28TH BRAZILIAN SYMPOSIUM ON DATABASES

LECTURES

PROCEEDINGS

September 30th – October 3rd, 2013
Recife, Pernambuco, Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Pernambuco – UFPE

Realization

Centro de Informática (CIn)

Lectures Chair

João Eduardo Ferreira, IME-USP

Editorial

The Brazilian Symposium on Databases (SBBD) comes to 28th Edition in 2013. This traditional event of Brazilian scientific community on Database Systems is supported by Brazilian Computer Society (SBC) and this year it will be hosted in Recife, Pernambuco State, Brazil. The SBBD 2013 is organized as set of many different technical events. More concretely, in this editorial text, we have a special attention for SBBD Lectures. SBBD Lectures are small courses about some specific subject in which the main goal is to provide fundamental issues and advanced discussion concerning with current trends on Database research and technologies. During these lectures, audience has the opportunity for opening discussions about relevant subjects and collaborate on building a collective knowledge with experts.

Because of some internal logistics decisions of SBBD 2013, only three (3) proposals that are organized as 3-hour presentation were accepted. This decision has considered the analysis of eight (8) initial proposals that were submitted by research groups from different Brazilian institutions. The evaluation was based on technical criteria, such as originality, relevance, format and connection to the SBBD audience. During this process, this committee had the essential collaboration of profs. Ana Carolina Salgado (UFPE), Fernanda Baião (UNIRIO), José Maria Monteiro (UFC), Márcio K. Oikawa (UFABC), Mirella Moro (UFMG), and Renato Fileto (UFSC).

We are very grateful to the unconditional support of SBBD 2013 Committees, authors, speakers and participants. The SBBD 2013 Lectures organization committee has worked hard to provide the best conditions to a very enjoyable event.

João Eduardo Ferreira
SBBD 2013, Lectures Chair

28TH BRAZILIAN SYMPOSIUM ON DATABASES

September 30th – October 3rd, 2013

Recife, Pernambuco, Brazil

Promotion

Brazilian Computer Society – SBC
SBC Special Interest Group on Databases

Organization

Universidade Federal de Pernambuco – UFPE

Realization

Centro de Informática (CIn)

SBBD Steering Committee

Marco A. Casanova, PUC-Rio (Chair)
Ana Carolina Salgado, CIn-UFPE
Cristina Dutra de Aguiar Ciferri, USP
José Palazzo Moreira de Oliveira, UFRGS
Mirella M. Moro, DCC-UFMG

SBBD 2013 Committee

Steering Committee Chair
Marco A. Casanova, PUC-Rio
Local Organization Chair
Bernadette Farias Lóscio, CIn-UFPE
Program Committee Chair
Cristina Dutra de Aguiar Ciferri, USP
Short Papers Chairs
Renato Fileto, UFSC and Marco Cristo, UFAM
Demos and Applications Chairs
Damires Souza, IFPB and Daniel Kaster, UEL
Thesis and Dissertation Workshop Chairs
Karin Becker, UFRGS and André Santanchè, UNICAMP
Tutorials Chair
Mirella M. Moro, DCC-UFMG
Lectures Chair
João Eduardo Ferreira, IME-USP

Local Organization Committee

Bernadette Farias Lóscio, CIn-UFPE (Chair)
Ana Carolina Salgado, CIn-UFPE
Agenor Marinho de Souza Neto, CIn-UFPE

Carlos Eduardo Santos Pires, DSC-UFCG
Danusa Ribeiro Cunha, CIn-UFPE
Priscilla Vieira, CIn-UFPE
Robson Fidalgo, CIn-UFPE

Lectures Program Committee

João Eduardo Ferreira, IME-USP (Chair)
Ana Carolina Salgado, UFPE
Márcio K. Oikawa, UFABC
Renato Fileto, UFSC
José Maria Monteiro, UFC
Fernanda Baião, UNIRIO
Mirella M. Moro, UFMG

28TH BRAZILIAN SYMPOSIUM ON DATABASES

LECTURES

Table of Contents

Minicurso 1

Análise em Big Data e um Estudo de Caso utilizando Ambientes de Computação em Nuvem

Ticiana L. C. da Silva, Flávio R. C. Sousa,
José Antônio F. de Macêdo e Javam C. Machado

Resumo

As novas aplicações de mineração de dados no contexto de Big Data ou análise em Big Data necessitam gerenciar grandes volumes de dados de diferentes tipos e estruturas. As técnicas de mineração aplicadas sob tais dados possibilitam a extração de novos conhecimentos ou a realização de predições, por exemplo. Entretanto, o processamento intensivo de dados está além da capacidade de qualquer máquina individual e requer que a computação seja realizada em clusters. Dessa forma, a utilização de técnicas de mineração para grandes volumes de dados exige adaptações em seus algoritmos de forma a paralelizar a execução e utilizar ambientes de larga escala para o armazenamento e processamento, atualmente disponíveis por meio de infraestruturas de computação em nuvem. Este minicurso tem como objetivo apresentar os principais algoritmos de mineração para Big Data considerando a utilização de ambientes de computação em nuvem, além de apresentar boas práticas para o desenvolvimento de soluções neste contexto. Também é apresentado um estudo de caso considerando o domínio de dados de monitoramento de tráfego, enfatizando o uso das técnicas de mineração e da infraestrutura de nuvem discutidas. Por fim, são apresentadas as considerações finais sobre o tema, destacando oportunidades e desafios encontrados.

1.1. Introdução

Nas duas últimas décadas, o aumento contínuo do poder computacional tem produzido um fluxo enorme de dados [QU et al. 2012]. A cada dia, 2.5 quintilhões de bytes de dados são criados e 90% dos dados no mundo hoje foram produzidos nos últimos dois anos [Wu et al. 2013]. Como exemplos deste cenário, pode-se citar modernos centros de pesquisa em física, tais como DZero, que normalmente geram mais de um terabyte de dados por dia. A rede social Facebook possui mais de um bilhão de usuário e cem horas de novos vídeos são armazenados a cada minuto no Youtube. Estes exemplos exigem arma-

zenamento eficiente, análise de grande volume de dados e tomada de decisão instantânea. A este contexto de gestão e análise de dados tem-se denominado "Big Data".

De acordo com [Begoli and Horey 2012], Big Data é a prática de coleta e processamento de grandes conjuntos de dados e sistemas e algoritmos utilizados para analisar estes conjuntos de dados massivos. Já [Wu et al. 2013] argumenta que Big Data refere-se a grandes volumes heterogêneos, fontes autônomas com controle distribuído e descentralizado, procurando explorar as relações complexas e em evolução entre os dados.

Apesar das enormes possibilidades do uso da informação aferidas ao fenômeno Big Data, não é trivial entender quais técnicas e métodos de gerenciamento de dados serão adequados para lidar com os desafios trazidos por este novo cenário. Apesar do termo Big Data ser explicado através dos famosos 3 V's: Volume, Velocidade e Variabilidade, não são claros os desafios trazidos por tais caracteristicas às tecnologias de banco de dados desenvolvidas ao longo dos últimos 40 anos. De fato, banco de dados tem como premissa lidar com grande volumes de dados por meio de inúmeras técnicas de processamento distribuído. A velocidade é tratada pelos bancos de dados que manipulam *stream* de dados, e a variabilidade foi amplamente pesquisada em métodos para integração de dados estruturados e semiestruturados, os quais visam lidar com a heterogeneidade de modelos e dados. Consequentemente, segundo Dan Suciu [Suciu 2013], o que Big Data traz de novo ao contexto de gerenciamento de dados são três aspectos: dados são armazenados em memória principal, necessidade de inúmeras iterações sobre os dados, a ocorrência de falhas é uma regra.

Com o atual uso de infraestruturas massivas de servidores virtualizados para processar um grande volume de dados distribuídos, podemos supor que a maior parte dos dados já processados estarão armazenados em memória, em suas respectivas máquinas virtuais. Desta forma, a nova métrica de complexidade para processamento de consultas deixa de ser a número de acessos a disco para se tornar quantidade de comunicação entre os servidores que processam as informações. Claramente, os métodos e técnicas de banco de dados deverão levar em consideração esta mudança de complexidade.

Diversos tipos de análise de dados no contexto Big Data exigem computação iterativa, incluindo clusterização usando K-Means, PageRank, consultas recursivas relacionais, análise de redes sociais, entre outros. Essas técnicas visam processar os dados iterativamente até que a computação satisfaça uma determinada condição de parada ou convergência. Este tipo de computação iterativa não é provido pelas tecnologias de banco de dados atuais. De fato, serão necessárias técnicas mais sofisticadas para lidar com computação iterativa, levando em conta os custos de comunicação.

Outro ponto importante a destacar é que banco de dados paralelos lidam com a falha dos nós de maneira excepcional. Em um cenário Big Data, onde uma grande quantidade de máquinas estão envolvidas, claramente as falhas deixam de ser uma exceção para se tornarem uma regra. Desta maneira, é necessário que o gerenciamento de dados em Big Data lide com a falha dos nós como um evento frequente, e não como uma exceção ao processamento.

Um outro ponto importante a destacar no cenário Big Data é a infraestrutura computacional necessária para lidar com grandes volumes de dados heterogêneos e distribuí-

dos. Considerando o grande volume a ser analisado e que o processamento intensivo dos dados está além da capacidade de qualquer máquina individual, o cenário Big Data demanda a disponibilização de novos modelos computacionalmente eficientes
[Lin and Dyer 2010]. Neste contexto, a computação baseada em *clusters* permite aproveitar grande número de processadores em paralelo para resolver um problema de computação [Pavlo et al. 2009]. Claramente, a disponibilização de uma infraestrutura de processamento em larga escala exige uma infraestrutura de software compatível, a qual possa tirar vantagem da grande quantidade de máquinas e mitigar o problema de comunicação entre essas máquinas. Com o interesse em *clusters*, aumentou a quantidade de ferramentas para utilizá-los, dentre as quais se destaca o framework MapReduce [Dean and Ghemawat 2008] e sua implementação de código aberto *Hadoop*, utilizado para gerenciar grandes quantidades de dados em *clusters* de servidores. Este framework é atraente porque oferece um modelo simples por meio do qual os usuários podem expressar programas distribuídos relativamente sofisticados.

A implantação e o gerenciamento de sistemas de *clusters* de larga escala envolvem grandes investimentos na aquisição e manutenção de equipamentos. Para melhorar o gerenciamento e reduzir os custos, as aplicações de Big Data têm utilizado ambientes de *Cloud Computing* ou Computação em Nuvem [Agrawal et al. 2011]. Estes ambientes permitem que empresas e indivíduos aluguem capacidade de computação e armazenamento sob demanda e com pagamento baseado no uso, em vez de fazerem grandes investimentos de capital necessários para a construção e instalação de equipamentos de computação em larga escala [Sousa et al. 2010]. Além disso, a Computação em Nuvem fornece ambientes com grande capacidade de armazenamento, escaláveis, elásticos, com elevado desempenho e alta disponibilidade. Assim sendo, a nuvem fornece uma alternativa viável para a construção de aplicações de gestão e análise de grandes volumes de dados [Agrawal et al. 2011].

As aplicações de Big Data estão se expandindo em todos os domínios de ciência e engenharia, incluindo física, biologia e medicina. Estas aplicações demostram que o gerenciamento de grandes volumes de dados está além da capacidade das ferramentas de software para armazenar e processar estes dados dentro de um intervalo de tempo aceitável. O desafio fundamental para as aplicações de Big Data é explorar os grandes volumes de dados e extrair informações úteis ou conhecimento para futuras ações [Rajaraman and Ullman 2012].

Em muitas situações, o processo de análise deve ser eficiente e quase em tempo real, pois o armazenamento de todos os dados observados é quase inviável [Wu et al. 2013]. Como resultado, um volume de dados sem precedentes necessita de análise eficaz. Para tanto, técnicas de mineração de dados podem ser utilizadas para analisar e entender os dados a serem manipulados. A análise é baseada em modelos capazes de sumarizar dados, extrair novos conhecimentos ou realizar predições. Estes modelos podem ser utilizados para construir um software que possibilite identificar o perfil de clientes para conceder empréstimos bancários, aplicações de recomendação de busca de amigos em redes sociais, que envolvem grafos com milhões de nós e arestas ou, ainda, sistemas de software que identifiquem possíveis ameaças terroristas [Rajaraman and Ullman 2012].

Os três principais componentes para sucesso de um projeto de mineração de dados

no contexto de Big Data são: (i) presença de um cenário de aplicação em que seja possível identificar a demanda por descoberta de conhecimento; (ii) um modelo que realize a análise desejada; (iii) uma implementação eficiente que seja capaz de gerenciar um grande volume de dados [Rajaraman and Ullman 2012]. Outros fatores devem ser considerados para resolver um problema com um grande volume de dados, tais como o tamanho e a complexidade dos dados, a facilidade com a qual os dados podem ser eficientemente transportados e principalmente se o algoritmo pode ser paralelizado de forma eficiente [Schadt et al. 2010].

Entretanto, a análise dos dados não é uma tarefa trivial. É preciso utilizar técnicas automatizadas que garantam que os dados serão explorados corretamente. Utilizar técnicas tradicionais permitindo, por exemplo, que o conhecimento seja obtido apenas por intervenções de especialistas humanos não é uma decisão viável, devido ao grande volume de dados. As técnicas de mineração podem ser utilizadas para extrair conhecimento tanto de tipos de dados convencionais, também chamados alfanuméricos (i.e. letras, números e datas), quanto de dados não convencionais, a exemplo de tipos de dados espaciais, espaço temporais, biológicos, assim como dados estruturados, bem como não estruturados, por exemplo, vídeos, fotos e som. Dessa forma, outro desafio relevante da análise de dados em Big Data é gerenciar dados de diferentes tipos e estruturas [Rajaraman and Ullman 2012].

Este minicurso apresenta e discute as principais questões relacionadas à análise para Big Data, destacando os pontos chaves e as boas práticas para a construção de novos algoritmos para Big Data. Além disso, como o ambiente de Computação em Nuvem pode auxiliar na gestão e no processamento destes dados. Este trabalho está organizado da seguinte forma: a Seção 1.2 apresenta as tecnologias e infraestruturas utilizadas para a análise de Big Data. A Seção 1.3 apresenta os principais algoritmos de mineração de dados em Big Data. A Seção 1.4 apresenta um estudo de caso baseado em dados de trânsito. Por fim, a Seção 1.6 apresenta as considerações finais sobre o tema, lições aprendidas e destaca tendências direcionamentos futuros, como as oportunidades e desafios encontrados.

1.2. Infraestrutura para Análise em Big Data

A necessidade em gerenciar e analisar uma grande quantidade de dados e fornecer alto desempenho tem sido requisitos comuns em muitas empresas. Para tratar estes problemas, diferentes soluções têm sido propostas, entre elas a migração/construção de aplicações para ambientes de computação em nuvem, além de sistemas baseados em *Distributed Hash Table* (DHT) ou estrutura de *arrays* multidimensionais [Sousa et al. 2010]. Dentre estas soluções, destaca-se o paradigma MapReduce [Dean and Ghemawat 2008], concebido para apoiar o processamento distribuído de grandes conjuntos de dados em *clusters* de servidores e sua implementação de código livre *Hadoop* [White 2012].

Computação em Nuvem provê soluções baratas e eficientes para armazenamento e análise de grandes volumes de dados [Li and Zhang 2011]. Existem diferentes definições e conceitos de computação em nuvem. Segundo [Mell and Grance 2009], computação em nuvem pode ser definido como um modelo que permite acesso sob demanda a um agrupamento de recursos computacionais que podem ser configuráveis, como CPU, armazenamento, memória, entre outros. Eles podem ser rapidamente fornecidos e liberados

com o mínimo esforço de gerenciamento ou assistência do provedor da nuvem.

Algumas propriedades fundamentais distinguem computação em nuvem dos sistemas distribuídos tradicionais (e.g. sistemas em grade, clusters, P2P, etc) e estão relacionadas ao seu caráter atrativo: (i) Auto-Serviço sob demanda, (ii) Elasticidade rápida, (iii) Pagamento à medida que o serviço é utilizado (*Pay-as-you-go*) (iv) Nível de qualidade de serviço (SLA), (v) Agrupamento ou *Pooling* de Recursos. A seguir, falaremos mais de cada uma dessas características.

Auto-serviço sob demanda diz respeito à possibilidade do consumidor utilizar os recursos computacionais (tempo de servidor, por exemplo) quando necessário sem qualquer intervenção humana com o provedor do serviço. Outra caracterítica é a elasticidade que permite escalar mais recurso com o aumento da demanda, e liberar recurso, na retração dessa demanda. Para o consumidor, os recursos disponíveis para provisionamento muitas vezes parecem ser ilimitados e podem ser alocados em qualquer quantidade e a qualquer momento. Os recursos em sistemas na nuvem são automaticamente controlados e o seu uso é medido em um nível de abstração apropriado para o tipo de serviço (como armazenamento, processamento, por exemplo). No modelo *Pay-as-you-go*, o consumidor só paga pelo que utiliza e pelo tempo de utilização. A utilização dos recursos pode ser monitorada, controlada e informada, gerando transparência tanto para o provedor como para o consumidor do serviço utilizado.

O SLA é a garantia mínima de qualidade de serviço. Os recursos de computação do provedor são agrupados para atender a múltiplos consumidores em modelo multi-inquilinos. Os diferentes recursos físicos e virtuais são dinamicamente atribuídos e reatribuídos conforme a demanda dos consumidores. Há uma certa independência de localização geográfica, uma vez que o consumidor em geral não controla ou conhece a localização exata dos recursos fornecidos (como armazenamento, processamento, memória e comunicação de rede), mas pode ser capaz de especificar a localização em um nível de abstração mais alto (como país, estado ou *datacenter*). Consequentemente, qualquer serviço computacional que seja executado em um ambiente em nuvem precisará estar em conformidade com tais propriedades [Coelho da Silva 2013].

O Hadoop implementa o modelo de programação MapReduce, juntamente com um sistema de arquivo distribuído chamado *Hadoop Distributed File System* (HDFS) para permitir o processamento de grandes volumes de dados em ambientes distribuídos. O HDFS foi projetado e otimizado para ser executado em centros de dados e fornecer elevada vazão, baixa latência e tolerância a falhas individuais de servidores.

Em clusters ou ambientes de computação em nuvem, é comum haver falhas nos nós ou na rede. Dessa forma, é necessário o uso de estratégias para tratar estas falhas, principalmente considerando que o processamento pode demandar horas para seu término. A cada falha pode ser necessário abortar e reiniciar todo o processamento. Um cenário possível é a execução nunca ser concluída com êxito. Algumas medidas poderiam ser adotadas como, por exemplo: (i) armazenar arquivos redundantes e (ii) dividir o processamento entre os nós computacionais. Assim sendo, quando uma tarefa é interrompida por uma falha, ela não prejudica o processamento de outras tarefas, basta que somente ela seja reiniciada. O HDFS implementa estratégias para o tratamento de falhas, como o uso de replicação e distribuição dos dados.

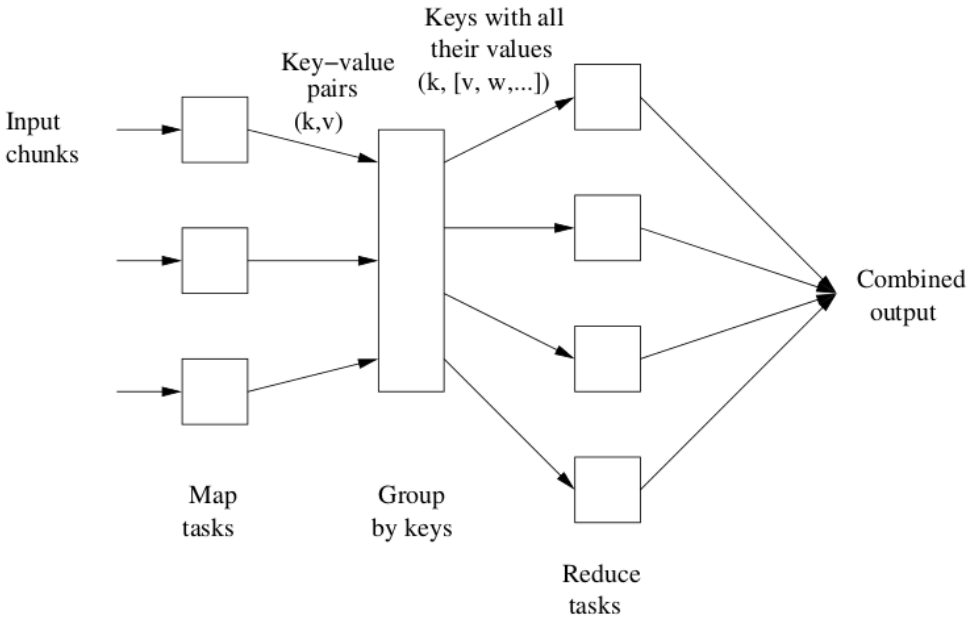

Figura 1.1. Funcionamento das funções Map e Reduce retirada do livro [Rajaraman and Ullman 2012]

O Hadoop utiliza o modelo de programação MapReduce para a execução de programas. Este modelo é baseado em duas primitivas da programação funcional: Map e Reduce. A execução MapReduce é realizada da seguinte forma: (i) A função Map recebe como entrada uma lista de pares chave-valor $\langle K_1, V_1 \rangle$ e sua saída também é uma lista de pares chave-valor intermediários $\langle K_2, V_2 \rangle$; (ii) Os pares chave-valor $\langle K_2, V_2 \rangle$ são definidos de acordo com a implementação da função Map fornecida pelo usuário, e ao final de cada tarefa Map, são coletados por um nó mestre. Em seguida, ordenados pela chave. As chaves são divididas entre todas as tarefas Reduce. Os pares chave-valor que possuem à mesma chave são designados a mesma tarefa Reduce; (iii) A função Reduce recebe como entrada todos os valores V_2 de uma mesma chave K_2 e produz como saída pares chave-valor $\langle K_3, V_3 \rangle$ que formam o resultado do processo MapReduce. As tarefas Reduce executam sobre uma chave por vez, e a forma de combinação dos valores é determinada pelo código da função Reduce dada pelo usuário. O funcionamento das funções Map e Reduce pode ser acompanhado como apresentado na Figura 1.1.

A arquitetura MapReduce é Mestre-Escravo. O nó Mestre possui várias responsabilidades, entre elas criar certo número de tarefas Map e tarefas Reduce (esses números podem ser dados de entrada pelo usuário). Além disso, o Mestre aloca as tarefas aos nós Escravos. No entanto, é desejável limitar o número de tarefas Reduce, pois cada tarefa Map cria um arquivo intermediário para cada tarefa Reduce, dessa forma se existem muitas tarefas Reduce, o número de arquivos intermediários pode ser muito grande. O nó Mestre guarda o estado de cada tarefa Map e Reduce (em espera, em execução ou finalizada). O nó Escravo reporta ao Mestre quando termina uma tarefa, e só então uma nova tarefa é escalonada pelo Mestre ao Escravo.

Os arquivos intermediários criados por uma tarefa Map (para cada tarefa Reduce)

são salvos no disco local do nó Escravo. O Mestre é informado da localização e do tamanho desses arquivos, e escalona uma ou mais tarefas Reduce que os recebem de entrada. Cada tarefa Reduce executa o código escrito pelo usuário e grava a saída do sistema de arquivo distribuído HDFS. O Hadoop gerencia a execução paralela, coordena a execução das tarefas Map e Reduce e trata as falhas durante a execução.

Um exemplo de computação MapReduce bastante difundido é a contagem do número de ocorrências que cada palavra aparece em uma coleção de documentos. Nesse exemplo, a entrada é um repositório de documentos. A função Map desse exemplo usa chaves que são do tipo Strings (as palavras) e os valores são inteiro. A tarefa Map lê o documento e o quebra em uma sequência de palavras w_1, w_2,..., w_n. Em seguida, emite uma sequência de pares chave-valor onde o valor é igual a 1. Dessa forma, a saída do Map para determinado documento do repositório é uma sequência de pares chave-valor: $\langle w_1, 1\rangle$, $\langle w_2, 1\rangle$,...,$\langle w_n, 1\rangle$.

Uma única tarefa Map pode processar muitos documentos. Então, a saída irá conter mais de uma sequência. Outro ponto importante é se uma palavra w aparece m vezes entre todos os documentos a serem lidos, então a saída conterá m pares chave-valor $\langle w, 1\rangle$. A função Reduce combina esses m pares em um único par $\langle w, m\rangle$.

1.3. Mineração de Dados em Big Data

1.3.1. Boas práticas para Algoritmos em Big Data

O processo de minerar dados é formado por um conjunto de técnicas para descoberta de conhecimento a partir de grandes bases de dados. Tais técnicas baseiam-se em modelos capazes de sumarizar dados, extrair novos conhecimentos ou realizar predições. Classificação é uma técnica de mineração de dados que está na categoria de aprendizagem supervisionada, ou seja, é fornecida uma classe a qual cada amostra do conjunto de dados de treinamento pertence. Os algoritmos que implementam essa técnica são preditivos, pois suas tarefas de mineração desempenham inferências nos dados com o intuito de fornecer previsões ou tendências, obtendo informações não disponíveis a partir dos dados disponíveis.

Outras duas técnicas também mais conhecidas, porém não supervisionadas, são Clusterização e Associação. Nestes algoritmos, o rótulo da classe de cada amostra do treinamento não é conhecido, e o número ou conjunto de classes a ser treinado pode não ser conhecido a priori, daí o fato de ser uma aprendizagem não-supervisionada. Clusterização é uma técnica de aprendizagem não supervisionada muito utilizada em mineração de dados. Assume papel importante em aplicações como reconhecimento de padrões e recuperação de informação. Os algoritmos de clusterização particionam os dados em diferentes grupos, tal que a similaridade entre elementos que pertencem ao mesmo grupo seja minimizada e a similaridade entre elementos que pertencem a grupos diferentes seja maximizada. A Associação é capaz de estabelecer regras entre os atributos. O próprio algoritmo elege atributos determinantes (lado esquerdo da regra) e os atributos resultantes (lado direito) na tarefa de mineração revelando associações entre valores dos atributos, tendo o algoritmo sua ênfase no compromisso entre precisão e cobertura [Silva 2004].

Conforme discutido anteriormente, os algoritmos de mineração de dados em sua

versão centralizada não são adequados para processar dados Big Data. Assim, podemos elencar um conjunto de boas práticas para que esses algoritmos sejam modificados e satisfatórios para o contexto de Big Data. Os trabalhos apresentados na próxima seção seguem esses princípios, o que os torna adequados para grandes volumes de dados.

Do nosso conhecimento, não há nenhuma publicação ou estudo que apresente quais os requisitos que um algoritmo deve ter para se tornar adequado ao contexto Big Data. O que conseguimos perceber foi um conjunto de boas práticas que estão presentes em muitos algoritmos que trabalham com grandes volumes de dados. Um dos nossos trabalhos futuros é caracterizar e descobrir quais outras características se enquadrariam também nesse cenário.

O primeiro passo para se obter um bom algoritmo que manipule dados Big Data é paralelizá-lo e utilizar o paradigma de divisão e conquista [Ji et al. 2012], isso pode ser alcançado utilizando MapReduce, que é capaz de manipular grandes volumes de dados e eficientemente executar o algoritmo em paralelo. Além disso, os dados devem estar distribuídos entre os nós de processamento. Essa distribuição ou particionamento dos dados deve ser balanceada entre os nós do cluster, a fim de evitar sobrecarga em um nó em particular. É necessário nesse caso observar o desvio da distribuição do dado. Além disso, é comum a utilização de replicação para paralelizar os algoritmos. No contexto de Big Data, é importante evitar replicação de dados, já que possivelmente pode resultar em crescimento exponencial da quantidade de dados a se processar. Outro ponto relevante é que os sistemas sobre os quais esses algoritmos executam também devem ser satisfatórios para a análise de dados Big Data.

Em [Cohen et al. 2009, Herodotou et al. 2011], são elencados 6 princípios que tornam um sistema adequado para Big Data, conhecido como MADDER: *Magnetic*, *Agile*, *Deep*, *Data-lifecycle-awareness*, *Elasticity* e *Robustness*. A seguir, explicamos cada um deles.

- *Magnetic*: O sistema é capaz de manipular qualquer fonte de dados independente da presença de possíveis *outliers*, de não haver esquema ou estrutura, além da ausência de valores para alguns atributos.
- *Agile*: O sistema se adapta à rápida evolução dos dados.
- *Deep*: O sistema suporta análises mais complexas, utilizando modelos estastísticos e técnicas de aprendizagem de máquina, por exemplo.
- *Data-lifecycle-awareness*: O sistema otimiza a movimentação, o armazenamento e o processamento de dados Big Data durante todo seu ciclo de vida.
- *Elasticity*: O sistema é capaz de ajustar o uso dos recursos e custos operacionais aos requisitos dos usuários e do processamento da carga de trabalho.
- *Robustness*: O sistema continua a prover serviço mesmo com possíveis adversidades como, por exemplo, falha de hardware e dados corrompidos.

Em [Begoli and Horey 2012], é possível encontrar alguns princípios para projetos em Big Data.

1.3.2. Exemplos de Algoritmos de Mineração de Dados modificados para tratar grandes volumes de dados

A seguir, reunimos os trabalhos dos 2 últimos anos das principais conferências e periódicos em banco de dados e mineração de dados. Os principais trabalhos que seguem as boas práticas apresentadas anteriormente são apresentados nas próximas subseções.

1.3.2.1. Itens Frequentes e Regras de Associação

A descoberta dos itens/palavras mais frequentes e Regras de Associação são técnicas bastante conhecidas em mineração e aplicações de banco de dados. Porém algumas dificuldades surgem quando a técnica é aplicada a um grande volume de dados, no caso do algoritmo Apriori [Agrawal et al. 1994], o tempo gasto pode ser exponencial [Yang et al. 2010]. Os dados possuem um conjunto de itens como, por exemplo, uma lista de compras de um consumidor, uma sequência de páginas visitadas por meio de um browser, uma coleção de vídeos avaliados por um espectador, um sequência de mutações nos genes de um amostra de DNA. Dado esse conjunto de dados, é interessante identificar padrões e regras de associação entre os itens que nele se encontram, ou seja, descobrir quais itens aparecem juntos frequentemente nesse conjunto de dados.

Os trabalhos [Agrawal et al. 1994] e [Yang et al. 2010] propõem uma implementação do algoritmo Apriori paralelizada via MapReduce. Porém, múltiplas iterações de MapReduce são necessárias durante a computação. Uma iteração produz um item frequente, e deve-se continuar iterando até que não existam mais itens novos a serem adicionados. Essa estratégia pode ser muito custosa para um grande volume de dados.

O trabalho [Riondato et al. 2012] também apresenta um algoritmo paralelizado usando MapReduce para mineração de itens frequentes e regras de associação, além disso não utiliza replicação de dados. Para isso, são criadas várias amostras randômicas e pequenas do conjunto de dados. O algoritmo de mineração é executado nessas amostras independentes e em paralelo. Os resultados gerados são coletados, agregados e filtrados para prover uma única saída. Como [Riondato et al. 2012] minera subconjuntos randômicos do conjunto de dados, o resultado final é uma aproximação da solução real. O artigo fez uma análise propabilística para mostrar que realmente provê garantias quanto a aproximação com a solução real. O usuário final deve fornecer a acurácia e a confiança como parâmetros e a estratégia computa uma aproximação da coleção de interesse que satisfaz esses parâmetros.

Embora [Riondato et al. 2012] não seja a primeira proposta (como vimos anteriormente) para solucionar tais problemas em mineração de dados utilizando MapReduce, o que o difere dos outros [Cryans et al. 2010, Ghoting et al. 2011, Hammoud 2011, Li et al. 2008, Li and Zhang 2011, Yang et al. 2010] são dois aspectos fundamentais. O primeiro, os outros trabalhos usam replicação de dados para paralelizar a computação, possivelmente resultando em crescimento exponencial da quantidade de dados. Ao contrário do [Riondato et al. 2012] que somente minera um número de amostras randômicas do conjunto de dados inicial. Como o MapReduce apresenta custo alto em relação à movimentação dos dados entre diferentes máquinas (denotado como “shuffling”), [Riondato et al. 2012] garante melhor desempenho que os anteriores. Outra diferença é

que os trabalhos anteriores criam as regras de associação sequencialmente após a coleta via MapReduce dos itens frequentes, ao contrário de [Riondato et al. 2012] que tanto a extração dos itens frequentes quanto a criação das regras são feitas via MapReduce. Considerando as boas práticas apresentadas anteriormente, tais diferenças mostram que [Riondato et al. 2012] é mais adequado para ser utilizado no contexto de Big Data do que os trabalhos [Cryans et al. 2010, Ghoting et al. 2011, Hammoud 2011, Li et al. 2008, Li and Zhang 2011, Yang et al. 2010].

1.3.2.2. Classificação

Os algoritmos de Classificação são capazes de realizar predição de categorias. Os mais conhecidos são árvore de decisão [Quinlan 1986], redes bayesianas [Michie et al. 1994] e os vizinhos mais próximos [Cover and Hart 1967]. Da mesma forma que para os outros algoritmos de mineração de dados apresentados anteriormente, as pesquisas também focaram em oferecer técnicas de classificação de processamento paralelo e distribuído para grandes volumes de dados.

O trabalho [He et al. 2010] propõe uma versão paralelizada dos algoritmos de árvore de decisão, redes bayesianas e vizinhos mais próximos utilizando MapReduce. Para o algoritmo dos k-vizinhos mais próximos, a computação para encontrar a similaridade entre os dados do conjunto de treino e teste é feita independentemente. Assim, o conjunto de dados fornecido como entrada pode ser particionado em n blocos que podem ser processados em diferentes nós. Além do conjunto de dados de entrada, também é fornecido o número k de vizinhos mais próximos. A função de Map calcula a similaridade das amostras dadas de entrada e a função Reduce ordena as similaridades e seleciona os k-vizinhos mais próximos.

Para o classificador utilizando redes bayesianas, o cálculo intensivo a ser realizado é o das probabilidades condicionais. Inicialmente, ocorre a fase de treino utilizando MapReduce. Na função de Map, os atributos e seus valores são identificados com valor de chave igual a 1. Na função de Reduce é contabilizada a frequência de cada atributo e seu valor. Na fase de teste também se utiliza MapReduce, a função de Map lê as probabilidades geradas pela fase de treino e calcula a probabilidade de amostras de testes pertencerem a cada categoria da classificação, bem como se a predição da categoria foi realizada corretamente ou não. Na função Reduce, calcula-se a quantidade de amostras categorizadas corretamente e erroneamente.

A ideia básica do algoritmo de árvore de decisão é recursivamente escolher o melhor atributo para dividir os nós da árvore. Após selecionar um atributo, o conjunto de treino é dividido em várias partições de acordo com o valor do atributo escolhido. Na fase de treino, o processo de MapReduce é iniciado para cada partição que recursivamente computa o melhor atributo para dividir os dados no nó corrente da árvore. As regras de decisão são armazenadas e novas regras são geradas. Na fase de teste, a função Map verifica para cada amostra, as regras de decisão e prevê uma categoria. Somente a função Map é necessária.

Alguns trabalhos como [Kim and Shim 2012], [Lu et al. 2012], [Ghoting et al. 2011], [Zhou et al. 2012], [Panda et al. 2009] apresentam a aplicação e implementação dos al-

goritmos de classificação discutidos acima utilizando MapReduce, ou ainda, aplicando algumas boas práticas para Big Data.

1.3.2.3. Clusterização

O k-means continua a ser o mais popular método de clusterização, além de ser identificado como um dos mais importantes dentre os algoritmos de mineração de dados [Wu et al. 2008]. Entretanto, em termos de qualidade e eficiência, a ordem de execução do k-means pode ser exponencial no pior caso. Além disso, a solução encontrada pode ser de um ótimo local, e possivelmente longe de se obter um ótimo global. O algoritmo k-means++ [Arthur and Vassilvitskii 2007] foi proposto como envolução do k-means e oferece melhor estratégia de inicialização dos centróides.

No k-means, são selecionados k centróides randomicamente em uma única iteração de acordo com uma distribuição específica, já em [Arthur and Vassilvitskii 2007] são k iterações e seleciona-se um ponto em cada iteração considerando que qualquer ponto não tem a mesma probabilidade de ser selecionado que os outros (pois esta probabilidade é constantemente alterada quando cada novo centróide é selecionado). Este é o ganho do algoritmo k-means++, porém devido a sua natureza sequencial, a aplicabilidade é limitada para grandes volumes de dados: deve-se iterar k vezes sobre os dados para encontrar um bom conjunto de centróides inicialmente. Idealmente, gostar-se-ia de ter um algoritmo que precise de um menor número de iterações, e que selecione mais de um ponto em cada iteração da mesma maneira que o K-means++. Em [Bahmani et al. 2012], é proposta uma estratégia paralela para reduzir o número de iterações necessárias, a fim de obter uma boa inicialização utilizando MapReduce. [Bahmani et al. 2012] segue essa intuição e encontra o ponto ideal de centróide (ou o melhor ponto de trade-off), definindo cuidadosamente o número de iterações e considerando que a probabilidade de selecionar cada ponto é diferente de qualquer outro ponto. Com esse cenário, a aplicabilidade do [Bahmani et al. 2012] é possível para grandes volumes de dados.

Um dos mais importantes algoritmos de clusterização é o DBScan(*Density-based Spatial Clustering of Application with Noise*) [Ester et al. 1996]. Suas vantagens em relação às outras técnicas de clusterização são agrupar dados em clusters de formato arbitrário, não requerer o número de clusters a priori, além de lidar com *outliers* no conjunto de dados. Em [He et al. 2011], é proposta uma implementação do DBScan em 4 estágios de MapReduce, utilizando particionamento baseado em grid. Como o conjunto de dados é particionado, o artigo também apresenta uma estratégia para junção de clusters que estão em partições diferentes e que contêm os mesmos pontos em suas fronteiras. Tais pontos estão replicados nas partições e a descoberta de quais clusters podem ser reunidos em um só cluster é averiguada a partir deles. Note que o número de pontos de fronteira replicados pode afetar a eficiência da clusterização, pois tais pontos não somente aumentam a carga de cada nó computacional, mas aumentam ainda o tempo de reunir os resultados dos diferentes nós computacionais.

Semelhante ao trabalho anterior, em [Dai and Lin 2012] também é proposta a implementação do DBScan utilizando MapReduce. O conjunto de dados é particionado de modo a manter o balanceamento de carga entre os nós computacionais, e ainda minimi-

zar os pontos de fronteiras entre as partições. Dessa forma, aumenta-se a eficiência da clusterização e do processo de junção dos clusters. O DBScan é executado em cada nó computacional sobre cada partição utilizando um índice Kd-tree. A junção de clusters que estão em partições distintas é feita quando um mesmo ponto estiver em tais partições e for designado como um *core point* em algum dos clusters. Se é detectado que dois clusters devem sofrer processo de junção, os clusters são renomeados. Tal processo também ocorre em [He et al. 2011].

Nosso estudo de caso se assemelha a ambos [He et al. 2011], [Dai and Lin 2012], pois considera o desafio de manipular grandes volumes de dados e ambas soluções utilizam MapReduce para paralelizar o algoritmo DBScan. Porém, nosso estudo de caso utiliza outra técnica de particionamento, propõe outra estratégia de junção de clusters que não necessita de replicação. Por isso, se torna mais adequado às boas práticas listadas anteriormente para Big Data.

1.4. Estudo de Caso

Para esse curso foi preparado um estudo de caso que aborda a aplicação do algoritmo de DBScan [Ester et al. 1996] em um grande volume de dados, utilizando uma plataforma de nuvem e o paralelismo obtido por meio do MapReduce. O enfoque é mostrar como a técnica de mineração foi modificada para ser aplicada em *Big Data*.

1.4.1. Descrição do Estudo de Caso

A divulgação de informações sobre o trânsito nas grandes cidades pode ser obtido de *tweets*, coletadas por meio de GPS nos veículos ou foto-sensores. A informação obtida pode ser usada tanto de forma complementar àquela gerada por câmeras e sensores físicos, orientando as ações dos agentes públicos a curto e longo prazo, quanto diretamente pelos motoristas, em tempo real ou quase-real, apoiando suas decisões quanto ao deslocamento pela cidade [Ribeiro Jr et al. 2012].

Por meio desses dados é possível analisar o comportamento do trânsito nas cidades e descobrir, por exemplo, padrões como: em quais trechos da cidade ocorrem mais congestionamentos? Tal descoberta pode auxiliar a busca por soluções eficazes de reengenharia de trânsito no contexto de cidades inteligentes. Dessa forma, a crescente popularidade desse tipo de canal de informação indica que, em pouco tempo, informações sensoriadas e transmitidas pelos próprios cidadãos podem se tornar a principal fonte de avaliação da situação do trânsito em tempo real [Ribeiro Jr et al. 2012]. Dessa forma, a obtenção de conhecimento a partir desses dados é uma tarefa importante.

O estudo de caso desse trabalho consiste em, a partir de dados de trânsito, detectar em que trechos ocorrem congestionamento em uma cidade com frequência. Para isso, o algoritmo de DBScan visto na Seção 1.3 é utilizado como solução, porém dado o grande volume de dados deve-se utilizar uma versão paralelizada. Dessa forma, outro problema consiste em como paralelizar tal algoritmo no nosso contexto de dados de trânsito. Para isso, o MapReduce é utilizado como plataforma para a execução do DBScan paralelizado. Outros trabalhos, tais como [He et al. 2011], [Dai and Lin 2012], também utilizam MapReduce para paralelizar o algoritmo de DBScan, porém apresentam estratégias e cenário diferentes do apresentado neste texto.

1.4.2. Descrição dos Dados de trânsito

Os dados utilizados nesse estudo de caso são da cidade de Fortaleza e foram coletados de sites de trânsito e de GPS. Tais dados são sobre o tráfego na cidade, informando a velocidade média em que os carros trafegam em determinado local (latitude e longitude) e o horário da coleta.

A manipulação desses dados nesse trabalho é considerada Big Data, (i) pela presença do grande volume de dados a ser gerenciado, (ii) pela variedade dos dados (dados obtidos a partir de diferentes fontes, os quais não estruturam a informação da mesma forma) e (iii) pela velocidade com que essas informações chegam aos conjuntos de dados, dado que a todo instante de tempo, os sensores coletam a posição espaço temporal do veículo, bem como a sua velocidade.

1.4.3. Infraestrutura de Computação em Nuvem

A infraestrutura utilizada nesse estudo de caso é o ambiente de nuvem privada da Universidade Federal do Ceará (UFC) que tem como plataforma o OpenNebula. O número de máquinas virtuais utilizadas foram 11 com sistema operacional Ubuntu, 8GB de memória RAM e 4 unidades de CPU. A versão do Hadoop instalada em cada máquina foi a 1.1.2 e as variáveis de ambiente foram setadas com valores como mostrados na Tabela 1.1. Em cada máquina também deve ser instalado o PostgreSQL 9.1, e ainda a biblioteca *Spatial and Geographic Objects for PostgreSQL* (PostGIS) para manipular objetos espaciais.

Tabela 1.1. Variáveis de configuração do Hadoop setadas.

Variável de Configuração	Valor
hadoop.tmp.dir	/tmp/hadoop
fs.default.name	hdfs://master:54310
mapred.job.tracker	master:54311
mapreduce.task.timeout	36000000
mapred.child.java.opts	-Xmx8192m
mapred.reduce.tasks	11
dfs.replication	5

1.4.4. Algoritmo DBScan para dados Big Data

O estudo de caso envolve diretamente a velocidade do fluxo de trânsito, uma vez que os congestionamentos se caracterizam principalmente por baixas velocidades. Assim, a primeira etapa é a partir dos dados coletados, inicialmente um filtro é aplicado para que apenas registros com velocidades abaixo de 20km/h (consideradas como congestionamento) sejam considerados no conjunto de dados e os outros sejam descartados.

A segunda etapa é considerar a dimensão espacial: latitude e longitude. Nessa fase ocorre a clusterização por proximidade geográfica dos pontos que foram selecionados de baixa velocidade. Dessa forma, será possível visualizar onde as velocidades baixas se concentram considerando a dimensão espacial, que representarão potencialmente os congestionamentos identificados. Porém, uma última etapa ainda é necessária. Além do processamento já realizado, a clusterização por tempo é fundamental para se identificar

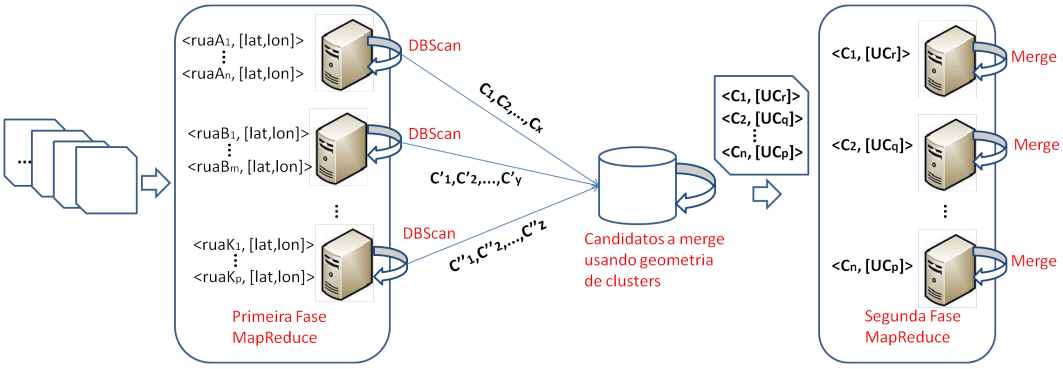

Figura 1.2. Arquitetura da nossa solução

em quais horários os congestionamentos ocorrem. No entanto, para caracterizar melhor a solução proposta, essa fase não é abordada nesse estudo de caso.

Dessa forma, a partir de um grande volume de dados de trânsito de uma cidade, o objetivo é encontrar os trechos da cidade onde há mais congestionamento. Assim, é necessário identificar grupos que possuem alta densidade de pontos nesse conjunto de dados, e ainda que apresentem baixas velocidades. A seguir, os passos para paralelizar o DBScan utilizando o modelo de programação MapReduce que constituem a implementação da solução proposta para o problema, também apresentado na Figura 1.2

- Na primeira fase de Map, cada registro do conjunto de dados será descrito como um par $\langle chave, valor \rangle$, tal que a chave diz respeito à rua em que a informação se refere, e o valor à posição geográfica (latitude e longitude) da coleta.

- Em seguida, ocorre o Reduce que recebe uma lista de valores de mesma chave, ou seja, as posições geográficas de uma mesma rua. Nessa fase, o algoritmo DBScan é aplicado. O resultado é armazenado em uma base de dados, são guardados o id de cada cluster e as informações dos seus pontos (latitude, longitude, se é *core point*, se é *noise*, por exemplo). Esses termos são definidos pela técnica do DBScan.

- Como as ruas de uma cidade se cruzam e o particionamento dos dados é feito por rua, nessa fase é necessário descobrir quais clusters de ruas diferentes se interceptam ou podem ser reunidos em apenas um cluster. Ou seja, dois clusters podem ter pontos a uma distância menor que *eps*, de tal sorte que se os dados fossem processados em um mesmo Reduce ou fossem de uma mesma partição, ter-ia-se apenas um único cluster. Dessa forma, o cluster é também armazenado como um objeto geométrico no banco de dados, e apenas objetos que estão a uma distância de no máximo *eps* poderão passar pela fase de *merge* que é a fase seguinte. Tuplas com pares de clusters candidatos a *merge* são passadas com mesma chave para o próximo MapReduce.

- Essa fase pode ser chamada de *merge* de clusters, ela também é descrita por um processo MapReduce. A função Map é a identidade. A função Reduce recebe como chave o menor id de clusters que devem ser reunidos em um só, e como valor os outros candidatos a *merge* com a chave. Nessa fase, se dois clusters devem

ser reunidos em um só, as informações dos pontos pertencentes aos clusters são atualizadas. A abordagem desse trabalho considera a possibilidade de um ponto que é *noise* em um cluster, com o merge de clusters, pode ter se tornado um *border* ou *core point*.

A seguir os algoritmos implementados da solução proposta.

1.4.5. Implementação dos algoritmos propostos

Na programação MapReduce é necessário definir duas classes que estendam de *MapReduceBase* e implementem uma das seguintes interfaces: *Mapper* e *Reducer*. Os tipos de entrada e saída das funções de Map e Reduce são parametrizadas, sendo que esses parâmetros devem implementar a interface Writable (o Hadoop já fornece alguns tipos básicos que seguem essa interface).

```
1  public static class  ClusterMapper extends MapReduceBase implements Mapper<LongWritable,
       Text, Text, Text>
2      {
3            //registro é formado por uma tripla <Rua, lat, long>
4            public void map(LongWritable key, Text value, OutputCollector<Text, Text>
                 output, Reporter reporter) throws IOException
5            {
6
7                Text street = new Text();
8                String line = value.toString();
9                String[] tokenizer = line.split(",");
10               street.set(tokenizer[0]);
11               String values=tokenizer[1]+" "+tokenizer[2];
12               value.set(values);
13           output.collect(street,value);
14           }
15     }
```

Algoritmo 1.1. Primeiro MapReduce-Fase de particionamento

O procedimento *map* no Algoritmo 1.1 *ClusterMapper* tem como objetivo converter os dados recebidos em arquivos, tal que cada registro é formado por uma tripla $\langle nomedarua, latitude, longitude\rangle$ em pares chave-valor $\langle K, V\rangle$, de tal sorte que K é o nome da rua e V é uma tupla $\langle latitude, longitude\rangle$. Tais pares são passados como entrada para a fase de Reduce a seguir. Isso é possível, pois o Hadoop por padrão particiona os arquivos a serem processados por linha do arquivo, assim cada linha é passada por vez ao procedimento *map*.

Ainda no primeiro *MapReduce* na fase de *Reduce*, o Algoritmo 1.2 recebe para uma mesma chave um conjunto de valores, sendo a chave o nome da rua e cada valor uma posição geográfica pertencente à rua que é uma tupla $\langle latitude, longitude\rangle$. O algoritmo de DBScan utilizando índice KD-tree [Bentley 1975] é aplicado aos pontos de uma mesma rua, sendo essa fase paralelizada para ruas distintas que pertencem ao conjunto de dados de entrada. Os resultados de cada cluster criado nessa fase são armazenados em uma base de dados, nesse estudo de caso utiliza-se o PostgreSQL.

A classe contendo o método principal é apresentada no Algoritmo 1.3 em que o objeto JobConf é construído. Por meio deste é possível determinar todas as configurações iniciais, além dos diretórios de entrada e saída de dados e dos parâmetros de *eps*

e *minPoints* necessários aos algoritmos anteriormente apresentados. Perceba que o Algoritmo 1.2 tem um método para obter o que foi passado de parâmetro para JobConf no método principal.

```
1  public static class ClusterReducer extends MapReduceBase implements Reducer<Text, Text,
       Text, Text> {
2          private static double eps;
3          private static int minPoints;
4          public void configure(JobConf job) {
5              eps = Double.parseDouble(job.get("eps"));
6              minPoints = Integer.parseInt(job.get("minPoints"));
7              super.configure(job);
8          }
9          public void reduce(Text key, Iterator<Text> values, OutputCollector<Text, Text>
               output, Reporter reporter) throws IOException {
10
11         String path = key.toString().replaceAll(" ", "");
12         BufferedWriter out = new BufferedWriter(new FileWriter(file));
13         String[] line;
14         while (values.hasNext()) {
15             line = values.next().toString().trim();
16             out.write(line[0]+","+line[1]+"\n");
17         }
18         out.close();
19         dbscan.loadPoints(file);
20         dbscan.createIndex();
21         dbscan.dbscan(key.toString(), eps, minPoints);
22         output.collect(key, new Text("points clustered"));
23        }
24   }
```

Algoritmo 1.2. Primeiro MapReduce-Aplicação do DBScan nos dados de uma mesma partição

```
1  public class FirstMapReduceKdTree {
2        public static void main(String[] args) throws IOException {
3            JobConf conf = new JobConf(FirstMapReduceKdTree.class);
4            conf.set("eps", args[2]);
5            conf.set("minPoints", args[3]);
6            conf.setJobName("ClusteringKdTree");
7            conf.setOutputKeyClass(Text.class);
8            conf.setOutputValueClass(Text.class);
9            conf.setMapperClass(ClusterMapper.class);
10           conf.setReducerClass(ClusterReducer.class);
11           conf.setInputFormat(TextInputFormat.class);
12           conf.setOutputFormat(TextOutputFormat.class);
13           FileInputFormat.setInputPaths(conf, new Path(args[0]));
14           FileOutputFormat.setOutputPath(conf, new Path(args[1]));
15           conf.setJarByClass(FirstMapReduceKdTree.class);
16           JobClient.runJob(conf);
17     }
18 }
```

Algoritmo 1.3. Classe que prepara os dados e que invoca as funções Map e Reduce

A fase seguinte é feita por meio de dois procedimentos armazenados no PostgreSQL. O SGBD escolhido poderia ter sido outro que tivesse suporte a dados espaciais como, por exemplo, Oracle. O primeiro procedimento quando invocado cria um objeto geométrico de cada cluster obtido da fase anterior. O segundo tem como objetivo encontrar quais clusters deveriam ser apenas um se estivessem em uma mesma partição durante a aplicação do DBScan. Assim, para descobrir quais clusters devem sofrer *merge*, é verificado quais desses objetos estão a uma distância menor ou igual que *eps*. A função utilizada para criar objetos geométricos é st-convexhull e para comparar se a distância

entre dois objetos geométricos é menor que *eps*, utiliza-se ST-Distance. Ambas fazem parte da biblioteca PostGIS.

Para descobrir se dois clusters devem sofrer merge, o Algoritmo 1 a seguir recebe dois clusters dados de entrada C_1 e C_2. Em seguida, para cada ponto $p_i \in C_1$ é verificado quais pontos $p_j \in C_2$ estão a uma distância menor ou igual a *eps* de p_i. Dessa forma, a vizinhança de cada ponto dos dois clusters é atualizada e, consequentemente, o número de vizinhos. Perceba que com isso, é possível que um ponto $p_i \in C_1$ que antes era *border*, pois em C_1 havia menos que *minPoints* vizinhos, com a junção de $C_1 \cup C_2$ pode virar *core* por ter aumentado sua vizinhança. Quando dois pontos quaisquer $p_i \in C_1$ e $p_j \in C_2$ são *core* e estão a uma distância menor que *eps*, isso é suficiente e necessário para que haja *merge* dos clusters. O Algoritmo 1 mostra como é verificado se dois clusters devem se unir, bem como atualiza o número de vizinhos e os *labels ehCore, ehBorder ou ehNoise*.

Se o retorno do Algoritmo 1 indica que deve ocorrer *merge*, os pontos pertencentes a cada cluster são renomeados para que tenham o mesmo identificador de cluster. Anteriormente a essa verificação de *merge*, o conjunto de clusters candidatos é dado de entrada para a segunda fase de MapReduce. Como toda entrada de uma função Map ou Reduce é dada por um par constituído por uma chave e a ela associada um conjunto de valores, a chave, nesse caso, é o identificador de um cluster que é chamado de representante. Os valores são todos os pares de clusters candidatos a *merge* com o representante.

```
1   public static class MergeReducer extends MapReduceBase implements Reducer<Text, Text
        , Text, Text>{
2
3       private static double eps;
4       private static int minPoints;
5       public void configure(JobConf job) {
6       eps = Double.parseDouble(job.get("eps"));
7       minPoints = Integer.parseInt(job.get("minPoints"));
8       super.configure(job);
9       }
10      public void reduce(Text key, Iterator<Text> value, OutputCollector<Text, Text>
            arg2, Reporter arg3) throws IOException {
11      String[] line;
12      String cluster;
13      ArrayList<String> clustersToMerge = new ArrayList<String>();
14      while(value.hasNext()){
15          clustersToMerge.add(value.next().toString());
16      }
17      for(int i = 0; i < clustersToMerge.size(); i++){
18          line = clustersToMerge.get(i).split(" ");
19          if(dbscan.mergeDBScan(line[0], line[1], eps, minPoints)){
20              for (int j = i+1; j < clustersToMerge.size(); j++) {
21                  if(clustersToMerge.get(j).startsWith(line[1]+" ")){
22                      cluster = clustersToMerge.get(j);
23                      cluster = cluster.replaceAll(line[1]+" ",line[0]+" ");
24                      clustersToMerge.set(j, cluster);
25                  }
26                  else if(clustersToMerge.get(j).endsWith(" "+line[1])){
27                      cluster = clustersToMerge.get(j);
28                      cluster = cluster.replaceAll(" "+line[1]," "+line[0]);
29                      clustersToMerge.set(j, cluster);
30                  }
31              }
32          }
33      }
34  }
35  }
```

Algoritmo 1.4. Segundo MapReduce-Verificação de Merge dos clusters candidatos

A segunda fase de MapReduce tem a função Map como identidade, porém a função Reduce apresentada no Algoritmo 1.4 é paralelizada de acordo com o número de representantes distintos de canditados a merge recebidos como chave. Tal algoritmo invoca o Algoritmo 1 para pares de clusters candidatos a merge e verifica se realmente ocorre merge. A atualização dos identificadores de clusters onde ocorre *merge* também é feita na lista de candidatos a *merge* do Algoritmo 1.4.

Algoritmo 1: MergeDBScan

Entrada: $C_i, C_j, eps, minPoints$
Saída: $ocorreMerge$

```
1  adj[][] ⟵ falso;
2  enquanto k ≤ C_i.tamanho faça
3      p_i ⟵ C_i[k];
4      enquanto w ≤ C_j.tamanho faça
5          p_j ⟵ C_j[w];
6          se distancia(p_i, p_j) ≤ eps então
7              p_i.vizinhanca ⟵ p_i.vizinhanca + 1;
8              p_j.vizinhanca ⟵ p_j.vizinhanca + 1;
9              adj[i][j] ⟵ true;
10         fim se
11         w ⟵ w + 1;
12     fim enqto
13     se p_i.vizinhanca ≥ minPoints ∧ ¬p_i.ehCore então
14         p_i.ehCore ⟵ true;
15     fim se
16     k ⟵ k + 1;
17 fim enqto
18 enquanto w ≤ C_j.tamanho faça
19     p_j ⟵ C_j[w];
20     se p_j.vizinhanca ≥ minPoints ∧ ¬p_j.ehCore então
21         p_j.ehCore ⟵ verdadeiro;
22     fim se
23     w ⟵ w + 1;
24 fim enqto
25 enquanto k ≤ C_i.tamanho faça
26     p_i ⟵ C_i[k];
27     enquanto w ≤ C_j.tamanho faça
28         p_j ⟵ C_j[w];
29         se p_j.ehCore ∧ p_i.ehCore ∧ adj[i][j] então
30             ocorreMerge ⟵ verdadeiro;
31         fim se
32         updatePontosNoise();
33         w ⟵ w + 1;
34     fim enqto
35     k ⟵ k + 1;
36 fim enqto
37 renomeiaClusters();
38 retornaocorreMerge;
```

Da mesma forma que apresentado no Algoritmo 1.3, o objeto JobConf é construído no método principal para invocar as funções Map e Reduce presentes nessa segunda fase de MapReduce. Por ser semelhante, omitimos tal método.

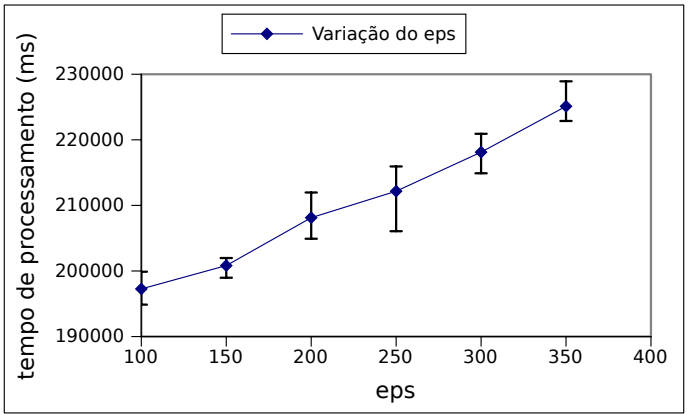

Figura 1.3. Variação do eps

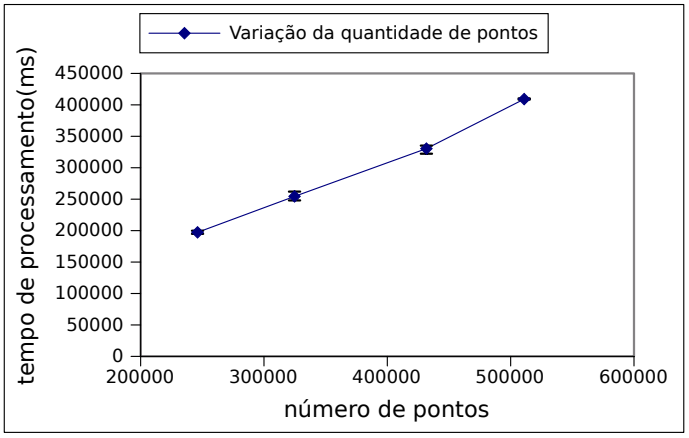

Figura 1.4. Variação da cardinalidade do conjunto de dados

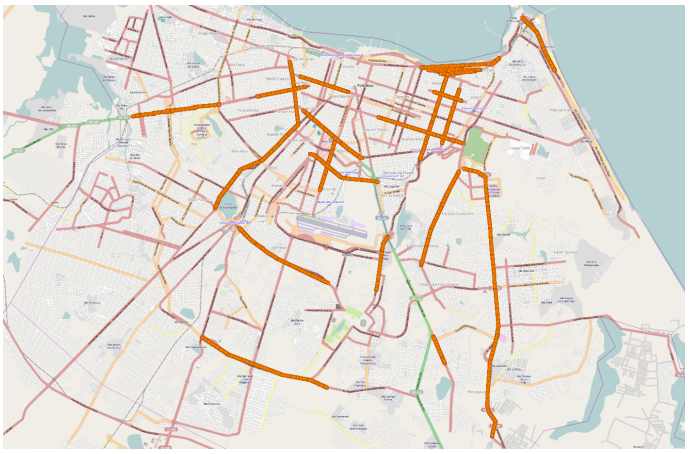

Figura 1.5. Um dos conjuntos de dados utilizados nos experimentos com 246142 pontos plotados

Figura 1.6. Clusters encontrados depois da fase de merge para 246142 pontos (eps=100 e minPoints=50)

1.4.6. Resultados da análise

O primeiro teste variou o valor de *eps* e manteve a mesma quantidade de pontos em cada rua utilizada, isto é, 246142 pontos. A variação de valores de *eps* foi 100,150,200,250,300 e 350, mantendo o valor de *MinPoints* como sendo 50. Como esperado, a Figura 1.3 mostra que com o aumento do *eps*, o tempo de processamento também aumenta, pois mais pontos na vizinhança de um *core point* podem ser expandidos.

A Figura 1.4 ilustra a variação do número de pontos do conjunto de dados de entrada versus o tempo de processamento. A quantidade de pontos usada durante os experimentos foram 246.142, 324.778, 431.698 e 510.952 pontos. Como esperado, quando a quantidade de pontos a serem processados pelo DBScan aumenta, o tempo de processamento também aumenta, mostrando que a nossa solução é escalável.

As Figuras 1.5 e 1.6 apresentam os 246.142 pontos plotados que pertencem ao conjunto de dados de entrada e os clusters encontrados após a aplicação do nosso método para *eps*=100 e *minPoints*=50, respectivamente. Na Figura 1.6, os clusters que sofreram o processo de merge estão com cor diferente de verde, note que o merge só ocorreu para os clusters ou ruas que se cruzam.

1.5. Desafios para Big Data

A utilização de Big Data apresenta diversas vantagens, mas também possui uma série de desafios que devem ser superados para sua ampla utilização. Diante do estudo realizado, alguns desafios para Big Data foram identificados. Estes desafios são oportunidades para trabalhos a serem desenvolvidos nesse contexto [Bertino et al. 2011] [QU et al. 2012].

Gartner afirmou que o principal desafio em Big Data está no volume dos dados gerados, seguido pelo desempenho do sistema e a escalabilidade necessária, além de lidar com o congestionamento na rede. Em [QU et al. 2012], são citados 3 aspectos desafiadores para o processamento em Big Data. Em primeiro lugar, Gerenciamento e Armazenamento para Big Data: as tecnologias atuais não são adequadas para satisfazer as necessidades de Big Data, um dos fatores é que a capacidade de armazenamento dos dis-

positivos cresce mais lentamente que o crescimento da quantidade de dados. Além disso, os algoritmos de gerenciamento não são eficazes para trabalhar com heterogeneidade de dados. Os processos de análise tradicionais esperam, em geral, dados homogêneos. Dessa forma, os dados devem ser estruturados como primeiro passo da análise. O processo de representação, acesso e análise eficiente de dados semiestruturados, que fazem parte do contexto de Big Data, tornam este processo mais complexo. É necessário lidar com erros e incompletude nos dados [Bertino et al. 2011].

Em segundo lugar, Computação e Análise para Big Data: no processamento de consultas em Big Data, a velocidade é fator significante, visto que não é possível percorrer todos os dados de uma base de dados Big Data em pequeno intervalo de tempo. Por exemplo, para descobrir fraude em uma transação de um cartão de crédito, é necessário detectá-la antes que a transação seja completada. Adicionalmente, a utilização de índices não é uma escolha apropriada para contornar o problema, já que estes são mais adequados para dados de tipo simples, ao passo que, em geral, Big Data apresenta estruturas de dados complexas. Assim sendo, a estratégia de paralelizar o processamento com o objetivo de diminuir o tempo de processamento é uma solução viável tratar este problema em Big Data. Se a aplicação é paralelizada, é possível aproveitar a infraestrutura dos provedores de nuvem composta por milhares de computares utilizados sob demanda e com baixo custo.

O terceiro desafio destacado em [QU et al. 2012] é a Segurança em Big Data: o volume dos dados cresce exponencialmente e acarretam grandes desafios no monitoramento e na segurança destes dados. Os métodos de segurança tradicionais devem ser repensados, pois deve ser possível realizar o processo de mineração dos dados para Big Data sem expor informações confidenciais de usuários. As tecnologias atuais para proteção da privacidade são baseadas principalmente no conjunto de dados estáticos, enquanto os dados em Big Data são alterados dinamicamente, incluindo padrão de dados, variação do atributo e adição de novos dados. Muitos dos serviços online atualmente requerem compartilhamento de informações privadas (como em aplicações como facebook), porém é importante estar atento a quem controla essa informação compartilhada e como ela pode ser usada para o casamento com outras informações. Assim, é um desafio alcançar de maneira eficaz a proteção da privacidade dos usuários neste contexto.

Um desafio citado por [Bertino et al. 2011] é a questão da escalabilidade. No passado, o objetivo era ter processadores cada vez mais rápidos, e prover os recursos necessários para lidar com o crescimento do volume de dados. Contudo, atualmente o volume de dados está aumentando em uma velocidade maior do que os recursos computacionais. Outra questão é a colaboração humana, alcançada atualmente por meio de *crowd-sourcing*, sendo a Wikipédia o melhor exemplo disso. É necessário, nesse contexto, lidar também com informações erradas por terem sido postadas por usuários mal intencionados.

Em [Fan et al. 2013], são citados os desafios apenas para análise em Big Data. Eles são decorrentes da alta dimensionalidade dos dados e pelo seu volume. O fato de haver várias dimensões pode ocasionar problemas, tais como o acúmulo de *outliers* e correlações incorretas dos dados. Alta dimensionalidade combinada com o grande volume de dados ocasiona outros problemas como, por exemplo, alto custo computacional e al-

goritmos complexos. O grande volume de dados é obtido a partir de múltiplas fontes, em diferentes pontos de tempo, utilizando diferentes tecnologias. Dessa forma, os problemas de heterogeneidade, de variações experimentais e estatísticas, nos obrigam a desenvolver procedimentos robustos e mais adaptáveis.

Em relação ao gerenciamento de grandes infraestruturas para armazenar e processar os dados em nuvem, faz-se necessário utilizar soluções eficientes para a gestão destas infraestruturas. Apesar da abordagem Hadoop apresentar bons resultados, a sua utilização é complexa e não existe uma relação entre as tarefas realizadas e o nível gerencial, por exemplo, garantir que o processamento seja realizado dentro de determinado intervalo. Assim sendo, um desafio é construir sistemas que facilitem a gestão de grandes infraestruturas e que permitam utilizar informações gerenciais. Nesta direção, a versão do Hadoop YARN aumenta o poder dos clusters Hadoop, melhorando a escalabilidade e a utilização de recursos baseados em SLAs [Hortonworks 2013].

1.6. Conclusão

A cada dia, um grande volume de dados é gerado por diferentes sistemas e dispositivos. Estes dados, chamados de Big Data, precisam ser armazenados e processados e podem ser analisados para obter tendências, por exemplo. Contudo, realizar estas tarefas não é simples, principalmente devido ao volume, velocidade e variedade. No tocante a análise para Big Data, os sistemas de gerenciamento de dados tradicionais não são adequados para lidar com uma escala tão grande. Algumas propostas exploram o paralelismo por meio de um grande número de máquinas para explorar o poder de armazenamento e computação. Neste contexto, a computação em nuvem fornece um ambiente escalável, elástico e com pagamento baseado no uso, o que permite construir soluções mais adequadas à gestão e análise de grandes volumes de dados.

Este trabalho apresentou os principais aspectos de Big Data, destacando a análise de dados. Foi apresentada a infraestrutura para análise em Big Data, com foco no Hadoop. Também foram apresentados algoritmos de mineração adaptados para Big Data e boas práticas para a concepção de novos algoritmos, assim como um estudo de caso em ambientes de computação em nuvem.

Por fim, foram discutidos alguns desafios importantes para Big Data, tais como armazenamento, processamento, análise, segurança, gerenciamento de grandes infraestruturas. Apesar de algumas soluções atenuarem estes desafios, ainda existe muito a fazer, principalmente porque o volume de dados e a velocidade com que estes dados são gerados cresce de forma exponencial. Estes desafios geram oportunidades de pesquisa, de forma que a análise para Big Data possa ser utilizada pelas empresas, melhorando seus produtos, serviços e a qualidade de vida de todos.

Referências

[Agrawal et al. 2011] Agrawal, D., Das, S., and El Abbadi, A. (2011). Big data and cloud computing: current state and future opportunities. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 530–533, New York, NY, USA. ACM.

[Agrawal et al. 1994] Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining asso-

ciation rules. In *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, volume 1215, pages 487–499.

[Arthur and Vassilvitskii 2007] Arthur, D. and Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms. Society for Industrial and Applied Mathematics*, pages 1027–1035.

[Bahmani et al. 2012] Bahmani, B., Moseley, B., Vattani, A., Kumar, R., and Vassilvitskii, S. (2012). Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633.

[Begoli and Horey 2012] Begoli, E. and Horey, J. (2012). Design principles for effective knowledge discovery from big data. In *Software Architecture (WICSA) and European Conference on Software Architecture (ECSA), 2012 Joint Working IEEE/IFIP Conference on*, pages 215–218. IEEE.

[Bentley 1975] Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. In *Communications of the ACM*, volume 18, pages 509–517. ACM.

[Bertino et al. 2011] Bertino, E., Bernstein, P., Agrawal, D., Davidson, S., Dayal, U., Franklin, M., Gehrke, J., Haas, L., Halevy, A., Han, J., et al. (2011). Challenges and opportunities with big data.

[Coelho da Silva 2013] Coelho da Silva, T. L. (2013). Processamento elástico e não-intrusivo de consultas em ambientes de nuvem considerando o sla. *Dissertação de Mestrado - Universidade Federal do Ceará*, page 55.

[Cohen et al. 2009] Cohen, J., Dolan, B., Dunlap, M., Hellerstein, J. M., and Welton, C. (2009). Mad skills: new analysis practices for big data. *Proceedings of the VLDB Endowment*, 2(2):1481–1492.

[Cover and Hart 1967] Cover, T. and Hart, P. (1967). Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27.

[Cryans et al. 2010] Cryans, J.-D., Ratte, S., and Champagne, R. (2010). Adaptation of apriori to mapreduce to build a warehouse of relations between named entities across the web. In *Advances in Databases Knowledge and Data Applications (DBKDA), 2010 Second International Conference on*, pages 185–189. IEEE.

[Dai and Lin 2012] Dai, B.-R. and Lin, I.-C. (2012). Efficient map/reduce-based dbscan algorithm with optimized data partition. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 59–66. IEEE.

[Dean and Ghemawat 2008] Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

[Ester et al. 1996] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231.

[Fan et al. 2013] Fan, J., Han, F., and Liu, H. (2013). Challenges of big data analysis. *arXiv preprint arXiv:1308.1479*.

[Ghoting et al. 2011] Ghoting, A., Kambadur, P., Pednault, E., and Kannan, R. (2011). Nimble: a toolkit for the implementation of parallel data mining and machine learning algorithms on mapreduce. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 334–342. ACM.

[Hammoud 2011] Hammoud, S. (2011). Mapreduce network enabled algorithms for classification based on association rules.

[He et al. 2010] He, Q., Zhuang, F., Li, J., and Shi, Z. (2010). Parallel implementation of classification algorithms based on mapreduce. In *Rough Set and Knowledge Technology*, pages 655–662. Springer.

[He et al. 2011] He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S., and Fan, J. (2011). Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce. In *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pages 473–480. IEEE.

[Herodotou et al. 2011] Herodotou, H., Lim, H., Luo, G., Borisov, N., Dong, L., Cetin, F. B., and Babu, S. (2011). Starfish: A self-tuning system for big data analytics. In *CIDR*, volume 11, pages 261–272.

[Hortonworks 2013] Hortonworks (2013). *Hadoop Yarn*. `http://hortonworks.com/hadoop/yarn/`.

[Ji et al. 2012] Ji, C., Li, Y., Qiu, W., Jin, Y., Xu, Y., Awada, U., Li, K., and Qu, W. (2012). Big data processing: Big challenges and opportunities. *Journal of Interconnection Networks*, 13(03n04).

[Kim and Shim 2012] Kim, Y. and Shim, K. (2012). Parallel top-k similarity join algorithms using mapreduce. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 510–521. IEEE.

[Li et al. 2008] Li, H., Wang, Y., Zhang, D., Zhang, M., and Chang, E. Y. (2008). Pfp: parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 107–114. ACM.

[Li and Zhang 2011] Li, L. and Zhang, M. (2011). The strategy of mining association rule based on cloud computing. In *Business Computing and Global Informatization (BCGIN), 2011 International Conference on*, pages 475–478. IEEE.

[Lin and Dyer 2010] Lin, J. and Dyer, C. (2010). Data-intensive text processing with mapreduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177.

[Lu et al. 2012] Lu, W., Shen, Y., Chen, S., and Ooi, B. C. (2012). Efficient processing of k nearest neighbor joins using mapreduce. *Proceedings of the VLDB Endowment*, 5(10):1016–1027.

[Mell and Grance 2009] Mell, P. and Grance, T. (2009). The nist definition of cloud computing. *NIST*, page 50.

[Michie et al. 1994] Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). Machine learning, neural and statistical classification.

[Panda et al. 2009] Panda, B., Herbach, J. S., Basu, S., and Bayardo, R. J. (2009). Planet: massively parallel learning of tree ensembles with mapreduce. *Proceedings of the VLDB Endowment*, 2(2):1426–1437.

[Pavlo et al. 2009] Pavlo, A., Paulson, E., Rasin, A., Abadi, D. J., DeWitt, D. J., Madden, S., and Stonebraker, M. (2009). A comparison of approaches to large-scale data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 165–178, New York, NY, USA. ACM.

[QU et al. 2012] QU, W., LI, K., XU, Y., AWADA, U., JIN, Y., QIU, W., LI, Y., and JI, C. (2012). Big data processing: Big challenges and opportunities. *Journal of Interconnection Networks*, 13(03n04):1250009.

[Quinlan 1986] Quinlan, J. R. (1986). Induction of decision trees. *Machine learning*, 1(1):81–106.

[Rajaraman and Ullman 2012] Rajaraman, A. and Ullman, J. D. (2012). *Mining of massive datasets*. Cambridge University Press.

[Ribeiro Jr et al. 2012] Ribeiro Jr, S. S., Rennó, D., Gonçalves, T. S., Davis Jr, C. A., Meira Jr, W., and Pappa, G. L. (2012). Observatório do trânsito: sistema para detecção e localização de eventos de trânsito no twitter. *SBBD*, pages 81–88.

[Riondato et al. 2012] Riondato, M., DeBrabant, J. A., Fonseca, R., and Upfal, E. (2012). Parma: a parallel randomized algorithm for approximate association rules mining in mapreduce. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 85–94. ACM.

[Schadt et al. 2010] Schadt, E. E., Linderman, M. D., Sorenson, J., Lee, L., and Nolan, G. P. (2010). Computational solutions to large-scale data management and analysis. *Nature reviews. Genetics*, 11(9):647–657.

[Silva 2004] Silva, M. (2004). Mineração de dados-conceitos, aplicações e experimentos com weka. *Livro da Escola Regional de Informática Rio de Janeiro-Espírito Santo. Rio Janeiro: SBC*.

[Sousa et al. 2010] Sousa, F. R. C., Moreira, L. O., Macêdo, J. A. F., and Machado, J. C. (2010). Gerenciamento de dados em nuvem: Conceitos, sistemas e desafios. In *SBBD*, pages 101–130.

[Suciu 2013] Suciu, D. (2013). Big data begets big database theory. In *BNCOD*, pages 1–5.

[White 2012] White, T. (2012). *Hadoop: the definitive guide*. O'Reilly.

[Wu et al. 2008] Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Philip, S. Y., et al. (2008). Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37.

[Wu et al. 2013] Wu, X., Zhu, X., Wu, G.-Q., and Ding, W. (2013). Data mining with big data. *IEEE Transactions on Knowledge and Data Engineering*, 99(PrePrints):1.

[Yang et al. 2010] Yang, X. Y., Liu, Z., and Fu, Y. (2010). Mapreduce as a programming model for association rules algorithm on hadoop. In *Information Sciences and Interaction Sciences (ICIS), 2010 3rd International Conference on*, pages 99–102. IEEE.

[Zhou et al. 2012] Zhou, L., Wang, H., and Wang, W. (2012). Parallel implementation of classification algorithms based on cloud computing environment. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 10(5):1087–1092.

Minicurso
2

Introdução à Mineração de Opiniões: Conceitos, Aplicações e Desafios

Karin Becker, Diego Tumitan

Programa de Pós-Graduação em Ciência da Computação - Instituto de Informática
Universidade Federal do Rio Grande do Sul (UFRGS)
{karin.becker,dctumitan}@inf.ufrgs.br

Abstract

Social networks, foruns, microblogs, on-line newspapers and sites for the evaluation of products and services are examples of the many platforms available in the web, which allow users to express their opinions. Opinion mining, or sentiment analysis, is a recent field that aims at identifying opinionative content, and determine the sentiment, perception or attitude of the public with regard to the target of the opinion. This Chapter discusses the motivation for the field and its main applications; presents the issues of opinion mining and related concepts; describes a set of techniques for extracting opinions and their target, classifying their sentiment, and summarizing opinions; and concludes with challenges and research perspectives for this field.

Resumo

Redes sociais, fóruns, tweets, jornais on-line, sites para avaliação de produtos e serviços, são alguns exemplos de plataformas através das quais usuários têm expresso suas ideias e opiniões. A mineração de opiniões, ou análise de sentimentos, é uma disciplina recente que busca identificar conteúdo de opinião, e determinar o sentimento, percepção ou atitude do público em relação ao alvo desta opinião. Este capítulo discute a motivação para a área e suas principais aplicações; apresenta o problema de mineração de opiniões e conceitos relacionados; descreve um conjunto de técnicas para extrair opiniões e seu alvo, classificar seu sentimento ou polarizá-las, e sumarizar as opiniões; e conclui com desafios e perspectivas de pesquisa na área.

1.1. Introdução

Opiniões têm grande influência sobre o comportamento das pessoas. Decisões simples como qual carro comprar, qual filme ver, ou em qual ação investir são frequentemente baseadas em opiniões de pessoas próximas, de especialistas, ou de estudos conduzidos por instituições especializadas. Organizações baseiam suas estratégias de negócio e investimentos na opinião de seus clientes sobre seus produtos ou serviços. A importância da opinião é tão grande que muitas empresas (e.g. marketing, relações públicas, pesquisas) têm seu negócio voltado à obtenção deste tipo de informação. Tradicionalmente, a resposta a questões envolvendo a opinião pública envolve técnicas como pesquisa de campo, telefonemas ou questionários escritos. Estas técnicas envolvem custos, são restritas a um grupo focal bem definido ou amostra, seu retorno é demorado e muitas vezes, pouco eficaz. A latência da opinião também é alta, devido ao longo tempo necessário entre a coleta dos dados brutos, sua análise e disponibilização dos resultados.

A explosão das mídias sociais alterou este cenário, disponibilizando a indivíduos e organizações conteúdo de opinião diversificado e em grandes volumes. Usuários da web têm a oportunidade de registrar e divulgar suas ideias e opiniões através de comentários, fóruns de discussão, blogs, Twitter, redes sociais, entre outros. Isto aumenta as opções dos indivíduos na busca de opiniões, pois não estão mais limitados a sua rede pessoal de contatos (e.g. familiares, amigos, conexões profissionais) ou a opiniões de especialistas disponíveis publicamente (e.g. revistas, jornais). No tocante às organizações, isto significa oportunidades de ampliar as fontes de opinião quantitativa ou qualitativamente, tornar mais baratas as formas de coleta, e reduzir o tempo necessário para disponibilização da informação. Contudo, o grande volume de informação produzido diariamente implica a necessidade de métodos e ferramentas capazes de processar automaticamente não apenas o conteúdo das publicações, mas também a opinião e sentimento que expressam.

A *mineração de opiniões*, também chamada de *análise de sentimento* ou *análise de subjetividade* [20, 34, 23], é uma disciplina recente que congrega pesquisas de mineração de dados, linguística computacional, recuperação de informações, inteligência artificial, entre outras. A mineração de opinião é definida em [19] como qualquer estudo feito computacionalmente envolvendo opiniões, sentimentos, avaliações, atitudes, afeições, visões, emoções e subjetividade, expressos de forma textual. O problema da mineração de opiniões pode ser estruturado em termos das seguintes tarefas genéricas [34]: a) identificar as opiniões expressas sobre determinado assunto ou alvo em um conjunto de documentos; b) classificar a orientação ou polaridade desta opinião, isto é, se tende a positiva ou negativa; e c) apresentar os resultados de forma agregada e sumarizada. A polaridade da opinião define o sentimento, percepção ou atitude do público em relação ao alvo da opinião.

Como mostra a Figura 1.1, boa parte dos trabalhos nesta área concentra-se no desenvolvimento de técnicas para detecção e sumarização automáticas de opinião sobre revisões de produtos e serviços [17, 24, 37, 12, 15, 2]. Posteriormente, o foco ampliou-se para entidades específicas (e.g. políticos, celebridades, marcas) em redes sociais [22, 10] ou notícias [16]. Várias ferramentas foram desenvolvidas com este objetivo, entre elas

TweetSentiments[1], Sentimonitor[2], Google Shopping[3], UberVU[4], etc. A mineração de opinião sobre textos menos estruturados, como notícias e blogs, também tem sido alvo de bastante atenção [6, 7, 18]. Outra vertente importante é a análise de opinião do Twitter, em particular visando estabelecer modelos preditivos [3, 8, 35].

O propósito deste capítulo é apresentar os conceitos subjacentes à mineração de opiniões, e caracterizar cada uma das etapas do processo, descrevendo os problemas envolvidos, e as técnicas que podem ser utilizadas. Etapas, problemas e técnicas serão ilustrados através de trabalhos representativos encontrados na literatura, considerando diferentes tipos de texto: revisões de produtos, notícias e mídias sociais.

O restante deste capítulo está estruturado como segue. Na Seção 1.2 é descrita a fundamentação teórica, com o detalhamento dos conceitos que caracterizam a opinião, dos diferentes níveis de análise de opinião, e dos problemas de análise textual relacionados. A Seção 1.3 descreve o processo de mineração de opiniões em termos de suas principais etapas: identificação, classificação de polaridade e sumarização. As diferentes abordagens para a classificação de polaridade são apresentadas na Seção 1.4. A mineração de opiniões sobre diferentes tipos de dados é então discutida através de exemplos na Seção 1.5, considerando revisões de produtos, notícias e mídias sociais. Finalmente, a Seção 1.6 apresenta conclusões e direções futuras.

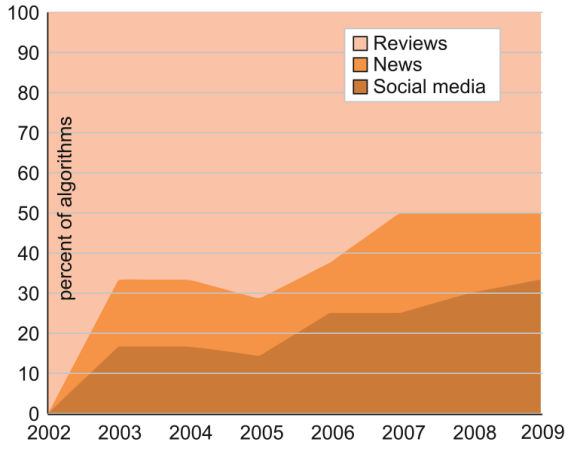

Figura 1.1. Distribuição dos trabalhos sobre domínios alvo (Fonte: [34]).

1.2. Conceitos

1.2.1. Definições

A mineração de opiniões opera sobre porções de texto de quaisquer tamanho e formato, tais como páginas web, posts, comentários, *tweets*, revisões de produto, etc. Toda opinião é composta de pelo menos dois elementos chave: um *alvo* e um *sentimento* sobre este alvo [20]. Um alvo pode ser uma entidade, aspecto de uma entidade, ou tópico, representando um produto, pessoa, organização, marca, evento, etc. Já um sentimento representa

[1]http://twittersentiment.appspot.com
[2]http://www.sentimonitor.com
[3]http://www.google.com/shopping
[4]http://www.ubervu.com

uma atitude, opinião ou emoção que o autor da opinião tem a respeito do alvo. A *polaridade* de um sentimento corresponde a um ponto em alguma escala que representa a avaliação positiva, neutra ou negativa do significado deste sentimento [34].

Mais formalmente, uma opinião corresponde a uma quíntupla $(\mathbf{e}_i, \mathbf{a}_{ij}, \mathbf{s}_{ijkl}, \mathbf{h}_k, \mathbf{t}_l)$ [19], onde:

- $\mathbf{e}_i$: é o nome de uma entidade;
- $\mathbf{a}_{ij}$: é um aspecto da entidade $\mathbf{e}_i$ (opcional);
- $\mathbf{s}_{ijkl}$: é a polaridade do sentimento sobre aspecto $\mathbf{a}_{ij}$ que tem como alvo a entidade $\mathbf{e}_i$;
- $\mathbf{h}_k$: é o detentor do sentimento (i.e. quem expressou o sentimento), também chamado de fonte de opinião;
- $\mathbf{t}_l$: é o instante no qual a opinião foi expressa por $\mathbf{h}_k$.

O conceito de *aspecto*, também denominado característica (*feature*) ou propriedade, permite que uma entidade seja vista através de diferentes perspectivas ou atributos, ou como uma hierarquia de partes e subpartes [19]. Por exemplo, considere o comentário abaixo sobre um hotel, retirado do *site* Booking.com. Os aspectos deste hotel são o quarto, a vista, e o *wi-fi*. Assim, o sentimento expresso neste comentário é representado por quatro quíntuplas, sendo que uma refere-se ao hotel, e as demais, a aspectos específicos deste.

- *Cláudio - 3 de setembro de 2013: "Adorei o hotel **Vida Mansa**. Os **quartos** do hotel são super espaçosos, com uma **vista** linda para o mar. Pena que não há **wi-fi** nos quartos"*.
  – (Vida Mansa, geral, positivo, Cláudio, 03/09/2013)
  – (Vida Mansa, quarto, positivo, Cláudio, 03/09/2013)
  – (Vida Mansa, vista, positivo, Cláudio, 03/09/2013)
  – (Vida Mansa, wi-fi, negativo, Cláudio, 03/09/2013)

Os termos *sentimento* e *opinião* frequentemente são usados como sinônimo neste contexto. A polaridade de um sentimento pode ser classificada em classes discretas (e.g. positiva, negativa ou neutra), ou como um intervalo que representa a intensidade deste sentimento, tipicamente [-1, 1]. Já o termo *emoção* é usado para designar as percepções e pensamentos subjetivos de uma pessoa, tais como raiva, desgosto, medo, alegria, tristeza e surpresa, não representando necessariamente um posicionamento ou uma atitude em relação ao alvo.

1.2.2. Níveis de Análise Textual

A detecção do sentimento em um texto pode ocorrer em diferentes granularidades, sendo que a decisão do nível está sujeita ao contexto e aplicação. A análise pode ser em nível de [20]:

- *Documento*: nesse nível, a tarefa é classificar se um documento, tratado como um todo, expressa um sentimento positivo ou negativo. Esta granularidade é adequada quando o documento trata de uma única entidade, por exemplo, um documento que forneça uma opinião sobre um dado produto;

- *Sentença*: determina o sentimento de uma sentença específica de um documento. Este nível é bastante utilizado quando um mesmo documento contém opiniões sobre várias entidades. Ele também permite identificar e distinguir sentenças objetivas (fatos) e subjetivas (opiniões). Alguns autores sugerem ir além do nível de sentença, dividindo-a em cláusulas (e.g. "A cidade é péssima, mas a população é muito simpática") [33];

- *Entidade e Aspecto*: este nível foca na opinião expressa, independentemente dos construtos utilizados para expressá-la (e.g. documentos, sentenças, orações). Neste caso, o alvo da opinião pode ser uma entidade, ou algum de seus aspectos. No exemplo "Adoro minha câmera X porque a qualidade de sua lente é excepcional. Pena que o preço seja tão alto", observa-se que existem três opiniões em 2 sentenças: sobre a câmera X, e sobre dois de seus aspectos (preço e lente). Apenas a opinião sobre o preço é negativa, sendo que a opinião sobre a lente, e sobre a câmera em geral são positivas. Este é o nível mais complexo de análise, o qual tem sido bastante estudado no contexto de revisões de produtos e serviços (e.g. [17, 33, 15]).

1.2.3. Tipos de Opiniões e Análise Linguística

Opiniões referem-se a conteúdo subjetivo, escrito em linguagem natural. A forma como as opiniões estão expressas influencia diretamente a habilidade de processá-las corretamente. A mineração de opiniões tem origens em comum com a linguística computacional, com a qual compartilha problemas e desafios [20].

Opiniões podem ser *regulares* ou *comparativas*; *diretas* ou *indiretas*, e *implícitas* ou *explícitas*. Em opiniões regulares, o autor da opinião expressa um sentimento, atitude, emoção ou percepção sobre um alvo ("Este filme é muito bom"). Já as opiniões comparativas expressam o sentimento com base na relação de similaridades ou diferenças entre duas ou mais entidades, ou preferência quanto a algum aspecto compartilhado ("O teclado deste telefone é muito melhor do que o do meu telefone antigo"). As opiniões podem ser diretas ("Este suco é muito bom"), ou indiretas ("minha gripe piorou depois que tomei este remédio" – implicando opinião negativa sobre o remédio através do seu efeito sobre a gripe). Finalmente, opiniões explícitas expressam diretamente o sentimento, enquanto que as implícitas sugerem-no indiretamente ("Formou-se um vale no colchão que comprei na semana passada"). A maioria dos trabalhos concentra-se em opiniões regulares, diretas e explícitas, por serem mais fáceis de serem tratadas.

Um problema enfrentado é a co-referência, onde diferentes tipos de menções designam a uma mesma entidade. Por exemplo, as expressões "Dilma", "Presidenta", "Presidenta Dilma Rousseff" referem-se à mesma pessoa, devendo ser reconhecidas e unificadas. Nesse mesmo contexto, outra dificuldade é a resolução de pronomes, com o objetivo de relacionar um pronome a uma determinada entidade. Por exemplo, no texto "Paris é uma cidade maravilhosa. Ela é um excelente lugar para se visitar. Seus restaurantes são

muito reconhecidos", os pronomes "ela" e "seus" referem-se a Paris. O tratamento da co-referência e dos pronomes é extremamente importante para a análise de sentimentos nos níveis de sentença e de aspectos, já que estes níveis analisam o sentimento de forma isolada (i.e. cada sentença ou opinião), com efeito direto sobre a revocação.

É comum o uso de palavras de sentimento (e.g. ótimo, péssimo) para detectar opiniões, mas seu uso não é condição necessária, nem suficiente, para detectar uma opinião, e classificar sua polaridade. Primeiro, as palavras de opinião podem ser positivas ou negativas de acordo com o contexto. Por exemplo, na sentença "este smartphone é muito caro", a palavra de sentimento "caro" é negativa, enquanto que em "este amigo me é muito caro", ela é positiva. Segundo, nem toda opinião é expressa com palavras de sentimento (e.g. "comprei este casaco na semana passada, e já está cheio de bolinhas"), ou vice-versa (e.g. "se encontrar um bom livro, vou lê-lo"). A negação é outra questão que deve ser tratada, já que inverte o sentido da opinião ("Este filme não é nada bom").

Finalmente, a ironia/sarcasmo é um dos problemas mais difíceis de se tratar ("Ontem vendo o horário político, vi propostas bem novas e inovadoras: melhorar a saúde, educação e emprego. Por que ninguém havia prometido isto antes?"). O uso de sarcasmo é muito comum em alguns domínios, como discussões políticas e esportivas, opiniões sobre arte (filmes, bandas), etc [27, 37, 5]. Os trabalhos encontrados na literatura para identificação de sarcasmo/ironia fazem uso de artifícios, como: frequência do sinal de exclamação e interrogação, palavras capitalizadas, interjeições (e.g. "ah, oh, yeah"), *emoticons* (e.g. ";-)") e superlativos [11, 20].

1.3. Etapas da Mineração da Opinião

A mineração de opinião pode ser caracterizada em termos de três grandes tarefas [34]: a) identificar (tópicos, sentenças opinativas), b) classificar a polaridade do sentimento, e c) sumarizar. Este processo é esboçado na Figura 1.2.

1.3.1. Identificação

Dado um conjunto de textos extraídos de alguma fonte (e.g. jornais, redes sociais, plataformas de revisão de produtos/serviços), a etapa de *identificação* consiste em encontrar os tópicos existentes (e possivelmente seus aspectos), e possivelmente associá-los com o respectivo conteúdo subjetivo. A forma de identificar as entidades, aspectos e sentimento são dependentes da granularidade escolhida para análise, e os algoritmos utilizados podem ser distintos daqueles propostos para recuperação de documentos opinionativos [23, 34].

A complexidade da identificação do alvo da opinião depende em muito da mídia considerada, e de seu grau de estruturação. A aplicação mais frequente em mineração de opiniões é a de revisão de produtos e serviços, porque o alvo pode ser mais facilmente identificado. Assume-se que todo o documento refere-se a uma única entidade, o alvo da revisão, sendo que o desafio está em identificar os aspectos desta entidade, se a análise for nesta granularidade.

Já em jornais, blogs ou posts, não se conhece *a priori* as entidades envolvidas, podendo inclusive envolver muitas entidades na mesma porção de texto. Na situação mais simples, pode-se restringir a identificação a entidades pré-definidas, como a busca

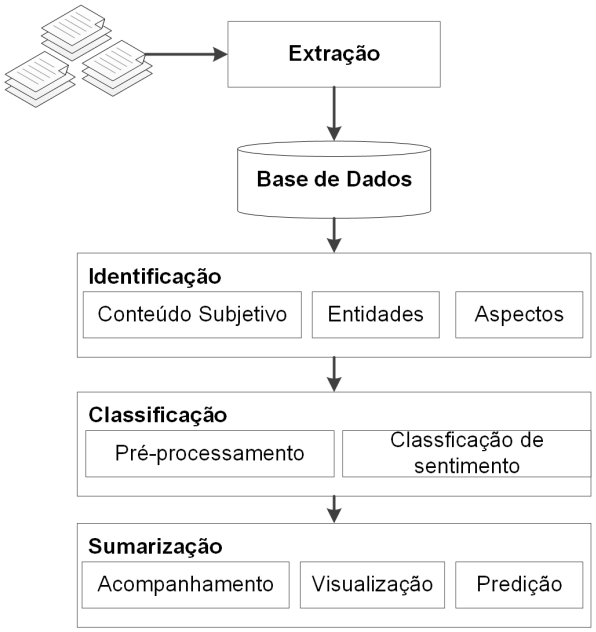

Figura 1.2. Etapas da Mineração de Opinião.

de celebridades, atletas, políticos ou marcas. Um dos problemas neste caso é resolver os problemas de co-referência e de pronomes, já mencionados na Seção 1.2.3. Em mídias sociais, a co-referência pode ser um problema acentuado, pois as menções podem ser muito informais (apelidos, gírias com significado local ou temporal, *hashtags*, etc). Por exemplo, o termo "tricolor" no estado de São Paulo refere-se ao São Paulo Futebol Clube, enquanto que no estado do Rio Grande do Sul, esse termo designa o Grêmio Foot-Ball Porto Alegrense. Se a identificação não for direcionada a alvos pré-definidos, pode-se ainda utilizar técnicas de identificação de entidades nomeadas da recuperação de informações [26, 1].

Finalmente, esta tarefa pode envolver também o discernimento entre conteúdo ou sentenças com ou sem opinião, visando melhorar os resultados da próxima etapa. Isto é bastante comum quando o nível de análise é de granularidade menor. O critério utilizado para determinar o conteúdo de opinião é quase sempre a identificação de palavras de sentimento (e.g. "Eu recomendo este filme"), ou de classes de palavras candidatas a expressar sentimento (e.g. adjetivos).

1.3.2. Classificação da Polaridade

O problema de *classificação de sentimento*, também denominado *classificação de polaridade*, é frequentemente um problema de classificação binário, isto é, que classifica um dado texto em uma de duas classes: *positivo* ou *negativo*. No entanto, classes adicionais podem ser consideradas para que a análise seja mais robusta, ou para aumentar o nível de detalhe dos resultados. Assim, estas classes podem ser desdobradas em classificações com diferentes graus de intensidade (e.g. muitoPositivo, moderadamentePositivo), ou em intervalos numéricos representando um grau de intensidade [34]. Neste último caso, a divisão do sentimento está relacionada à capacidade de definir algum limiar para distinguir os níveis de sentimento.

Outra abordagem é considerar a categoria neutra, que engloba textos sem uma tendência clara quanto a sua polaridade, ou simplesmente sem sentimento. Neste último caso, é a etapa de classificação de polaridade que tem como responsabilidade identificar textos sem sentimento de acordo com suas propriedades. Mas, como já mencionado, é mais frequente que este tipo de texto já tenha sido descartado na etapa anterior de identificação, porque a qualidade dos resultados da classificação costuma ser maior [34].

Para a classificação da polaridade, diferentes abordagens são propostas na literatura, as quais são discutidas com maiores detalhes na Seção 1.4. Cada técnica pode necessitar de operações de pré-processamento e transformação específicas, tais como reconhecimento de construtos sintáticos, reconhecimento de n-gramas, extração de *features*, eliminação de termos irrelevantes, transformação em vetor de termos, etc.

Independente da abordagem empregada, a classificação da polaridade não é um problema trivial. Entre os principais desafios estão:

- o uso de palavras de sentimento pode ser enganoso, como discutido na Seção 1.2.3, já que existem opiniões sem o uso de palavras de sentimento, e vice-versa. Ainda, a polaridade de alguns termos são dependentes de contexto;
- muitos domínios são caracterizados pelo uso frequente de ironias ou sarcasmo, onde o sentido implícito é exatamente oposto ao sentimento expresso explicitamente. Outros domínios (e.g. debates políticos, críticas culturais) estabelecem uma opinião positiva por oposição a uma argumentação negativa (ou vice-versa) [5, 24, 37];
- a opinião pode depender do observador. Por exemplo, a opinião representada na sentença "As ações da Petrobrás subiram" é positiva para quem detém este tipo de ação, mas pode ser péssima para quem deixou de investir nelas;
- a polaridade de conteúdo subjetivo nem sempre é objeto de consenso. Por exemplo, em anotações feitas por humanos, dificilmente o consenso é maior que 75% [9, 18, 39, 24];
- a classificação é bastante dependente da extração das *features* do texto, a qual deve lidar com as várias questões da língua natural já discutidas na Seção 1.2.3.

É importante ressaltar que um texto neutro é diferente de um texto não polarizado. Um texto não polarizado é aquele no qual não há elementos suficientes para poder classificá-lo, e consequentemente, para o qual a tarefa de classificação não consegue chegar à conclusão sobre sua polaridade. Isso geralmente acontece quando o conteúdo analisado possui ruídos, tais como erros tipográficos e sentenças incompletas [13].

1.3.3. Sumarização

Para poder identificar opinião média ou prevalecente de um grupo de pessoas sobre um determinado tópico/entidade, a opinião expressa por uma única pessoa não é suficiente, sendo necessário analisar uma grande quantidade de opiniões [20]. É necessário a criação de métricas e sumários que quantifiquem a diversidade de opiniões encontradas a respeito

um mesmo alvo. Este é o objetivo desta etapa, onde são criadas métricas que representem o sentimento geral, as quais podem ser visualizadas ou servir de entrada para outras aplicações.

Em revisão de produtos, um sumário de um determinado produto pode ajudar um consumidor a identificar seus respectivos pontos fortes e fracos, levando em consideração a experiência prévia de outras pessoas, expressas em suas opiniões. Esse tipo de recurso pode ser encontrado, por exemplo, no Google Shopping, que automaticamente extrai, analisa e agrega os aspectos de revisões de produtos disponibilizados por diferentes lojas de comércio eletrônico (e.g. Best Buy). A Figura 1.3 ilustra um exemplo de resultado desta ferramenta, onde aspectos como facilidade de uso, *design* e tamanho, foram identificados e exemplificados com uma sentença que demonstra o sentimento predominante sobre o mesmo. Além disso, a ferramenta ainda mostra uma nota geral sobre o produto, baseada em uma classificação de estrelas.

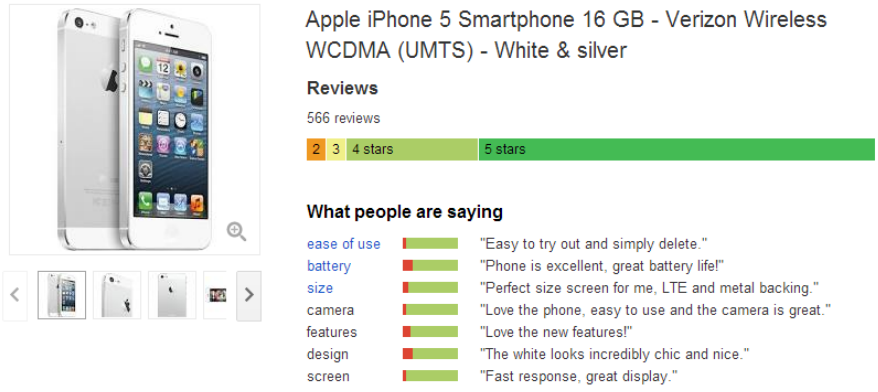

Figura 1.3. Exemplo de sumarização de opiniões de aspectos produto extraídas de revisões (Fonte: Google Shopping).

Outra forma de sumarização, comum em aplicações que extraem de mídias sociais o sentimento do público em geral sobre uma determinada entidade (e.g. uma marca, produto, político, celebridade), é apresentar o sentimento na forma de relógios, ou associá-lo a informações temporais ou geográficas. Normalmente este tipo de mídia reflete o que as pessoas pensam sobre o alvo, dado algum evento. Por exemplo, o lançamento de um novo produto terá impacto nas redes sociais, que expressarão reações a esse acontecimento através de posts, comentários, *tweets*, endossos, etc. A empresa pode aproveitar-se disto para avaliar se este produto foi bem recepcionado pelo mercado. Um exemplo é a ferramenta UberVB, que sumariza e acompanha as menções e o sentimento em relação a uma determinada marca através do tempo. Os dados analisados são provenientes de várias mídias sociais, como o Twitter, Facebook, YouTube, blogs, etc. Na Figura 1.4 é mostrado o sentimento em relação à marca Microsoft, medindo-se o sentimento positivo, negativo e neutro através de um relógio, e sua evolução temporal.

O sentimento sumarizado também pode ser utilizado para diversas aplicações, como prever eleições [35], comportamento da bolsa de valores [8], arrecadação de bilheterias de filmes [3], definição de preços [2], etc. No entanto, o sentimento puro (positivo ou negativo) não pode refletir de maneira correta o contexto analisado. Portanto, é im-

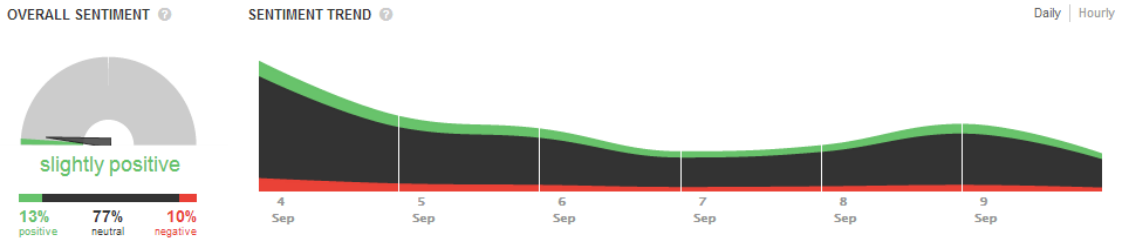

Figura 1.4. Sentimento extraído de mídias sociais em relação à Microsoft (Fonte: UberVU).

portante criar métricas para representar o sentimento em relação ao alvo. Boa parte dos trabalhos na área utilizam a média do sentimento, ou a razão entre o sentimento positivo e negativo. Em certos casos, a predição pode ser feita somente com base na quantidade de menções às entidades, independente do sentimento sobre elas (e.g. [35]).

Por exemplo, Social Market Analytics[5] é uma ferramenta para analisar a bolsa de valores, e ações de uma determinada companhia utilizando o Twitter. Na Figura 1.5 é mostrada a análise temporal do sentimento em relação às ações da *APPL - Apple Inc*. Esta ferramenta criou suas próprias métricas para análise, baseando na representação normalizada e ponderada da série de tempo do sentimento ao longo de um período retrospectivo (*S-Score*), e também uma média suavizada da métrica anterior (*S-Mean*). Estas métricas são comparadas com o valor de fechamento de uma determinada ação.

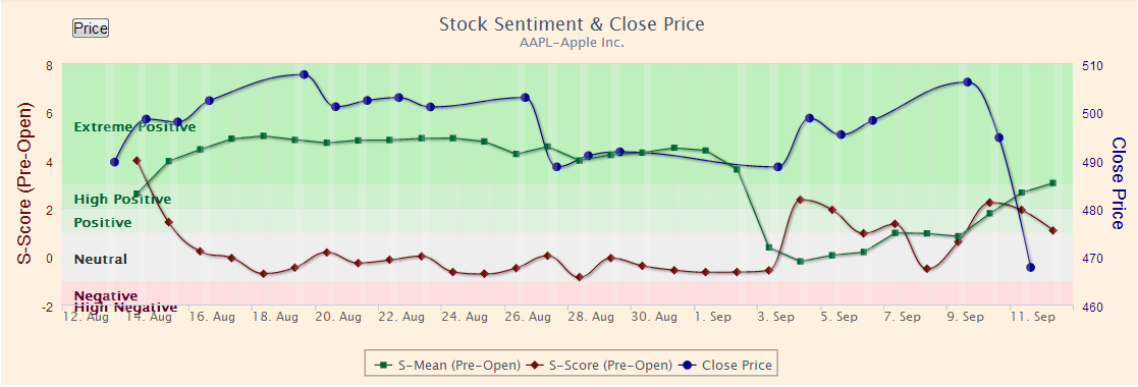

Figura 1.5. Relação entre sentimento e preço de ações (Fonte: Social Market Analytics).

1.4. Abordagens de Classificação de Polaridade

As abordagens de classificação podem ser divididas em quatro grandes grupos: a) léxicas, com o uso de dicionários de sentimentos; b) aprendizado de máquina, com o uso predominante de técnicas de classificação ou de regressão; c) estatísticas, que valem-se de técnicas para avaliar a co-ocorrência de termos, e d) semânticas, que definem a polaridade de palavras em função de sua proximidade semântica com outras de polaridade conhecidas. Técnicas destas diferentes abordagens podem ser combinadas para melhoria de resultados. Uma revisão sistemática descrita em [34] aponta uma predominâncias

[5]http://socialmarketanalytics.com

das duas primeiras abordagens, sem que nenhuma técnica se sobressaia em termos de desempenho.

1.4.1. Abordagem Baseada em Dicionário

A abordagem baseada em *dicionário* é também denominada *léxica* ou *linguística*. O aspecto central desta abordagem é o uso de léxicos (dicionários) de sentimentos, que são compilações de palavras ou expressões de sentimento associadas à respectiva polaridade.

Um dos métodos mais utilizados na abordagem linguística é o da co-ocorrência entre alvo e sentimento, que não leva em consideração nem a ordem dos termos dentro de um documento (*bag-of-words*), nem suas relações léxico-sintáticas. Para a classificação do sentimento em um texto, basta que exista uma palavra de sentimento, onde sua polaridade é dada por um léxico de sentimentos. Esse método é extensamente empregado para o atrelamento de um sentimento a uma entidade em uma sentença. Por exemplo, na sentença "o iPhone é muito bom", a polaridade positiva da palavra "bom" é associada à entidade iPhone. O método por co-ocorrência apresenta bons resultados quando o nível de análise textual é de granularidade pequena, pois a palavra detentora do sentimento está próxima à entidade que qualifica. Sendo assim, este método é usualmente utilizado em análises de nível de sentença, cláusula ou até em documentos com poucos caracteres, como um *tweet*.

Quando aplicada em nível de maior granularidade, estabelece-se algum tipo de média sobre as palavras de sentimento encontradas. A Equação 1 mostra uma função genérica de determinação de polaridade em um documento D, onde S_w representa a polaridade de uma palavra w em um dicionário. A agregação pode levar em conta funções de peso e de modificação. A função $peso()$ pode ser, por exemplo, alguma medida de distância entre a palavra de sentimento e o alvo, ou de importância da palavra no texto (e.g. frequência). A função $modificador()$ pode ser usada para tratar negações, palavras de intensidade (e.g. muito), etc. Esta função de agregação também pode ser estendida a sentenças, cujas cláusulas podem combinar diferentes palavras de sentimento.

$$\mathrm{S(D)} = \frac{\sum_{w \in D} S_w \cdot peso(w) \cdot modificador(w)}{\sum peso(w)} \tag{1}$$

Existem métodos linguísticos mais complexos, como a utilização de *parsers* linguísticos, que têm como propósito analisar o texto e aumentar a qualidade da classificação com base em informações morfossintáticas ali presentes (e.g. sujeito, predicado, dependências, funções sintáticas, etc.). No entanto, ferramentas de processamento de linguagem natural são em sua maioria restritas a determinado idioma. Recursos para a língua portuguesa são escassos, quando comparada à língua inglesa, situação esta comum a outra línguas.

A composição básica de um léxico de sentimento é a palavra de sentimento com suas possíveis flexões (e.g. bonito, bonita, bonitos), e sua respectiva polaridade, expressa como uma categoria, ou como um valor em uma escala. Muitos dicionários possuem adicionalmente: o lema e o *stem* de cada entrada; a categoria gramatical (*Part-Of-Speech - POS*); e o alvo do sentimento (predicado ou sujeito). A maioria dos léxicos existentes são dependentes de idioma e foram feitos estritamente para a língua inglesa, como General

Inquirer [30], OpinionFinder [38], SentiWordNet [4] e WordNetAffect [31]. Já para a língua portuguesa estão disponíveis o OpLexicon [29], e o SentiLex-PT [28], sendo o primeiro para português do Brasil e o último, para português de Portugal. Outro exemplo, é o Linguistic Inquiry and Word Counts (LIWC) [25], que é um software de análise de texto desenvolvido para avaliar os componentes estruturais, cognitivos e emocionais de amostras de texto, sendo que essa análise pode ser feita em vários idiomas disponibilizados na ferramenta. A Tabela 1.1 mostra uma comparação entre os léxicos mencionados.

Tabela 1.1. Tabela comparativa de léxicos de sentimentos.

Dicionário	Pos	Neg	PoS	Stem	Lema	Idioma
General Inquirer	1.915	2.291	S	N	N	Inglês
OpinionFinder	2.718	4.912	S	S	N	Inglês
OpLexicon	8.675	14.469	S	N	N	Português
SentiLex-PT	82.347 entradas		S	N	S	Português
SentiWordNet	117.659 entradas		S	N	S	Inglês

A maioria dos dicionários disponíveis são genéricos, ou seja, auxiliam na tarefa de classificação, independentemente do domínio dos textos sendo considerados. Entretanto, os melhores resultados obtidos na tarefa de classificação foram baseados em dicionários dependentes de contextos [17], criados a partir de palavras semente e expandidos utilizando o WordNet [14] ou tesauros. Esta abordagem também é classificada como *semântica*, e será melhor discutida na Seção 1.4.3. Finalmente, léxicos são de pouca valia quando considerados em textos gerados em mídias informais (e.g. redes sociais, *tweets*), onde expressões regionais, gírias, abreviaturas típicas da internet, etc, são fartamente empregados. A velocidade na criação de um novo vocabulário ultrapassa a capacidade de verificação de sanidade dos termos [10].

1.4.2. Abordagem baseada em Aprendizado de Máquina

O objetivo principal das técnicas de aprendizado de máquina é descobrir automaticamente regras gerais em grandes conjuntos de dados, que permitam extrair informações implicitamente representadas. De modo geral, as técnicas de aprendizado de máquina podem ser divididas em dois tipos: aprendizado supervisionado e aprendizado não supervisionado [32].

Na área de mineração de opiniões, nota-se um predomínio do uso de métodos supervisionados de aprendizagem, mais especificamente, *classificação* e *regressão*. Neste contexto, o problema de classificação é dividido em dois passos: (1) aprender um modelo de classificação sobre um corpus de treinamento previamente rotulado com as classes consideradas (e.g. positivo, negativo); e (2) prever a polaridade de novas porções de texto com base no modelo resultante. Dentre os algoritmos de classificação mais usados nesta área estão Support Vector Machine, Naïve Bayes, Maximum Entropy e algoritmos baseados em redes neurais [12, 8, 24, 23, 34].

A qualidade do modelo preditivo resultante da etapa de aprendizagem é medida em termos de métricas como *acurácia* (capacidade do modelo de prever corretamente), *precisão* (número de instâncias previstas corretamente em uma dada classe), ou *revocação* (número de instâncias de uma dada classe previstas na classe correta). Alguns trabalhos

obtêm precisões muito maiores na classificação da polaridade negativa, do que na positiva [27, 5, 36]. Além das dificuldades próprias ao domínio, esta situação pode ser explicada pela dificuldade em tratar ironia e sarcasmo.

Os dados de treino para a classificação/regressão correspondem a um conjunto de instâncias caracterizadas por atributos. O rótulo é denominado atributo *alvo*, enquanto que os demais são designados como atributos discriminantes ou *features*. O atributo alvo na classificação é discreto, enquanto que na regressão é numérico. Em termos de pré-processamento, é necessário extrair de cada porção de texto analisada, as *features* relevantes para a tarefa de classificação e representá-las na forma de um vetor de termos, como ilustra a Tabela 1.2.

Tabela 1.2. Exemplo de uma entrada de classificador com vetor binário de termos.

gosto	produto	ruim	grande	prático	não	facilidade	uso	polaridade
1	1	0	0	1	0	0	0	pos
0	1	1	1	1	1	0	0	neg
1	1	0	0	0	1	1	1	neg

Os tipos de *feature* mais frequentemente considerados são [24, 20, 34]:

- Palavras de sentimento: somente as palavras de sentimento de um corpus são utilizadas como *feature*. Não existe ordem entre os termos, e estes são caracterizados de forma binária, isto é, presente ou ausente no texto;
- Termos e sua frequência: são usados n-gramas (de sentimento ou não), junto com sua frequência absoluta ou relativa (e.g. TF-IDF), como peso dos termos;
- *Part-of-Speech* (POS): as classes morfológicas das palavras também podem ser usadas, em complementação às palavras de sentimento ou termos;
- Dependência sintática: as dependências sintáticas entre as palavras podem ser utilizadas, com o intuito de auxiliar na definição do alvo e fonte do sentimento.

Uma das grandes limitações no uso de aprendizado supervisionado para definição de polaridade é a necessidade de dados rotulados para treino. O desempenho destes métodos é afetado não somente pela quantidade, mas igualmente pela qualidade dos dados de treino disponíveis. Ainda, cada conjunto de treino é fortemente vinculado ao seu domínio. Nos trabalhos envolvendo revisões de produto, a classificação dada pelos usuários na forma de notas ou estrelas é utilizada como o rótulo para o texto correspondente [24, 37]. Tal facilidade não está disponível para outros domínios, implicando a necessidade de anotações manuais, as quais são trabalhosas e com alto teor de subjetividade. Nestes casos, as alternativas costumam ser os métodos léxicos, ou probabilísticos/semânticos, discutidos na próxima seção.

Como uma possível solução ao problema de anotação, foi proposto um método de classificação incremental baseado em regras [38], que utiliza regras genéricas sobre estruturas sintáticas para gerar texto subjetivo rotulado, o qual é usado para treinar classificadores e gerar novas regras mais específicas. A geração automática de texto rotulado no

domínio de política também é proposta em [27]. Esta alternativa é contudo impraticável para fontes que geram dados em *streaming*, onde além de grandes volumes de dados que devem ser analisados com baixa latência, existe uma volatilidade enorme nos tópicos e termos utilizados. Uma abordagem semi-supervisionada voltada à análise de sentimentos em tempo real [10] determina a polaridade de sentimento de *tweets*, tomando como ponto de partida opositores e apoiadores conhecidos, e inferindo através de regras de associação a propagação deste sentimento nas conexões sociais.

1.4.3. Abordagens Estatísticas e Semânticas

As abordagens estatísticas, por vezes denominadas *não supervisionadas* [19, 20], baseiam-se na premissa de que palavras que traduzem opiniões frequentemente são encontradas juntas no corpus dos textos. Se a palavra ocorre mais frequentemente junto a palavras positivas (negativas) no mesmo contexto, então é provável que seja positiva (negativa); já se ocorre em igual frequência, a palavra deve ser neutra. A polaridade de uma palavra desconhecida pode ser determinada calculando a co-ocorrência com uma palavra notadamente positiva (negativa), tal como "excelente" ou "péssimo". A técnica mais representativa nesta categoria é a Pointwise Mutual Information (PMI) [37].

O PMI entre dois termos quaisquer x e y é calculado segundo a Equação 2, onde Pr(x e y) é a probabilidade de co-ocorrência dos termos x e y, enquanto que Pr(x).Pr(y) é a probabilidade de co-ocorrência se são estatisticamente independentes. Esta razão é portanto a medida de grau de independência estatística entre os dois termos, e o logaritmo desta razão é a quantidade de informação ganha se os termos são observados juntos. A polaridade do sentimento de um termo x, dada pela Equação 3, é a diferença entre os valores de PMI calculados a partir de duas listas opostas: termos positivos (e.g. excelente), e termos negativos (e.g. péssimo). Na proposta original deste método [37], as co-ocorrências foram obtidas a partir da internet, utilizando uma mecanismo de busca existente na época (AltaVista).

$$PMI(x,y) = \log_2\left(\frac{Pr(x \wedge y)}{Pr(x)Pr(y)}\right) \tag{2}$$

$$\textit{PMI-IR}\ (x) = \sum_{p \in pWords} PMI(x,p) - \sum_{n \in nWords} PMI(x,n) \tag{3}$$

Outras técnicas na mesma abordagem são o Semantic-orientation Latent Semantic Analysis (SO-LSA) [37], que usa a LSA para calcular a força da associação semântica entre dois termos, através da análise estatística entre eles; e a Latent Dirichlet Allocation (LDA), muito utilizada para a extração de tópicos em textos [1]. A abordagem estatística é menos suscetível à dependência de contexto quando comparada às abordagens por dicionário e por aprendizado de máquina, mas pode ser utilizada para complementá-las [34].

A abordagem *Semântica* é bastante parecida com a estatística, exceto que a polaridade é calculada em termos de alguma medida de distância entre termos. O princípio das técnicas nesta categoria (e.g. [17, 16]) é que palavras semanticamente próximas devem ter a mesma polaridade. Por exemplo, o WordNet [14] provê diferentes relacionamen-

tos entre palavras que podem ser usadas para calcular a polaridade do sentimento, tais como sinônimos e antônimos. De forma similar à abordagem estatística, palavras semente com sentimento positivo e negativo são utilizadas como ponto de partida de um processo de comparação ou expansão, que busca determinar a polaridade dos termos. Isto pode ser feito através da mensuração da distância entre palavras usando as relações como caminho (e.g. [16]), ou da contagem da frequência com que são associadas a sinônimos positivos/negativos (e.g. [17]). A abordagem semântica também pode ser usada como complemento às outras abordagens, como forma de expansão ou de aquisição de vocabulário específico, na ausência de bons dicionários de sentimento. No entanto, ela carece ainda de métodos outros que manual, para validação da polaridade atribuída.

1.5. A Mineração de Opiniões e suas Fontes de Dados

1.5.1. Revisão de Produtos

Uma parcela significativa dos trabalhos na área de mineração de opiniões concentra-se na revisão de produtos, como já mencionado. Por um lado, este foco é justificado pelo interesse comercial nesta classe de aplicação, e a pela grande disponibilidade de dados. Mas do ponto de vista computacional, revisões de produtos apresentam uma série de propriedades que facilitam a mineração de opiniões, quando comparadas a textos em geral, tais como predominância de conteúdo de opinião, bom volume de opinião sobre uma mesma entidade ou relativa a um domínio, e definição clara da entidade alvo. Ainda, o uso de vocabulário muito coloquial, gírias, emoticons e mesmo de ironia é bem mais limitado, se comparado a outros tipos de mídia (e.g. Twitter, comentários). Assim, dado o foco claro, e a menor quantidade de ruído, os problemas computacionais inerentes podem ser mais facilmente identificados, estruturados e tratados [20]. Nesta seção, examinaremos trabalhos pioneiros nesta área [37, 24, 17], sendo que muitos outros resolvendo aspectos mais pontuais ou que melhoram resultados prévios são discutidos em [20, 34].

1.5.1.1. *Analise de Opinião em Nível de Documento*

Um dos trabalhos precursores nesta área foi desenvolvido por Turney em [37]. A motivação deste trabalho é agregar ao resultado de uma busca sobre revisões de um produto/serviço, informação sobre a recomendação (*Thumbs up*) ou não (*Thumbs down*) deste. Suas principais características são: a) análise em nível de documento, b) uso de abordagem estatística para classificação de polaridade, c) identificação de porções de texto com sentimento usando informação morfológica.

No tocante à etapa de identificação, assume-se que uma mecanismo de busca retorna uma coleção dos documentos referentes à entidade consultada, a qual é considerada como alvo de todas opiniões expressas nos documentos. A etapa de classificação envolve, para cada documento, seu pré-processamento para identificação *features* de interesse com base nas classes morfológicas das palavras, e a definição de sua polaridade. Finalmente, esta proposta não envolve explicitamente a etapa de sumarização, mas os documentos polarizados podem ser a entrada para várias aplicações, tais como geração de estatísticas, identificação de fóruns com posts abusivos (*flames*), etc.

Um documento de entrada é analisado usando um *parser* de POS, que rotula cada

palavra do texto com sua classe morfológica. São então selecionados apenas pares de palavras que seguem determinados padrões sintáticos. Mais especificamente, retém-se pares de palavras onde um dos elementos é um adjetivo ou advérbio, e o outro, uma palavra que lhe dá contexto. Com o contexto, a polaridade de palavras de sentimento pode ser melhor avaliada, tais como o adjetivo "inesperado", que possui conotação positiva no contexto de filme ("final inesperado"), ou negativo em uma revisão sobre carros ("defeito inesperado").

Para cada expressão resultante do pré-processamento, é calculado o PMI-IR, com base no PMI da expressão analisada com as palavras *excelente* e *ruim*, respectivamente. Para cálculo do PMI são submetidas consultas à internet com a mecanismo de busca AltaVista, onde o número de documentos retornados é utilizado como cálculo de frequência. Esta disponibiliza o operador *near*, que restringe a busca a documentos onde os termos estão próximos dentro de uma janela de no máximo 10 palavras. A polaridade final do documento, denominada *orientação semântica*, é dada pela média da polaridade de todas as expressões consideradas.

Como avaliação, foram usadas 410 revisões extraídas do *site* Epinions sobre filmes, automóveis, bancos e destinos de viagem, as quais estão associadas com uma classificação de estrelas. Foi obtida uma acurácia média de 74,39% em relação à recomendação do produto/serviço avaliado. É interessante notar que os piores resultados obtidos foram no domínio de filmes, já que mesmo que o sentimento geral sobre o filme tenha sido bom, os usuários sempre criticaram negativamente aspectos específicos. Ainda, não foi possível distinguir a opinião sobre o filme, do seu conteúdo objetivo. Por exemplo, na revisão *"Ele falava de um jeito devagar e metódico. Eu amei! Isto fez ele mais arrogante e mais perverso."*, as expressões "mais arrogante" e "mais perverso" referem-se ao personagem, mas são interpretadas como opiniões sobre o filme. Essa situação não aconteceu nos outros contextos analisados.

O desempenho devido à grande quantidade de buscas necessárias, e os restritos padrões gramaticais considerados, são apontados como os fatores mais limitantes desta proposta.

Uma alternativa à etapa de classificação foi apresentada na mesma época por Pang et al. em [24], usando métodos de aprendizagem de máquina. Mais especificamente, o trabalho concentrou-se em comparar experimentos com três classificadores conhecidos (Support Vector Machine - SVM, Naïve Bayes e Maximum Entropy [32]), e diferentes alternativas de pré-processamento. Na preparação de um vetor de termos, foram consideradas as seguintes alternativas de pré-processamento, algumas delas combinadas:

- unigramas com representação binária: adoção de qualquer termo (após remoção das *stop words*, e sem *stemming*). Quando uma negativa aparecia próxima ao termo, foi adicionado um prefixo NOT_ para representar a negação. Os termos mais frequentes na coleção (aproximadamente 18.000) foram representados como um vetor binário de termos;

- unigramas com peso: semelhante aos unigramas acimas, mas com uma representação vetorial com pesos (TDF-IF);

- bigramas: foram selecionados os bigramas mais frequentes, sem negação, os quais foram representados como um vetor binário;
- uso de POS: os termos foram rotulados usando um *parser*. Experimentos foram feitos considerando somente adjetivos como *features*, ou prefixando cada termo com classe gramatical para dar mais contexto;
- uso de posição: uso de unigramas com indicação de sua posição no texto (início, meio e fim). A intuição é que pessoas iniciam o texto, ou o concluem com a opinião mais importante, e que portanto a posição das palavras de opinião pode ser importante.

Experimentos foram conduzidos sobre uma base de revisões sobre filmes, da qual foram extraídas 700 revisões positivas e 700 negativas. Em termo de qualidade, os resultados foram bastante semelhantes aos obtidos por Turney em [37]. Os experimentos não revelaram de forma consistente a superioridade de nenhum dos classificadores adotados, e em termos de preparação de dados, o pré-processamento de vetor binário de unigramas foi, de uma forma geral, o mais eficaz.

1.5.1.2. *Análise de Opinião em Nível de Aspecto*

A análise de um produto em nível de documento permite derivar uma avaliação geral do mesmo. Em revisões onde existem sentimentos mistos expressos sobre a mesma entidade, pode-se chegar a uma situação de neutralidade em caso de médias (e.g. [37]), onde as expressões positivas e negativas se anulam; ou de incapacidade de classificação correta, pela falta de *features* discriminantes (e.g. [24]). A análise em nível de aspecto permite detalhar o alvo do sentimento, de tal forma que possam ser detectados seus pontos fortes e fracos.

Um trabalho pioneiro nesta área foi proposto por Hu e Liu em [17], que se caracteriza por: a) identificação de aspectos e de sentenças de opinião na etapa de identificação; b) uso de POS para identificação tanto de aspectos, quanto de palavras de sentimento; c) emprego da abordagem semântica para classificação da polaridade; e d) criação de sumários contendo o sentimento positivo e negativo sobre cada aspecto do produto.

A fase de identificação reconhece tópicos e sentenças de opinião. É primeiramente realizada a marcação das categorias gramaticais usando um *parser* de POS sobre os documentos. Para identificar os aspectos, são usadas regras de associação para encontrar termos frequentemente associados, os quais são podados segundo algumas regras de pré-processamento. Para encontrar as sentenças de opinião, são encontrados os adjetivos, considerados como palavras de opinião, e se estão próximos a aspectos, são designados como *opiniões efetivas*.

A etapa de classificação polariza as opiniões sobre os aspectos usando uma abordagem semântica, e refina o conjunto de tópicos utilizando as palavras de opinião. Para a polarização, o ponto de partida é um conjunto de palavras semente com polaridade definida manualmente. Usando o WordNet, a polaridade destas palavras é propagada para sinônimos (mesma polaridade) e antônimos (polaridade inversa), resultando assim em um

dicionário específico de sentimentos para o domínio das revisões. Este algoritmo de propagação é iterativo, de tal forma que a polaridade antes desconhecida de uma determinada palavra, acaba sendo encontrada em uma iteração futura, já que uma vez que um adjetivo é polarizado, ele passa a incorporar o conjunto de sementes. Para relacionar o sentimento ao aspecto, são usadas apenas as palavras de sentimento efetivas, na mesma sentença ou na sentença próxima.

As palavras de sentimento polarizadas são também utilizadas para identificar os aspectos de produtos menos frequentes e previamente desconhecidos. Por exemplo, se a palavra "excelente" é uma palavra de sentimento conhecida e na sentença, perto dessa palavra, existe uma frase nominal, assume-se que esta seja um aspecto de produto.

Finalmente, os sumários são criados em dois passos: 1) para cada aspecto, é feita uma contagem de quantas revisões opinam de maneira positiva/negativa; 2) aspectos são ordenados pela frequência com que aparecem nas revisões de produtos. Sentenças que contribuíram à pontuação positiva/negativa são mostradas junto com os sumários

A validação envolveu experimentos com vários aparelhos eletrônicos, usando revisões extraídas dos *sites* Amazon.com e Cnet.com. Estas foram manualmente anotadas quanto ao seus aspectos, sentenças opinativas e sua respectiva polaridade. Como resultado, obteve-se uma precisão média de 68,25% para aspectos identificados, de 64,2% para identificação de sentenças opinativas e uma acurácia média de 84,2% no tocante à polaridade. Os autores apontam como oportunidade de melhorias a resolução de pronomes, o uso de palavras de sentimento de classes outras que adjetivos, e o tratamento da intensidade da opinião sobre os aspectos extraídos.

1.5.2. Mineração de Opiniões em Notícias e Blogs

A mineração de opiniões em textos não estruturados é bem mais complexa que em revisões de produtos. Existem muitos desafios a serem enfrentados, entre eles: a) o texto pode conter opiniões sobre múltiplos alvos, e não é fácil reconhecê-los; b) o conteúdo de opinião é mais esparso no texto; c) dados os dois problemas anteriores, a associação da entidade alvo com a opinião fica ainda mais complexa; d) alguns tipos de texto, como notícias, tendem a não explicitar a opinião diretamente, fazendo-o através de artifícios (e.g. frases atribuídas a outras pessoas citadas na notícia); e e) é difícil distinguir entre conteúdo ruim (e.g. um terremoto), e uma opinião boa sobre um conteúdo ruim (e.g. elogiar o socorro às vítimas de um terremoto). Nesta seção são discutidos dois trabalhos que mineram opiniões sobre notícias: um que utiliza menções a entidades específicas [16], e outro que aborda os desafios de encontrar sentimento explícito em notícias [6, 7].

1.5.2.1. *Monitoramento de entidades em jornais e blogs*

Godbole *et al.* em [16] têm como objetivo desenvolver um sistema de análise de sentimentos em notícias e blogs que monitore o sentimento do público geral em relação a determinadas entidades, como pessoas, locais ou marcas. Assume-se que as entidades analisadas possuem características singulares, tais como atletas, celebridades, políticos, criminosos, etc, e que as opiniões emitidas devem ser interpretadas dentro de cada contexto. O método propõe uma forma algorítmica de construir dicionários de sentimentos

voltados a cada contexto, e métricas para medir o sentimento expresso sobre essas entidades. As principais características desta proposta são: a) análise em nível de sentença; b) uso de menções específicas para identificação de entidades; c) uso de abordagem semântica de criação de léxicos voltados a cada domínio, os quais são usados na polarização; e d) representação do sentimento através de diferentes métricas, cuja evolução pode ser monitorada.

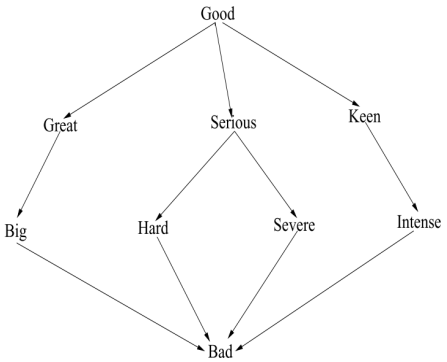

Figura 1.6. Expansão da palavra "*good*" até uma inversão de sentimento (Fonte: [16]).

Na fase de identificação, são encontradas as menções às entidades de interesse, e as respectivas sentenças. O método de co-ocorrência é utilizado para encontrar variações nas menções à mesma entidade, e uma ferramenta é utilizada para resolver pronomes. Usando os léxicos específicos relativos ao domínio da entidade em questão, cada sentença é polarizada. Se a entidade e uma palavra do léxico estiverem na mesma sentença, sua polaridade é atribuído ao alvo.

Para a fase de sumarização, são propostas métricas de sentimento, envolvendo *polaridade* e *subjetividade*. A primeira busca determinar a razão entre sentimento positivo e negativo, enquanto que a segunda, a razão entre sentimento positivo e o total de sentimento (positivo e negativo). Com isto, podem ser medidas a polaridade e subjetividade relativa a uma dada entidade, ou em relação ao conjunto de todas entidades (denominado mundo). A intuição é que determinadas épocas (e.g. natal, eleições) implicam que mais sentimentos sejam expressos. A evolução destas métricas pode ser acompanhada visualmente em um gráfico.

Os autores ainda fazem uma comparação entre o conteúdo de blogs e jornais para as mesmas entidades, concluindo que o sentimento relacionado a certas entidades pode diferenciar-se em notícias e blogs, devido ao viés do meio de publicação.

O método foi validado utilizando indicadores externos existentes (e.g. sentimento detectado sobre o presidente, e pesquisas de intenção de votos) mostrando que os resultados são bastante próximos.

1.5.2.2. *Análise de sentimentos em notícias*

O foco do trabalho de Balahur *et al.* [6, 7] é a identificação de conteúdo subjetivo em notícias. Exceto por jornais sensacionalistas, este tipo de texto evita expressar explicitamente

opiniões, com objetivo de manter a seriedade e a suposta neutralidade da notícia. Opiniões neste meio são expressas de formas bem mais sutis, e que são linguisticamente difíceis de serem identificadas: argumentações tendenciosas; omissão ou destaque de fatos em detrimento de outros; citações de outras pessoas que expressam opiniões (e.g. "reiniciar este julgamento é um verdadeiro absurdo", disse o Ministro da Justiça); entre outros.

Dada a complexidade da análise de conteúdo implícito, a análise da subjetividade restringe-se a opiniões explicitamente expressas em citações, nas quais o conteúdo potencialmente de opinião é associado a um autor. Balahur *et al.* enfatizam a complexidade da mineração de opinião neste tipo de texto, e a restrição a citações como um primeiro passo para a compreensão dos diferentes problemas envolvidos. Este trabalho caracteriza-se por: a) nível de análise em citação, composta de uma ou mais sentenças; b) abordagem orientada a léxicos; c) uso de menções específicas para identificação de entidades.

A principal contribuição destes trabalhos é uma melhor caracterização da tarefa de mineração de opiniões em citações, através da realização de experimentos. As tarefas são definidas como:

- a definição do alvo da opinião: mesmo restringindo a citações, o alvo da opinião nem sempre é claro e explícito, podendo existir mais de um alvo. Utilizam-se entidades específicas como alvo, que estão em uma dada janela de distância das citações. As entidades são termos de busca de notícias (e.g. tsunami, Obama);

- a separação entre conteúdo ruim/bom, da opinião positiva ou negativa sobre este: a abordagem é baseada em léxicos, dos quais são removidos termos específicos que podem representar fatos com conotação negativa/positiva (e.g. crise, desastre, carnaval). Nos experimentos, foram utilizados rótulos (*tags*) que caracterizam categorias de notícias, e que possuem um sentimento associado. Os autores apontam que outras abordagens poderiam ter sido utilizadas com melhores resultados, inclusive escolha manual de termos.

- interpretação da polaridade sem informação contextual: os autores discorrem que em notícias existe uma grande distinção entre a opinião do autor, expressa na citação, e a opinião do leitor, de acordo com sua interpretação dos fatos. Esta divergência é uma grande dificuldade para o consenso em processos de anotação. Eles focam na classificação do primeiro tipo, usando apenas as informações explicitamente presentes no texto. Para a polarização, que foi baseada na abordagem de dicionário, foram experimentados 4 léxicos distintos, utilizados para polarizar todos os termos de sentimento encontrados na citação a uma dada distância do alvo. Diferentes abrangências de janela foram testadas.

Os experimentos envolveram 1.292 citações sobre as quais houve concordância entre os anotadores sobre a polaridade e o alvo. O melhor resultado (86% de acurácia) foi obtido com o uso combinado de dois léxicos (entre eles JRCTonality, desenvolvido pelo próprio grupo) e uma janela de 6 palavras entre alvo e termo de sentimento. Na análise de erros, os principais problemas detectados foram as citações polarizadas erroneamente como neutras (devido à falta de palavras explícitas), o uso de ironias, emprego de abstrações para referir-se ao alvo, e múltiplos alvos.

1.5.3. Mídia Social

A mineração em mídias sociais têm a informalidade deste tipo de meio como um dos seus principais desafios. Não apenas o vocabulário pode ser bem específico e volátil, como o número de erros de digitação, de ortografia ou gramaticais pode invalidar a contribuição de análises linguísticas. Por outro lado, o volume da dados gerados sobre cada tópico é tão grande, quando comparado com outras fontes, que tais erros podem não ser relevantes. Um dos focos de trabalhos de mineração de opinião no Twitter, é o da capacidade de previsão, justamente com base neste grande volume. Dois destes trabalhos [3] [8] são melhor detalhados no restante desta seção.

1.5.3.1. Previsão de indicadores de rentabilidade de filmes

A popularidade do Twitter é tal que organizações têm utilizado esta plataforma para divulgação e marketing de suas marcas e produtos. Este é o caso de filmes, onde produtores têm investido massivamente em publicidade e marketing voltado aos usuários do Twitter. Asur *et al.* em [3] realizam experimentos para verificar se o interesse que um filme desperta, particularmente na época de pré-lançamento, correlaciona-se com indicadores econômicos deste domínio. Mais especificamente, a partir do volume de *tweets* e do sentimento neles expressos, deseja-se prever: a arrecadação da primeira semana de bilheteria; os preços dos índices do *Hollywod Stock Exchange* (HSX); e o rendimento de todos os filmes de uma semana específica. Este trabalho caracteriza-se por: a) análise em nível de documento (*tweet*); b) uso de termos pré-definidos para determinação de entidades alvo; c) uso da abordagem de aprendizagem de máquina para polarização, e d) definição de métricas de agregação de sentimento utilizadas para previsão.

A proposta consiste de um estudo de caso onde foram analisados *tweets* de filmes específicos, sobre um período de três meses. Palavras-chaves, como URL's de material promocional e o título dos filmes, foram utilizadas para extrair os *tweets* relevantes e identificar os alvos. Juntamente com o conteúdo de cada *tweet*, também extraiu-se a data e hora de postagem, e o respectivo autor.

Os autores comparam dois tipos de previsão: quantitativo (quantidade de *tweets* e *retweets* contendo URL's de material promocional sobre o filme), e baseado em opinião expressa nos *tweets*. A segunda implica a necessidade de classificação da polaridade dos *tweets*, que foi realizada utilizando o classificador DynamicLMClassifier, disponível no pacote análise linguístico LingPipe[6]. Este classificador é ternário, i.e. classifica os *tweets* como positivo, negativo ou neutro. Os dados de treino foram rotulados manualmente utilizando-se o Amazon Mechanical Turk. Somente *tweets* com polaridade classificada de forma unânime pelos anotadores foram utilizados para treino. A acurácia do classificador foi elevada (98%) com features representando 8-gramas.

Para a previsão baseada em número de *tweets*, os autores definem a métrica taxa de *tweets* de cada filme por hora (Equação 4). Esse métrica permite criar uma série temporal de menções de cada filme, e correlacioná-la usando regressão linear com as variáveis alvo, ou seja, previsão de bilheteria e previsão de valor dos índices HSX. Uma forte correlação

[6]http://www.alias-i.com/lingpipe

foi encontrada (R^2 ajustado de 0,8, e de 0,9, respectivamente), significando alto poder de previsão.

$$\textit{Taxa-de-tweets(filme)} = \frac{|\textit{tweets(filme)}|}{|\textit{tempo (em horas)}|} \quad (4)$$

Já para a previsão baseada em sentimento, foram definidas duas outras métricas: a razão entre o total de *tweets* positivos e negativos e o total de *tweets* neutros, chamada pelos autores de *subjetividade*; e a razão entre *tweets* positivos e negativos. Os resultados obtidos foram inferiores aos obtidos com a métrica quantitativa de taxa de *tweets* por hora. No entanto, quando as métricas de sentimento são associadas à da taxa de *tweets*, houve uma melhoria no poder de predição.

É interessante notar que resultados semelhantes foram encontrados no domínio político, onde experimentos revelaram que a quantidade de *tweets* tiveram maior poder preditivo sobre o resultado de eleições, que o sentimento ou emoção neles expressos [21, 35].

1.5.3.2. Previsão do comportamento da bolsa de valores

O objetivo deste trabalho é semelhante ao da seção anterior, no domínio da bolsa de valores. Bollen *et al.* [8] realizam experimentos para verificar se sentimento expresso no Twitter, chamado no trabalho de humor, tem influência sobre a bolsa de valores, e pode ser utilizado para prever seu comportamento usando o índice Dow Jones. Este trabalho caracteriza-se por: a) análise em nível de documento (*tweet*); b) uso de expressões específicas para determinar conteúdo de sentimento; c) uso da abordagem de léxica para polarização, onde um léxico de emoções também foi usado, e d) definição de métricas de agregação de sentimento utilizadas para previsão.

Foi realizado um estudo de caso envolvendo *tweets* correspondentes a um período de onze meses (Fevereiro, 2008 - Dezembro, 2008). A fase de identificação selecionou apenas *tweets* que continham sentimento explícito. Como critério, foram utilizadas expressões pré-definidas como “eu sinto”, “eu estou sentindo”, “eu não sinto”, etc.

A classificação envolveu a abordagem léxica, mas dois tipos de classificação foram feitos: a) polaridade de sentimento (positivo e negativo), utilizando o léxico OpinionFinder [38]; e b) classificação da emoção (vide Seção 1.2.1), utilizada uma ferramenta denominada Google-Profile of Mood States (GPOMS). GPOMS analisa e classifica a emoção em seis diferentes dimensões: *calma, alerta, certeza, vitalidade, gentileza* e *felicidade*.

A polaridade da opinião foi sumarizada através de uma métrica que representa a razão entre a quantidade de termos positivos encontrados nos *tweets* e a de termos negativos, em um determinado dia. Já cada dimensão da emoção GPMOS foi totalizada por dia. Para todas estas métricas, foram criadas séries temporais, as quais foram comparadas com eventos conhecidos do mesmo período: eleições e Ação de Graças. Exceto pela *gentileza* e *alerta* em relação às eleições, foi demonstrado que as técnicas utilizadas caracterizavam o humor típico destas datas. Uma análise estatística mostrou ainda correlação entre a po-

laridade da opinião e as dimensões de emoção *certeza*, *vitalidade*, e *alegria*, mas não com as demais.

Finalmente, foi usado um método baseado em redes neurais fuzzy (*self-organizing fuzzy neural network* - SOFNN) para correlacionar as séries temporais das polaridades das opiniões e das emoções, com a série temporal Down Jones Industrial Average (DIJA), e desenvolver um modelo preditivo. A abordagem utilizando o OpinionFinder não obteve bons resultados. No entanto, as emoções *calma* e *felicidade* demonstraram correlação com o DIJA em certos trechos do período analisado. O sentimento de calma foi o que obteve um melhor poder preditivo com um atraso (*lag*) de 3 dias. No entanto, o sentimento de calma só conseguiu prever corretamente o DIJA em períodos em que não há eventos inesperados (e.g. anúncio da Reserva Federal Americana). Isto significa dizer que consegue prever quando os índices se mantêm devido à falta de eventos externos importantes. Os autores concluem que a polaridade da opinião pode ser muito abrangente, e esconder aspectos cobertos pela subjetividade das emoções.

1.6. Conclusões e Direções Futuras

A mineração de opiniões é uma área de crescente interesse. Neste capítulo, discutimos seus conceitos básicos, os desafios na detecção de sentimento e de seu alvo, e técnicas que podem ser usadas para identificar, classificar a polaridade e agregar o sentimento expresso. Alguns trabalhos clássicos envolvendo diferentes fontes de opiniões foram apresentados a título de ilustração. O volume crescente de conteúdo subjetivo disponível diariamente, em particular nas redes sociais, motiva o crescimento da área com novas técnicas capazes de processar automaticamente textos, de forma escalável, robusta, precisa e independente de domínio e de linguagem. Muitas são as aplicações centradas na sumarização e visualização do sentimento, ou na predição de comportamentos com base no sentimento existente. Empresas, eventos (e.g. Olimpíadas), personalidades, estão interessadas na compreensão de como são percebidas pelo público em geral em tempo real, e nas mais variadas mídias.

A área ainda apresenta muitos problemas e oportunidades. Muitos esforços estão voltados à detecção de outros tipos de opinião: comparativa, implícita, dependente do observador, contraditórias, spams, ironias e sarcasmos, etc. A identificação de alvos e opiniões em mídias sem grau de estruturação, como notícias, é também outra área bastante importante. O desenvolvimento de recursos multilíngues permitirão o avanço do estado da arte, permitindo tratar corpus para os quais hoje não existem recursos (e.g. dados rotulados, léxicos, recursos para tratamento da língua natural, etc.). A evolução das técnicas de classificação para métodos escaláveis, menos sensíveis ao contexto ou a ruídos, e que combinam abordagens já existentes em um *framework* único é uma outra meta. Novas aplicações devem estabelecer soluções para *streaming* de dados, apoio a decisão baseado em sentimento, predições, entre tantas outras.

Referências

[1] Aggarwal, C. C. and Zhai, C. (2012). *Mining text data*. Springer.

[2] Archak, N., Ghose, A., and Ipeirotis, P. G. (2007). Show me the money!: deriving the pricing power of product features by mining consumer reviews. In *Proceedings*

of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 56–65. ACM.

[3] Asur, S. and Huberman, B. A. (2010). Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE.

[4] Baccianella, S., Esuli, A., and Sebastiani, F. (2010). Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*, volume 10, pages 2200–2204.

[5] Balahur, A., Kozareva, Z., and Montoyo, A. (2009a). Determining the polarity and source of opinions expressed in political debates. In *Computational Linguistics and Intelligent Text Processing*, pages 468–480. Springer.

[6] Balahur, A., Steinberger, R., Goot, E. v. d., Pouliquen, B., and Kabadjov, M. (2009b). Opinion mining on newspaper quotations. In *Web Intelligence and Intelligent Agent Technologies, 2009. WI-IAT'09. IEEE/WIC/ACM International Joint Conferences on*, volume 3, pages 523–526. IET.

[7] Balahur, A., Steinberger, R., Kabadjov, M., Zavarella, V., Van Der Goot, E., Halkia, M., Pouliquen, B., and Belyaeva, J. (2010). Sentiment analysis in the news. In *Proceedings of LREC*, volume 10.

[8] Bollen, J., Mao, H., and Zeng, X. (2011). Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8.

[9] Bruce, R. F. and Wiebe, J. M. (1999). Recognizing subjectivity: a case study in manual tagging. *Natural Language Engineering*, 5(2):187–205.

[10] Calais Guerra, P. H., Veloso, A., Meira Jr, W., and Almeida, V. (2011). From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158. ACM.

[11] Carvalho, P., Sarmento, L., Silva, M. J., and de Oliveira, E. (2009). Clues for detecting irony in user-generated contents: oh...!! it's so easy;-). In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM.

[12] Dave, K., Lawrence, S., and Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM.

[13] Dey, L. and Haque, S. (2009). Opinion mining from noisy text data. *International journal on document analysis and recognition*, 12(3):205–226.

[14] Fellbaum, C. (2010). *WordNet*. Springer.

[15] Ghani, R., Probst, K., Liu, Y., Krema, M., and Fano, A. (2006). Text mining for product attribute extraction. *ACM SIGKDD Explorations Newsletter*, 8(1):41–48.

[16] Godbole, N., Srinivasaiah, M., and Skiena, S. (2007). Large-scale sentiment analysis for news and blogs. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM)*, volume 2.

[17] Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

[18] Ku, L., Liang, Y., and Chen, H. (2006). Opinion extraction, summarization and tracking in news and blog corpora. In *Proceedings of AAAI-2006 Spring Symposium on Computational Approaches to Analyzing Weblogs*, number 2001.

[19] Liu, B. (2010). Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:568.

[20] Liu, B. (2012). *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers.

[21] O'Connor, B., Balasubramanyan, R., Routledge, B. R., and Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM*, 11:122–129.

[22] Pak, A. and Paroubek, P. (2010). Twitter as a corpus for sentiment analysis and opinion mining. In *LREC*.

[23] Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135.

[24] Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

[25] Pennebaker, J. W., Chung, C. K., Ireland, M., Gonzales, A., and Booth, R. J. (2007). The development and psychometric properties of liwc2007. *Austin, TX, LIWC. Net*.

[26] Sarawagi, S. (2008). Information extraction. *Foundations and trends in databases*, 1(3):261–377.

[27] Sarmento, L., Carvalho, P., Silva, M., and de Oliveira, E. (2009). Automatic creation of a reference corpus for political opinion mining in user-generated content. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 29–36. ACM.

[28] Silva, M., Carvalho, P., and Sarmento, L. (2012). Building a sentiment lexicon for social judgement mining. *Computational Processing of the Portuguese Language*, pages 218–228.

[29] Souza, M., Vieira, R., Busetti, D., Chishman, R., and Alves, I. M. (2011). Construction of a portuguese opinion lexicon from multiple resources. In *The 8th Brazilian Symposium in Information and Human Language Technology (STIL 2011), Cuiabá, Brazil*.

[30] Stone, P. J., Dunphy, D. C., and Smith, M. S. (1966). The general inquirer: A computer approach to content analysis.

[31] Strapparava, C. and Valitutti, A. (2004). Wordnet affect: an affective extension of wordnet. In *LREC*, volume 4, pages 1083–1086.

[32] Tan, P.-N., Steinbach, M., and Kumar, V. (2006). *Introduction to data mining*. Addison Wesley.

[33] Thet, T., Na, J., and Khoo, C. (2010). Aspect-based sentiment analysis of movie reviews on discussion boards. *Journal of Information Science*, 36(6):823–848.

[34] Tsytsarau, M. and Palpanas, T. (2012). Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, 24(3):478–514.

[35] Tumasjan, A., Sprenger, T., Sandner, P., and Welpe, I. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proceedings of the Fourth International aaai conference on weblogs and social media*, pages 178–185.

[36] Tumitan, D. and Becker, K. (2013). Tracking sentiment evolution on user-generated content: A case study in the brazilian political scene. In *Brazilian Symposium on Databases (SBBD)*, page 6. SBC. A ser publicado.

[37] Turney, P. D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics.

[38] Wiebe, J. and Riloff, E. (2005). Creating subjective and objective sentence classifiers from unannotated texts. In *Computational Linguistics and Intelligent Text Processing*, pages 486–497. Springer.

[39] Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

Minicurso
3

Modelagem Conceitual de Bancos de Dados Geográficos: Modelo OMT-G

Bruno Rabello Monteiro, Clodoveu A. Davis Jr.

Resumo

Este minicurso apresenta conceitos relacionados à modelagem conceitual, de representação e de apresentação para bancos de dados geográficos, utilizando uma metodologia baseada no modelo OMT-G. O objetivo do minicurso é propiciar uma visão introdutória dos tipos de problemas que ocorrem quando componentes geográficos precisam ser incorporados a um banco de dados, e aborda soluções de projeto capazes de levar em consideração as características específicas dos dados geográficos. O minicurso é dividido em 2 momentos distintos: o teórico e o prático. Na parte inicial teórica, são apresentados conceitos sobre dados geográficos e o modelo OMT-G. Na parte prática, são apresentados e discutidos dois estudos de casos, problemas para modelagem utilizando o modelo OMT-G para produzir diagramas de classes e promover seu mapeamento até as estruturas físicas.

Borges, K. A., Davis Jr., C. A., Laender, A. H. F. (2005) Modelagem de Dados Geográficos. In Casanova, M. A., Câmara, G., Davis Jr., C. A., Vinhas, L., Queiroz, G. R. Bancos de Dados Geográficos. MundoGeo, Curitiba (PR), 2005.
Disponível em http://www.dpi.inpe.br/livros/bdados/cap3.pdf