável**

  A arquitetura adotada neste tipo de sistemas tem de ser uma arquitetura de fácil alteração devido aos diferentes tipos de dados (dados estruturados, não estruturados e semiestruturados) que podem estar a ser processados.

- **Transformação dos Dados**

  Este processo desempenha um papel crucial, pois permite a identificação de anomalias ou de discrepâncias que possam comprometer a veracidade dos resultados obtidos. Esta transformação é aplicada usando métodos estatísticos, que avaliam os desvios e a relevância dos dados para a análise a fazer.

- **Escabilidade**

  Quando falamos que a escabilidade está presente num sistema estamos a ter em conta uma implementação com uma arquitetura de fácil manutenção, com inúmeras funcionalidades e com a possibilidade de existir trabalho em paralelo. Assim sendo, a escabilidade é a capacidade de aumentar a memória, os processos, resolver problemas de alta complexidade de processamento e de concorrência.

- **Armazenamento dos Resultados**

  Para o armazenamento dos dados temos de ter sempre em consideração o volume, a variedade e a velocidade de armazenamento para os guardar de forma eficiente usando para isso algoritmos previamente implementados.

- **Comunicação entre vários blocos de processamento**

  A leitura deste tipo de dados deve ser realizada de forma eficiente para que estes possam ser processados num tempo muito próximo do real.

- **Recolha de dados muito próxima do tempo real**

  Decidir que dados são úteis para responder a uma questão é um desafio nos dias de hoje, porque a dificuldade não está na recolha, mas sim na identificação de quais vão ser os dados úteis e importantes.

2.4 ÉTICA E PRIVACIDADE

Com o aumento contínuo da produção de dados surge uma questão bastante oportuna que é até que ponto é que podemos aceder a dados gerados livremente, uma vez que muitos deles são de cariz pessoal? Para defesa do utilizador foi criado o *Regulamento Geral de Proteção de Dados* (RGPD). Mas será que isso é suficiente? A grande maioria das pessoas sentem-se incomodadas quando um sistema parece saber demais sobre si, o que faz com que surjam sentimentos de vigilância e de desconfiança dado considerarem tal situação uma violação da sua privacidade. Isto pode ser visto como sendo um peso de duas medidas. Por um lado, pode ser usado para promover uma vida melhor para uma pessoa, mas, por outro lado, pode ser usado pelas empresas para desenvolverem, os seus negócios e aumentarem os seus lucros, com o conhecimento das atividades e comportamento de um utilizador comum. Essas situações levam-nos a um debate entre o livre arbítrio e uma "ditadura" de dados.

Para podermos tirar uma conclusão geral sobre este tema precisamos de ver respondidas duas perguntas relativamente à privacidade e à autoria dos dados. Quem deve controlar o acesso aos dados? E a quem pertencem os dados e quais são as obrigações de quem os usa? Estas e outras questões ainda não foram respondidas, o que revela que ainda temos muito a discutir sobre este tema. Porém, sabemos que, como qualquer outra tecnologia, esta pode ser usada para o bem e para o mal, cabendo-nos decidir como usá-la. Em conclusão, podemos verificar que uma das principais utilidades do *Big Data* é a possibilidade de conseguirmos adaptar serviços e produtos em função do cliente. Esta característica permite-nos impingir um serviço ou um produto estimulando as pessoas a realizarem um consumo excessivo do mesmo. Quanto mais sabemos sobre as pessoas, mais podemos oferecer e as chances de acertar também aumentam.

2.5 O FUTURO DO *big data*

Todos nós sabemos que é altamente improvável que o crescimento da produção de dados estanque ou diminua. Cada vez mais vivemos num mundo digital e, assim sendo, é muito improvável que a influência e a utilidade do *Big Data* diminuam num futuro próximo. Para o futuro podemos esperar por mais discussões e controvérsias sobre as vantagens e os potenciais perigos do *Big Data* para o nosso quotidiano. Prevê-se que a evolução do *Big Data* se desenvolva em três vertentes distintas. De referir, o estudo académico, o desenvolvimento de estruturas legais, especialmente para os meios empresariais, e uma participação responsável por parte da sociedade. As pesquisas académicas ou de investigação neste domínio continuarão a ser realizadas de forma muito ativa, expandindo assim o conhecimento adquirido até ao momento e contribuindo para a evolução do *Big Data*. A sociedade também vai possuir um papel importante na evolução do conceito, porque as pessoas são um dos maiores geradores de dados e isso remete-nos a debater questões éticas na utilização dos mesmos. Os desenvolvimentos de estruturas legais para o setor empresarial em conjunto com os valores éticos definidos pela sociedade vão regular de que forma os dados gerados poderão ser utilizados.

O debate sobre a ética do *Big Data* irá originar o desenvolvimento de *frameworks* e padrões-normativos e legais, bem de como novas técnicas para recolher, armazenar, processar e partilhar fluxos de *Big Data*. Os avanços técnicos neste domínio irão ser úteis, por exemplo, permitindo o uso de criptografia nos dados para dificultar a identificação ou a previsão de comportamentos que são abrangidos por eles.

O uso de *Big Data* levanta inúmeras questões éticas, especialmente quando as empresas utilizam os dados fornecidos para finalidades diferentes daquelas para as quais os dados foram inicialmente recolhidos. Para isso devemos ter em consideração os seguintes princípios (Tera, 2018):

- Os dados e as identidades devem permanecer privados
- Informações privadas partilhadas devem permanecer privadas
- Clientes devem ter uma visão clara do uso dos seus dados
- O *Big Data* não deve interferir na vontade humana
- O *Big Data* não deve tirar partido de questões preconceituosas

As empresas e as instituições deveriam discutir abertamente sobre estes assuntos com o objetivo de nos sensibilizar para termos outras considerações humanísticas, como a privacidade e a igualdade, bem como a elaboração de um *design* centrado nas pessoas baseados num conjunto de princípios definidos previamente.

Uma outra influência, será a capacidade que o *Big Data* terá para se desenvolver e para evoluir paralelamente com o uso ou não dos dados livres. Estes dados não têm nenhuma restrição em termos de acesso e de utilização, podendo ser facilmente manipulados por máquinas e estarem disponíveis de forma gratuita, sem limitações, para serem processados e analisados por qualquer instituição. Num futuro próximo, o *Big Data* e os dados livres serão os dois fatores principais para originar uma nova revolução nos sistemas de dados. Ambos aparecem num contexto de um mundo que procura mais abertura, agilidade e transparência. Mas, por outro lado, as limitações políticas estão um bocado esquecidas, uma vez que todos podem usufruir das suas vantagens em benefício próprio sem pensar no impacto que isso poderá causar na sociedade. Deste modo, seria necessário que acontecesse uma verdadeira revolução no *Big Data* para impor algumas restrições ao uso dos dados e não pondo em causa a vida de terceiros.

3

ALGORITMOS DE *SKETCHING*

3.1 ALGORITMOS DE *sketching*

O conceito de *Sketching* teve origem há várias décadas, e com a evolução deste surgiram algoritmos que possuem a capacidade de produzir resumos com base na aproximação dos dados. Esta abordagem tem demonstrado uma grande utilidade na forma como lidamos com o fluxo crescente dos dados, usando para isso menos recursos computacionais. Desta forma, podemos verificar que esta abordagem é bastante utilizada em redes neuronais, e em problemas de escalonamento de atividades, como por exemplo, na atribuição de tarefas a n máquinas numa linha de produção.

Em vários tipos de sistemas, hoje, temos a necessidade de utilizar uma classe de estruturas de dados particular, com uma complexidade sublogarítmica, que tem a capacidade de representar uma grande quantidade de dados. Essas estruturas têm o nome de *sketchs* e permitem-nos analisar um grande volume de dados no nosso computador, sem que que para isso tenhamos de aumentar o poder computacional das nossas plataformas de análise, de forma expedita muito próxima do tempo real.

Para conseguir resolver problemas NP-Complexos, tivemos a necessidade de utilizar algoritmos de *Sketching*, uma vez que estes problemas não podem ser resolvidos em tempo polinomial. Muitos destes problemas aparecem em aplicações no dia-a-dia, originando um forte apelo económico para resolvê-los de forma eficiente. Em termos práticos, podemos não precisar da solução ótima do problema, uma solução boa obtida por um algoritmo deste género pode ser o suficiente. Estes algoritmos utilizam a aleatoriedade como princípio do seu funcionamento. Assim sendo, para além do conjunto de dados que servem de *input* para o problema, é inserido um conjunto de *bits* aleatórios que determinará o comportamento do algoritmo, e consequentemente o resultado do problema. Isso significa que, usando o mesmo *input*, se o algoritmo for executado algumas vezes, podemos obter resultados diferentes.

A necessidade de implementarmos este tipo de algoritmos surgiu devido ao facto de nos dias de hoje a produção de dados estar a aumentar exponencialmente, de forma generalizada, a sua análise só vem trazer benefícios aos mais diversos setores. Este tipo de algoritmos permite responder de forma rápida às mais diversas interrogações que possam ser lançadas sobre repositórios de dados de grande dimensão. Em termos genéricos,

o aparecimento deste tipo de algoritmos permitiu reduzir os custos de análise de dados, dada a sua menor exigência em termos de capacidade de processamento.

Todavia, estes algoritmos não são perfeitos em termos de satisfação de resultados, uma vez que trabalham sobre esboços (*sketchs*) do conteúdo real dos dados que temos disponíveis. Por isso temos que ter sempre em conta uma certa probabilidade de erro. Assim, quando definimos uma determinada interrogação temos que ter um "equilíbrio" entre a precisão dos resultados que queremos obter e a eficiência do processo relacionada com a sua obtenção. De referir que, em muitos processos de tomada de decisão, os resultados contendo algumas falhas ou erros são admissíveis, desde que a percentagem de erro esteja abaixo de um dado valor considerado aceitável.

De seguida, vamos abordar alguns dos algoritmos mais relevantes para a manipulação de *sketchs*, nomeadamente: *count-min*, *hyper log log* e *t-digest*. A seleção de algoritmos de *sketching* que realizámos teve em conta o facto de estes usarem menos memória, possuírem um tempo de consulta constante e pela capacidade de serem facilmente paralelizados.

3.2 O ALGORITMO *count min sketch*

O algoritmo *Count-Min Sketch* foi concebido em 2003 por Muthukrisham e Cormode (Cormode and Muthukrishnan, 2005). Em termos gerais, este algoritmo trabalha com base num conjunto de elementos probabilísticos, de forma a conseguir apresentar uma resposta válida para os diferentes tipos de consultas que possam ser realizadas sobre um dado fluxo contínuo de dados – *data stream*. O algoritmo *Count-Min* pertence à família dos algoritmos de aproximação, que permitem estimar propriedades relacionadas com a frequência de um dado conjunto de dados num repositório de informação, por exemplo, estimar as frequências de elementos ou realizar consultas de intervalos de valor nas quais se pretende encontrar a soma das frequências dos vários elementos que estão contidos dentro dos intervalos considerados.

A estrutura deste algoritmo consiste numa matriz de contadores, com largura w e profundidade d. Desta forma, podemos definir cada entrada como sendo um contador, que é sempre inicializado com o valor 0. Posteriormente, em cada linha da matriz possuímos uma função de *Hash* diferente associada. Por outro lado, esta função possui a capacidade de mapear valores no intervalo {1,2,...,w}.

Posto isto, podemos concluir quanto à importancia do *UPDATE(i,c)* e do *ESTIMATE(i)* como sendo as principais operações deste algoritmo. Tendo isso em conta, é importante salientar que, enquanto o *UPDATE(i,c)* incrementa a contagem associada ao valor i, o *ESTIMATE(i)* retorna uma estimativa de contagem associada ao valor i.

De seguida, vamos fazer uma pequena explicação do fluxo de utilização deste algoritmo. Numa primeira fase, antes de começarmos a utilizar o algoritmo precisamos de criar e inicializar a matriz dos contadores com as respetivas funções de *Hash* (Figura 7), passando como argumento o número de colunas e linhas para a matriz que vai ser gerada.

```
const sketch = new CountMinSketch(2048, 1)
```

Figura 7: Inicialização do algoritmo *Count Min Sketch*

Se pretendermos adicionar um novo registo na tabela, usamos o *UPDATE(item, int count).* O item corresponde á ocorrência que pretendemos adicionar e o count correspode ao número de vezes que a queremos adicionar (se não tiver valor é considerado 1 por *default*), como demonstra a Figura 8.

```
update(x):
    for i = 1 ... d:
        count[i][hi(x)]++

sketch.update('alice')
sketch.update('alice')
sketch.update('bob')
sketch.update('carlos',2)
```

Figura 8: Exemplo de utilização do update no algoritmo *Count Min Sketch*

Finalmente, para estimarmos a contagem total de um item específico, usamos o *ESTIMATE (item).* Como pode ser visto na Figura 9.

```
estimate(x):
    result = 0
    for i = 1 ... d:
        result = min(result, count[I][hi(x)])
    return result

console.log(sketch.estimate('alice')) // output: 2
console.log(sketch.estimate('bob')) // output: 1
console.log(sketch.estimate('daniel')) // output: 0
console.log(sketch.estimate('carlos')) // output: 2
```

Figura 9: Exemplo de utilização do estimate no algoritmo *Count Min Sketch*

Para explicar de uma forma mais clara este algoritmo, de seguida, vamos fazer uma pequena demonstração (e explicação) do seu funcionamento ilustrando um caso específico de aplicação do algoritmo. Começaremos por considerar um pequeno caso, bastante simples, relativo à contagem de visualizações de um vídeo. Tipicamente, este caso aborda um problema da deteção da ocorrência de um evento num fluxo (uma sequência contínua de registos) de eventos, o qual queremos contabilizar o número de vezes que este evento ocorre.

Consideremos, uma tabela em que todas as colunas são representadas por uma função de *Hash*. Inicialmente todas as posições dessa tabela contém o valor 0. Quando uma nova visualização acontece as posições na tabela correspondestes ao vídeo, que são calculadas usando as funções de *Hash* de cada coluna, vão ser incrementadas.

De forma a efetuarmos a contagem de um dado evento, assumimos o mínimo calculado pelas funções de *Hash* das contagens do mesmo, como demonstra a Figura 10.

$$count(X) = min_i < h_i(x) > \quad \boxed{count(x) = min_i < h_i(x) >}$$

Figura 10: Expressão de cálculo utilizada pelo algoritmo *Count Min Sketch*

Suponhamos, agora, que usamos um *sketch* com 3 colunas (o número de funções de *hash*) e 4 linhas, e que o primeiro utilizador decide assistir ao vídeo "História do Japão". Desta forma, o identificador do vídeo vai ser dividido pelas 3 funções de *hash*, o que faz com que se obtenham os valores 1,2 e 1. Isto indica que devemos incrementar os contadores relativos às posições (1,1), (2,2) e (1,3), como podemos observar na Figura 11.

Hash 1	Hash 2	Hash 3
0	0	0
0	0	0
0	0	0
0	0	0

→

Hash 1	Hash 2	Hash 3
1	0	1
0	1	0
0	0	0
0	0	0

put(Japan)
hash(Japan) = (hash1(Japan), hash2(Japan), hash3(Japan)) = (1,2,1)

Figura 11: Ilustração do resultado da aplicação do algoritmo *Count Min Sketch* - Figura extraída de (Shukla, 2018)

Agora, suponhamos que um outro utilizador vai assistir ao vídeo "*Hello* – Adele", que possui 1,1 e 4 como valores de *Hash*, e que, de seguida, outro utilizador assiste ao vídeo "História do Japão". Para cada um destes casos, os valores correspondentes às *hashs* de cada vídeo vão ser incrementados. Estas alterações na tabela, podem ser visualizadas por ordem cronológica nas Figuras 12 e 13 respetivamente.

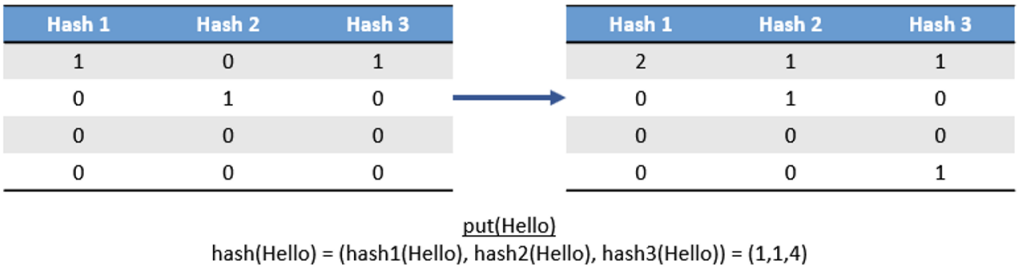


Figura 12: Exemplo 2 execução do Count Min Sketch - Figura extraída de (Shukla, 2018)

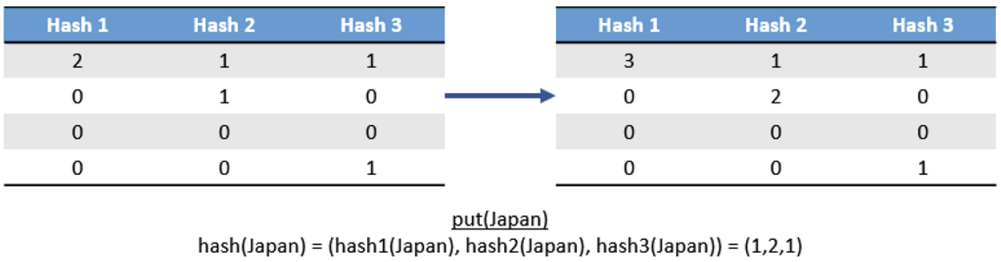


Figura 13: Exemplo 3 execução do Count Min Sketch - Figura extraída de (Shukla, 2018)

No fim das iterações realizadas anteriormente, podemos analisar quantas visualizações teve o vídeo "*Hello* – Adele". Para este vídeo, calculamos as respetivas *hashs* que são 1,1 e 4. Desta forma, devemos examinar as posições (1,1), (1,2) e (4,3), que pela análise da Figura 14, podemos constatar que são os valores 3,1 e 1.

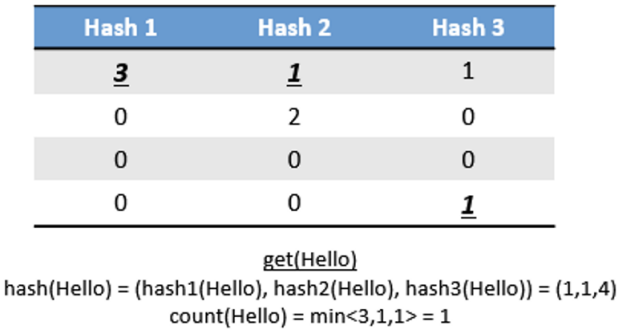


Figura 14: Exemplo 4 execução do *Count Min Sketch* - Figura extraída de (Shukla, 2018)

Podemos concluir que o "*Hello* – Adele" foi assistido apenas uma vez, apesar de uma das posições conter o valor 3 (Figura 14). Este valor verifica-se pelo facto de o valor resultante do cálculo das *hashs* serem coincidentes para diferentes vídeos. Com base na equação 1, podemos concluir que o vídeo foi visualizado apenas uma vez.

$$count(X) = min\, i < 1, 1, 3 >= 1 \tag{1}$$

O algoritmo *Count Min* tem grande aplicação em situações nas quais é necessário contar o número de vezes que um dado valor, ou conjunto de valores, ocorre num fluxo de dados, sendo bastante eficiente na realização desse trabalho. Em termos gerais, podemos aplicar este algoritmo em casos como:

- A compressão de dados de sensores. Neste tipo de casos, um sinal ou uma imagem podem ser adquiridos usando *hardware* (analógico ou digital), que com maior ou menor taxa de precisão calcula um produto escalar de cada linha de uma matriz com um sinal, possuindo um custo unitário por operação. Geralmente, o número de medições obtidas usando um *hardware* de deteção compactado é muito menor do que os dispositivos tradicionais, por causa deste ser um processo mais complexo, mais dispendioso e com maior taxa de acerto. Assim sendo, podemos amostrar um sinal analógico de banda limitada a uma taxa sub-*Nyquist* e ainda conseguimos recuperar frequências significativas no sinal ou num dado espectro do mesmo. A taxa de *Nyquist* é usada no algoritmo de compressão de modo a melhorar o *hardware*. Desta forma, verificamos que o algoritmo *Count Min* possui as seguintes propriedades que o tornam bastante útil na agregação eficiente dos dados em cenários de *streaming*, como é o caso das redes de sensores: o algoritmo é insensível a duplicações, ou seja, a reinserção de dados não o irá afetar, fornecendo assim garantias aproximadas a nível do cálculo. Também consegue aliar a agregação de dados com o nível de precisão sem nunca pôr em causa a objetividade do resultado final.

- A realização de *queries* complexas em bases de dados. Em situações nas quais existe um grande número de pedidos de informação (*queries*) sobre uma base de dados, o algoritmo *Count Min* pode ser utilizado para evitar uma sobrecarga na base de dados, para permitir a realização de consultas agregadas simples, tais como o número de entradas, a média ou a mediana de uma tabela.

- O controlo de aplicações em redes sociais (*networking*). Com o crescente aumento do uso de aplicações em rede tornou-se altamente imprescindível efetuar um controlo mais efetivo da execução dessas aplicações, controlando, por exemplo, o número de visitas que um site teve num dia ou o número de vezes que uma dada ação foi efetuada na aplicação. Uma das empresas que utiliza este algoritmo é a YouTube, com o objetivo de conseguir contabilizar de forma eficiente todas as visualizações dos seus vídeos. A análise deste tipo de situações melhora a experiência do utilizador, permitindo-lhe receber conteúdo de acordo com as suas preferências e para reajustar o foco da empresa no sentido de cumprir um maior número de objetivos.

3.3 O ALGORITMO *hyper log log*

O algoritmo *Log Log* enquadra-se na classe dos algoritmos probabilísticos. Como em muitos algoritmos semelhantes, tem por base o uso de uma função de *Hash* para aleatorizar os dados e colocá-los todos uniformes e independentes. São estes conjuntos de dados em *Hash* que são convertidos em estimativas de cardinalidade pelo algoritmo. Desta forma, o algoritmo *Log Log* utiliza m "pequenos *bytes*" de memória auxiliar para conseguir estimar em uma única passagem, o número de elementos distintos (cardinalidade) num conjunto de dados, com uma precisão de 1 / $\sqrt[2]{m}$.

O melhoramento do algoritmo *Log Log* deu origem ao *Hyper Log Log* (Dur and Flajolet). É usado quando pretendemos calcular a cardinalidade de um fluxo de dados de grande dimensão sem termos a necessidade de guardarmos todos os valores. Isso permite resolver os problemas causados pelos dados em termos de recursos computacionais. Podemos aplicar este algoritmo, por exemplo quando pretendemos representar os endereços IP visitados pelos utilizadores de um dado site ou quando tencionamos calcular a cardinalidade de um dado conjunto.

O algoritmo *Hyper Log Log* caracteriza-se por usar estruturas de tamanho fixo, que, dependendo da forma como são implementadas, podem ocupar valores mais baixos do que 16KB. Como a medição da cardinalidade realizada por este algoritmo é probabilística, pode suceder a ocorrência de erros nos resultados na ordem dos 2%. Porém, podemos verificar que esta percentagem de erro não é muito grande para grandes volumes de dados. Vejamos o seguinte exemplo: se considerarmos uma *stream* de dados com 1 000 000 de registos únicos, um eventual erro de 2% significaria que, desses 1 000 000 registos, perderíamos 20 000. Para valores de registos desta grandeza, pensamos que uma taxa de erro como esta é comportável.

Após a apresentação da utilidade do algoritmo *Hyper Log Log*, vejamos agora como é que ele funciona a nível interno. Como já foi referido anteriormente, este algoritmo é responsável por determinar a cardinalidade de uma *stream* de dados (Fraguela and Bozkus, 2017). Com base na Figura 15, podemos verificar que essa stream de dados vai ser utilizada no fluxo de entrada de forma a ser processada por blocos no primeiro *loop* do algoritmo. Depois disso, o *loop* mais interno vai realizar o processamento de todos os itens de um dado bloco, aplicando-lhes uma função de *hash* numa primeira fase. O valor resultante do passo anterior em binário vai ser dividido em duas partes. A primeira corresponde a uma sequênciade *bits* que vai identificar o indice da matriz. Aos restantes *bits* vai ser aplicada uma função que calcula o número de zeros á esquerda e, em seguida, o intervalo é atualizado com este valor se for maior do que o valor que está na variável. Quando o processo termina este valor é usado para calcular a estimativa.

input: Input file *inputFile*
input: Number of estimator buckets N
data: Vector of N buckets $\overrightarrow{M} = (M_0, \ldots, M_{N-1})$
output: Estimation E of number of different items in the input file
$M_{\text{index}} = 0,\ 0 \leq \text{index} < N$
while *not at end of file inputFile* **do**
  dataChunk = `readChunk`(*inputFile*)
  foreach *item in dataChunk* **do**
    hash = `hashFunction`(*item*)
    [*remainder*, *index*] = `splitHash`(*hash*)
    nleadingZeros = `countLeadingZeros`(*remainder*)
    $M_{\text{index}} = \texttt{max}\left(M_{\text{index}}, nleadingZeros\right)$
  end
end

$$E = \text{Alpha} \times N^2 \times \left(\sum_{i=0}^{N-1} 2^{-M_i} \right)^{-1}$$

Figura 15: Pseudocódigo do algoritmo *Hyper Log Log* - Figura extraída de (Fraguela and Bozkus, 2017)

De forma a compreendermos melhor o seu funcionamento, vejamos um exemplo da sua aplicação. Suponhamos que queremos contar o número de lançamentos seguidos de uma moeda ao ar nos quais calhou 'Cara'. Se ocorrerem, no máximo, 3 lançamentos nos quais calhou 'Cara' podemos concluir que a moeda foi lançada poucas vezes. Porém, se nesse mesmo conjunto de lançamentos ocorrerem 10 vezes 'Cara' isso pode significar que a moeda foi lançada bastantes vezes.

Basicamente, o algoritmo *Hyper Log Log* utiliza o resultado da função de *hash* em vez de contar o número seguido de vezes em que ocorreu a situação 'Cara'. Um bom algoritmo de *hashing* garante-nos que qualquer posição da tabela tem a mesma probabilidade de ocorrência e a mesma distribuição uniforme. Isto permite ao algoritmo estimar a quantidade de elementos com base no resultado da função de *Hash* com maior número de '0' à esquerda. Consideremos uma função de *hashing* que, dado um certo valor, devolve uma representação binária de um número compreendido entre 0 e 15, por exemplo:

- 0000 – 0100 – 1000 – 1100
- 0001 – 0101 – 1001 – 1101
- 0010 – 0110 – 1010 – 1110
- 0011 – 0111 – 1011 – 1111

Olhando para os números anteriores verificamos que todos eles têm a mesma probabilidade de ocorrência. Por outro lado, consideremos o número de zeros à esquerda e a sua probabilidade presentes na representação binária de números no intervalo de 0 a 15:

- 0 -> 50% (1 em 2)
- 00 -> 25% (1 em 4)
- 000 -> 12,5% (1 em 8)
- 0000 -> 6,25% (1 em 16)

Pela análise das probabilidades acima referidas, verificamos que 1 em cada 8 elementos começa por '000'. Se generalizarmos, precisaríamos de 2ˆ k elementos para obtermos todos os possíveis k zeros à esquerda. Ou seja, para representarmos o intervalo de números de 0 a 15 precisaríamos de 2ˆ 4 combinações em que 4 seria o número máximo de zeros à esquerda que podemos ter seguidos. Desta forma, o algoritmo *Hyper Log Log* (HLL) na sua execução divide o resultado da *Hash* em 2, onde na primeira parte coloca a representação dos números em binário e na segunda coloca a contagem de zeros à esquerda.

Podemos observar aplicações deste algoritmo de uma forma mais concreta em aplicações realizadas pela Google e pela Reddit. A Google teve a necessidade de aplicar este algoritmo ao seu processo de consultas de informação. Isto deveu-se ao facto de ter a necessidade de lidar com mais de 1 bilião de utilizadores em serviços como o Android, o Chrome, o Gmail ou o YouTube. Por exemplo, se fosse necessário contabilizar o número de utilizadores únicos utilizando o processo *standard* (o processo utilizado até então), essa contabilização seria um processo lento, uma vez que seria necessário percorrer todos os identificadores dos utilizadores armazenados na base de dados da Google, que iria originar um elevado consumo de RAM. O processo de *Big Query* (Hoffa, 2017) implementado pela Google baseia-se na execução de *queries* com base numa aproximação ao resultado, com uma maior ou menor taxa de aproximação que é inversamente proporcional ao número de recursos utilizados. Para entendermos melhor o funcionamento da aplicação do *Big Query*, consideremos que pretendemos saber quantos utilizadores únicos teve o Github durante o ano de 2016. Para se conseguir obter essa informação efetuámos uma análise, e a respetiva comparação do tempo de execução, de uma *query standard* e outra baseada em aproximação.

Assim, se usássemos SQL *standard* (Figura 16) para o exemplo anterior, obteríamos a informação relativa a 6610026 utilizadores distintos (3,39GB processados) resultantes da análise de 320825029 registos que foram processados em 4,1 segundos.

```
#standardSQL
SELECT COUNT(DISTINCT actor.login) exact_cnt
FROM 'githubarchive.year.2016'
```

Figura 16: Exemplo de *query* usando *Standard* SQL

Usando SQL com base em aproximações (Figura 17), teríamos obtido 6643627 utilizadores distintos no total de 320825029 registos processados em 2,6 segundos.

```
#ApproximationSQL
SELECT APPROX_COUNT_DISTINCT( actor.login) approx_cnt
FROM 'githubarchive.year.2016'
```

Figura 17: Exemplo de *query* usando SQL baseado em aproximações

Assim, podemos verificar que para o caso de utilizarmos uma contagem aproximada obtivemos um melhor tempo de execução (quase metade de uma execução *standard*) e um resultado que foi calculado considerando uma taxa de erro 0,5%. Este resultado permite-nos ver a grande aplicação do processo *Big Query* em situações que envolvam um elevado número de dados.

Outro exemplo da aplicação do *Hyper Log Log* é a contagem do número de visualizações de um *post* por parte do Reddit (Chra, 2017) . Anteriormente, os votos e o número de comentários eram os principais indicadores de atividade de um determinado *post*. No entanto, o Reddit tem muitos utilizadores que visualizam um *post* sem votar ou comentar. Para poder capturar essa atividade, o Reddit desenvolveu um sistema para contabilizar o número de visualizações de *posts*. Depois, esse número é mostrado aos criadores e moderadores de um *post* para lhes fornecer uma visão mais concreta do alcance da sua publicação. Na Figura 18 podemos ver o número de visualizações de um *post*.

Figura 18: Exemplo de um *post* no Reddit - Figura adaptada de (Chra, 2017)

Para que seja possível contabilizar todos os utilizadores que viram um dado *post*, precisamos de ter uma lista dos utilizadores que o visitaram anteriormente e, de seguida, verificar se o utilizador atual está na lista anterior, no caso de não estar o número de utilizadores distintos que visualizaram o *post* vai ser incrementado, de outra forma irá permanecer igual. Porém, dado o elevado número de utilizadores, esta solução não é aplicável, porque não é eficiente. Como o fornecimento de contagens exatas é inviável, o Reddit utilizou um algoritmo de estimativa linear para realizar esse trabalho.

Uma abordagem baseada no algoritmo *Hyper Log Log*, não fornece tanta precisão como outros algoritmos, mas no entanto, é a alternativa mais eficiente. Por exemplo, a publicação da imagem que apresentámos na Figura 18 obteve mais de 1 milhão de utilizadores únicos. Se armazenássemos esta informação usando um algoritmo linear, considerando que um identificador de imagem tem 8 bytes, precisaríamos de cerca de 8 MB de memória para guardar todos os identificadores dos utilizadores únicos de um *post*. Por outro lado, se usássemos o *Hyper Log Log* para efetuar a contagem, usaríamos em média 12KB, o que representaria apenas 0,15% do espaço referido anteriormente.

3.4 O ALGORITMO *t-digest*

O algoritmo *Q-Digest* (Etrl and Dunning, 2015) foi desenvolvido para lidar com o consumo de memória e deteção de anomalias. Posteriormente, foi concebido o *T-Digest* com o objetivo de melhorar o desempenho do anterior. A ideia base deste algoritmo assenta em abdicar um pouco da precisão de cálculo de maneira a reduzir os recursos computacionais usados, uma vez que estes são limitados. O algoritmo *T-Digest* é usado quando temos como objetivo final encontrar anomalias numa amostra de dados, por exemplo, numa situação na qual é necessário fazer uma análise às medições de uma propriedade de um químico, de modo a evitar situações inconvenientes. Também o podemos aplicar no momento em que pretendemos classificar algo em função dos percentis, como é o caso de quando fornecemos resultados de testes padronizados ou quando medimos o desenvolvimento físico das crianças (Cormode et al.).

O *T-Digest* caracteriza-se por utilizar uma estrutura de dados com base probabilística para estimar a mediana (2º percentil) ou qualquer outro percentil de uma *stream* de dados contínua. Esta estrutura caracteriza-se por ser uma representação esparsa da função de distribuição acumulada. Após a inserção dos dados o algoritmo consegue determinar onde eventualmente vão estar os pontos "interessantes" a ter em conta para análise. Estes pontos são designados por centróides. Assim sendo, de seguida vamos explicar o fluxo de funcionamento das operações do algoritmo. Numa primeira fase, vamos supor que possuimos uma amostra de dados X = {x1,x3,...,xn}. De seguida, formamos o *t-Digest* a partir de uma sequência de pontos agrupados em intervalos X = {s1,s2,...,sn}, usando uma distribuição empírica. É importante referir que o algoritmo força os intervalos mais próximos das extremidades a serem mais pequenos, enquanto que os intervalos do meio são maiores para permitir uma maior precisão nos cálculos. Depois de agruparmos os dados em subconjuntos, podemos estimar os quartis usando a interpolação entre os pontos de cada intervalo. Desta forma, cada coluna tem associada a si um centróide que contém a média e a soma dos valores do intervalo, que são utilizados quando pretendemos determinar um quartil ou percentil.

Com base no excerto de código da Figura 19 podemos visualizar o fluxo de processamento do algoritmo para o cáculo dos percentis.

```
# inicializar um novo t-Digest
digest = TDigest()

# adicionar valores ao t-Digest
digest.update('1')
digest.update('1')
digest.update('2')
digest.update('3')
digest.update('8')
digest.update('5')
digest.update('5')
digest.update('4')
digest.update('3')
digest.update('2')
digest.update('2')
digest.update('5')
digest.update('4')

#Calculo dos percentis
digest.percentile(50)

#e o 15 percentil da distribuicao uniforme (0,1)
digest.percentile(15)
```

Figura 19: Exemplo de utilização do algoritmo *T-Digest*

Vejamos agora um caso de aplicação do algoritmo *T-Digest* (Davidson-Pilon, 2015). Consideremos uma distribuição normal (empírica) aleatória de um conjunto de dados qualquer (Figura 20).

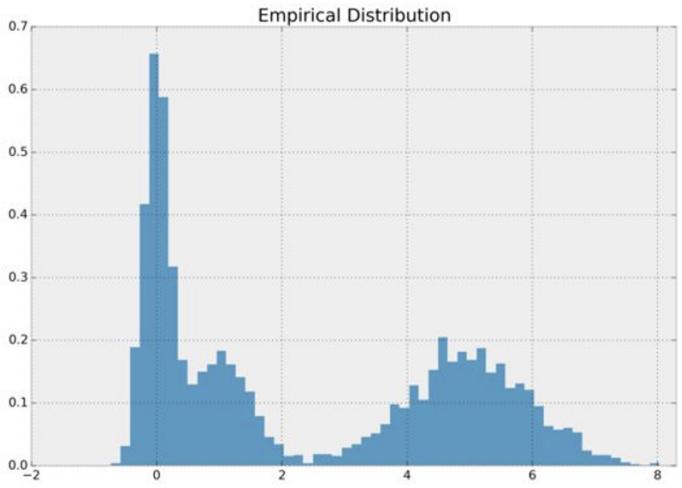


Figura 20: Exemplo de uma Distribuição Empírica - Figura extraída de (Davidson-Pilon, 2015)

Com base na distribuição empírica anterior, numa fase inicial do algoritmo geramos uma distribuição probabilística acumulativa (Figura 21).

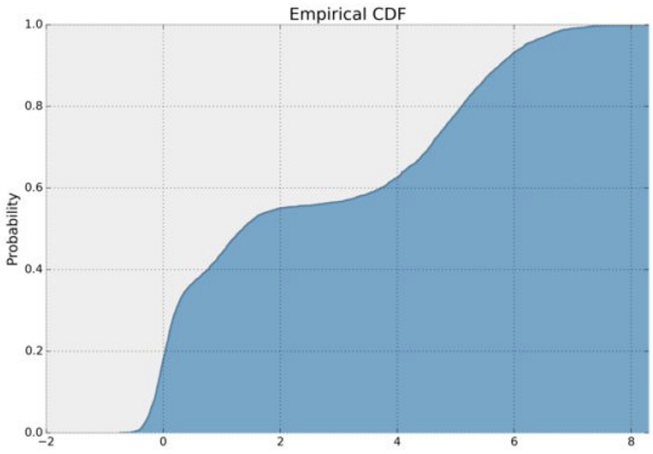


Figura 21: Exemplo de uma Distribuição probabilística - Figura extraída de (Davidson-Pilon, 2015)

De seguida, o algoritmo *T-Digest* irá processar o resultado da função probabilística acumulativa e tentará representar os pontos "interessantes" da distribuição. Essa representação pode ser observada na Figura 22.

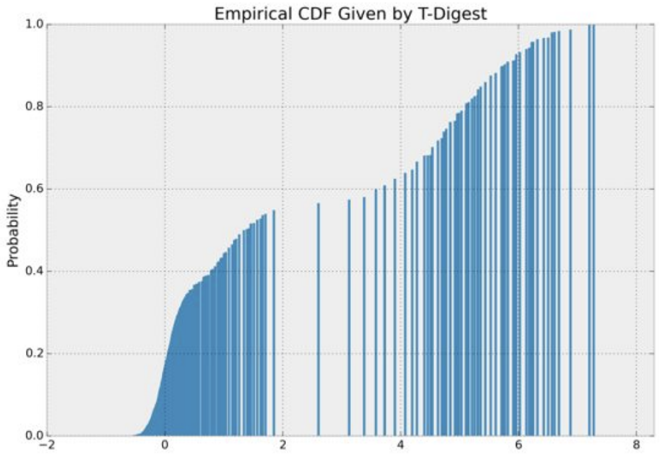


Figura 22: Exemplo de uma Distribuição Acumulativa - Figura extraída de (Davidson-Pilon, 2015)

As linhas verticais que se podem visualizar na Figura 22, representam os pontos onde o algoritmo "pensa" os centróides como sendo interessantes e relevantes para o caso em estudo. Tendo em conta a Figura 20, podemos analisar que os centróides representam as partes da função de distribuição acumulativa onde esta varia mais rapidamente. Por sua vez, nestes pontos as quantidades como o percentil também mudam mais rápido. Por exemplo, para os valores próximos de x=3 e x=8 podemos observar que estes podem ser resumidos apenas por alguns pontos, fazendo com que estes depois do processo de compressão não aparecem devido á sua baixa frequência. Em função disso, concluímos com maior exatidão, com maior eficiência e com menor margem de erro no cálculo dos valores ideais para cada percentil conforme o caso de estudo previamente definido.

O *T-Digest* costuma ser muito aplicado em casos em que o foco principal é a deteção de anomalias em fluxos de dados. Este processo, consiste em identificar em tempo real eventos incomuns ou suspeitos em fluxos de dados contínuos, permitindo assim reagir proactivamente, economizando os recursos envolvidos. De uma forma geral, os valores anómalos referem-se a algum tipo de problema, como, por exemplo, uma fraude bancária, problemas médicos ou defeitos estruturais. Para podermos adaptar o algoritmo *T-Digest* à deteção de anomalias, é essencial compreendermos os diferentes tipos de anomalias que podem ser estudadas. Segundo Banerjee (Banerjee et al., 2009), as anomalias podem-se dividir em 3 categorias:

1. Anomalias Pontuais - Este tipo de anomalias ocorre quando um dado fluxo de dados é considerado anómalo em relação aos restantes dados. Este é o tipo mais simples de anomalia, sendo aquela que mais processos de deteção teve implementados. Um exemplo de aplicação de um processo de deteção deste tipo de anomalia pode ser a deteção de fraudes em transações de *e-commerce*, na qual se assume a característica valor da transação para detetar se uma transação com um valor muito acima da média de gastos de uma pessoa pode ser uma anomalia pontual.

2. Anomalias de Contexto – Podendo ser reconhecida como uma anomalia condicional, este tipo de anomalias caracteriza-se pelo fato de algo ser anómalo dentro de um dado contexto. Um exemplo deste tipo de anomalias é o número de sessões do Google Analytics num dado intervalo de tempo. Dois dias podem ter o mesmo volume de sessões, mas o contexto de um deles pode ser uma anomalia. Se obtivermos, um pico de sessões numa *Black Friday* não consideramos como sendo uma anomalia, mas um pico numa sexta feira dita normal já é considerado como uma anomalia.

3. Anomalias Coletivas – Esta anomalia caracteriza-se por conter um conjunto de dados anómalos. Um exemplo desta anomalia ocorre, por exemplo, com a queda repentina de vendas de um produto durante um dado intervalo de tempo. É normal termos horas do dia em que as vendas são baixas, por exemplo, durante a madrugada, mas quando esses valores permanecem baixos a outras horas do dia, isso pode ser indicativo de uma anomalia.

Estes diferentes tipos de anomalias podem ser aplicados em vários domínios como, por exemplo, na deteção da presença de intrusos, fraudes, ou falhas e monitorização da integridade de um sistema.

3.5 COMPARAÇÃO DAS PROPRIEDADES DOS ALGORITMOS

Na Tabela 1 apresentamos um resumo das principais propriedades dos algoritmos de *sketching* que estudámos neste capítulo. Desta forma, podemos fazer uma breve comparação entre esses algoritmos, de modo a conseguirmos diferenciá-los objetivamente e sabermos escolher o mais adequado para uma dada aplicação.

	T-Digest	*Count-Min Sketch*	*Hyper Log Log*
Objetivo	Estimar quartis	Cálculo de frequências	Determinar cardinalidade
Tipo Algoritmo	Probabilístico	Probabilístico	Probabilístico
% de Erro	0.5 %	2 * n / w	2 %
Estrutura	Função esparsa acumulativa	Matriz na qual as colunas são funções de *Hash*	Função de *Hash*
Memória Usada	1.5 kb	Linhas * Colunas	16 kb
Prós	Não requer alocação dinâmica	Bom tempo de execução para um volume elevado de dados	
Contras		Em caso de erro, este algoritmo só origina valores superiores ao real	Uma má escolha da função de *Hash* provoca maus resultados

Tabela 1: Comparação das propriedades dos algoritmos estudados

4

CASO DE ESTUDO

4.1 CONTEXTUALIZAÇÃO

A evolução contínua da tecnologia fez com que todos os setores mudassem os seus paradigmas, usando para isso a produção sistemática de dados a seu favor e fazendo com que o processamento e análise de *streams* de dados se tornasse num tema com elevada pertinência no seio das empresas. Praticamente, todas as empresas com uma atividade regular produzem grandes volumes de dados, cuja análise é essencial, muitas das vezes num tempo muito próximo do real. Quando conseguida, essa análise traz inúmeras vantagens às empresas aos mais diversos níveis tais como, por exemplo, na redução dos custos operacionais ou na descoberta de falhas numa linha produtiva.

Hoje, a produção desses dados acontece em larga escala e de forma contínua, originando vários problemas na forma de como fazer o seu processamento, especialmente em termos dos recursos computacionais necessários para o realizar. Tais problemas são bastante comuns, praticamente em muitos sistemas operacionais empresariais, uma vez que as empresas não possuem a capacidade computacional necessária para processar tão elevado fluxo de dados em tempo útil. São situações complicadas que afetam muitas empresas, principalmente aquelas que pretendem tratar as *streams* de dados que produzem, de forma contínua, para conseguir selecionar os dados mais pertinentes.

A título ilustrativo, consideremos uma empresa cuja atividade permite gerar um elevado volume de negócios. De acordo com a natureza dos seus negócios, esta empresa pode ter que lidar com diferentes tipos de dados, oriundos das suas diversas atividades, como, por exemplo: vendas, gestão de clientes, gestão interna, gestão de preços, etc. Assim sendo, é praticamente impossível para um dado conjunto de pessoas identificar padrões nestes dados tão diversos, especialmente quando aparecem em grandes volumes. Para este tipo de casos é fácil reconhecer a importância e a utilidade que os processos de extração de conhecimento podem ter na análise desses dados. Através da aplicação de regras, bem estabelecidas e integradas num sistema específico de análise, é possível encontrar padrões frequentes (e invulgares, também), que revelem a ocorrência de situações particulares, como, por exemplo, horas de pico em redes de distribuição, procura excessiva de bens em sistemas de supermercados, nível de satisfação de um cliente num sistema de *streaming*, etc.

Nesta dissertação, abordamos este tipo de problemática através de um caso de estudo específico, tratando e analisando dados de forma eficiente, em tempo próximo do real, utilizando técnicas de *sketching*, para tirarmos algumas conclusões a partir das anomalias encontradas nos processos de análise que realizarmos.

4.2 APRESENTAÇÃO DO CASO

Dia após dia, a quantidade de dispositivos que estão ligados à *Internet* tem vindo a aumentar exponencialmente, contribuindo para o constante desenvolvimento do termo *Internet of Things* (IoT) (Chui). Em inúmeros aspetos, a IoT veio facilitar o dia a dia de uma pessoa em muitas áreas de trabalho, como sejam, por exemplo, a automação de sistemas, a assistência médica, os sistemas de transporte. No controlo e gestão de tais sistemas, podem ser usados diversos tipos de sensores, para a aquisição de dados em tempo real, se necessário, que têm a capacidade de angariar diferentes tipos de dados, que, dependendo do sistema instalado, poderão recolher elementos de dados bastante pertinentes, relativos, por exemplo, à localização de dispositivos, à temperatura ambiente ou à operação de sistemas operacionais, coletando esses dados e armazenando-os em *logs* de eventos específicas (nControl). Tais dispositivos de controlo costumam ter incorporados no seu sistema um interface de rede, que tem a capacidade de comunicar com outros dispositivos de modo a conseguirem prestar serviços. Esta capacidade de comunicação garante grande eficiência no processo de análise de grandes fluxos de dados. Muitos destes dispositivos são usados em cidades inteligentes, em sistemas inteligentes, no setor dos transportes, em medidores de energia, em dispositivos remotos de monotorização na saúde e em sistemas de controlo de qualidade.

Inspirados em alguns desses sistemas, idealizámos e desenvolvemos um caso de estudo relativo a um sistema de controlo da qualidade da água, que assumimos que utilizaria uma série de sensores que, de forma contínua, faz a recolha de valores relativos a um conjunto de parâmetros usados regularmente para fazer a avaliação da qualidade da água num dado sistema de abastecimento. Os valores recolhidos por este sistema de sensores, ao longo do tempo, constituem uma *stream* de dados, cuja análise permitirá fazer a monitorização da qualidade da água e identificar vários tipos de situações, nomeadamente, água com valores de *potencial hidrogeniónico* (pH) elevado, oxigénio dissolvido elevado ou excesso de condutividade, entre outras coisas. Ao podermos identificar situações como esta, quase em tempo real, podemos minimizar as consequências que daí possam advir, pois, assim, seremos capazes de estudar soluções para os problemas detetados e avisar as pessoas sobre o nível da qualidade da água num determinado momento podendo prevenir, ou mesmo evitar, situações desagradáveis. Assumimos, também, que os dados são recolhidos e guardados em formato JSON. Para podermos caracterizar e tratar as situações anómalas na qualidade da água analisada, tivemos que realizar um estudo prévio para identificarmos quais os parâmetros para avaliação da qualidade da água, bem como os seus valores padrão, que a podem influenciar de forma positiva ou negativa.

4.3 TECNOLOGIAS UTILIZADAS

Na realização dos trabalhos desta dissertação fomos utilizando diferentes tecnologias ao longo do desenvolvimento do nosso laboratório de teste de algoritmos de *sketching*. Todas as tecnologias que adotámos estão sob licenças *open source*. Em termos gerais, o laboratório foi implementado em *Python*, incluindo os módulos dos diferentes algoritmos de *sketching* e os módulos de comunicação entre os clientes e o servidor. Estes últimos, utilizaram a biblioteca *sockets*. Os dados que foram sintetizados e que servem de base de trabalho ao sistema de processamento e análise de *streams* estão em formato JSON. O valor ideal de cada um dos parâmetros de análise com que trabalhámos teve como base os valores de referência emitidos por decreto-lei pelo Estado Português (Estado Português, 2007). A comunicação entre o cliente e o servidor teve uma atenção especial da nossa parte, uma vez que teve um papel muito importante no nosso sistema. Na Tabela 2 podemos ver uma breve descrição de cada uma das tecnologias que foram utilizadas na conceção e implementação do nosso sistema de processamento e análise de *streams* de dados.

Tecnologia	Descrição
Python 3	O laboratório foi implementado usando a linguagem *Python*. Os processos de criação dos dados, a comunicação entre cliente e servidor, a implementação dos algoritmos e a análise dos resultados foram totalmente desenvolvidos usando esta linguagem de programação.
JSON	Formato utilizado para armazenamento dos dados utilizados nos testes de *sketching* realizados.
Selectors	É um modulo *Python*, que nos permite saber o que fazer em caso de escrita ou de leitura, tornando a conexão mais simples.

Tabela 2: Tecnologias utilizadas no desenvolvimento do laboratório de *sketching*

4.4 A ARQUITETURA DO SISTEMA

No processo de implementação do laboratório de teste para algoritmos de *sketching* definimos uma arquitetura específica (Figura 23), que fosse capaz de acolher as diversas tarefas que são realizadas no seu ambiente. A arquitetura do sistema está organizada em três módulos distintos:

1. **Modelação dos dados**, é a etapa responsável por gerar os dados que vão servir de *input* para o laboratório. Estes são gerados seguindo uma distribuição normal baseada nos valores recomendados pela *Direção Geral da Saúde* (DGS).

2. **Processamento de dados**, No qual os dados gerados no modulo anterior são distribuídos de forma eficiente pelos clientes e posteriormente, são processados através da aplicação de um algoritmo de *sketching*. No final desta etapa, os clientes juntam todos os resultados para serem analisados no próximo módulo.

3. **Análise de resultados**, é onde o resultado final vai ser analisado e apresentado num tempo muito próximo do real.

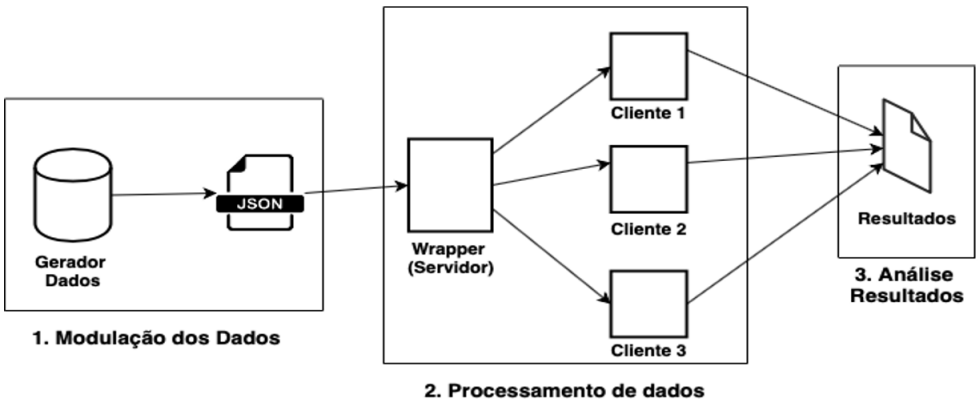


Figura 23: Arquitetura do sistema de *sketching*

Vejamos, agora, com mais detalhe cada um destes módulos. Antes de procedermos ao desenvolvimento do módulo de modulação de dados, primeiramente, tivemos que encontrar uma solução que garantisse a fiabilidade dos dados a gerar, de modo a que o resultado final fosse o mais fidedigno possível. Assim, para garantirmos a veracidade dos dados, realizámos um estudo prévio, com o intuito de saber quais eram as gamas dos valores

ótimos e dos valores discrepantes para todos os parâmetros que iríamos tratar e analisar no nosso caso de aplicação, para teste dos algoritmos de *sketching* que selecionámos. Esse estudo teve como base os valores ideais definidos pela DGS em conjunto com o Estado Português (Seção 4.5).

O processo de ingestão de dados num sistema de processamento de *streams* em tempo real não consiste apenas em reunir ou selecionar um conjunto de dados que vão ser processados, mas também implica a realização de uma dada formatação por causa da diversidade de elementos que podemos encontrar nesses dados. Basicamente, a formatação de dados consiste numa uniformização do formato dos dados, tanto em termos de domínios como de valores e na eliminação de dados que estejam corrompidos ou que não sejam necessários para o caso de estudo. Com a formatação de dados realizada, o seu processamento será mais simples.

No sentido de possuirmos uma melhor organização nos dados e uma estruturação padronizada decidimos guardá-los em formato JSON. Isso facilitou bastante a realização de um *parser* específico para analisar e preparar os dados que queríamos trabalhar no servidor do sistema. Na Figura 24 podemos ver um pequeno exemplo de um elemento de dados no formato adotado.


```
{
"condutividade" : 1.94,
"id": 4,
"oxigenio dissolvido": 8.25,
"pH": 7.73,
"posicao": 4,
"temperatura": 16.4,
"timestamp": "2019-09-24 15:10:25",
"turbidez": 4.09
}
```


Figura 24: Exemplo de um elemento de dados em formato JSON

O segundo módulo da arquitetura – Processamento de Dados – do nosso sistema de *sketching*, o mais importante do sistema, no nosso ponto de vista, acolhe as tarefas de processamento de *streams* de dados em tempo real. É aqui que a informação presente nos dados será processada, através da utilização de um algoritmo de *sketching*. Neste módulo, os dados são recolhidos pelo *Wrapper* (Servidor), que tem a responsabilidade de distribuir esses dados por um conjunto de clientes, de modo a que cada cliente livre ou com menor taxa de ocupação tenha sempre prioridade em termos de acolhimento de dados. Os dados são enviados em intervalos de 1 segundo. O servidor tem, também, a responsabilidade de fazer a seleção dos algoritmos a usar em cada momento e de passar essa informação a cada um dos clientes. Por sua vez, os clientes aplicam o algoritmo selecionado aos dados recebidos e, no final, juntam toda a informação obtida para que esta possa ser posteriormente analisada.

Depois de processada, a informação será analisada no módulo seguinte – Análise de Resultados – e os resultados serão apresentados quase em tempo real. Ao analisarmos esta informação, proveniente da fase anterior, podemos prevenir ou evitar situações anómalas numa rede de distribuição de água como por exemplo, quando encontramos uma sequência de valores de pH muito baixos. Indicando-nos que a água tem um grau de acidez alto e assim sendo está imprópria para consumo humano.

4.5 GAMAS DE VALORES PADRÃO

Para podermos detetar anomalias nos valores recebidos em tempo real tivemos que realizar um estudo prévio para identificarmos a gama de valores para quais se verificam situações anómalas. De seguida, enunciamos as diretivas que forneceram os valores padrão que foram utilizados no nosso caso de estudo e que foram extraídas do Decreto-Lei nº 306/2007 de 27 de Agosto (Estado Português, 2007). Como base nesse decreto, sabemos que conhecer:

- O pH permite minimizar os fenómenos de corrosão nos sistemas de abastecimento.
- A condutividade na água, que, quando o valor é excessivo, pode originar a formação de depósitos ou causar um sabor desagradável.
- A temperatura influencia uma série de parâmetros, que podem ter um impacto sobre a corrosão, o nível de resíduos, a atividade microbiana e os fluxos de água; o aumento da temperatura contribui para a precipitação de sais dissolvidos, como é o caso dos carbonatos e bicarbonatos.
- A turbidez presente na água permite identificar os “pontos mortos” dos cursos de água para que sejam removidos com a renovação da água.
- O valor paramétrico para o oxigénio dissolvido (5 mg/L), cuja análise permite-nos estimar a poluição orgânica e, como tal, pode ser útil como identificador da eficiência dos tratamentos oxidativos, tais como a adição de cloro ou ozono.

Na Tabela 3 podemos ver a gama de valores considerados ótimos para os diversos parâmetros que acabámos de apresentar. Todos os valores apresentados foram retirados de normas da DGS, como sendo os valores adequados que cada um dos parâmetros indicados deveriam apresentar para o caso particular de uma rede de abastecimento de água. Ao nos guiarmos por estes valores, definidos por uma entidade certificada, conseguimos garantir uma maior veracidade aos nossos próprios dados, uma vez que assim poderemos retirar do

nosso trabalho conclusões mais adequadas, com base em cenários reais, que eventualmente no futuro poderão acontecer.

Parâmetros	Gama de Valores
pH	Valores entre 6,5 a 9.
Oxigénio Dissolvido	Valores próximos de 5 mg/L (valor ideal tabelado).
Temperatura	Valores entre os 15ºC e os 23ºC.
Condutividade	Valores próximos de 2,5 S/cm a 20ºC (valor ideal tabelado).
Turbidez	Valores próximos de 4 UNT.

Tabela 3: Valores padrão definidos para uma rede de abastecimento de água

4.6 O *workflow* DE TRABALHO

Depois de explicarmos a arquitetura utilizada, é essencial perceber-se como é que os dados do sistema são trabalhados ao longo das diferentes etapas até ao momento no qual são apresentados os resultados. As etapas realizadas ao longo do nosso *workflow* de trabalho (Figura 25) são as seguintes:

1. **Geração dos dados** - etapa na qual são gerados os dados para análise de *sketchs*, tendo em conta os valores padrão estudados previamente.

2. **Seleção do algoritmo** - etapa na qual o servidor do sistema de análise indica a todos os clientes qual vai ser o algoritmo de *sketching* que devem utilizar durante um determinado processo de análise.

3. ***Parsing* dos dados** - nesta etapa o servidor do sistema recebe como *input* o ficheiro JSON gerado anteriormente (Etapa 1), guardando os dados numa estrutura que vai ser utilizada conjuntamente com uma *queue* para os distribuir de forma eficiente por todos os clientes.

4. **Distribuição dos dados** - a distribuição dos dados é feita através de algoritmos de filas de espera, de modo a tornar este processo o mais eficiente possível – os clientes que estejam livres ou com uma baixa taxa de sobrecarga têm sempre prioridade perante os outros.

5. **Aplicação do algoritmo** - para cada linha de dados recebida, o algoritmo de *sketching* escolhido guarda-a na sua estrutura para que, no fim da execução, possa juntar todos esses dados e fazer a aplicação do algoritmo.

6. **Análise de resultados** - após a receção dos resultados produzidos pelos algoritmos de *sketching* estes são analisados para podermos tirar as conclusões sobre o caso que temos em estudo.

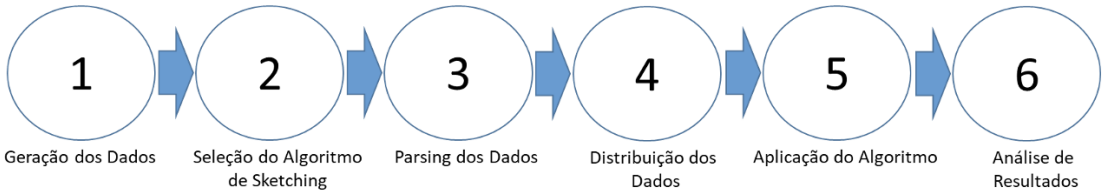


Figura 25: Ilustração do *workflow* de trabalho

5

TESTE E ANÁLISE DOS ALGORITMOS

5.1 CENÁRIOS E AMBIENTE DE TESTE

Para avaliarmos o desempenho de cada um dos algoritmos que estudámos, identificámos e preparámos cinco cenários de teste, cada um deles orientado para a análise de uma propriedade específica da qualidade da água. Com esse objetivo, identificámos cinco propriedades de qualidade da água, para suportar, cada uma delas, um teste específico para a aplicação e estudo do comportamento dos algoritmos *T-Digest* (T-Dig), *Count-Min Sketch* (CMS) e *Hyper Log Log* (HLL). Assim, criámos e implementámos os seguintes cenários de teste:

- Cenário 1 - Controlo do Valor do pH da Água.
- Cenário 2 - Análise da Concentração de Oxigénio Dissolvido na Água.
- Cenário 3 - Análise da Radiação Solar Incidente.
- Cenário 4 - Análise da Condutividade da Água.
- Cenário 5 - Análise da Turbidez.

Para cada um dos cenários apresentados executámos um conjunto fixo de testes, que foram realizados três vezes. Todos os testes foram realizados na mesma máquina com as seguintes características:

- Placa Gráfica: *Intel Iris Plus Graphics* 645 1536MB
- Processador 1.4 GHz *Intel Core* i6
- Memória RAM 8 GB
- Disco SSD de 256GB

Esses testes consistem na aplicação dos três algoritmos em estudo em cinco cenários distintos, envolvendo um conjunto de 500 registos de dados, que são enviados e distribuídos de forma continua por 8 clientes, em intervalos de 1 segundo.

Com base no diagrama ilustrado na Figura 27, podemos ter uma melhor perceção de todas as fases no processo de testes, sendo que este pode ser dividido em duas fases distintas. Numa primeira fase, tivemos que definir uma fonte de dados de forma a garantirmos um maior grau de veracidade nos resultados gerados. Essa fonte de dados, serviu como *input* para a realização dos testes, tendo como base os valores ideais para cada parâmetro seguindo uma distribuição normal no momento em que os valores foram gerados. Na Figura 26, temos um exemplo da estrutura que cada um dos 500 registos possui, registos esses que se caracterizam por serem um conjunto de parâmetros.


```
{
"condutividade" : 1.94,
"id": 4,
"oxigenio dissolvido": 8.25,
"pH": 7.73,
"posicao": 4,
"temperatura": 16.4,
"timestamp": "2019-09-24 15:10:25",
"turbidez": 4.09
}
```


Figura 26: Exemplo de registo de *input*

Na segunda fase, os registos são enviados para o laboratório de dados de forma continua em intervalos de 1 segundo. É no decorrer desta fase que é medido o tempo de execução para cada algoritmo em todos os cenários. A medição ocorre no intervalo de tempo, em que o primeiro registo entra no laboratório até todos os registos tiverem sido processados. Estes são distribuídos de forma eficiente por todos os clientes, onde vai ser aplicado em cada um o algoritmo em estudo. Por fim, todos os resultados anteriores vão ser compactados para dar origem ao resultado final.

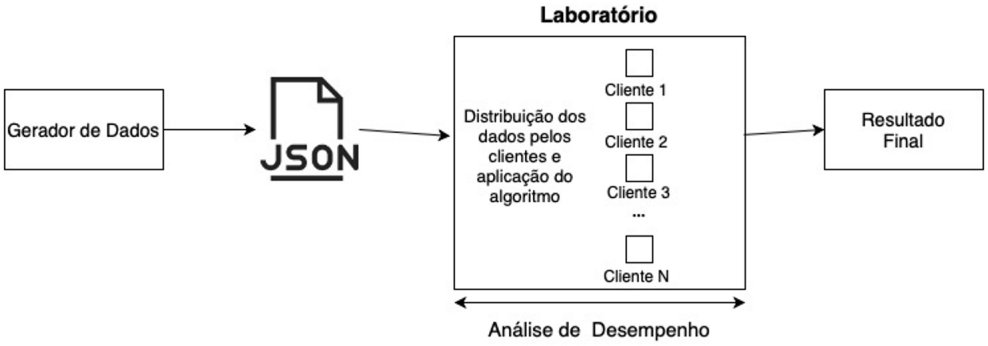


Figura 27: Cenário de testes

De seguida, vamos analisar o desempenho de cada um dos algoritmos em cada um destes cenários de teste, tendo particular atenção à velocidade de processamento de cada um deles. Para cada cenário, usamos o mesmo *input* com 500 registos e para cada um desses selecionamos o parâmetro mais adequado para o caso em estudo. Para que pudéssemos garantir uma maior fiabilidade aos cenários de teste, previamente, realizámos 5 execuções para cada caso e consideramos o valor médio do tempo de execução de cada algoritmo como o valor ideal a ter em conta para a elaboração dos diversos gráficos que produzimos. O uso do mesmo *input* e da mesma máquina na realização dos testes serviu para minimizar as interferências externas na análise do desempenho dos algoritmos.

Apesar de analisarmos o desempenho dos três algoritmos, eles são aplicados com finalidade distinta sobre o conjunto de dados em estudo. O *Count Min Sketch*, é utilizado quando pretendemos determinar a frequência de um parâmetro com um valor específico, por exemplo, no caso de querermos precisar o número de vezes em que o pH da água toma o valor 7 (sete). Por sua vez, o *Hyper Log Log* é executado em situações que tencionamos estimar a cardinalidade, ou seja, verificar o número de elementos distintos relativos a um parâmetro. Por último, o *t-Digest* pode ser aplicado em casos em que pretendemos balizar os quartis, por exemplo, ao determinar em que gama de valores se encontra o primeiro quartil.

5.2 Cenário 1 - Controlo do Valor do pH da Água

Neste primeiro cenário realizámos o controlo do valor do pH da água de forma contínua. pH significa Potencial Hidrogeniónico, que é a representação por meio de uma escala logarítmica do grau de acidez, neutralidade ou alcalinidade de uma dada solução, que neste caso é a água. A escala possui valores entre 0 (zero) e 14 (catorze), sendo que o 7 (sete) representa o valor neutro. O valor 0 (zero) representa a acidez máxima e o valor 14 a alcalinidade máxima. Ao efetuarmos o controlo do pH sobre um dado volume de água podemos garantir e identificar os momentos nos quais a água tem a qualidade requerida para poder ser consumida. Em Portugal, o Decreto-lei n.º 306/2007 (Estado Português, 2007) prevê que a água para consumo humano é considerada de boa qualidade quando o seu valor de pH se situa entre os 6,5 e os 9 (e idealmente entre os 7 e os 8), podendo ser classificada como “ligeiramente ácida”, “neutra” ou “alcalina” de acordo com esses valores. O último relatório da Entidade Reguladora dos Serviços de Águas e Resíduos (ERSAR), referente a 2015, afirma que 99% da água para consumo humano a nível nacional é de boa qualidade, pois preenche os limites dos parâmetros considerados no referido Decreto-Lei. Sendo assim, quando estas condições estiverem reunidas podemos beber água da torneira de forma segura.

No gráfico apresentado na Figura 28, podemos ver o resultado dos testes realizados ao desempenho do três algoritmos em estudo para o primeiro cenário de teste. Estes resultados foram obtidos através do valor médio de cinco execuções da ferramenta desenvolvida para o processamento de dados.

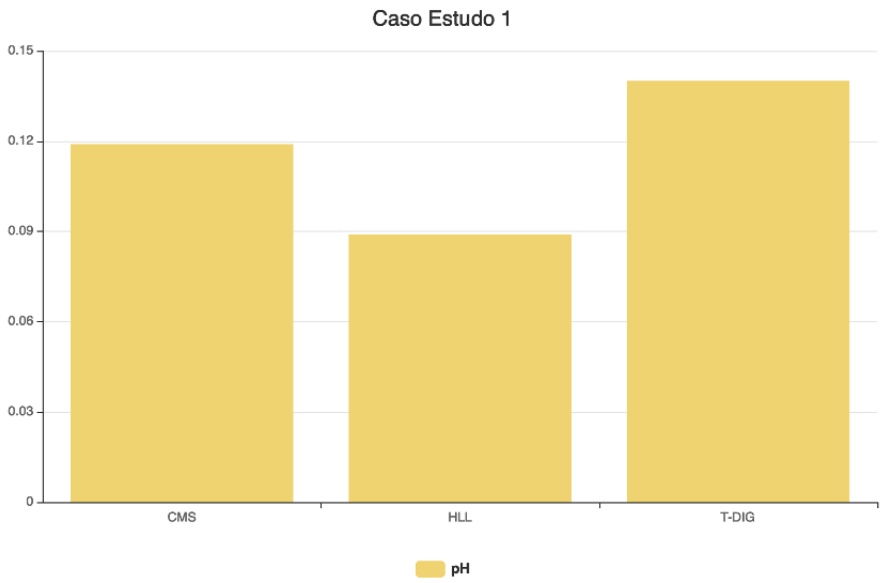


Figura 28: Resultados do Cenário 1 - Controlo do Valor do pH da Água

Neste primeiro cenário de teste, para determinarmos os valores do pH anómalos, usámos três formas diferentes de deteção de uma irregularidade, em que cada corresponde a uma implementação de um algoritmo. Por exemplo, no caso do CMS verificamos que um dado valor de pH tomava inúmeras vezes o mesmo valor atípico, ou no caso do HLL quando verificámos uma cardinalidade elevada significava que os valores do pH tomavam valores demasiado oscilantes, o que é indicativo de que estamos perante uma anomalia, ou no caso do *T-Digest* pudemos verificar quais eram os valores médios para o pH, bem como determinar a gama de valores nos quais estes se enquadravam e verificar se estes se encontram ou não numa zona crítica. Pela descrição da aplicação, no caso do uso dos algoritmos para o estudo do pH, constatámos que podemos aplicar qualquer um deles para determinarmos anomalias. A forma de como estas são detetadas varia de algoritmo para algoritmo devido à objetividade de cada um. Assim sendo, tivemos apenas em conta o tempo de execução para a análise de eficiência que efetuamos. Pela análise da Figura 28, conseguimos verificar que a implementação do algoritmo HLL é a mais eficiente, no que diz respeito ao tempo de execução dos algoritmos. Por isso, podemos concluir que para este primeiro cenário o algoritmo mais eficiente para o estudo da variância dos valores do pH é o HLL.

5.3 CENÁRIO 2 – ANÁLISE DA CONCENTRAÇÃO DE OXIGÉNIO DISSOLVIDO NA ÁGUA

A *concentração de oxigénio dissolvido* (COD) na água é um parâmetro muito importante para analisar as características químicas e biológicas da água (Matos, 2008a). No meio ambiente, o *oxigénio dissolvido* (OD) é originado por processos de fotossíntese aquática ou pela propagação do oxigénio que está presente na superfície da água. A COD pode variar por causa de vários aspetos, nomeadamente:

- Temperatura.

  A capacidade de dissolução do oxigénio na água diminui com o aumento da temperatura. Portanto, águas quentes retém menos oxigénio do que as águas frias.

- Salinidade.

  Quanto maior o nível de sal dissolvido na água, menor será o nível do OD. Assim sendo, podemos afirmar que a água do mar contém menos OD do que "outras" águas.

- Pressão.

  A solubilidade do oxigénio, é diretamente proporcional à pressão, ou seja, quanto maior o valor da pressão, maior será a solubilidade do oxigénio na água.

O OD é essencial para a sobrevivência das espécies aquáticas, uma vez que promove a respiração branquial dos peixes. No entanto, em locais poluídos pelo despejo de esgotos domésticos e industriais em rios e lagos, provoca o aparecimento em excesso de matéria orgânica o que origina uma grande diminuição do COD. Ao efetuarmos um controlo regular sobre este parâmetro, conseguimos identificar quanto a água está poluída e, assim, conseguimos evitar situações desagradáveis em tempo real. Através da análise do gráfico apresentado na Figura 29, podemos comparar os diferentes tempos de execução dos algoritmos tendo em conta a aplicação específica de cada um sobre os valores do oxigénio dissolvido. Os resultados para cada algoritmo foram obtidos tendo em conta o tempo médio de cinco execuções, de forma a amenizar os intervenientes externos aquando a obtenção dos resultados.

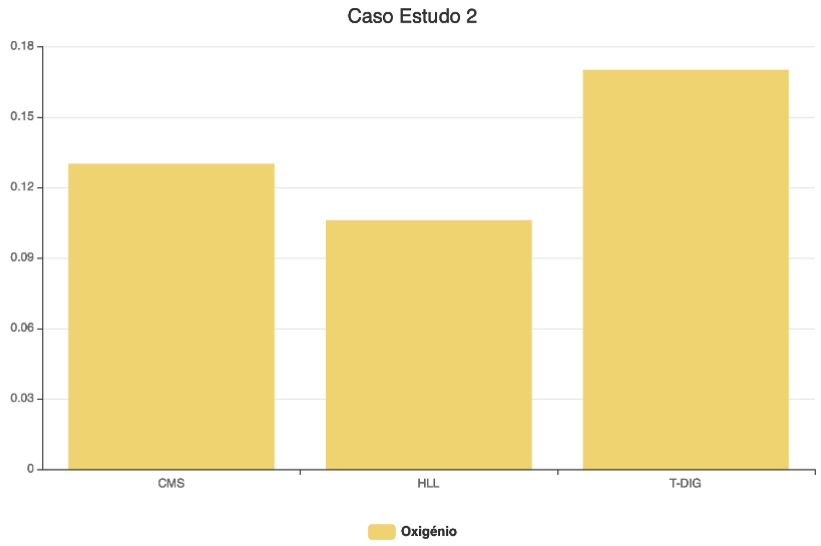


Figura 29: Resultados do Cenário 2 – Análise da Concentração de Oxigénio Dissolvido na Água

Neste segundo cenário, efetuámos uma análise á concentração de oxigénio dissolvido na água, de forma a identificarmos facilmente quais seriam os valores prejudiciais para o meio ambiente. Ao aplicar qualquer um dos três algoritmos estudados, conseguimos identificar quais eram os valores incomuns para o oxigénio dissolvido na água. No entanto, apesar dos algoritmos serem aplicados com o mesmo objetivo, este é obtido de maneira diferente devido à diversidade de cada algoritmo. No caso do CMS calculámos o número de vezes que o parâmetro teve um certo valor específico de 5 mg/L, enquanto que para o HLL verificámos a cardinalidade do parâmetro oxigénio dissolvido. Neste caso, uma baixa cardinalidade enquadrada num intervalo ideal significa que os valores estão dentro da normalidade. Por último, no caso do *T-Digest*, pudémos calcular o valor médio do oxigénio dissolvido e analisar o desvio padrão deste com o valor considerado ideal. Tendo em conta a Figura 29, verificámos que o algoritmo HLL apresentou um menor tempo de execução. Como tal, considerámo-lo como o melhor algoritmo a aplicar neste caso de estudo.

5.4 CENÁRIO 3 – ANÁLISE DA RADIAÇÃO SOLAR INCIDENTE

O nível da radiação solar incidente indica-nos qual é a temperatura da água, exceto em estações termoelétricas ou quando usamos fontes de calor ou de arrefecimento para variarmos a temperatura. A temperatura da água é o parâmetro que está relacionado direta ou indiretamente com os outros parâmetros em estudo, já que esta consegue influenciar os valores de cada um desses parâmetros. Por exemplo, se a temperatura da água for elevada, o oxigénio dissolvido diminui. A água usada para consumo próprio deve apresentar, geralmente, uma temperatura idêntica à temperatura ambiente, uma vez que não existe uma gama de valores tabelada que regule

esse parâmetro. Mas, no entanto, as águas frias são as recomendadas para consumo, já que as águas quentes possuem dificuldades em manter os níveis de cloro, originando, assim, o aparecimento de micróbios, que podem causar problemas relacionados com a cor, o sabor ou o odor da água.

Através da Figura 30, podemos analisar e comparar os diferentes tempos de execução dos algoritmos no cenário de teste relativo ao nível da radiação solar incidente, ou seja, sobre a temperatura da água. Optamos por uma representação gráfica dos resultados, com a intenção de tornar o processo de análise mais intuitivo, que vai ser realizado numa fase posterior.

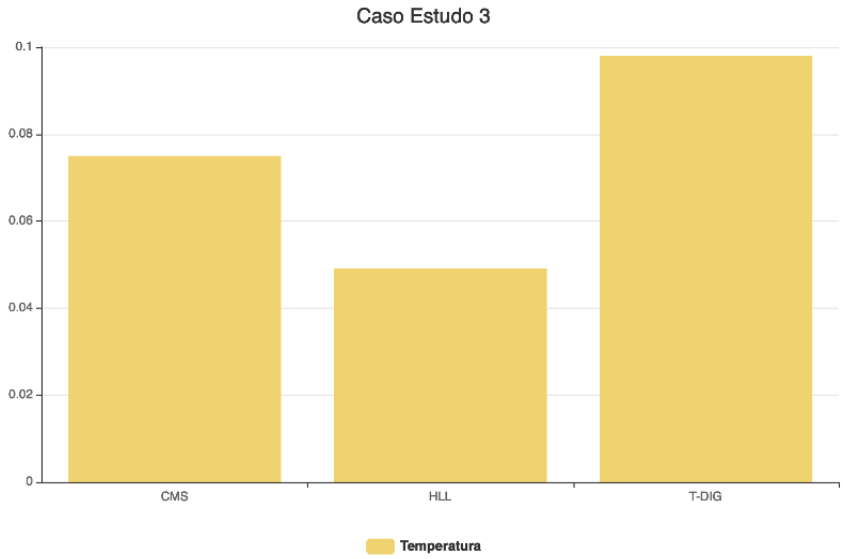


Figura 30: Resultados do Cenário 3 – Análise do Nível da Radiação Solar Incidente

Neste cenário, tivemos em conta os valores da temperatura da água. O estudo do comportamento deste parâmetro é dos mais importantes, uma vez que a variação dos seus valores influencia positivamente ou negativamente os valores dos outros parâmetros estudados. Na Figura 30, podemos verificar que o algoritmo HLL foi o mais eficiente e por sua vez, tal como nos casos anteriores, constatamos que o *T-Digest* foi o menos eficiente. Isto aconteceu, devido às diferentes estruturas utilizadas por cada algoritmo.

5.5 CENÁRIO 4 – ANÁLISE DA CONDUTIVIDADE DA ÁGUA

Ao analisarmos a condutividade presente na água, conseguimos avaliar de forma eficiente o seu grau de mineralização. Este grau resulta da relação entre o teor de sais minerais dissolvidos na água e a resistência que ela oferece à passagem da corrente elétrica. Essa resistência pode ser analisada pela condutividade, que depende da quantidade de substâncias solúveis na água, podendo ser, por vezes, influenciada também pela temperatura. A origem dos sais dissolvidos na água é muito diversa. Parte deles surge na realização de processos de carbonatos, sulfatos, nitratos, cálcio, magnésio, sódio entre outros sais dissolvidos presentes no solo. O valor da condutividade aumenta com o aumento do teor de sólidos dissolvidos. No entanto, a condutividade não pode ser considerada um problema para a saúde do consumidor (Pacheco, 2009). Porém, algumas das suas propriedades podem apresentar certos riscos. Uma mineralização excessiva da água pode causar um sabor desagradável, evidenciar marcas de corrosão ou formar pequenos depósitos, o que por exemplo pode originar um fraco serviço prestado ao cliente.

Ao realizarmos uma análise constante da condutividade da água podemos prevenir ou minimizar as consequências ao utilizador quando detetarmos em tempo real um excesso de condutividade na água. A Figura 31, revela-nos os resultados que obtivemos na execução dos testes ao desempenho para o quarto cenário, relativamente ao valor da condutividade da água. O estudo deste parâmetro ajuda-nos a ter uma melhor perceção do nível ideal de mineralização que devemos ter num sistema de controlo de água. De forma a sabermos qual o melhor algoritmo a aplicar neste caso, efetuamos uma análise de desempenho para este cenário.

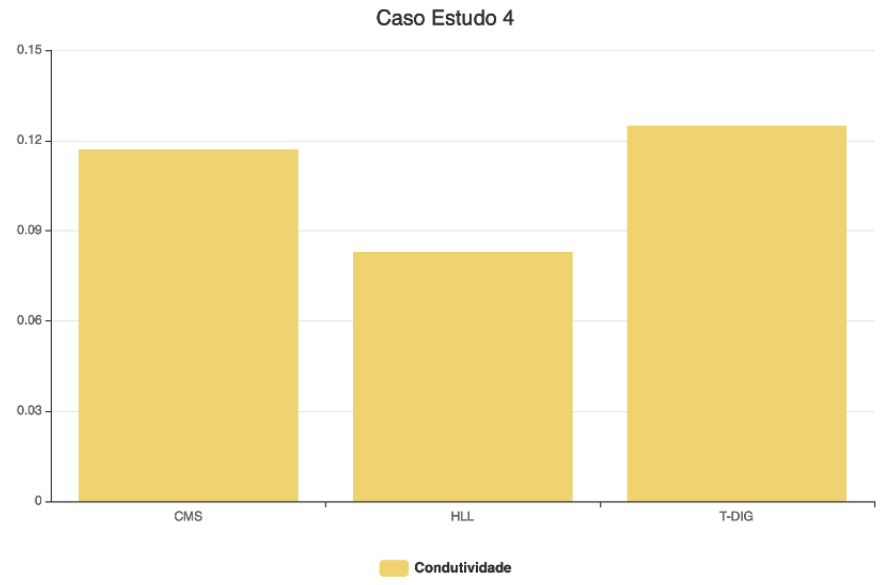


Figura 31: Resultados do Cenário 4 – Análise da Condutividade da Água

Para este quarto cenário tivemos em consideração a variação dos valores da condutividade presentes na água, usando para isso os três algoritmos com o objetivo de identificar valores distantes do valor tabelado. Pela análise dos elementos disponibilizados pela Figura 31, podemos ver que o algoritmo HLL neste cenário voltou a ser o mais eficiente. Verificámos, também, que os algoritmos CMS e *T-Digest* apresentam tempos de execução similares. Porém, o último o *T-Digest* apresenta um tempo de execução um pouco superior.

5.6 CENÁRIO 5 – ANÁLISE DE TURBIDEZ

A turbidez é uma propriedade física da água que se traduz na dificuldade que a luz tem em atravessar a água, devido á presença de partículas em suspensão (Matos, 2008b). Essas partículas podem ser de areia, restos de vegetação ou seres vivos como algas ou bactérias. A erosão e o escoamento da água para locais impermeáveis nas cidades, a contaminação devido às indústrias e às zonas de mineração são os principais fatores que alteram a turbidez da água. As chuvas fortes também fazem com que as águas superficiais fiquem turvas por causa de transportarem pequenos sedimentos na enxurrada. Maioritariamente, a turbidez encontra-se em águas superficiais. Quando a quantidade de partículas ultrapassa os valores máximos permitidos ocorre uma diminuição da penetração da luz na água e consequentemente existe uma redução na fotossíntese realizada pelos seres vivos.

Outra consequência, é a sedimentação entre as pedras no fundo, que acabam por eliminar os locais de desovas dos peixes e o seu *habitat*. Tendo em conta estas consequências, que advêm do excesso de turbidez na água, temos de resolver este problema o mais rápido possível, de modo a este não ter grandes repercussões no ecossistema.

Com base na Figura 32, podemos analisar o tempo de execução dos três algoritmos para o quinto cenário de teste, que é responsável por fazer uma análise da turbidez da água. Efetuando essa análise ao desempenho dos algoritmos, podemos identificar qual o melhor a aplicar neste cenário em específico. Os resultados foram obtidos através do cálculo da média de cinco medições efetuadas ao laboratório de modo a evitar a interferência de fatores que possam influenciar o resultado.

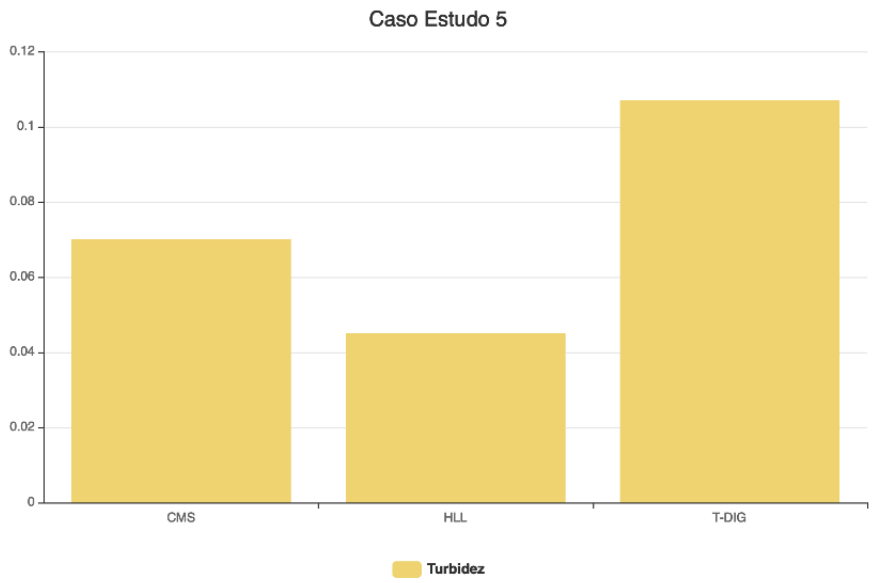


Figura 32: Resultados do Cenário 5 – Análise da turbidez da água

Neste último cenário, efetuamos a análise dos valores da turbidez de modo a identificarmos, quase de forma instantânea, quais os casos em que eles variavam para fora da gama de valores definidos como ideais. Tendo em conta a Figura 32, podemos concluir que, tal como em todos os outros cenários, o algoritmo HLL é aquele que apresenta um melhor tempo de execução, sendo também aquele que é mais eficiente. Isto verificou-se para todos os cenários, pois estes têm a mesma fonte e tipo de dados, mudou apenas o parâmetro estudado de cenário para cenário.

5.7 ANÁLISE GERAL DOS RESULTADOS

Após a realização dos diversos testes de eficiência a cada um dos algoritmos, procedemos a uma análise geral dos resultados obtidos. Em todos os testes utilizámos três algoritmos determinísticos, cada um deles com objetivos diferentes, nomeadamente, o algoritmo *Count Min Sketch* tem a função de determinar quais são os valores que se encontram numa gama de valores, o algoritmo *Hyper Log Log* tem a função de considerar todos os elementos sem repetições e por fim, o algoritmo *t-Digest* tem como objetivo saber quais os quartis (o segundo quartil representa a mediana) de um dado parâmetro.

Os três algoritmos foram aplicados em cinco cenários de teste diferentes, tal como foi referido anteriormente, envolvendo um conjunto de 500 registos de dados, que lhes foram sendo enviados, de forma continua, em intervalos de 1 segundo.

De uma forma geral, tendo em conta os resultados obtidos concluímos que o algoritmo HLL é para todos os cenários em estudo o mais eficiente em termos de tempo de execução e por sua vez o *T-Digest* é o menos eficiente. Apesar de todos os algoritmos estudados possuírem o mesmo objetivo, eles diferem na estrutura implementada para identificarem os valores discrepantes para cada cenário e daí resultam os diferentes tempos de execução para cada um. A estrutura do HLL é a mais eficiente uma vez que esta usa apenas uma função de *Hash* para o cálculo da cardinalidade, enquanto que a estrutura do *T-Digest* é a menos eficiente. Isso pode ser justificado por este algoritmo utilizar uma função acumulativa e realizar cálculos matemáticos complexos responsáveis por calcular os quartis, requerendo assim um maior poder computacional para processar os dados em tempo muito próximo do real. Num tempo de processamento intermédio entre os dois algoritmos anteriores está o CMS que usa na sua estrutura uma matriz de funções de *hash* para calcular o número mínimo de repetições de um dado parâmetro.

Na globalidade, os resultados obtidos foram satisfatórios uma vez que os tempos de execução gerais para cada cenário se encontram entre os 0,05 e os 0,20 milissegundos o que nos possibilita tirar conclusões muito próximas do tempo real, satisfazendo assim o objetivo principal desta dissertação. Assim sendo, a aplicação de qualquer um destes algoritmos só traz benefício para a análise da água uma vez que obtemos em tempo real situações onde são precisas intervenções humanas rápidas evitando assim efeitos secundários maiores.

Cenários	*T-Digest*	*Count-Min Sketch*	*Hyper Log Log*
Cenário 1	0,140	0,119	0,081
Cenário 2	0,170	0,120	0,106
Cenário 3	0,097	0,075	0,049
Cenário 4	0,122	0,117	0,075
Cenário 5	0,107	0,072	0,045

Tabela 4: Comparação do desempenho dos algoritmos utilizados

Na Tabela 4, podemos observar o tempo de execução em milissegundos para todos os cenários e algoritmos implementados. Embora estes resultados tenham sido muito próximos entre si, os algoritmos aplicados possuem métodos diferentes na deteção de anomalias. O *Count-Min Sketch* caracteriza-se por conseguir determinar a frequência de um valor específico para um dado parâmetro. Enquanto isso, o *Hyper Log Log* pode ser aplicado em situações em que pretendemos estimar a cardinalidade de um parâmetro, ou seja, quando tencionamos determinar o número de elementos distintos num conjunto de dados. Por sua vez, o *T-Digest* pode ser utilizado quando planeamos definir os quartis, a média ou a mediana de um conjunto.

6

CONCLUSÕES E TRABALHO FUTURO

6.1 CONCLUSÕES

Nos dias de hoje, o processamento de *streams* de dados em tempo real (ou em tempo útil) traz-nos inúmeras vantagens na análise de dados provenientes de sensores em aplicações nos mais diversos domínios de negócio. Nesses domínios, a análise dos dados contidos em *streams* é um processo bastante importante que contribui com elementos relevantes para a gestão de várias ações de trabalho, muitas delas relacionadas com tarefas de tomada de decisão, que têm grande impacto no seio das empresas, uma vez que permitem melhorar, de forma significativa, as suas ações de trabalho bem como o seu próprio desempenho.

Dependendo do seu objetivo final, um processo de análise de dados contidos em *streams* pode utilizar diversos tipos de algoritmos de *sketching*, de forma a permitir aproveitar todos os dados que sejam gerados por uma fonte, de forma contínua, constituindo, na grande maioria dos casos, uma ferramenta de auxilio bastante útil em processos de tomada de decisão, permitindo acompanhar a evolução de um processo ao longo do tempo, tendo em conta o tipo de variações que vão ocorrendo nos elementos de dados das *streams* em análise. Podemos aplicar técnicas de *sketching* nas mais diversas áreas, como é o caso da saúde, das companhias aéreas, dos sistemas de recomendações, ou mesmo em processos de otimização ou de deteção de anomalias de uma empresa.

Na realização desta dissertação estudámos e aplicámos um conjunto de diferentes algoritmos de *sketching* no processamento de *streams* de dados em tempo muito próximo do real, com o intuito de analisar o desempenho de cada um dos algoritmos estudados. Para isso idealizámos e desenvolvemos um sistema de análise de qualidade da água, com o objetivo de fornecer os dados para teste de cada um dos algoritmos de *sketching* que escolhemos, tendo como propósito encontrar valores discrepantes dos parâmetros (pH, oxigénio dissolvido, turvação, etc.) que usualmente são analisados para verificar a qualidade da água numa rede de distribuição. Para isso, implementámos um laboratório de análise (um sistema de *software*) específico, capaz de simular um fluxo de dados contínuo, contendo os diversos valores de cada um dos parâmetros de qualidade escolhidos, que, no caso de uma aplicação prática real seria gerado por um conjunto de sensores específicos.

Para montar o referido laboratório de dados, numa primeira fase tivemos que realizar um estudo concreto sobre quais os parâmetros (variáveis) utilizados na determinação da qualidade da água, de forma a sabermos quais seriam os seus valores ideais segundo uma entidade certificada e, assim, estabelecermos a maneira de como detetar eventuais valores anómalos. Dessa forma, poderíamos encontrar uma solução adequada para eliminar, ou pelo menos minimizar, danos que podiam vir a ser causados. Para obter os dados necessários para testar cada um dos algoritmos de *sketching* que escolhemos, tivemos de simular um fluxo de dados contínuo que deveria ser captado em tempo real por diversos sensores, em intervalos de tempo de 1 segundo. Depois, esse fluxo serviu como fonte de dados para o laboratório. A análise dos dados gerados foi realizada utilizando três diferentes algoritmos de *sketching*, sendo estes o *Count Min Sketch*, o *t-Digest* e *Hyper Log Log*, que produziram um *output* com a identificação das várias situações anómalas que detetaram, em tempo muito próximo do real.

Todos os algoritmos utilizados são aplicados com objetivos diferentes no processo de deteção de anomalias. Esta diferença deve-se às estruturas usadas na implementação de cada algoritmo. Assim sendo, podemos verificar que o algoritmo *Count Min Sketch* tem a função de determinar a frequência de um dado valor ou intervalo num conjunto de dados, por sua vez o algoritmo *Hyper Log Log* tem a função de calcular a cardinalidade de um conjunto (considerar todos os elementos distintos) e por fim, o algoritmo *t-Digest* tem como objetivo definir os quartis relativos a um parâmetro num conjunto de dados.

Para cada um dos diferentes algoritmos efetuámos uma medição do seu tempo de execução, para mais tarde fazermos uma análise da sua eficiência e identificar qual seria o algoritmo com as melhores características para atingir o objetivo inicialmente definido. O estudo comparativo realizado permiti-nos concluir que para os cinco cenários em estudo, o algoritmo HLL é o que apresenta um tempo de execução mais baixo e que por outro lado, o algoritmo *t-Digest* apresenta um tempo mais elevado.

A realização desta dissertação permitiu-nos consolidar o conhecimento acerca da importância que o processamento de *streams* de dados apresenta nos dias de hoje e do que poderá vir a representar no futuro. No decorrer do processo de implementação desta dissertação ficamos com uma melhor perceção dos fatores que podem implicar alterações importantes em sistemas operacionais, tendo dado particular atenção a um caso de aplicação concreto relacionado com uma rede de distribuição de água. Além disso, com a aplicação de técnicas específicas de *sketching* alicerçamos os nossos conhecimentos sobre a exploração, tratamento e processamento de fluxos de dados. O conhecimento obtido sobre o funcionamento e implementação dos algoritmos de *sketching* permitiu-nos perceber, e perspetivar, o impacto que o processamento e análise de *streams* de dados em tempo real pode ter no seio de uma organização. Ao podermos efetuar, de forma expedita, a análise contínua de *streams* de dados temos a possibilidade de prevenir ou minimizar eventuais danos causados por qualquer sistema ou até mesmo maximizar a eficiência ou o lucro associados com um dado sistema de análise. Termos a capacidade de análise de grandes volumes de dados em tempo real, representa uma mais valia não só em termos de análise como também da própria operacionalidade do sistema de controlo dos fluxos de dados, fazendo com que estes possam evoluir de uma forma mais sustentável e eficiente.

6.2 TRABALHO FUTURO

Ao longo deste trabalho de dissertação demonstrámos como poderíamos aplicar o processamento de *streams* de dados em tempo real para efetuar uma análise contínua à qualidade da água numa rede de distribuição de água, de forma a evitar ou prevenir situações anómalas que possam acontecer na rede, no imediato. Para isso, tivemos que efetuar a simulação de uma rede de distribuição de água, munida com diversos sensores para captar dados relativos à qualidade da água tal como o pH, a condutividade, a temperatura, a turbidez e o oxigénio dissolvido de forma contínua, através da implementação de um laboratório de dados que fosse capaz de sustentar a abordagem que foi definida.

Para isso, implementamos um programa com a capacidade de produzir dados de forma ininterrupta e com a capacidade de os analisar em tempo real. Com a aplicação deste laboratório realizámos uma prova de conceito sobre processamento de *streams* de dados em tempo muito próximo do real. No entanto não nos foi possível aplicar e validar o sistema implementado num caso no mundo real. Assim, uma das próximas tarefas a realizar seria aplicar e testar esta a solução de processamento de *streams* desenvolvida no âmbito desta dissertação a um caso de aplicação real, abordando de forma detalhada cada uma das suas etapas, desde a recolha e tratamento dos dados em tempo real até ao processo da tomada de decisão, com o objetivo de analisar a eficiência do sistema e de identificar potenciais situações anómalas em tempo real durante o funcionamento do sistema. Através da aplicação deste estudo num caso real poderíamos verificar, em concreto e de um modo mais preciso, quais seriam as vantagens adquiridas pelos utilizadores que têm a responsabilidade de gerir e controlar este tipo de sistemas e a análise dos enormes volumes de dados que produzem.

Complementarmente, seria também bastante útil a conceção e o desenvolvimento de um interface gráfico que permitisse visualizar em tempo real o fluxo de dados e as respetivas variações de cada parâmetro estudado, através de um conjunto específico de *dashboards* de gestão e controlo pertinentes, bem como implementar e gerir alertas para situações não previstas que pudessem eventualmente acontecer ao longo dos processos de *sketching* e de análise de dados. Além disso, poderíamos também criar um conjunto de serviços adicionais que nos permitisse guardar e gerir o histórico da realização das várias ações de processamento de *streams*, bem como providenciar um relatório sobre eventos para auxiliar em ações futuras que possam ser realizadas sobre o problema que já tenham sido processados anteriormente pelo sistema. Numa perspetiva de utilização, a combinação destas duas possibilidades de evolução do atual sistema, numa aplicação com uma interface intuitiva, melhoraria imenso a sua qualidade de serviço, já que facilitaria muito o processamento e a análise de *streams* de dados com o sistema implementado.

BIBLIOGRAFIA

A. Banerjee, V. Chola, and V. Kumar. *Anomaly detection: A survey*. 2009.

G. Bower. Which apple watch running app deserves to log your sweaty miles?, 2017. URL https://www.cultofmac.com/486176/best-running-apps-apple-watch/.

K. Chra. View counting at reddit, 2017. URL https://redditblog.com/2017/05/24/view-counting-at-reddit/.

M. Roberts R. Chui, M. Loffler. The internet of things. URL http://dln.jaipuria.ac.in:8080/jspui/bitstream/123456789/3111/1/The%20Internet%20of%20Things.pdf.

G. Cormode and S. Muthukrishnan. *An improved data stream summary: the count-min sketch its applications*. Journal of Algorithms 55, 2005.

G. Cormode, Yi K. Luo, G., and L. Wang. Quantiles over data streams: Experimental comparisons, new analyses, further improvements. URL http://archive.dimacs.rutgers.edu/~graham/pubs/papers/nquantvldbj.pdf.

A. Cross. The importance of scalability in big data processing, 2018. URL https://www.ngdata.com/the-importance-of-scalability-in-big-data-processing/.

C. Davidson-Pilon. Percentile quantile estimation of big data: The t-digest, 2015. URL https://dataorigami.net/blogs/napkin-folding/19055451-percentile-and-quantile-estimation-of-big-data-the-t-digest.

DrChrono. URL https://www.drchrono.com/electronic-health-record-ehr/.

M. Dur and P. Flajolet. Loglog counting of large cardinalities. URL http://algo.inria.fr/flajolet/Publications/DuFl03-LNCS.pdf.

Estado Português. Decreto lei de 27 de agosto 2007, 2007. https://dre.pt/application/conteudo/640931.

O. Etrl and T. Dunning. *Percentile Quantile Estimation of Big Data: The t-Digest*. 2015.

FIA. Big data: o que é, como aplicar, a importância e exemplos, 2018. URL https://fia.com.br/blog/big-data/.

B. Fraguela and C. Bozkus. *Accelerating the HyperLogLog Cardinality Estimation Algorithm*. 2017. Article ID 2040865.

N. Galdino. *Big Data: Ferramentas e Aplicabilidade*. IESSA, 2016.

F. Halper and K. Krishnan. *TDWI Big Data Maturity Model Guide Interpreting Your Assessment Score*. 2013.

J. Hecht. Fixing a broken record, 2019. URL https://media.nature.com/original/magazine-assets/d41586-019-02876-y/d41586-019-02876-y.pdf.

F. Hoffa. *Counting uniques faster in BigQuery with HyperLogLog++*. 2017.

J. Howell. Extrair, transformar e carregar (etl) em escala, 2020. URL https://docs.microsoft.com/pt-pt/azure/hdinsight/hadoop/apache-hadoop-etl-at-scale.

S. Chung J. Kin, G. Trimi. Big-data applications in the government sector, 2014. URL https://cacm.acm.org/magazines/2014/3/172509-big-data-applications-in-the-government-sector/fulltext.

K. Krishnan. *Datawarehousing in the age of big data*. Morgan Kaufmann, 1st edition edition, 2013.

V. Marcelino. Sistema de recomendação. Master's thesis, 2014.

A. Matos. *Riscos para a saúde humana resultantes da exposição à oxidabilidade*. 2008a.

A. Matos. *Riscos para a saúde humana resultantes da exposição à turvação*. 2008b.

W. Mira. Netflix: Big data e os algoritmos de recomendação. Master's thesis, 2019.

R. Munné. *Big Data in the Public Sector*. 2016.

Natalie, 2017. URL https://digital.hbs.edu/platform-digit/submission/netflix-a-personalized-viewing-experience/.

nControl. O que é rfid ? URL https://www.ncontrol.com.pt/o-que-e-rfid.html.

C. Nietner. The panana pappers: How data science can help to effectively identify the wrongdoers. URL https://blog.unbelievable-machine.com/en/blog/panama-papers--data-science/.

NiuSi. Racca's pizzeria napoletana - restaurant review tripadvisor. URL https://www.tripadvisor.com/Restaurant_Review-g60437-d10689307-Reviews-Racca_s_Pizzeria_Napoletana-Casper_Wyoming.html.

P. Pacheco. *Riscos para a saúde humana resultantes da exposição à condutividade*. 2009.

A. Sabtu and S. A. Ismail. *The Challenges of Extract, Transform Load (ETL) for Data Integration in Near Real-Time Environment*. Journal of Theorical Applied Information Technology, 2017.

K. Saikali. Iot apps for smart cities with ge predix and scriptr.io, 2016. URL https://blog.scriptr.io/implement-smart-cities-applications/.

S. Sheriff. Understing the 5 vs of big data, 2019. URL https://acuvate.com/blog/understing-the-5vs-of-big-data/.

K. Shukla. Big data with sketchy structures, part 1 — the count-min sketch, 2018. URL https://towardsdatascience.com/big-data-with-sketchy-structures-part-1-the-count-min-sketch-b73fb3a33e2a.

C. Taurion. *Big Data*. 2013a.

C. Taurion. *O papel do Big Data na Transformação da Sociedade, Negócios e Governo*. IBM Corporation, 2013b.

M. Techlabs. 5 advantages recommendation engines can offer to businesses, 2017. URL https://towardsdatascience.com/5-advantages-recommendation-engines-can-offer-to-businesses-10b663977673.

Tera. 5 principios de ética em big data, 2018. URL https://medium.com/somos-tera/5-principios-de-etica-em-big-data-4cada3d86687.

A

LISTA DE SIGLAS

- ETL - Extract Transform Load
- SGBD - Sistemas de Gestão de Base de Dados
- SQL - Structured Query Language
- NoSQL - Not Only SQL
- BI - Business Intelligence
- RGPD – Regulamento Geral sobre a Proteção de Dados
- IoT - Internet of Things
- PH - Potencial Hidrogeniónico
- OD - Oxigénio Dissolvido
- COD - Concentração de Oxigénio Dissolvido
- HLL - Hyper Log Log
- CMS – Count Min Sketch
- T-Dig – T-Digest
- DGS - Direção Geral da Saúde