ROGER SANTOS FERREIRA

BIGFEEL - UM AMBIENTE DE PROCESSAMENTO DISTRIBUÍDO PARA INTEGRAÇÃO DE MÉTODOS DE ANÁLISE DE SENTIMENTOS

LAVRAS – MG

2018

ROGER SANTOS FERREIRA

BIGFEEL - UM AMBIENTE DE PROCESSAMENTO DISTRIBUÍDO PARA INTEGRAÇÃO DE MÉTODOS DE ANÁLISE DE SENTIMENTOS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para obtenção do título de Mestre em Ciência da Computação.

Prof. Dr. Denilson Alves Pereira (DCC/UFLA)
Orientador

LAVRAS – MG

2018

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Ferreira, Roger Santos.
BigFeel : Um ambiente de processamento distribuído para integração de métodos de análise de sentimentos / Roger Santos Ferreira. - 2018.
100 p. : il.

Orientador(a): Denilson Alves Pereira.
.
Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2018.
Bibliografia.

1. Análise de sentimentos. 2. Spark. 3. Processamento distribuído. I. Pereira, Denilson Alves. . II. Título.

ROGER SANTOS FERREIRA

BIGFEEL - UM AMBIENTE DE PROCESSAMENTO DISTRIBUÍDO PARA INTEGRAÇÃO DE MÉTODOS DE ANÁLISE DE SENTIMENTOS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para obtenção do título de Mestre em Ciência da Computação.

APROVADA em 19 de Outubro de 2018.

Prof. Dr. Leonardo Andrade Ribeiro	INF/UFG
Prof. Dr. André Luiz Zambalde	DCC/UFLA

Prof. Dr. Denilson Alves Pereira (DCC/UFLA)
Orientador

LAVRAS – MG
2018

AGRADECIMENTOS

Agradeço a Deus, por estar sempre ao meu lado, me guiando, incentivando e protegendo a cada minuto de vida.

À minha esposa, Lívia, e ao meu filho, Heitor, os quais foram pacientes e longânimos em seu amor incondicional durante essa minha jornada acadêmica.

Aos meus pais e irmãos, Alzira, Cleber, Antar e Heber, que amorosamente compreenderam minha ausência constante em prol da realização dos meus sonhos acadêmicos.

A meu sogro, sogra e cunhado, Paulo, Gislaine e Paulo Henrique, os quais carinhosamente me ajudaram e apoiaram durante os estudos.

Ao professor Denilson, pelo companheirismo, dedicação e compromisso para a realização deste trabalho. Os ensinamentos e conhecimentos repassados nunca serão esquecidos.

A todos os amigos do IFMG, os quais de forma direta ou indireta me auxiliaram durante os estudos, apoiando tanto o meu trabalho, quanto os meus estudos.

Ao coordenador do Programa de Pós-Graduação em Ciência da Computação, professor Luiz Henrique, pela sua seriedade e compromisso para com a qualidade de nossa formação.

Aos amigos do Laboratório de Recuperação da Informação e demais contemporâneos no curso, por todo o auxílio, companheirismo, amizade e suporte durante essa jornada.

À Universidade Federal de Lavras e ao Programa de Pós-Graduação em Ciência da Computação, pelo excelente trabalho na construção de um curso de qualidade.

A todos aqueles que de alguma forma me auxiliaram durante o mestrado.

Obrigado!

RESUMO

A análise de sentimentos tem sido foco de muita pesquisa, devido principalmente à sua importância comercial, tanto para consumidores quanto para empresas. Muitos métodos têm sido propostos, e os principais têm sido comparados em termos de eficácia. Entretanto, há uma carência na literatura da avaliação de eficiência desses métodos para processamento de grandes volumes de dados, os quais são gerados em grande velocidade, volume e variedade, conhecidos como *Big Data*. O presente trabalho apresenta uma abordagem para integração de métodos de análise de sentimentos de forma a processar grandes volumes de dados em um ambiente distribuído, usando para tanto das plataformas Hadoop e Spark, ambas da fundação Apache. Desenvolveu-se uma aplicação protótipo em ambiente distribuído, denominada BigFeel, a qual oferece suporte ao uso de 22 métodos de análise de sentimentos, além de alguns métodos de processamento de linguagem natural e pré-processamento textual em grandes volumes de dados. O BigFeel oferece serviços adaptados ao uso em redes de computadores, locais e na *web*, além de oferecer uma API para desenvolvedores Scala/Java. A eficiência dos métodos integrados foi avaliada experimentalmente, demonstrando ganho em comparação à execução na implementação não distribuída dos métodos. Usando os recursos oferecidos pelo BigFeel, é apresentado ainda um estudo de caso de detecção de sugestões de inovação com base em revisões de produtos e serviços.

Palavras-chave: Análise de sentimentos. Aprendizagem de máquina. Hadoop. Spark. *Big Data*. Processamento de linguagem natural

ABSTRACT

Sentiment analysis has been the main focus of plenty of research efforts, particularly justified by its commercial significance, both for consumers and businesses. Thus, many methods have been proposed, and the main ones have been compared in terms of effectiveness. Nonetheless, the literature is deficient when it comes to assessing the efficiency of these methods for processing large volumes of data, which are generated at great speed, volume and variety, known as Big Data. The present work presents an approach for integrating methods of sentiment analysis in order to process large volumes of data in a distributed environment, using both the Apache Hadoop and Spark platforms. A distributed application prototype was developed, named BigFeel, which supports the use of 22 methods of sentiment analysis, as well as some methods of natural language processing and textual preprocessing in large volumes of data. BigFeel offers services tailored to the use of computer networks, local and web, as well as offering an API for Scala/Java developers. The efficiency of the integrated methods was evaluated experimentally, demonstrating gain in comparison to the execution in the non-distributed implementation of the methods. Using the features offered by BigFeel, a case study of detection of innovation suggestions based on product and service reviews is also presented.

Keywords: Sentiment analysis. Machine learning. Hadoop. Spark. Big Data. Natural language processing

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

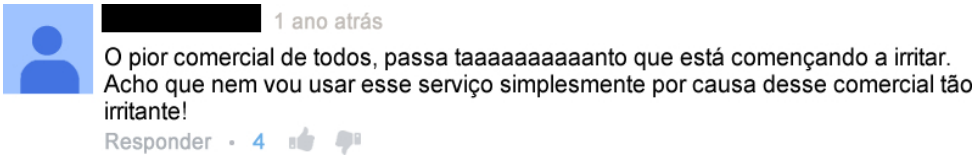
Este capítulo introduz o tema do presente trabalho, contextualizando o problema abordado, apresentando a motivação para sua execução, uma proposta de solução para o problema, os objetivos, a justificativa e as contribuições para a área da Ciência da Computação.

1.1 Contextualização e motivação

No processo de tomada de decisão, saber a opinião das pessoas tem sido uma valiosa peça de informação. Com o aumento do uso da Internet e das atividades *online* como mensageiros instantâneos, conferências *online*, vigilância, *e-commerce*, redes sociais e microblogs, as técnicas de extração, transformação, carregamento e análise de grandes volumes de dados textuais não estruturados e em tempo hábil se tornam cada vez mais necessárias (RAVI; RAVI, 2015). Nesse contexto, a análise de sentimentos busca identificar a aprovação do público quanto às várias características de um determinado item ou assunto, usando para tanto da análise de sentenças opinativas, conteúdo esse ricamente encontrado na *web* (LIU, 2012).

A Figura 1.1 exemplifica um típico caso onde se pode utilizar a análise de sentimentos: um comentário de cunho opinativo de um usuário sobre determinado vídeo comercial de um serviço.

Figura 1.1 – Comentário sobre produto em mídia social.



Fonte: Adaptado de Expedia SEA (2017).

Para um leitor atento, verificar a opinião expressa textualmente pelo usuário como positiva ou negativa é um tarefa simples, porém quando há a necessidade de se avaliar milhões de sentenças como as da Figura 1.1, o tempo se torna um problema, fazendo cada vez mais necessárias abordagens que desenvolvam tal análise em tempo hábil. O principal problema da manipulação de *Big Data* é o seu grande tamanho e o rápido fluxo de geração dos dados (FAN; BIFET, 2013).

A aplicação prática da análise de sentimentos se dá em geral nos seguintes contextos: 1) empresas e organizações que requerem as opiniões dos consumidores como meio de desenvolvimento de inovação sobre os produtos e serviços comercializados; 2) análise de opinião

pública, a qual auxilia a decisão de compra daqueles que têm dúvidas sobre qual produto ou serviço adquirir; 3) publicidade *on-line*, utilizando-se da opinião pública para direcionamento do *marketing* em mídias sociais; e 4) definição de opinião pública, para suporte à tomada de decisão (FELDMAN, 2013).

A análise de sentimentos é definida computacionalmente como uma técnica de processamento textual para se encontrar opiniões de um autor sobre um determinado objeto ou um de seus aspectos, identificar os sentimentos expressos nessas opiniões e classificar sua polaridade (MEDHAT; HASSAN; KORASHY, 2014). Seu principal foco é a análise textual, que em simples termos pode ser definida como a detecção da polaridade de texto, que pode ser positiva, negativa ou neutra. A análise de sentimentos expressos na opinião de usuários pode ser classificada entre opinião direta ou por comparação, e também pode ser subdividida segundo a granularidade da análise: em nível de documento (todo o documento ou revisão expressa uma única opinião), em nível de sentença (cada sentença de uma revisão pode expressar uma opinião distinta) e em nível de aspecto (cada aspecto de um determinado produto ou serviço pode ter uma opinião distinta, dada pela perspectiva do usuário) (TSYTSARAU; PALPANAS, 2011; LIU, 2011).

Esse tema de pesquisa teve grande influência do trabalho de Pang e Lee (2008), intitulado *Opinion Mining and Sentiment Analysis*. Com mais de cinco mil citações, segundo a *Web of Science* [1], esse estudo abriu uma porta no meio acadêmico, levantando quais seriam os grandes desafios envolvidos na mineração de opiniões e análise de sentimentos. Nos últimos anos, muito se tem pesquisado nessa área envolvendo aplicações de tempo real (NODARAKIS et al., 2016; PAGE, 2014; SHEELA, 2016) como Twitter e redes de *feeds*. Contudo, abordagens de ambientes que ofereçam métodos de análise de sentimentos para a experimentação variada sob *Big Data* ainda são escassas. O trabalho de Ribeiro et al. (2016), por exemplo, oferece vários métodos para comparação de resultados em análise de sentimentos, porém eles não avaliaram o comportamento desses métodos para grandes volumes de dados de entrada.

1.2 Problema, Objetivo e Hipótese

O problema da análise de sentimentos tem sido extensivamente estudado e abordado nos últimos anos. A maior parte das soluções existentes está ligada a abordagens que utilizam técnicas de processamento de linguagem natural e aprendizagem de máquina. Contudo, tais técnicas

[1] https://www.webofknowledge.com

têm conhecidamente alto custo computacional em tempo de processamento (LIN, 2010; BANERVELD; LE-KHAC; M., 2014), fazendo com que poucas milhares de sentenças sob análise ultrapassem o limite de processamento de um único servidor (NODARAKIS et al., 2016).

Após uma revisão sistemática da literatura, surgem as seguintes questões: 1) Os métodos de análise de sentimentos existentes são eficientes no processamento de grandes volumes de dados? 2) Caso seja negativa a resposta à primeira questão: a adaptação dos métodos de análise de sentimentos para execução distribuída oferece eficiência significativa?

O objetivo deste trabalho é a proposta de um ambiente distribuído de código-fonte aberto que integra e possibilita outras integrações de métodos de análise de sentimentos para processamento de grandes volumes de dados, denominado BigFeel. Como objetivos específicos: a) apresentação da arquitetura que permite ao BigFeel a solução para problemas de integração de métodos não distribuídos à execução distribuída; b) análise e discussão da eficiência e da eficácia na avaliação de resultados experimentais sob os métodos integrados ao ambiente BigFeel.

A hipótese é que métodos de análise de sentimentos existentes na literatura e implementados para executar em ambientes não distribuídos podem ser adaptados e integrados a uma plataforma de processamento distribuído como forma de melhorar sua eficiência. Foram escolhidas as plataformas Apache Spark (KARAU et al., 2015) e Apache Hadoop (WHITE, 2015) para implementação de um ambiente, denominado BigFeel, o qual permite a integração de métodos de análise de sentimentos. Foram avaliados experimentalmente os principais métodos existentes na literatura cuja implementação está disponível. As plataformas Hadoop e Spark garantem escalabilidade, resiliência e eficiência para processamento distribuído, afinal vários autores apontam o processamento distribuído como boa opção para a solução em eficiência (ONETO et al., 2016; BURDORF, 2016; KANAVOS et al., 2017; GUPTA, 2017).

1.3 Justificativa e Contribuição

A avaliação feita neste trabalho se justifica pelo fato de não terem sido encontrados na literatura trabalhos que comparam a eficiência de métodos de análise de sentimentos. As propostas de métodos são avaliadas normalmente em relação à eficácia, comparando-as com *baselines*. Ribeiro et al. (2016) compara a eficácia de diversos métodos, mas não foi feita uma avaliação de eficiência desses métodos. A criação do BigFeel se justifica pelo fato de não ter sido encontrado na literatura um ambiente que permita a integração dos métodos implemen-

tados naturalmente não distribuídos para execução distribuída, bem como a possibilidade de integração de métodos distribuídos e criação nativa de métodos na plataforma Apache Spark.

O BigFeel, em sua primeira versão, integra 22 métodos de análise de sentimentos, os quais podem ser invocados diretamente via console em um *cluster* Hadoop/Spark, podem ser utilizados via *Application Programming Interface* (API) por desenvolvedores Java ou Scala, podem também ser usados via API de *web services*, tanto para redes locais quanto remotas (Internet, por exemplo), além de uma interface *web* para experimentação mais simples por leigos.

Foi avaliada, experimentalmente, a eficiência dos 22 métodos integrados. Mediu-se o tempo médio de execução e analisaram-se métricas distribuídas em experimentos com *datasets* sobre livros, eletrônicos e cinema/televisão da Amazon (MCAULEY; HE, 2016). Também foi feita uma avaliação da eficácia usando-se as métricas acurácia, precisão, revocação e $F1$ em um *dataset* de revisões sobre cinema e televisão do *Internet Movie Database* (IMDB) (CHEN et al., 2016), buscando se comparar os métodos integrados com dois métodos supervisionados implementados em Spark. Busca-se nessa avaliação as respostas às questões de pesquisa deste trabalho.

Utilizando-se dos recursos oferecidos pelo ambiente BigFeel, o presente trabalho também propõe um estudo de caso que objetiva a detecção automatizada de sugestões de inovações por análise textual de revisões de produtos. Foi construído um *dataset* constituído de revisões de produtos no domínio de eletrônicos, coletados da Amazon (MCAULEY; HE, 2016). É proposto o uso de heurísticas como a ocorrência de padrões sintáticos, o uso de *lexicons* e também técnicas de aprendizagem de máquina para a detecção automatizada de sugestões de inovação.

As contribuições deste trabalho para a área de Ciência da Computação podem ser sintetizadas a:

- Apresentação do artefato BigFeel, ambiente de código-fonte aberto para experimentação variada pela comunidade acadêmica em análise de sentimentos para grandes volumes de dados;
- Análise e discussão da eficiência e da eficácia na avaliação dos resultados experimentais sob os métodos integrados ao BigFel;
- Apresentação e discussão de estudo de caso sobre a detecção de sugestões de inovações em produtos/serviços pela análise automatizada de um grande volume de revisões realizadas por consumidores.

1.4 Estrutura do Trabalho

Este trabalho é organizado como se segue: o referencial teórico, apresentado no Capítulo 2, aborda os principais conceitos necessários à compreensão deste trabalho, como análise de sentimentos, processamento distribuído de dados e pré-processamento textual. São ainda apresentados no Capítulo 2 alguns trabalhos relacionados, selecionados por uma revisão sistemática de literatura. O Capítulo 3 detalha a arquitetura e as características técnicas da construção do ambiente BigFeel. O Capítulo 4 aborda a análise experimental conduzida para validação da hipótese apresentada. O Capítulo 5 apresenta e discute os resultados obtidos nos experimentos. O Capítulo 6 apresenta um estudo de caso de detecção automatizada de sugestões de inovação por análise de textual de revisões de produtos. Por fim, o Capítulo 7 discute as considerações finais sobre todo o trabalho.

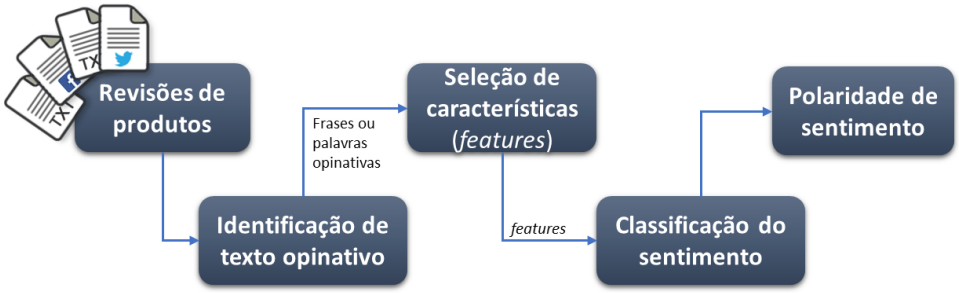
2 REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos fundamentais para o desenvolvimento do trabalho. São definidos conceitos básicos ao entendimento da análise de sentimentos, abordando os passos preliminares à análise, a coleta de dados, o pré-processamento textual e a mineração de dados. Por fim, a análise de sentimentos e o processamento distribuído de dados são abordados, ficando a ultima seção com os conceitos sobre serviços *web*.

2.1 Análise de Sentimentos

A análise de sentimentos é um processo complexo, composto por várias etapas que objetivam a identificação de sentimentos expressos em texto, não somente a análise de sua opinião entre classificações pré-determinadas, denominada mineração de opiniões (TSYTSARAU; PALPANAS, 2011). Segundo Medhat, Hassan e Korashy (2014), seus objetivos são: encontrar as opiniões, identificar os sentimentos que elas expressam e classificar sua polaridade, conforme o exemplo de revisões de produtos mostrado na Figura 2.1:

Figura 2.1 – Processo de análise de sentimentos em revisões de produtos.



Fonte: adaptado de Medhat, Hassan e Korashy (2014).

O processo descrito na Figura 2.1 apresenta as etapas para a classificação do sentimento expresso em texto. Revisões de produtos são coletadas de mídias sociais formando um *dataset*. Utilizando-se *lexicons*[1] ou abordagens de aprendizagem de máquina, palavras ou frases com conteúdo opinativo são destacadas ou extraídas. Em seguida, são selecionadas as características mais salientes na expressão do sentimento da opinião do autor, sendo portanto utilizadas em classificadores para a predição da polaridade de sentimento associada.

[1] *Lexicon*: coleção pré-compilada de termos conhecidos que conotam sentimento (MEDHAT; HASSAN; KORASHY, 2014). Exemplos: ruim, bom, excelente, gostei, ínfimo.

2.2 Passos Preliminares à Análise de Sentimentos

Análise de sentimentos, mineração de opinião e análise de subjetividade são áreas de pesquisa inter-relacionadas que utilizam várias técnicas derivadas do Processamento de Linguagem Natural, Recuperação de Informação e Mineração de Dados Estruturados e não Estruturados (RAVI; RAVI, 2015). A análise de sentimentos não é um problema único, pois se apresenta como um problema de múltiplas faces, necessitando vários passos para de fato se realizar a análise de sentimentos de determinados textos, uma vez que as fontes de dados podem ser heterogêneas e apresentadas em diversos formatos (LIU, 2012). A aquisição de dados e seu pré-processamento são os principais passos preliminares à análise de sentimentos.

Visto que existem variados recursos *online*, a origem dos dados é subjetiva quanto ao tipo de mídia e à aplicação da análise de sentimentos. Algumas das fontes como o Twitter, Facebook e ferramentas para a construção de microblogs oferecem uma API, o que permite acesso facilitado aos desenvolvedores que necessitam dos dados trafegados em tais serviços (KHAN; BASHIR; QAMAR, 2014). Outas fontes exigem o uso (sob autorização) de ferramentas de coleta de tais dados, denominadas *scrapers* (GO; BHAYANI; HUANG, 2009).

2.2.1 Pré-processamento

Na etapa de pré-processamento, o texto é tratado objetivando-se sua futura representação computacional, permitindo assim sua análise. Esse passo comumente é subdividido entre separação em *tokens*, remoção de *stopwords*, radicalização de termos, marcação de *Parts of Speech* (POS) e extração de características e representações. A separação em *tokens* é a etapa onde são quebradas as sentenças do texto em palavras, frases, símbolos ou outros lexemas significativos (como *smiles* e *emoticons*), removendo pontuação e caracteres inúteis à análise. Em seguida são removidos aqueles *tokens* que não contribuem para a análise, as *stopwords*, como exemplo em português: “a”, “o”, “da”, “do”, “também”, dentre outros, os quais compõem uma lista negra de termos a se desconsiderar (*blacklist*). Radicalização de termos é a etapa onde os termos são transformados à sua forma raiz, reduzindo-se assim o número de termos indexados, o que torna a análise de palavras como “andar”, “andou” e “andando” semelhante, analisando-se portanto “and”, o radical comum aos três exemplos (NLTK, 2018). A marcação de POS, por sua vez, é realizada para se definir as diferentes atribuições sintáticas de cada termo em uma oração, essenciais ao processamento de linguagem natural (RAVI; RAVI, 2015).

Por razão da variedade e heterogeneidade dos dados textuais, normalmente é necessário um nível de extração de características. Nesse passo são extraídas as várias propriedades que distinguem uma opinião de outra, as quais serão usadas no processo para a classificação de polaridade positiva, negativa ou neutra de uma sentença (GAUTAM; YADAV, 2014).

O passo seguinte, da seleção de características, viabiliza intrinsecamente a execução da análise de sentimentos buscando selecionar as características relevantes pela conversão de trechos de texto em vetores de características ou outra representação que torne mais salientes os aspectos disponíveis nos dados (PANG; LEE, 2008).

Medhat, Hassan e Korashy (2014) apontam várias técnicas de seleção de características, como por exemplo: *Pointwise Mutual Information* (PMI), teste qui-quadrado e *Latent Semantic Indexing* (LSI). PMI provê uma maneira formal de modelar a informação mútua entre características de um item e as classes existentes no processo de classificação. O teste estatístico qui-quadrado de Pearson é usado como método de seleção das características mais proeminentes (MESLEH, 2007) e o método LSI, que permite a transformação de várias características em um conjunto menor de representação.

2.2.2 Mineração de Dados

A mineração de dados detecta conteúdo útil em grandes volumes de dados como tendências e padrões de mercado, capturando e analisando dados como meio de buscar padrões de interesse, relacionando-os entre si. Duas atividades básicas estão envolvidas: preditivas e descritivas. As atividades preditivas incluem classificação, regressão e detecção de desvios, enquanto as atividades descritivas incluem aglomeração (*clustering*), sumarização, aprendizagem por associação e padrões sequenciais. Várias técnicas de mineração de dados são aplicadas para resolver diferentes problemas da extração de dados em redes sociais, como detecção de influência, detecção de comunidades, buscas especializadas, predição de links, sistemas de recomendação, confiança na predição, dentre outros (GOLE; TIDKE, 2015).

2.3 O Estudo da Análise de Sentimentos

A análise de sentimentos é um campo de estudo que analisa as opiniões públicas, sentimentos, avaliações, atitudes e emoções relacionados com produtos, serviços, organizações, indivíduos, eventos, tópicos e seus atributos. Representa um grande espaço de problema e se apresenta na literatura também com outros nomes, como: extração de opinião, mineração de

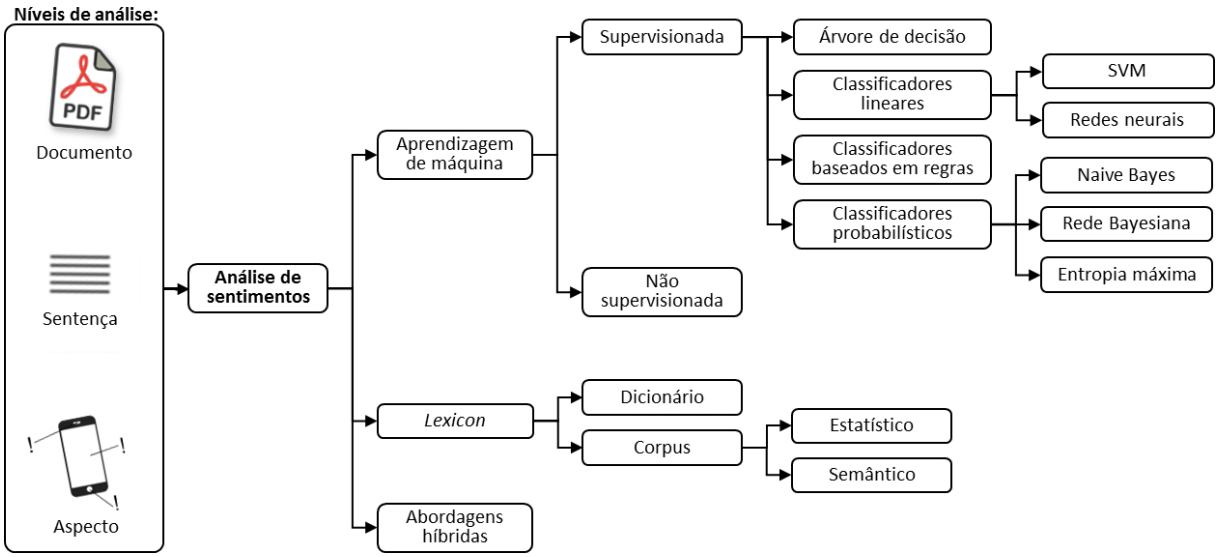
sentimento, análise de subjetividade, análise de afeto, análise de emoções, mineração de revisão, dentre outros (LIU, 2012).

Em grande parte dos estudos, tem se explorado o termo análise de sentimentos principalmente na questão da classificação de polaridade da opinião pública sobre determinado assunto ou item, tornando esse um problema focado principalmente no discernimento automatizado entre as expressões positivas e negativas sobre o assunto (RIBEIRO et al., 2016; NODARAKIS et al., 2016; RAVI; RAVI, 2015).

A classificação de polaridade de um determinado texto, sentença ou nível de característica é a atividade básica de um processo de análise de sentimentos. Exemplos de classificação de polaridade são opiniões sobre algo usando curti ou não curti, no Facebook (CAMBRIA, 2016).

Existem abordagens que extrapolam a polaridade de uma classificação, definindo o objeto analisado entre classes pré-definidas, como exemplo: triste, feliz ou satisfeito. Ressalta-se que a análise de sentimentos pode ser realizada não somente em textos, mas em todo tipo de mídia, como vídeo e áudio (BORTH et al., 2013).

Figura 2.2 – Abordagens de análise de sentimentos.



Fonte: Adaptado de Medhat, Hassan e Korashy (2014).

A Figura 2.2 apresenta o fato de que as técnicas de classificação de sentimentos podem ser divididas em abordagens por aprendizagem de máquina, abordagens baseadas em *lexicons* e também abordagens híbridas entre essas duas. No primeiro caso, são aplicados algoritmos da área de aprendizagem de máquina utilizando-se de recursos linguísticos, como PLN. Os métodos de classificação textual supervisionados fazem o uso de um conjunto de dados, normalmente classificados previamente de forma manual, ao qual se dá o nome de dados de treina-

mento. Com base nesses dados de treinamento, a classificação é feita sob os dados de análise. O caso não supervisionado é aplicado para os casos nos quais não se consegue criar tal conjunto de treinamento (MEDHAT; HASSAN; KORASHY, 2014).

Os métodos de análise de sentimentos existentes podem ser agrupados em três categorias: técnicas baseadas em conhecimento, métodos estatísticos e abordagens híbridas. Técnicas baseadas em conhecimento classificam o texto por categorias de afeto baseadas na presença de palavras não ambíguas como feliz, triste, medo, receoso e entediado (CAMBRIA, 2016). Algumas bases de dados de conhecimento não incluem apenas palavras óbvias de afeto, mas também atribuem as probabilidades das palavras estarem associadas por afinidade a determinadas emoções (STEVENSON; MIKELS; JAMES, 2007).

Os métodos estatísticos se aproveitam de elementos da área de aprendizagem de máquina, como a LSI, as máquinas de vetores de suporte (do inglês: *support vector machines* (SVMs)) e de PMI. Algumas abordagens híbridas se baseiam na mistura de aprendizagem de máquina e de elementos da representação de conhecimento, como ontologias e redes semânticas, detectando-se assim o significado expressado sobre determinado tema (KIM; HOVY, 2006).

O problema computacional da análise de sentimentos está no uso de termos simples para tentar expressar sentimentos sobre produtos ou serviços que, muitas das vezes, têm sua classificação difícil até mesmo para os seres humanos. Fatores culturais, nuances linguísticas e diferentes contextos tornam esse processo extremamente difícil. O simples fato de que os seres humanos frequentemente divergem suas opiniões sobre assuntos simples exemplifica bem o problema computacional, pois quanto menor o texto, como exemplo um *tweet*, mais difícil é sua análise (WRIGHT, 2009).

2.4 Processamento Distribuído de Dados

A geração de dados em mídias sociais e na Internet das coisas (do inglês: *Internet of Things*) expande em grande escala e, segundo estudos, continuará exponencialmente crescendo. A imensa quantidade de dados, normalmente conhecida como *Big Data*, tende a crescer com o tempo, acompanhando a evolução tecnológica. O Facebook tem 600 milhões de usuários ativos, os quais utilizam o serviço cerca de 9,3 bilhões de horas por mês, gerando a cada mês, mais de 30 milhões de itens de conteúdo, incluindo fotos, anotações, postagens, *links* e notícias. *Big Data* não pode ser mantido e nem manipulado por bancos de dados relacionais convencionais

(*Relational Database Systems* - RDBMS), assim Sistemas Gerenciadores de Bancos de Dados (SGBDs) NoSQL e plataformas como o Apache Hadoop e o Apache Spark, foram desenvolvidos para lidar com tal problema de forma reduzida (GOLE; TIDKE, 2015).

Muito embora a dicotomia SQL - NoSQL esteja deixando de ser relevante (BACON et al., 2017), as plataformas Hadoop e Spark despontam como o estado da arte no que tange ao processamento de grandes volumes de dados. Esta seção introduz os conceitos relacionados à essas plataformas.

2.4.1 Hadoop

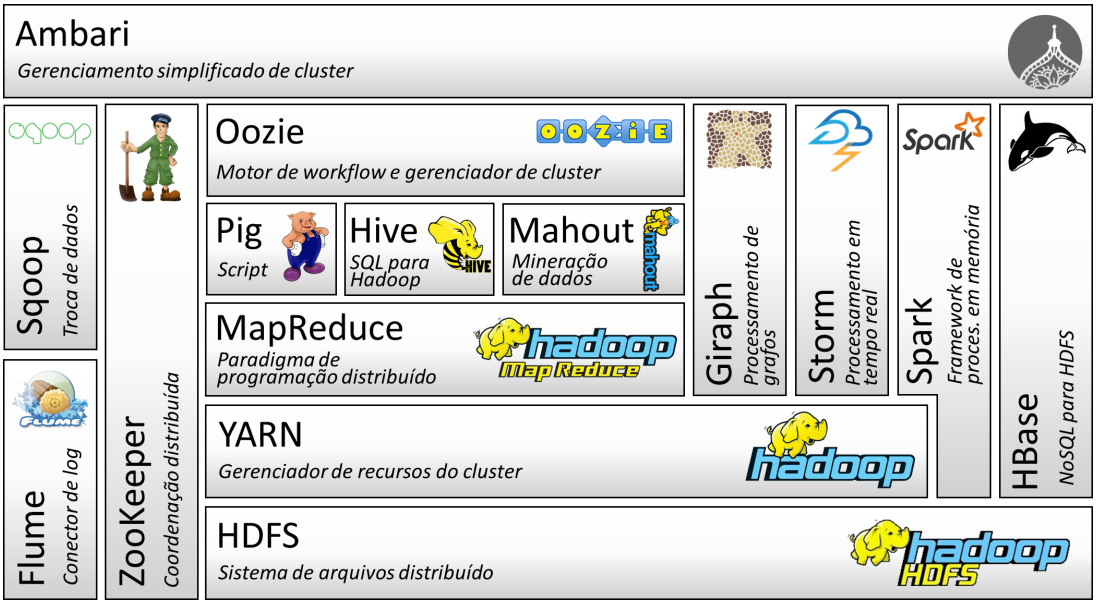
Apache Hadoop é uma plataforma que permite o processamento de grandes volumes de dados em *clusters* formados por computadores padrão *Personal Computer* (PC). A distribuição do processamento de dados é implementada pelo paradigma de programação MapReduce, o qual foi introduzido por Dean e Ghemawat (2004), está implementado no Hadoop e trouxe novas capacidades para a comunidade de *software* livre (WHITE, 2015).

Os dois principais componentes do Hadoop são o *Hadoop Distributed File System* (HDFS) e o modelo de programação MapReduce. O HDFS é um sistema de arquivos distribuído altamente configurável, tolerante a falhas, escalável e escrito em linguagem Java. Um *cluster* Hadoop tem um nó mestre, que gerencia a sincronização e coordena o acesso e escrita de dados nos demais nós do HDFS, armazenando também metadados necessários à resiliência dos dados, afinal um mesmo dado é replicado entre vários nós. Um *job* implementado em MapReduce possui dois componentes principais: *map* e *reduce*. O Hadoop divide os dados de entrada em múltiplas partições, assim múltiplas rotinas *map* podem processar cada partição em paralelo. A saída das rotinas *map* são coletadas e processadas pelas rotinas *reduce*. Logo, como as rotinas de *map* e *reduce* operam em pares <chave, valor>, as entradas e saídas também necessitam desse mesmo valor, sendo portanto armazenadas em HDFS em ambos os casos (HOLMES, 2015).

Com a utilização do Hadoop, o sistema toma conta dos detalhes de partição dos dados de entrada, gerenciando a execução da aplicação em um conjunto de computadores em paralelo, tratando falhas de nós do *cluster*, e gerenciando a comunicação entre as máquinas. Além disso, o Hadoop proporciona várias outras soluções focadas no processamento paralelo distribuído, formando assim seu ecossistema (HADOOP, 2018). A Figura 2.3 demonstra por um diagrama de blocos em camadas a disposição e a hierarquia básica de alguns dos componentes oferecidos pela plataforma Hadoop, apresentando por exemplo que Giraph, Storm e Spark são

implementações independentes do paradigma de programação MapReduce, cada qual acrescentando serviços e funcionalidades próprias.

Figura 2.3 – Pilha de componentes da plataforma Hadoop.



O Hadoop é capaz de oferecer uma arquitetura altamente escalável e flexível para o processamento paralelo, pois ao contrário de confiar em *hardware* para proporcionar alta disponibilidade, se baseia na ideia de detectar e tratar falhas na camada de aplicação, de modo a fornecer um serviço de alta disponibilidade para um conjunto de computadores, cada um dos quais propenso a falhas, se tornando assim independente de *hardware* de alta capacidade de processamento ou armazenamento para concluir grandes tarefas (WHITE, 2015; RAJURKAR; GOUDAR, 2015).

Embora o Hadoop ofereça o Yarn como opção de gerenciador de recursos do *cluster* e escalonador de tarefas, a abordagem adotada neste trabalho se fixou no uso de sua estrutura comum de suporte aos demais módulos Apache e em seu sistema de arquivos distribuído HDFS, ficando portanto o gerenciamento de recursos e escalonamento de tarefas do *cluster* à cargo do Spark, conforme é melhor detalhado na subseção a seguir.

2.4.2 Spark

Apache Spark é uma plataforma de propósito geral para processamento de dados de forma distribuída em larga escala. Considerado como uma evolução do Hadoop (NODARAKIS et al., 2016), apesar de ambos possuírem implementações distintas do paradigma de programa-

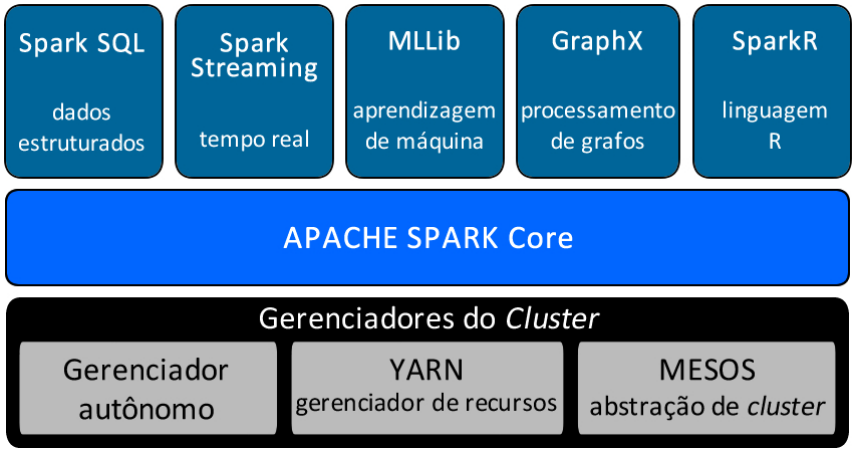
ção MapReduce, para algumas atividades o Spark obtém um desempenho até dez vezes superior ao Hadoop (SHI et al., 2015).

O desempenho do Hadoop reduz radicalmente para certos tipos de problemas, como exemplo quando há a manipulação iterativa de algoritmos baseados em grafos. O componente de manipulação de grafos acíclicos dirigidos (DAG), junto à capacidade de fluxo de dados em memória, resolvem os problemas ocasionados no Hadoop para esse tipo de algoritmo, resultando assim em programas com execução até 100 vezes mais rápidos em memória e até dez vezes mais rápidos em disco (KARAU et al., 2015; NODARAKIS et al., 2016).

Possuindo uma implementação própria do modelo MapReduce otimizado para o processamento em memória, o Spark se integra nativamente ao sistema de arquivos distribuído oferecido pelo Hadoop, o HDFS. Oferece ainda suporte à implementação de programas em linguagem Java, Scala, Python e R (KARAU; WARREN, 2017). A implementação do ambiente BigFeel foi realizada em linguagem Scala, e isso se deu justamente pela implementação original do Spark ser realizada nessa linguagem, tornando mais simples, versátil e produtivo o desenvolvimento sob Spark nessa linguagem (EDER, 2014; SPANN, 2016).

A Figura 2.4 apresenta a pilha de componentes Spark. Destes, foram usados no BigFeel o gerenciador autônomo do *cluster*, como gestor de recursos de memória, nós, partições de processamento e tratamento de falhas; o Core, núcleo de gerenciamento de aplicações Spark; recursos do Spark SQL, para manipulação de dados estruturados por meio da API de *DataFrames*/*Datasets*; e MLLib, biblioteca de métodos e técnicas envolvidos com a aprendizagem de máquina.

Figura 2.4 – Pilha de componentes da plataforma Spark.



Além do Core, componente essencial para uso do Spark, os demais componentes apresentados na Figura 2.4, segundo Karau et al. (2015), são:

- Spark SQL: pacote para manipulação de dados estruturados via *Structured Query Language* (SQL) ou Apache Hive SQL, denominado *Hive Query Language* (HQL).
- Spark *Streaming*: componente que habilita o processamento de *streams* de dados em tempo real. Exemplos: arquivos de *log* gerados por servidores *web* em produção ou filas de atualização de status de serviços acessados via *web services*.
- MLLib: biblioteca de funções comuns para aprendizagem de máquina, incluindo algoritmos de classificação, regressão, *clustering*, filtragem colaborativa, assim como suporte a funcionalidades como avaliação de modelos e importação de dados.
- GraphX: biblioteca para manipulação de grafos e computação paralela em grafos, como exemplo, grafo que represente a relação de amizades em redes sociais.
- Gerenciadores do *cluster*: componentes responsáveis pela garantia de escalabilidade eficiente de um a milhares de nós que compõem o *cluster*. Para garantir flexibilidade, o Spark permite sua implantação em uma variedade de gerenciadores de *cluster*, como Hadoop YARN, Apache Mesos ou até mesmo uma opção de gerenciador de *cluster* autônomo incluído no próprio Spark, o Gerenciador Autônomo.

Da mesma forma que ocorre no caso do Hadoop, diversas são as fontes instruindo a melhor maneira de se instalar, configurar e usar um *cluster* Apache Spark. Um passo-a-passo seguido para a instalação e configuração do ambiente proposto neste trabalho (BigFeel) pode ser obtido no repositório Git. Tal material aborda a instalação da versão 2.1.2 do Apache Spark junto ao Hadoop 2.8, além das instruções de instalação e uso do Spark *History Server*, componente indispensável para *log* de execução de aplicações no *cluster*.

2.5 Serviços Web

Serviços *web* são considerados como aplicações modulares auto-contidas e auto-descritas que podem ser publicadas, localizadas e invocadas pela *web* (RAO; SU, 2005). Para Koehler e Srivastava (2003), a tendência de evolução da arquitetura de *software* está na construção de aplicações independentes de plataforma, as quais suportam a interação de entidades de negócio e

seus processos, oferecendo um modelo de troca de mensagens de forma síncrona ou assíncrona atômico (auto-contido) e sem estado (nada é armazenado sobre o estado de cada requisição após seu término).

A especificação de serviços *web* no modelo *Simple Object Access Protocol* (SOAP) é expressada em linguagem de descrição de serviços *web* (WSDL), a qual especifica o padrão de entrada e saída de dados[2]. Uma abordagem mais recente se apresenta na arquitetura REST ou serviço *web* RESTFul, o qual dispensa a necessidade de uma linguagem para descrever os serviços oferecidos, ainda que exista a linguagem de descrição de aplicações *web* (WADL) análoga à WSDL. Ao contrário da arquitetura SOAP, REST define serviços na *web* com uma interface uniforme usando os verbos já implementados em HTTP: GET, POST, PUT e DELETE, fazendo assim com que cada recurso disponibilizado como um serviço *web* RESTful tenha tais métodos implementados (ASNIKA; VASUDEVA; SUDHINDRA, 2014; RAO; SU, 2005). Como exemplo: "GET /servicoweb/recurso" retornaria uma lista do recurso solicitado.

As operações definidas para cada recurso têm as seguintes atribuições, segundo Richardson e Ruby (2007):

- GET: recupera informações do recurso solicitado em forma de entidades ou objetos a serem manipulado pela aplicação cliente. Não gera alteração de dados.
- POST: requisição que envia dados ao serviço *web*, o qual os reconhece como dados para a inserção de nova instância do recurso solicitado (ex.: POST /servicoweb/cliente). Gera alteração de dados.
- PUT: requisição que envia dados ao serviço *web*, o qual os reconhece como dados para atualização de uma instância já existente do recurso solicitado.
- DELETE: requisição para exclusão de uma instância existente do recurso solicitado.

Dentre as vantagens introduzidas pela arquitetura REST, destacam-se a velocidade dos serviços (serviços RESTful são projetados para serem rápidos), simplicidade de implementação e baixo custo (baixo consumo de banda e curva de aprendizagem para a implementação menor), além de recursos de *caching* em *HyperText Transfer Protocol* (HTTP) e fácil integração com aplicações existentes via *eXtensible Markup Language* (XML) e *JavaScript Object Notation* (JSON) (ASNIKA; VASUDEVA; SUDHINDRA, 2014).

[2] https://www.w3.org/TR/wsdl

2.6 Trabalhos Relacionados

Alguns estudos abordaram parte do problema tratado na proposta deste trabalho. Baldominos et al. (2014) propõem uma arquitetura de aprendizagem de máquina com API RESTFul para processamento de tempo real em *Big Data*. O BigFeel também utiliza uma API de *web services* como interface para acesso aos serviços distribuídos de um *cluster* Hadoop/Spark, podendo assim, por exemplo, encapsular o depósito de grandes arquivos em sistema distribuído de forma transparente ao usuário final da aplicação, se resumindo portanto sua atitude ao *upload* de dados, assim como também abordado por Rajurkar e Goudar (2015).

O ambiente iFeel[3], proposto por Gonçalves et al. (2013), oferece a análise de sentimentos de vários métodos, porém restrito a pequenos *datasets* ou até mesmo a uma análise de demonstração de uma sentença. A versão mais recente do iFeel, com mais métodos, é apresentada no trabalho de Ribeiro et al. (2016), os quais ainda oferecem a implementação Java não distribuída de seus métodos em um repositório Git[4]. Isso permitiu a adaptação e integração de variados métodos ao ambiente distribuído proposto neste trabalho, o BigFeel. Contudo, ressalta-se que o foco do BigFeel está na possibilidade de experimentação em análise de sentimentos escalável, ou seja, para grandes volumes de dados.

Na área de pré-processamento textual, bem como processamento de linguagem natural (PLN), os trabalhos de Dasgupta et al. (2015) e Woldemariam (2016) abordam técnicas de limpeza do texto, remoção de palavras indesejadas (*stopwords*), radicalização de termos, além da análise de sentimentos propriamente dita. Woldemariam (2016) ainda realiza a comparação entre as abordagens baseadas em *lexicons* (dicionário) e aprendizagem de máquina, tratando de forma distribuída a aquisição de múltiplas fontes, ideia essa apontada como desejável para trabalhos futuros no BigFeel.

Como referência de uso do Spark na implementação de soluções de análise de sentimentos em *Big Data*, Oneto et al. (2016) apresentam uma abordagem envolvendo *Statistical Learning Theory* (SLT) fundamentada em *Extreme Learning Machine* (ELM).

Como não existem, até o momento, trabalhos diretamente relacionados a um ambiente de integração de variados métodos de análise de sentimentos para processamento distribuído de grandes volumes de dados, as referências aqui citadas configuram parcialmente algumas das ideias adotadas e aplicadas ao BigFeel. Muitas dessas, relacionadas em semelhante contexto

[3] iFeel: http://blackbird.dcc.ufmg.br:1210
[4] Repositório Git iFeel: https://bitbucket.org/matheusaraujo/methodsjava

de uso, compostas por soluções envolvendo Hadoop, Spark e ferramentas de processamento de linguagem natural, assim como os trabalhos comentados a seguir.

Burdorf (2016) apresenta um trabalho em progresso de uma solução distribuída para análise de sentimentos de uso industrial, baseado em tecnologia recente e de acesso livre. Usando a linguagem de programação Scala e um ambiente composto pelo HDFS, HBase, NLTK e Stanford Core NLP, o autor apresenta o objetivo alcançado da distribuição do algoritmo de classificação implementado sob o *framework* Akka, porém aponta o gargalo de desempenho do trabalho, citando o problema do tempo de criação dos conjuntos de dados para treinamento.

Bakırov, Çoğalmış e Bulut (2016) demonstram os passos para a criação de um simples classificador textual usando Spark, Hadoop e Mahout em sete milhões de revisões de cinema e televisão. Seu trabalho demonstra uma eficiência até dez vezes superior entre as implementações em Spark comparadas às implementadas em Hadoop MapReduce.

Como ferramentas para o pré-processamento textual e de linguagem natural, Manning et al. (2014) apresentam de forma simplificada seu trabalho e o modo de uso da ferramenta, denominada Stanford CoreNLP. Dentre os métodos oferecidos pelo ambiente, destacam-se o rotulador de análise sintática de sentenças, conhecido como *Parts-Of-Speech Tagger* (POS), o Reconhecimento de Entidades Nomeadas (NER), além de um método de análise de sentimentos, que compõe as integrações realizadas para o BigFeel.

3 O AMBIENTE BIGFEEL

Neste capítulo é apresentada a arquitetura e os detalhes envolvidos na construção do ambiente BigFeel. A Seção 3.1 apresenta detalhes de alto nível da arquitetura proposta. As Seções 3.2 e 3.3 abordam os métodos integrados ao BigFeel. A Seção 3.4 apresenta os detalhes da composição de métodos. A Seção 3.5 apresenta informações técnicas para a integração de novos métodos ao portifólio de serviços do BigFeel e para o seu uso. A Seção 3.6 apresenta os aspectos técnicos das soluções adotadas nos problemas encontrados durante o projeto e a Seção 3.7 discute ideias alternativas à solução proposta para a construção do BigFeel.

3.1 Descrição do Ambiente e Arquitetura

O BigFeel é um ambiente distribuído de código-fonte aberto, implementado em Apache Spark na linguagem Scala. Integra várias ferramentas e métodos envolvidos com a área de estudo da análise de sentimentos, como processamento de linguagem natural e aprendizagem de máquina. O ambiente oferece quatro possibilidades de uso de seus métodos: 1) por desenvolvedor local em linguagem Java ou Scala, integrando seu projeto em desenvolvimento à API do BigFeel; 2) por desenvolvedor *web*, em qualquer linguagem de programação *server-side*, consumindo *web-services* oferecidos pelo BigFeel via rede local ou remota; 3) por administrador de *cluster* Hadoop/Spark, via execução de tarefas por console; e 4) por leigos em programação, via interface *web*[1] simplificada.

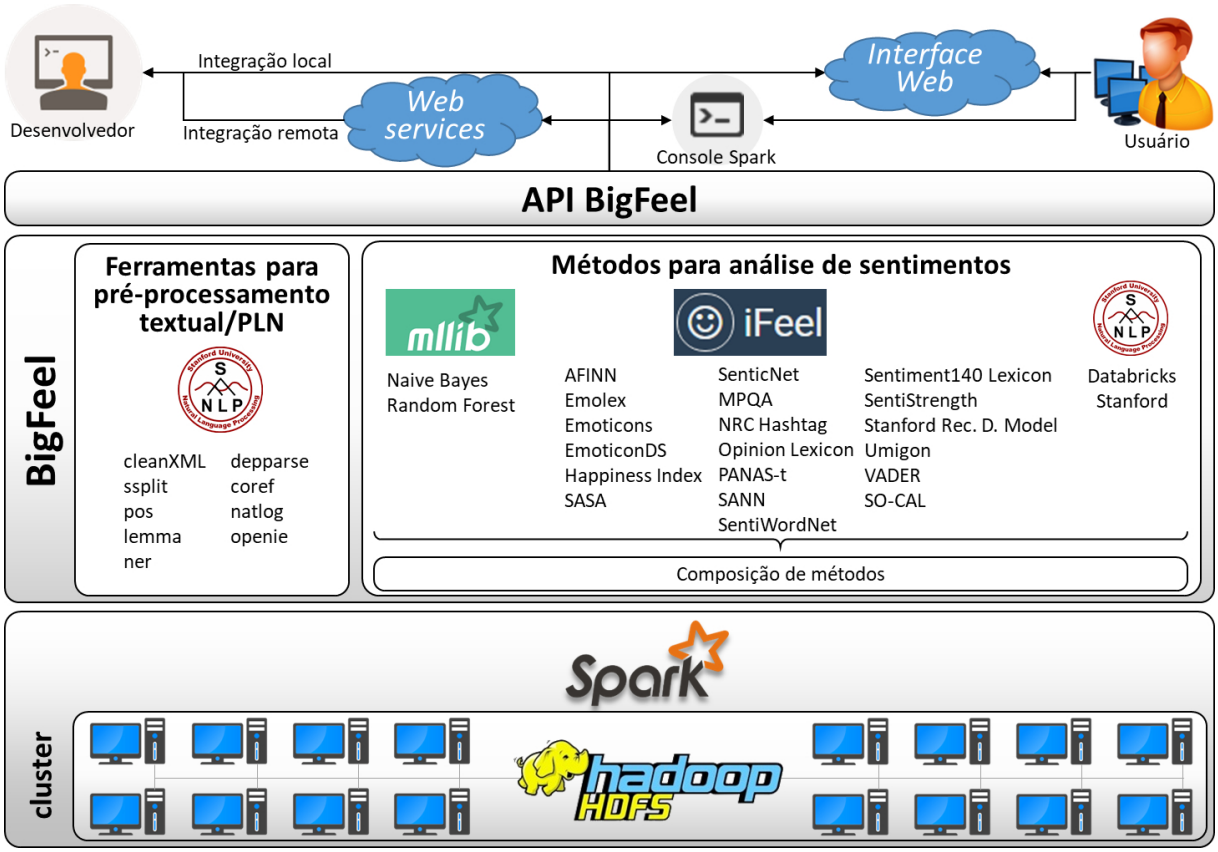
Em sua primeira versão, foram integrados ao BigFeel métodos de pré-processamento textual, implementações de classificadores do Spark, além de 19 métodos implementados pelo trabalho de Ribeiro et al. (2016), composto por abordagens de *lexicons* e aprendizagem de máquina. A implementação do encapsulamento de Meng (2016) do método de Stanford CoreNLP (MANNING et al., 2014) também foi integrada para comparação de resultados com a implementação do mesmo método, Stanford *Recursive Deep Model*, implementado pelo trabalho de Ribeiro et al. (2016).

O ambiente BigFeel possibilita a integração e/ou substituição de novos métodos ao seu portfólio de serviços, possibilitando assim a continuidade ou extensão de suas capacidades. A modificação deve ser orientada pelas instruções técnicas constantes no documento "*Extending BigFeel*", disponível no repositório Git do projeto.

[1] Repositório Git da interface *web* do BigFeel: https://bitbucket.org/rogersantosferreira/bigfeel_web

A implantação do ambiente BigFeel envolve a pré-existência de um *cluster* de computadores executando a plataforma Apache Spark, empilhado sob um *cluster* Hadoop e seu sistema de arquivos distribuído HDFS. A Figura 3.1 apresenta a infraestrutura de *cluster* necessária que permitiu a construção e integração dos variados métodos e serviços oferecidos pelo BigFeel.

Figura 3.1 – Diagrama de pilha da arquitetura do BigFeel.



A camada de API oferece serviços para chamadas externas aos métodos do BigFeel, por meio das quatro interfaces de acesso. Para uso dessa API, definiu-se a entrada de dados como arquivos de tamanho indiscriminado previamente depositados em *cluster* com sistema de arquivos distribuído HDFS. Para isso, existe um método dentre os *web services* do BigFeel, denominado *upload*, que é responsável pelo envio e alocação no HDFS do arquivo submetido, retornando um identificador único para que possa ser usado nas chamadas de outros métodos da aplicação. No caso do uso do BigFeel no modo console Spark, essa responsabilidade da prévia alocação dos dados em HDFS é atribuída ao usuário.

A camada de integração de métodos de pré-processamento textual, de processamento de linguagem natural e de análise de sentimentos é o principal componente do ambiente BigFeel, pois nessa camada estão localizados os adaptadores para cada uma das ferramentas e métodos integrados ao ambiente.

A camada do *cluster* Spark/Hadoop é representada na Figura 3.1 como a base de toda a arquitetura, pois as principais dependências do BigFeel são o processamento em memória, as bibliotecas e as estruturas de dados oferecidas pelo Spark, além do sistema de arquivos distribuído HDFS, oferecido pelo Hadoop. Essa arquitetura oferece recursos de resiliência e escalabilidade de dados, o que torna possível o processamento do ambiente proposto em *Big Data*.

3.2 Integração de Métodos de Análise de Sentimentos

Esta seção descreve os métodos de análise de sentimentos e sua integração ao BigFeel, sendo dois deles métodos de implementação própria utilizando a biblioteca MLLib por meio dos classificadores Naive Bayes e Random Forest, ambos treinados com *dataset* contendo revisões de usuários na internet sobre o domínio de cinema e televisão (CHEN et al., 2016). A escolha desses dois métodos é arbitrária e não implica que outros métodos oferecidos pela MLLib não possam ser adicionados ao BigFeel. Para mais detalhes e instruções técnicas sobre a adição de novos métodos ao BigFeel, verificar o documento "*Extending BigFeel*" no repositório Git do projeto e, para treinar novamente os modelos sob outro domínio, basta se executar novamente os passos constantes no método main de src/main/scala/adapters/SparkML4_Trainning_Testing, observando os comentários presentes a cada passo do método.

Dos métodos implementados por Ribeiro et al. (2016), foram integrados 19 métodos daqueles que possuem licença permissiva para propósitos de pesquisa ou são completamente livres. Foi ainda adicionado um método de análise de sentimentos de Stanford CoreNLP (MANNING et al., 2014), implementado por Meng (2016).

Observando na listagem abaixo, onde [L] indica o uso de uma abordagem por *Lexicon* e [ML] por aprendizagem de máquina, os métodos integrados ao BigFeel são (adaptado de Ribeiro et al. (2016)):

- **AFINN [L]:** *Lexicon* de sentimentos baseado em *tweets* incluindo gírias da internet e palavras obscenas, com um total de 2.477 termos (AFINN-111). AFINN pode ser considerado como uma expansão do ANEW, um dicionário criado para prover pontuação emocional para as palavras do inglês. O dicionário ANEW avalia as palavras em termos de prazer, excitação e dominância (NIELSEN, 2011).

- **Emolex [L]:** *Lexicon* de sentimentos geral baseado em *crowdsourcing*. Cada entrada do dicionário lista a associação de um *token* com oito sentimentos básicos: alegria, tristeza, raiva, dentre outros definidos por Plutchik (1980). O *lexicon* proposto ainda inclui unigramas e bigramas de Macquarie Thesaurus e palavras do General Inquirer e WordNet (MOHAMMAD; TURNEY, 2013).

- **Emoticons [L]:** Mensagens contendo *emoticons* negativos/positivos são avaliadas como negativas/positivas, enquanto que mensagens sem *emoticons* não são avaliadas. Dividido em três dicionários, positivo, neutro e negativo (GONÇALVES et al., 2013).

- **EmoticonDS [L]:** *Lexicon* pontuado com base em um grande *dataset* de *tweets*. É baseado na frequência em que cada termo aparece associado a emoções positivas ou negativas utilizando abordagem *Distant Supervised* (HANNAK et al., 2012).

- **Happiness Index [L]:** Método baseado no uso do dicionário ANEW. Usando os valores atribuídos pelo dicionário ANEW os autores propuseram o cálculo de uma valência geral, a qual pode ser usada como uma sinalização do grau de felicidade atribuído a uma determinada sentença (DODDS; DANFORTH, 2010).

- **MPQA [L][ML]:** Análise de subjetividade por meio de um *framework* com análise léxica e posterior abordagem de aprendizagem de máquina (WILSON et al., 2005; WILSON; WIEBE; HOFFMAN, 2005).

- **NRC Hashtag [L]:** *Lexicon* usando a abordagem *Distant Supervised*. Utiliza *hashtags* conhecidas como #alegria, #feliz, etc. para classificar um *tweet*. Verifica a frequência em que cada *n*-grama específico implica emoção, calculando sua força de associação com aquela emoção (MOHAMMAD, 2012).

- **Opinion Lexicon [L]:** *Lexicon* para a predição de polaridade de sentenças sobre características de produtos, as quais são sumarizadas para prover uma pontuação geral sobre aquela característica (HU; LIU, 2004).

  PANAS-t [L]: Detecção de flutuações de humor de usuários no Twitter. O método consiste de uma versão adaptada de *Positive Affect Negative Affect Scale* (PANAS), conhecido método da psicologia com um grande conjunto de palavras, cada uma associada com um dos 11 humores, como surpreso, medo, culpado, etc (GONÇALVES; BENEVENUTO; CHA, 2013).

- **SANN [L][ML]:** Proposta de um modelo *Sentiment-Aware Nearest Neighbor* (SANN) utilizando-se de uma extensão do classificador baseado em regras de Wilson et al. (2005), que utiliza o *lexicon* MPQA (PAPPAS; POPESCU-BELIS, 2013).

- **SASA [ML]:** Método usado para a detecção de sentimentos públicos no Twitter durante a eleição presidencial dos E.U.A. de 2012. Baseado no modelo estatístico obtido do classificador Naive Bayes de *features* unigramas. Explora ainda *emoticons* e exclamações (WANG et al., 2012).

- **SenticNet [L]:** Utiliza da redução de dimensionalidade para inferir a polaridade de conceitos de senso comum, além de prover recursos para a mineração de opinião textual em nível semântico, e não só em nível sintático (CAMBRIA; OLSHER; RAJAGOPAL, 2014).

- **Sentiment140 Lexicon [L]:** Sentiment140 Lexicon (conhecido anteriormente como "*Twitter Sentiment*") foi proposto como um *ensemble* de três classificadores (Naive Bayes, Maximum Entropy e SVM) construído com um grande conjunto de *tweets* contendo *emoticons* coletados pelos autores. Após vários avanços, foi transformado em uma ferramenta proprietária. O *lexicon* usado no projeto é baseado no mesmo *dataset* usado para treinamento do método Sentiment140. Foi construído de modo similar ao método do NRC *Hashtag* de Mohammad (2012), mas os autores usaram a ocorrência de *emoticons* para classificar o *tweet* como positivo ou negativo. A pontuação por termo foi calculada baseada na frequência de ocorrência em cada classe de *tweets* (MOHAMMAD; KIRITCHENKO; ZHU, 2013).

- **SentiStrength [L][ML]:** Dicionário léxico anotado por humanos e melhorado pelo uso de técnicas de aprendizagem de máquina (THELWALL, 2017).

- **SentiWordNet [L][ML]:** *Lexicon* para mineração de opinião baseado no dicionário WordNet. Os autores agruparam adjetivos, substantivos, e outros em conjuntos de sinônimos (*synsets*) e associaram três polaridades de pontuação (positivo, negativo e neutro) para cada conjunto desses (ESULI; SEBASTIANI, 2006; BACCIANELLA; ESULI; SEBASTIANI, 2010).

- **SO-CAL [L]:** *Lexicon* com unigramas (verbos, advérbios, substantivos e adjetivos) e multigramas (*phrasal verbs* e intensificadores) manualmente rotulados em uma escala

entre +5 (fortemente positivo) e -5 (fortemente negativo). Os autores também incluíram processamento de *Parts-of-Speech* (POS), negação e intensificadores (TABOADA et al., 2011).

- **Stanford *Recursive Deep Model* [L][ML]:** Proposta de um modelo chamado *Recursive Neural Tensor Network* (RNTN) que processa todas as sentenças em um texto, lidando com sua estrutura sintática e calculando a interação entre elas. Essa abordagem considera a ordem das palavras na sentença, fato ignorado pela maioria dos métodos de análise de sentimentos (SOCHER; PERELYGIN; WU, 2013).

- **Umigon [L]:** Desambiguação de *tweets* usando um *lexicon* com heurística para detecção de negações, palavras alongadas e avaliação de *hashtags* (LEVALLOIS, 2013).

- **VADER [L]:** Método de análise de sentimentos com validação humana desenvolvido para o Twitter e contextos de mídias sociais. Foi criado de um *lexicon* de sentimentos padrão rotulado por especialistas e baseado em valências, generalista (HUTTO; GILBERT, 2014).

- **Spark Naive Bayes [ML]:** Implementação Spark em linguagem Scala do classificador Naive Bayes dos próprios autores. Treinada com *dataset* sob o domínio no contexto de cinema e televisão.

- **Spark Random Forest [ML]:** Implementação Spark em linguagem Scala do classificador Random Forest dos próprios autores. Treinada com *dataset* sob o domínio no contexto de cinema e televisão.

- **Databricks Stanford [L][ML]:** Implementação da Databricks que mede o sentimento expresso por uma sentença na escala entre 0 (fortemente negativo) a 4 (fortemente positivo) (MENG, 2016).

- **Composition:** Método de implementação própria dos autores que oferece a composição entre os outros métodos disponibilizados pelo BigFeel por meio de atribuição de pesos aos métodos selecionados para a composição, resultando assim uma única análise baseada na mistura dos resultados de vários outros métodos. A seção 3.4 apresenta detalhes e um exemplo de uso da composição.

Para permitir a comparação de resultados, seguindo a abordagem de Ribeiro et al. (2016), os resultados apresentados pelo BigFeel foram normalizados e adaptados para o formato numérico de -1, 0 ou 1, respectivamente significando uma análise resultante negativa, neutra ou positiva, análoga à polaridade do sentimento expresso pelo autor da revisão sobre um produto ou serviço.

3.3 Integração de Técnicas de Pré-processamento Textual e PLN

O uso dos serviços oferecidos pelo BigFeel não limita o desenvolvedor às ferramentas por ele oferecidas, todo o conjunto de métodos e ferramentas do Spark podem ser usados em comum acordo com o código-fonte do ambiente BigFeel. Várias ferramentas envolvidas nas técnicas de aprendizagem de máquina como extratores, transformadores e seletores de *features*, além de classificadores, customização e seleção de modelos estão disponíveis para uso de desenvolvedores diretamente pelas APIs do Spark (SPARK, 2018a), porém algumas técnicas envolvidas no pré-processamento textual, como métodos de PLN são necessários em análise de sentimentos. Assim, o BigFeel oferece os métodos encapsulados pela DATABRICKS (2018) do motor de PLN Stanford CoreNLP (MENG, 2016), os quais são descritos no Quadro 3.1.

Quadro 3.1 – Ferramentas de pré-processamento textual e PLN oferecidas pelo BigFeel.

Método	Descrição
cleanxml	Limpeza de *tags* XML de documentos inteiros (entrada: texto com *tags* XML; saída: texto sem *tags* XML).
ssplit	Divisão de texto em sentenças (entrada: texto; saída: lista de sentenças).
pos	Análise sintática de sentenças - *Parts of Speech* (entrada: sentença; saída: lista de *tags* POS).
lemma	Forma básica para cada palavra de uma sentença (entrada: sentença; saída: lista de palavras em sua forma básica).
ner	Reconhecimento de entidades nomeadas - *Named Entities Recognition* (entrada: sentença; saída: lista de entidades nomeadas reconhecidas na sentença).
depparse	Dependências semânticas de uma sentença (entrada: sentença; saída: tuplas no formato [origem, índice_origem, relação, objetivo, índice_objetivo, peso]).
coref	Cadeias de correferência em um documento (entrada: sentença; saída: tuplas encadeadas no formato [rep, menções], onde as menções são formatadas como [número_sentimento, índice_inicial, menção]).
natlog	Noção lógica natural de polaridade para cada *token* em uma sentença (entrada: sentença; saída: *up*, *down* ou *flat*).
openie	Lista de triplas Open IE (entrada: sentença; saída: triplas no formato [sujeito, relação, objetivo, confidência]).

Fonte: adaptado de Meng (2016).

3.4 A Composição de Métodos

O BigFeel também fornece a opção de composição de métodos, a qual possibilita a execução de tarefas de análise de sentimentos em lote por meio da atribuição de pesos ao resultado de cada um dos métodos, de acordo com a Fórmula 1. Nessa fórmula, met_i e w_i são, respectivamente, o nome e o peso do método i. O resultado final da composição é o valor máximo obtido entre POS, NEU e NEG, representando o valor ponderado dos métodos que resultaram a polaridade positiva, neutra e negativa, respectivamente. Em caso de empate, retorna-se o resultado do método de maior peso entre os que retornaram a polaridade do empate. Em caso de novo empate, retorna-se o resultado do método de menor índice i dentre aqueles que retornaram a polaridade do empate.

Fórmula 1 – Regras de cálculo da composição de métodos de análise de sentimentos. met_i e w_i são, respectivamente, o nome e o peso do método i.

$$Composição((met_1,w_1),(met_2,w_2),...,(met_n,w_n)) = max(POS,NEU,NEG);$$

$$\sum_{i=1}^{n} w_i = 1;$$

$$POS = \sum_{i=1}^{n} (m_i * w_i); m_i = \begin{cases} 1, & \text{se } met_i \text{ retornou positivo.} \\ 0, & \text{caso contrário.} \end{cases}$$

$$NEU = \sum_{i=1}^{n} (m_i * w_i); m_i = \begin{cases} 1, & \text{se } met_i \text{ retornou neutro.} \\ 0, & \text{caso contrário.} \end{cases}$$

$$NEG = \sum_{i=1}^{n} (m_i * w_i); m_i = \begin{cases} 1, & \text{se } met_i \text{ retornou negativo.} \\ 0, & \text{caso contrário.} \end{cases}$$

O Quadro 3.2 apresenta um exemplo de composição com os métodos de análise de sentimentos AFINN, Emolex e VADER, cujos pesos atribuídos como exemplo são respectivamente 0,2; 0,4; e 0,4. Observa-se que a sentença (a) apresenta um empate entre os sentimentos neutro e negativo, necessitando assim da utilização das regras de desempate definidas na Fórmula 1, o que resulta no sentimento negativo. A sentença (b) apresenta um resultado ponderado positivo e a sentença (c) apresenta um resultado ponderado neutro.

Quadro 3.2 – Exemplo da composição de métodos de análise de sentimentos. Os pesos para os métodos AFINN, Emolex e VADER são 0,2; 0,4; e 0,4, respectivamente.

Sentença	Resultados da análise de sentimento de cada método da composição			Cálculo de pesos dos resultados dos métodos			Resultado
	AFINN	Emolex	VADER	POS	NEU	NEG	
(a) Talvez esse seja o mais ···	positivo	negativo	neutro	0,2	0,4	0,4	negativo
(b) O presidente Xi Jinp Young ···	positivo	positivo	neutro	0,6	0,4	0,0	positivo
(c) A tela desse dispositivo ···	neutro	negativo	neutro	0,0	0,6	0,4	neutro

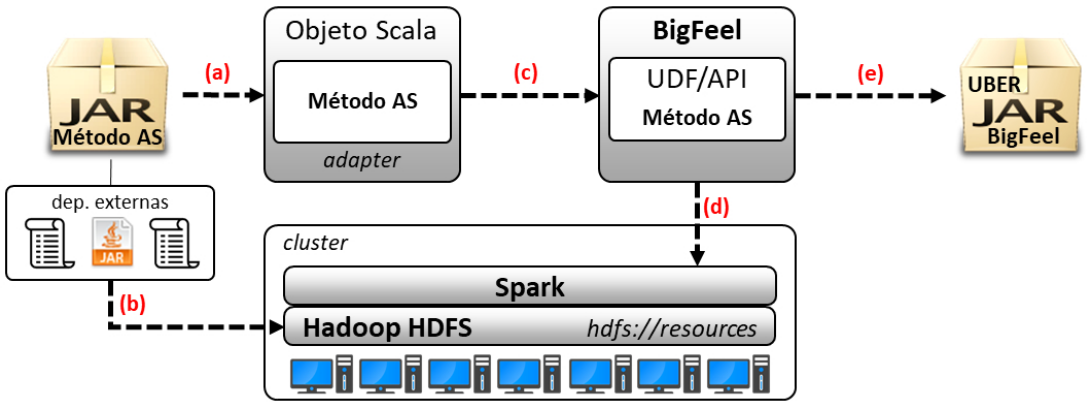
3.5 Detalhes de Implementação do BigFeel

O modo de uso do ambiente BigFeel é dependente de quem o utiliza, seja usuário final ou desenvolvedor. A forma de acesso pode ser optada como acesso a interface web simplificada (por usuários finais), acesso a serviço *web* consumindo uma API (por desenvolvedores), seja em rede local ou na internet. A seguir são detalhadas as formas de integração de novos métodos ao BigFeel e os detalhes de seu funcionamento.

3.5.1 Integração de Novos Métodos ao BigFeel

A Figura 3.2 apresenta um exemplo da integração de métodos ao BigFeel por um desenvolvedor em linguagem Java.

Figura 3.2 – Abordagem para integração de métodos ao BigFeel.



Considere a implementação de um método de análise de sentimentos denominado "método AS", implementado em linguagem Java e oferecido livremente pelo seu autor. Caso o código-fonte do método AS esteja disponível, há a possibilidade de adaptá-lo ao Spark, porém caso o código-fonte não esteja disponível, mas apenas um arquivo JAR e talvez alguma(s) de-

pendência(s), a simples integração de tal arquivo à estrutura do Spark impede que o mesmo funcione em execuções distribuídas pelo *cluster* (FERRARO, 2016).

Seguindo o exemplo da Figura 3.2, a abordagem adotada pelo BigFeel para solucionar esse problema implica na adição do JAR do método AS à pasta *lib* como dependência do projeto BigFeel (como por exemplo, projeto em IDE Eclipse ou Scala IDE) e criação da sua instância em um objeto da linguagem Scala, o qual funciona como um adaptador entre o método original e o BigFeel (Figura 3.2 (a)). As eventuais dependências do método original são todas adicionadas à pasta *resources* do projeto e também encaminhadas para alocação no sistema de arquivos distribuído HDFS (Figura 3.2 (b)). É então criada uma *User Defined Function* (UDF) para fazer uso do objeto Scala do método AS junto à API de *DataFrames* do Spark (Figura 3.2 (c); LASKOWSKI, 2018).

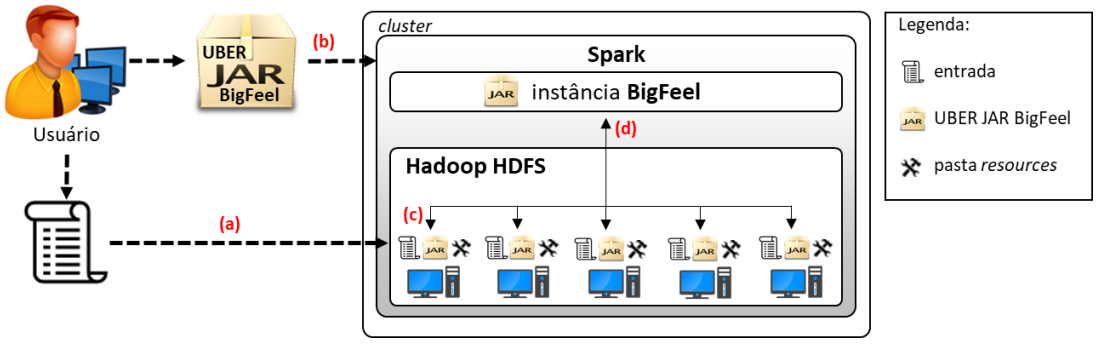
Desse ponto em diante o método AS já pode ser usado diretamente para testes locais não distribuídos junto ao projeto BigFeel (Figura 3.2 (d)). Entretanto, para uso pleno distribuído e para todas as interfaces de acesso aos serviços, é necessário se encapsular em um arquivo JAR o projeto BigFeel completo, junto às suas dependências, transformando-o em apenas um único e grande arquivo denominado UBER JAR ou FAT JAR (Figura 3.2 (e); KARAU; WARREN, 2017; ALMEIDA, 2017). Mais informações e detalhes técnicos serão abordados na seção 3.6 deste trabalho. Detalhes sobre o uso de cada interface também podem ser conferidos no documento “APIs BigFeel”, disponível no repositório Git do projeto.

Ressalta-se que o BigFeel é um ambiente dependente do Spark, afinal de contas é uma aplicação Spark. Fazer o uso dos serviços oferecidos pelo BigFeel implica necessariamente executar um *job* Spark, o qual, independente da interface utilizada, executa uma submissão ao Spark (*spark-submit*) com a localização do nó mestre, o caminho para o UBER JAR do BigFeel, o caminho para o arquivo de entrada, que pode ser local ou distribuído e o método desejado (Figura 3.2 (b)).

3.5.2 Funcionamento do BigFeel

A Figura 3.3 (a) exemplifica a execução Spark com o BigFeel em um ambiente distribuído.

Figura 3.3 – Execução do ambiente BigFeel em *cluster*.



Convenciona-se que a entrada como uma coleção de sentenças (uma por linha), a qual é carregada e alimentada como um *DataFrame* em uma aplicação Spark. Caso ela seja um grande conjunto de dados, inicialmente há a necessidade de se depositá-la no sistema de arquivos distribuído HDFS (Figura 3.3 (b)), caso contrário, o arquivo pode ser passado localmente. A seguir, a submissão de um *job* Spark é realizada passando-se os parâmetros de execução, o que cria uma instância BigFeel na plataforma Spark (Figura 3.3 (b)).

A execução Spark procede com o escalonamento feito pelo nó mestre para se definir quais nós escravos utilizar, priorizando-se a localidade dos dados a se processar (SPARK, 2018b). Após esse passo, cada nó escravo (*executor*) recebe uma cópia do UBER JAR do BigFeel e uma cópia das dependências externas localizadas no HDFS, deixando disponíveis localmente todos os recursos necessários à execução (Figura 3.3 (c)). É iniciada a execução do método selecionado em cada nó para cada fatia de dados alocada a ele, permitindo assim a paralelização na execução do trabalho de processamento. Tal paralelização é realizada pela possibilidade agregada pelo BigFeel de se executar as UDFs dos métodos integrados no *DataFrame* gerado pelo arquivo de entrada. Por fim, cada nó executor responde ao nó mestre o término de seu trabalho, armazenando no mesmo caminho do arquivo de entrada (local ou distribuído) o resultado de seu processamento.

O fim do processamento do *job* submetido ao Spark é então repassado novamente à instância BigFeel, a qual se encarrega de montar a devida resposta dependente de qual interface de acesso iniciou o processo, como por exemplo no caso de *web services*, o resultado é um arquivo JSON de saída e o *log* de processamento.

Como se percebe pela descrição das Figuras 3.2 e 3.3, a abordagem de distribuição adotada pelo ambiente BigFeel se resume à aplicação de técnicas que permitem a distribuição eficaz de métodos com implementação original não distribuída, permitindo que a execução do

método em cada nó seja local, encarregando-se portanto o Spark e o BigFeel da divisão inicial de trabalho e o tratamento da união da solução ao final (Figura 3.3 (d)).

3.6 Problemas e Soluções Adotadas

Nesta seção, são discutidos os problemas encontrados na implementação do ambiente BigFeel usando a arquitetura proposta (Figura 3.1). São também comentadas as soluções adotadas durante o desenvolvimento do projeto.

3.6.1 Implantação e Configuração de Cluster Hadoop/Spark

Várias são as implementações disponíveis previamente construídas para a implantação e/ou experimentação de plataforma distribuída Apache Hadoop/Spark, das quais foram experimentadas algumas opções na seguinte ordem:

1. **Cloudera CDH:** distribuição livre construída e mantida pela empresa Cloudera, a qual inclui a plataforma Hadoop e os principais projetos livres da Apache, como por exemplo Spark, Flume, Hive, Impala e Kafka. Por se tratar de um ambiente previamente construído e, como sua instalação implica a instalação do gerenciador de *cluster* da Cloudera (*Cloudera Manager*), além de vários outros projetos desnecessários ao desenvolvimento do BigFeel, esta não foi a opção selecionada para a construção da versão atual do projeto (CLOUDERA, 2018). Assim, optou-se neste trabalho pela implantação de um ambiente limpo com apenas o Hadoop, o Spark e o Spark *History Server*.

2. **Databricks *Notebooks*:** plataforma de *Software* como Serviço (SaaS) que permite a criação virtual de *clusters* Spark em poucos minutos. Apesar da facilidade de uso, sua versão *community* (gratuita) é voltada para estudantes com o objetivo de se iniciar o aprendizado em Spark, limitando-se portanto à execução de Notebooks (nome dado a cada execução Spark - *job*) de até 6 GB de memória RAM. Por fugir ao escopo da implementação de análise de sentimentos em larga escala, tal plataforma apenas auxiliou no entendimento dos primeiros passos com o Spark (DATABRICKS, 2018).

3. **Virtualização de *cluster* em servidor próprio:** utilizando-se da instalação da plataforma VMware vSphere Hypervisor (ESXi) é possível se criar computadores virtuais em um servidor de forma a se construir um *cluster* Spark. Para experimentação, um *cluster*

foi construído, composto por um nó mestre e oito nós escravos com quatro núcleos de processamento e 8 GB de memória RAM cada. Após a instalação e configuração do ambiente, os primeiros experimentos se mostraram viáveis até o ponto onde se exigiu maior desempenho do *cluster*, momento esse em que o gargalo do *cluster* estava nos discos rígidos do servidor usado, que, mesmo otimizados (interface SAS), não suportaram a escrita simultânea de oito nós comunicando com um mestre que registra *logs* o tempo inteiro sobre o processamento do *cluster* (VMWARE, 2018). Pela identificação desse problema, essa não foi a solução adotada para a experimentação do BigFeel neste trabalho.

4. **Implementação em laboratório próprio:** foi definida como melhor opção para o projeto a construção de um *cluster* Spark formado por nove computadores padrão PC, AMD Athlon X2 com 8 GB RAM e 320 GB de disco. Nos primeiros experimentos várias falhas de execução foram obtidas o que acabou sendo atribuído ao baixo desempenho do equipamento *switch* utilizado para interligação física de rede do *cluster*. A substituição do *switch* por um modelo de melhor desempenho resolveu tais problemas. Mais detalhes e um tutorial passo-a-passo sobre a instalação e as configurações realizadas podem ser observados no documento "*Cluster Installation Tutorial.pdf*", disponível no repositório Git do projeto BigFeel.

3.6.2 Integração de Métodos Originalmente não Serializáveis

É comum se encontrar ferramentas e métodos de análise de sentimentos e processamento de linguagem natural em sua implementação original na forma não serializável (BURDORF, 2016), como é o caso de Stanford CoreNLP. A tentativa de se executar tais métodos de forma distribuída no Spark resulta na exceção identificada pelo Spark como "org.apache.spark.Spark Exception: *Task not serializable*".

Caso o método a se integrar ofereça seu código-fonte de forma aberta, a solução para o problema é se adicionar às classes do projeto a implementação da classe Java *Serializable*, porém quando não se possui o código-fonte, a solução para esse problema é o encapsulamento de qualquer objeto em um bloco Scala do tipo *Object*, o que garante que os objetos instanciados serão serializáveis em meio à aplicação sob execução no *cluster* (FERRARO, 2016).

3.6.3 Gestão e Tratamento de Conflitos de Dependência dos Métodos Integrados

Ao se integrar métodos de terceiros ao ambiente BigFeel, além da inclusão dos arquivos JAR, é frequente a necessidade de recursos externos ao método que está sendo integrado, como por exemplo, dicionários. A passagem dessas dependências para o *cluster* Spark em tempo de execução pode ser realizada utilizando-se os parâmetros "--jars" e "--files" na submissão do *job*. Porém, uma lista muito grande de dependências internas e externas pode tornar a submissão um comando de difícil gestão (SPARK, 2018a). Outro problema frequente está no conflito de dependências dos métodos em versões diferentes.

A solução adotada para gestão das dependências está no uso do *Scala Build Tool* (SBT) como ferramenta de *build* junto ao seu *plugin sbt-assembly*, capaz de construir um UBER JAR, também denominado FAT JAR, ou seja, um JAR único que contém toda a aplicação e suas dependências internas. Assim, o problema de transporte da aplicação como um todo é resolvida com a chamada de apenas um JAR principal via *spark-submit*. O conflito de bibliotecas que possuem dependências iguais em versões diferentes pode ser tratado pelo *plugin sbt-assembly*. Nele é possível se estabelecer regras de mesclagem em caso de conflito, chamadas de *Merge Strategies*. Ações como renomeação, remoção, realocação, priorizar a mais antiga ou a mais nova, são todas opções das estratégias de mesclagem oferecidas pelo *sbt-assembly* (KARAU; WARREN, 2017; ALMEIDA, 2017).

Para o caso das dependências externas dos métodos integrados, a solução adotada está na prévia alocação desses arquivos em um diretório do HDFS, convencionado no projeto BigFeel como: hdfs://resources/. Assim, em tempo de execução do *job* no *cluster*, o BigFeel verifica tal diretório distribuído e o copia para os nós escravos que farão o processamento, assim como também o faz com o próprio UBER JAR do BigFeel, possibilitando que cada nó tenha todos os recursos necessários localmente no momento de seu processamento (SPARK, 2018a).

3.6.4 Adaptação de Métodos para Integração com a API de DataFrames e DataSets do Spark

A convenção adotada como estrutura distribuída de dados padrão para processamento em larga escala do BigFeel é o uso da nova API de *DataFrames*, disponível para aplicações Spark 2+. Apesar dos benefícios da adoção dessa API como estrutura de dados padrão (XIN; ARMBRUST; LIU, 2015), as ações possíveis de serem realizadas em um conjunto de dados são limitadas às ações que a própria API pode oferecer, como por exemplo as capacidades do *Spark*

Structured Query Language (Spark SQL). Logo, a criação de métodos próprios no BigFeel, como é o caso da aplicação das adaptações de métodos integrados para execução no *cluster* usando *DataFrames* e *Datasets*, se torna um problema.

Para solucionar esse caso foram usadas UDFs, adaptando assim os métodos criados de forma independente de implementação, versão ou linguagem. A versatilidade agregada à nova API de *DataFrames* pelas UDFs permite ao usuário desenvolvedor criar suas próprias funções e anexá-las à API, tornando transparente o código produzido, como por exemplo, transformar em maiúsculas toda uma sentença, seria algo como: *DataFrame* teste.maiusculas(). Nesse caso, o *DataFrame* "teste" poderia ser uma coleção contendo milhões de sentenças, sob a qual o UDF maiusculas() executaria para cada uma dessas sentenças (LASKOWSKI, 2018).

3.7 Alternativas à Solução Proposta

Em tempo de projeto do ambiente BigFeel, decidiu-se pelo modelo de um ambiente integrável onde variados métodos, seja de pré-processamento textual, PLN ou análise de sentimentos, pudessem ser adaptados e integrados sem nenhum tipo de alteração no código-fonte original, necessitando-se no máximo de um adaptador para isso. Essa decisão, apesar de se mostrar factível, acarreta uma perda no uso de recursos do Spark como estruturas de dados otimizadas para o processamento distribuído e também a dependência de recursos locais como dicionários e índices.

Uma proposta de ambiente similar ao BigFeel com implementação própria dos métodos integrados em Spark traria resultados diferentes e, provavelmente melhores, para os experimentos realizados neste trabalho. Prova disso está no comportamento dos métodos Spark Naive Bayes e Spark Random Forest, implementados junto a este trabalho para comparação com os demais resultados. A implementação de métodos nativos em Spark ainda acarreta uma sobrecarga menor de dados de dependência transitando pelo *cluster* a cada execução, afinal os métodos integrados necessitam de suas dependências localmente, o que implica na cópia de aproximadamente 700 Megabytes do HDFS para um diretório local em cada tarefa a ser executada no BigFeel.

Como outra alternativa à solução proposta, em um nível mais técnico, as soluções adotadas para adaptação de todos os métodos integrados ao BigFeel se basearam no uso de UDFs, assim como ocorre em outras adaptações como é o caso de Meng (2016). Porém, existe uma alternativa às UDFs, as *Custom Unary Transformers*, as quais permitem o mesmo tipo de en-

trada e saída atualmente utilizada - os *DataFrames* (LASKOWSKI, 2018). Por meio de uma pesquisa pela literatura recente não foi encontrada uma comparação entre essas duas técnicas que responda se existe diferença de desempenho entre elas.

Outra alternativa plausível seria a não dependência do sistema de arquivos distribuído HDFS e, consequentemente, da plataforma Hadoop como um todo. Atualmente o BigFeel utiliza do Hadoop apenas o sistema de arquivos distribuído HDFS e suas dependências, tornando todos os seus demais componentes subutilizados. Existem alternativas ao HDFS, como: Cleversafe, Amazon S3 e GPFS (HARRIS, 2012; GUPTA, 2017). Por razão da integração nativamente simplificada com o Spark, foi adotado no BigFeel o uso do HDFS, sendo que outros sistemas de arquivos distribuídos poderiam oferecer resultados diferentes aos aqui expostos.

A interface *web* disponibilizada pelo BigFeel utiliza o *framework* PHP Laravel, em sua versão 5.5 (OTWELL, 2018), porém poderiam ser utilizados outros *frameworks* de desenvolvimento *web* para tal.

4 AVALIAÇÃO EXPERIMENTAL

Nesta seção, são apresentados os passos e os resultados da condução dos experimentos realizados de forma a se obter as respostas às questões de pesquisa deste trabalho: (i) os métodos de análise de sentimentos existentes são eficientes para processamento de grandes volumes de dados? (ii) a adaptação de métodos em um ambiente distribuído implica eficiência significativa? A análise da eficiência de métodos distribuídos foi realizada por meio das métricas distribuídas de *SpeedUp*, *SizeUp* e *ScaleUp* (HAI; ZHANG; ZHANG, 2017). Foram também conduzidos experimentos para se avaliar a eficácia dos métodos supervisionados implementados nativamente em Spark, utilizando-se para tanto dos classificadores da MLLib Naive Bayes e Random Forest, comparando-os com os métodos externos integrados ao BigFeel.

4.1 Fontes de Dados

Para a avaliação da eficiência do ambiente proposto, foram utilizados três grandes *datasets* de revisões coletadas da loja eletrônica Amazon durante anos (MCAULEY; HE, 2016). Essas revisões foram categorizadas e disponibilizadas na *web*, sendo utilizados neste trabalho os três maiores *datasets*: *Books*, *Electronics* e *Movies_TV*. A Tabela 4.1 resume as informações sobre os *datasets* usados nos experimentos do BigFeel.

Tabela 4.1 – *Datasets* usados para avaliação das métricas de eficiência e eficácia.

Dataset	Número revisões	Número classes	Número itens	Tamanho	Revisões negativas	Revisões neutras	Revisões positivas	Tamanho médio (carac./revisão)
IMDB	67.426	3	1.635	107 MB	9.285	14.319	43.822	1.956,28
Books	22.507.155	3	2.370.585	13,3 GB	-	-	-	637,11
Electronics	7.824.482	3	498.196	3,37 GB	-	-	-	461,81
Movies_TV	4.607.047	3	2.208.321	2,63 GB	-	-	-	612,46

Ressalta-se que nesta seção não está incluído o *dataset* utilizado para o desenvolvimento do estudo de caso, o qual será abordado em detalhes no Capítulo 6.

O *dataset* de revisões de usuários sobre cinema e televisão coletadas do site IMDB por Tang, Qin e Liu (2015) e disponibilizado por Chen et al. (2016), foi utilizado na avaliação da eficácia dos métodos supervisionados integrados ao ambiente BigFeel. O *dataset* oferece um total de 67.426 revisões de 1.310 usuários distintos expressando sua opinião sobre 1.635 produções cinematográficas.

Esse *dataset* do IMDB possui rótulos atribuídos manualmente entre dez classes no intervalo entre 1 e 10. Para padronização e possível comparação com todos os métodos integrados ao BigFeel, o *dataset* foi pré-processado atribuindo-se negativo para as revisões classificadas entre 1 e 4, neutro para as revisões entre 5 e 6 e positivo para o intervalo entre 7 e 10. Essa divisão de classes segue o padrão de polaridades já adotado pela literatura: (1,2) muito negativo, (3,4) negativo, (5,6) neutro, (7,8) positivo e (9,10) muito positivo (RANGANATHAN; IRUDAYARAJ; TZACHEVA, 2017).

4.2 Métricas de Avaliação

As métricas utilizadas para avaliação da eficiência distribuída dos resultados experimentais são *SpeedUp*, *SizeUp* e *ScaleUp* (HAI; ZHANG; ZHANG, 2017):

$$SpeedUp = \frac{T_1}{T_n} \tag{4.1}$$

$$SizeUp(dados, m) = \frac{T_m}{T_1} \tag{4.2}$$

$$ScaleUp(dados, m) = \frac{T_1}{T_{mm}} \tag{4.3}$$

onde

- *SpeedUp*: métrica para avaliação do tempo de execução de uma aplicação distribuída à medida que se varia o número de nós de processamento, fixando-se o *dataset*. T_1 é o tempo de execução em um nó e T_n o tempo de execução em n nós. Teoricamente, o valor esperado de *SpeedUp* é: aumentando-se os nós de processamento, diminui-se na mesma proporção o tempo de processamento, o que caracteriza um *SpeedUp* linear.

- *SizeUp*: métrica para avaliação do tempo de execução de uma aplicação distribuída à medida que se varia o tamanho do *dataset* de entrada, fixando-se o número de nós de processamento. T_m é o tempo de execução de uma constante m multiplicada pelo tamanho dos dados (m * *dados*), e T_1 é o tempo de execução de dados. Teoricamente, o valor esperado de *SizeUp* é: aumentando-se os dados a processar, aumenta-se na mesma proporção o tempo de processamento, caracterizando-se um *SizeUp* linear.

- *ScaleUp*: métrica para avaliação do tempo de execução de uma aplicação distribuída à medida que são variados o número de nós de processamento e, proporcionalmente, o

tamanho do *dataset* de entrada. T_1 é o tempo de execução de dados em um nó e T_{mm} é o tempo de execução de *m* * *dados* em *m* nós. Teoricamente, o valor esperado de *ScaleUp* é: aumentando-se na mesma proporção os nós de processamento e os dados a se processar, nenhuma alteração deve ocorrer no tempo de processamento, caracterizando-se um *ScaleUp* constante.

Se o *SpeedUp* aumenta linearmente com o respectivo aumento de nós de processamento, isso significa que múltiplos nós efetivamente podem reduzir o tempo de processamento do algoritmo. Contudo, obter um *SpeedUp* linear com o aumento do número de nós de processamento é difícil, afinal o acréscimo de nós introduzem sobrecarga adicional em comunicação no *cluster*. No caso da métrica *ScaleUp*, como o aumento do número de nós de processamento também implica sobrecarga de comunicação no *cluster*, isso reduz a taxa de utilização de cada nó. Assim, para determinados casos, a efetividade de uso de todos os nós do *cluster* por um algoritmo distribuído pode se tornar limitada (HAI; ZHANG; ZHANG, 2017).

Juntas, as três métricas de desempenho demonstram o ganho obtido pela execução distribuída em comparação à execução em um único nó, bem como sua variação em função do tamanho da entrada de processamento.

Para avaliar a eficácia dos métodos supervisionados implementados em Spark, foram aplicadas as métricas comumente adotadas pela literatura: a acurácia (A) e a média aritmética das três classes (negativo, neutro, positivo) da precisão (P), da revocação (R) e da $F1$ (também denominada macro-$F1$).

$$A = \frac{TP}{TP+FP} \tag{4.4}$$

$$P(c) = \frac{TP(c)}{TP(c)+FP(c)} \tag{4.5}$$

$$R(c) = \frac{TP(c)}{TP(c)+FN(c)} \tag{4.6}$$

$$F1(c) = \frac{2*R(c)*P(c)}{R(c)+P(c)} \tag{4.7}$$

onde

- *TP* (*true positives*): número de instâncias corretamente preditas como positivas.
- *FP* (*false positives*): número de instâncias incorretamente preditas como positivas.

- *FN* (*false negatives*): número de instâncias positivas, mas incorretamente preditas como negativas.

A acurácia (EQUAÇÃO 4.4) é a razão entre o número de instâncias precisamente preditas (TP) e o número total de instâncias analisadas ($TP + FP$), medindo portanto o quão frequente um classificador está correto. A precisão (EQUAÇÃO 4.5) mede a exatidão de um classificador, o que significa que uma precisão alta aponta menos falsos positivos (*FP*), enquanto que uma precisão baixa aponta mais falsos positivos (*FP*). Por outro lado, a revocação (EQUAÇÃO 4.6) mede a completude ou sensitividade de um classificador, sendo que um valor alto significa menos falsos negativos (*FN*), e valores baixos de revocação significam mais falsos negativos. A *F-measure* (EQUAÇÃO 4.7) é uma média harmônica entre a precisão e a revocação, onde 1 é o valor ideal e 0 é o valor mínimo (WOLDEMARIAM, 2016).

4.3 Hardware e Software Utilizados

Esta seção apresenta os materiais de *hardware* e *software* envolvidos na execução do presente trabalho. Para implementação da maior parte deste trabalho foi utilizado um *notebook* pessoal de processador Intel Core i7 @2.4 GHz, 4 MB de memória *cache*, 8 GB de memória RAM, HD de 1TB e SSD de 480 GB, usado para o estudo e implementação em primeira instância do ambiente proposto. Foi também utilizado um computador padrão PC de processador Intel Core i5 @2.9 GHz, 6 MB de memória *cache* e 4 GB de memória RAM, usado para testes e simulações em segunda instância, disponível no LabRI do DCC/UFLA.

Para a experimentação e análise experimental do ambiente proposto, foi utilizado um *cluster* disponível no DataCenter do Instituto Federal de Minas Gerais (IFMG) - Campus Formiga, composto por nove computadores de configuração igual, os quais constituem um *cluster* com um nó mestre e oito nós escravos. Suas características são: computador padrão PC, processador AMD Athlon II X2 240 de dois núcleos @ 2.8GHz, placa-mãe Asus M3N78-VM, disco rígido Samsung 320 GB SATA-II, memória de 8 GB DDR2 @ 800MHz - CL5, interface de rede *on-board* comunicando a 1 Gbps. Java versão 1.8.0 (151), sistema operacional Linux Ubuntu 16.04.3 LTS, Hadoop 2.8.0 e Spark 2.1.2. Como equipamento centralizador da comunicação em rede no *cluster* foi utilizado um Switch HP A5120 de 48 portas, modelo JE069A.

Com relação aos itens de *software*, as seguintes linguagens, ferramentas e tecnologias livres foram utilizadas:

- Java (JDK 8 (Oracle) 1.8.0_102). Para desenvolvimento do ambiente proposto majoritariamente foi utilizada a linguagem Scala, a qual oferece recursos de alto nível e estruturas de dados escaláveis para a linguagem Java, sendo o processamento todo executado sob a *Java Virtual Machine* (JVM).

- Scala 2.12.1. Linguagem de programação para desenvolvimento em Spark para uso na integração dos módulos e implementações externas ao trabalho. Recursos dessa linguagem foram utilizados como um adaptador entre tecnologias de terceiros e o Spark, o qual foi construído nativamente em Scala.

- PHP 5.6 - 7.0. Para uso na construção do protótipo da interface *web* do BigFeel em *framework* Laravel.

- Spark. Para processamento em memória, análise e apresentação de resultados de forma distribuída no *cluster*.

- Especificamente sobre os componentes da plataforma Hadoop, foram utilizados apenas o *Core* do Hadoop e seu sistema de arquivos distribuído, o HDFS. O gerenciamento do *cluster* ficou sob a responsabilidade do Gerenciador autônomo do Spark.

- *Eclipse NEON IDE*. Ambiente integrado de desenvolvimento com ferramentas para implementação em linguagem Scala.

4.4 Condução dos Experimentos

Para avaliação dos resultados experimentais da eficiência dos métodos integrados, foram utilizados os *datasets Books*, *Electronics* e *Movies_TV*, os quais variam em tamanho, possibilitando assim a comparação de eficiência em função do tamanho do *dataset* de entrada. Foram realizadas cinco execuções de cada método para cada *dataset*, para cada uma das métricas, sendo o resultado apresentado aqui a média aritméticas das cinco execuções. Agregado a esses experimentos de eficiência distribuída de *SpeedUp*, foram ainda realizadas as mesmas cinco execuções em implementação original não distribuída de cada método.

Para avaliação experimental da eficácia dos métodos implementados em Spark, foram selecionadas 10.000 revisões do *dataset* IMDB de forma a se obter os melhores parâmetros (*tuning*), ficando as demais 57.426 para as fases de treinamento e testes por validação cruzada com dez partições (*folds*) de todos os métodos integrados ao BigFeel. Ressalta-se que o *dataset*

não possui nenhuma ordenação ou tendência dos dados, sendo portanto considerada aleatória a seleção das 10.000 revisões. O treinamento ocorreu apenas nos métodos supervisionados implementados em Spark, nos demais, apenas a fase de testes foi aplicada.

As 10.000 revisões selecionadas do *dataset* IMDB foram divididas aleatoriamente em dez partes, usando-se nove partes para treinamento e uma para teste, em dez execuções alternando-se as partições de treinamento e teste. Isso foi realizado para todas as combinações de parâmetros de um *grid* de opções. De posse dos parâmetros com melhores resultados, os modelos treinados foram integrados ao BigFeel a título de testes. Ressalta-se aqui o domínio de treinamento desses dois métodos: revisões de cinema e televisão.

Para se comparar os dois classificadores implementados da MLlib/Spark com os demais métodos, a eficácia de todos os métodos integrados foi calculada sob o *dataset* IMDB, o qual foi dividido aleatoriamente em dez partes, sendo portanto executado cada método em cada uma das partes, as mesmas usadas no testes dos classificadores supervisionados. Os resultados apresentados neste documento representam a média das dez execuções para cada método.

Tanto nos experimentos de eficiência quanto de eficácia, foi realizado o teste estatístico de intervalo de confiança, permitindo assim a avaliação e comparação entre os métodos com uma confiança de 95%.

Ressalta-se que todos os métodos integrados ao BigFeel não tiveram nenhuma alteração em seu código-fonte, afinal foi priorizada a integração das implementações de terceiros ao BigFeel via arquivos JAR, utilizando-se somente adaptadores para acesso aos mesmos. Isso implica na observação de que a eficácia dos métodos integrados permanece inalterada, restando aos experimentos presentes neste trabalho a avaliação da eficiência de todos os métodos e também a comparação entre a eficácia dos métodos integrados com os métodos implementados em Spark.

O ambiente sob o qual foram executados os experimentos apresentados nesta seção é composto por nove computadores de configuração igual, os quais constituem um *cluster* com um nó mestre e oito nós escravos.

Salienta-se que o *cluster* Spark utilizado para todos os experimentos realizados neste trabalho foi construído isolando-se o computador mestre daqueles que realmente fazem o processamento, os computadores escravos (*workers*). Isso foi definido para se garantir que as atividades alheias à execução, atribuídas ao nó mestre, não interferissem na avaliação do tempo de processamento. São algumas das atividades atribuídas ao mestre: gerenciamento do *cluster*,

armazenamento de *logs*, verificação e tratamento de falhas do *cluster*, armazenamento das configurações e histórico de cada execução submetida (*Spark History Server*). Há ainda o fato de que, no caso do *cluster* implementado para este trabalho, o mestre também assume as responsabilidades de mestre do *cluster* Hadoop (LASKOWSKI, 2018; MORITZ et al., 2016; SPARKS et al., 2015).

Um exemplo de comparação injusta seria processar um *dataset* de 40 GB em uma máquina local com 8 GB de memória RAM executando apenas o Sistema Operacional e a *Java Virtual Machine* (JVM) e comparar seu tempo de processamento com uma execução semelhante em um *cluster* Hadoop/Spark com apenas um nó que acumula as funções de mestre e escravo, afinal de contas ele não faria uso de todos os 8 GB de memória RAM, pois parte desse montante estaria comprometido com as atividades de gerenciamento do *cluster*. Borsos (2017) e Hu (2017) apresentam exemplos de construção do *cluster* Spark separando o nó mestre dos *n* nós escravos, o que de fato otimiza a execução distribuída e contribui com o objetivo da criação do ambiente BigFeel.

5 RESULTADOS E DISCUSSÕES

Esta seção apresenta as médias aritméticas de todos os experimentos executados, sendo cinco execuções sob cada um dos vinte e dois métodos para cada uma das três métricas para análise da eficiência, em cada *dataset*, resultando aproximadamente 4.290 execuções, e dez execuções sob cada método para a avaliação da eficácia, resultando aproximadamente 220 execuções.

Para todas as tabelas apresentadas nesta seção, um caractere * indica que o experimento executou por um período superior a 24 horas (o que representa 86.400 segundos na escala utilizada) e, devido ao longo tempo, sua execução foi interrompida. O caractere – representa situações onde a execução não foi bem sucedida, seja por erros do tipo de estouro de pilha de execução em memória ou na execução de alguma dependência, como é o caso do excesso de uso do *Garbage Collector*. Além disso, a sigla ND representa os experimentos onde não está disponível um valor, afinal não havia a possibilidade de execução, como é o caso de uma implementação original não distribuída do Spark Naive Bayes, uma vez que esse método foi construído nativamente no ambiente Spark e, portanto, é naturalmente distribuído.

Em todas as tabelas desta seção convencionou-se o destaque em negrito do cálculo das métricas de *SpeedUp*, *SizeUp* e *ScaleUp* que obtiveram melhor valor a cada modificação das configurações do experimento, respectivamente número de nós de processamento para *SpeedUp*, tamanho do *dataset* para *SizeUp* e ambos para o caso de *ScaleUp*. Exceção a essa convenção está nos casos com apenas um nó ou 1/8 do *dataset* usado, afinal o cálculo da métrica equivale sempre a 1,0.

Para todos os valores médios apresentados nas tabelas desta seção, é fornecido à direita o intervalo de confiança de 95% entre os valores medidos.

Para a execução dos experimentos de *SizeUp* os *datasets Books*, *Electronics* e *Movies_TV* foram aleatoriamente particionados em oito partes iguais, unindo-se assim essas partes para a obtenção dos *datasets* de tamanhos: 1/8, 2/8, 4/8 e 8/8. Assim, mantendo-se o *cluster* fixo em oito nós, o tamanho do *dataset* foi sendo incrementado a fim de se testar o comportamento dos métodos. Os experimentos de *ScaleUp* para os *datasets* foram conduzidos incrementando-se o número de nós do *cluster* para processamento e, proporcionalmente, incrementando-se também o *dataset* de entrada, sendo as mesmas partições utilizadas nos experimentos de *SizeUp*.

Para todos os gráficos de linhas apresentados nesta seção, convencionou-se o uso de uma linha tracejada (**- - -**) para representar o valor teórico esperado para os resultados de *SpeedUp*, *SizeUp* e *ScaleUp*.

Os resultados são apresentados e discutidos nas seções seguintes desta seção.

5.1 Eficiência

Por convenção, os gráficos apresentados nesta seção representam os resultados da eficiência média medida em tempo (segundos). Tais gráficos apresentam apenas dados de execuções comparáveis, ou seja, no caso de falha ou extrapolação do limite superior estipulado de 24 horas, a representação da execução foi removida do gráfico, podendo ser verificado em detalhes nas tabelas em anexo aos gráficos.

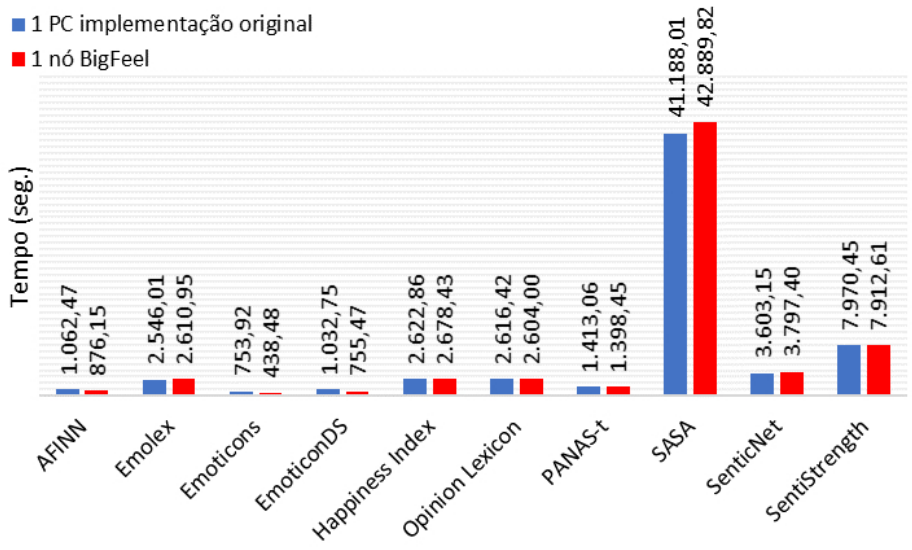
São também abordados nesta seção os resultados das métricas distribuídas de *SpeedUp*, *SizeUp* e *ScaleUp* para os *datasets* definidos como alvo de experimento, além dos resultados que permitem a comparação entre a execução original dos métodos integrados ao BigFeel com sua execução em Spark utilizando apenas um nó de processamento do *cluster*. Salienta-se a forma correta de verificação dos gráficos de resultados das métricas calculadas, sendo que para *SpeedUp* e *ScaleUp*, valores acima da linha tracejada de valor teórico esperado são melhores e, no caso do experimento de *SizeUp*, valores abaixo da linha tracejada de valor esperado são melhores.

Esta seção está subdividida em três subseções, respectivamente uma para cada *dataset* utilizado: *Books* ($\sim$22,5 milhões de revisões), *Electronics* ($\sim$8 milhões de revisões) e *Movies_TV* ($\sim$4,5 milhões de revisões).

5.1.1 Dataset Books

O Gráfico 5.1 apresenta a comparação entre o tempo médio de execução dos experimentos no *dataset Books* em implementação original não distribuída e o tempo de execução pelo BigFeel em apenas um nó de processamento.

Gráfico 5.1 – Tempos de execução em um PC na implementação original e em um nó no *cluster* para o *dataset Books*.



Os resultados apontam uma pequena diferença no tempo de execução dos métodos, tanto em implementação original, quanto em um único nó de processamento BigFeel (Gráfico 5.1). Esse resultado demonstra que o custo computacional de tempo não é consideravelmente maior para se executar o método de análise de sentimentos pelo BigFeel, ainda que existindo a sobrecarga de um nó mestre gerenciando a execução de um nó escravo. De fato, algumas execuções no BigFeel se mostraram um pouco melhores que a própria implementação original, conforme pode ser observado na Tabela 5.1, onde as execuções com melhor tempo (analisadas pelo intervalo de confiança) foram destacadas em negrito.

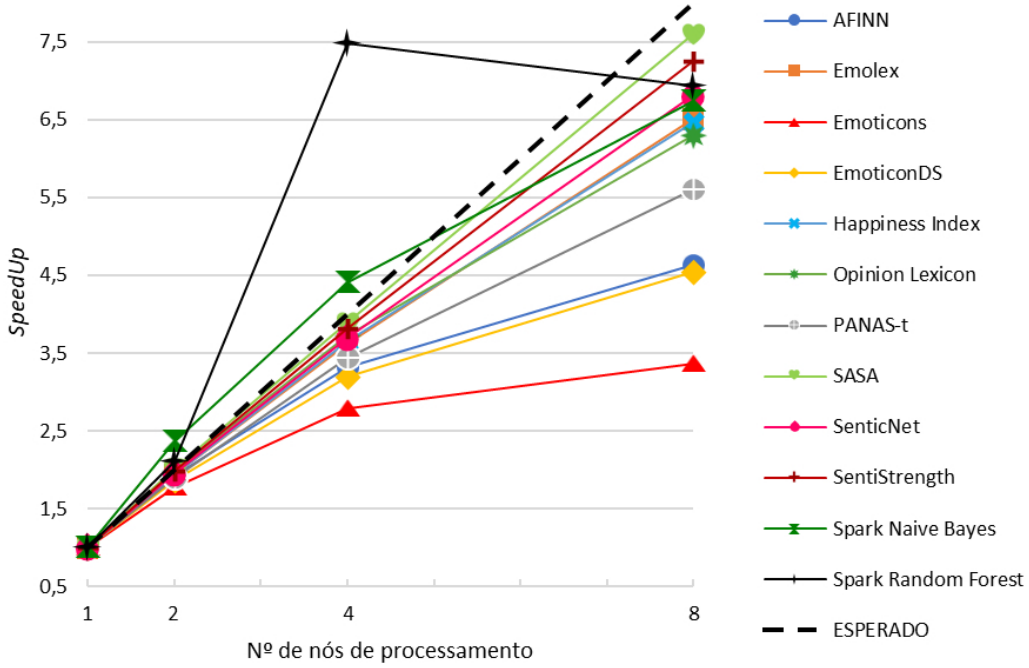
Tabela 5.1 – Tempo médio de execução e o respectivo intervalo de confiança em segundos na implementação original e em um nó no BigFeel - *dataset Books*.

Método	1 PC (impl. original)	1 nó BigFeel
AFINN	1.062,47 ± 3,40	**876,15 ± 0,88**
Emolex	**2.546,01 ± 27,38**	2.610,95 ± 1,27
Emoticons	753,92 ± 5,18	**438,48 ± 0,62**
EmoticonDS	1.032,75 ± 2,15	**755,46 ± 0,51**
Happiness Index	**2.622,86 ± 2,73**	2.678,43 ± 0,84
MPQA	*	*
NRC Hashtag	*	*
Opinion Lexicon	2.616,42 ± 2,59	**2.604,00 ± 1,45**
PANAS-t	1.413,06 ± 2,78	**1.398,45 ± 3,43**
SANN	475,01 ± 1,41	–
SASA	**41.188,01 ± 52,32**	42.889,82 ± 60,28
SenticNet	**3.603,14 ± 33,62**	3.797,39 ± 49,59
Sentiment140 Lexicon	*	*
SentiStrength	7.970,44 ± 15,95	**7.912,61 ± 15,37**
SentiWordNet	*	*
SO-CAL	*	*
Stanford Recursive Deep Model	*	*
Umigon	–	–
VADER	*	*
Spark Naive Bayes	ND	4.675,18 ± 39,23
Spark Random Forest	ND	9.399,59 ± 33,94
Databricks Stanford	ND	*

Exemplos onde o tempo de execução médio no BigFeel se mostrou um pouco melhor podem ser observados na Tabela 5.1, nos métodos AFINN, Emoticons, EmoticonDS, Opinion *Lexicon*, PANAS-t e SentiStrength. Nos demais casos ocorreu o comportamento esperado, de que a implementação original se mostrasse mais eficiente, afinal não tem a sobrecarga do gerenciamento de um *cluster*.

Os demais experimentos realizados no *dataset Books* seguiram as métricas de *SpeedUp*, *SizeUp* e *ScaleUp*, definidas na seção anterior.

Gráfico 5.2 – *SpeedUp* da execução dos métodos integrados para o *dataset Books*.



O Gráfico 5.2 demonstra uma proximidade de resultados ao valor esperado da execução de métodos como SASA e SentiStrength. Os resultados observados de *SpeedUp* para o *dataset Books* se resumem a (Gráfico 5.2; Tabela 5.2):

- 2 nós - melhor: Spark Naive Bayes; pior: Emoticons.
- 4 nós - melhor: Spark Random Forest; pior: Emoticons.
- 8 nós - melhor: SASA; pior: Emoticons.

Os seguintes métodos não aparecem no Gráfico 5.2 pois levaram mais de 24 horas ou apresentaram falhas para serem executados: MPQA, NRC *Hashtag*, Sentiment140 *Lexicon*, SANN, SentiWordNet, SO-CAL, Stanford *Recursive Deep Model*, Umigon, VADER e Databricks Stanford CoreNLP.

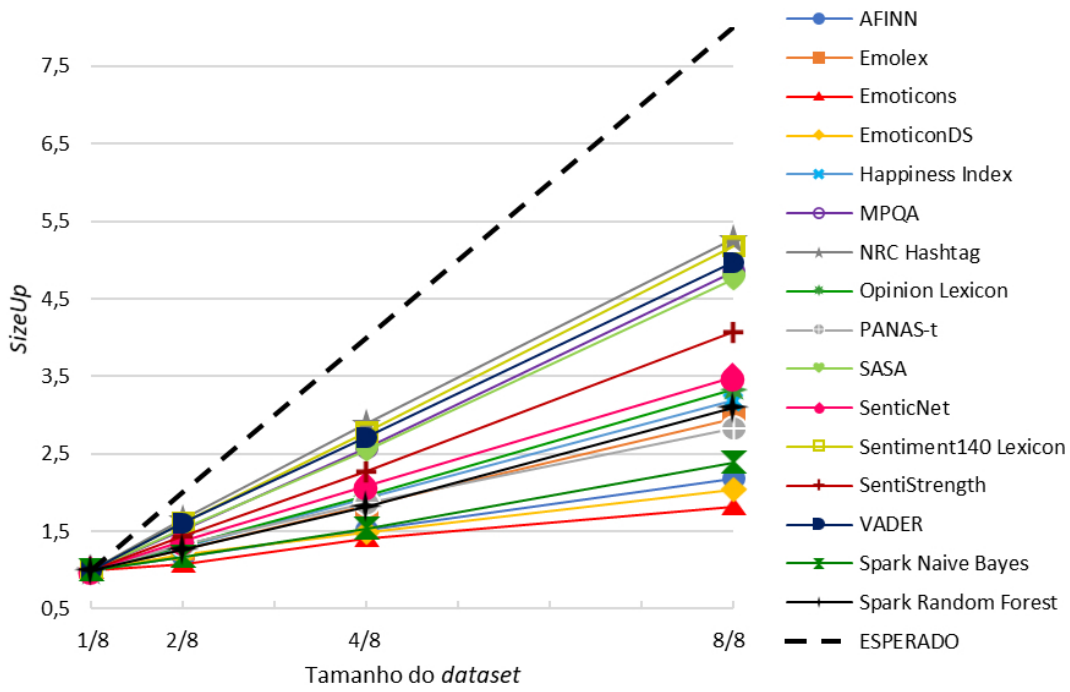
Todos os métodos analisados no Gráfico 5.2 indicaram crescente *SpeedUp* até oito nós, com exceção do método Spark Random Forest, o qual apresentou uma queda entre quatro e oito nós. Isso aponta um maior custo computacional na distribuição do método/*dataset* do que seu próprio cômputo para o *dataset Books*.

A Tabela 5.2 apresenta o resultado em tempo médio da execução de todos os experimentos de *SpeedUp* no *dataset Books*.

Tabela 5.2 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *SpeedUp - dataset Books*.

Método	1 nó BigFeel	2 nós BigFeel	4 nós BigFeel	8 nós BigFeel
AFINN	876,15 ± 0,88	454,06 ± 0,99	264,06 ± 0,98	188,73 ± 1,66
Emolex	2.610,95 ± 1,27	1.342,33 ± 15,40	720,29 ± 1,44	401,31 ± 0,50
Emoticons	438,48 ± 0,62	247,84 ± 1,04	157,40 ± 0,34	130,39 ± 1,19
EmoticonDS	755,46 ± 0,51	403,74 ± 1,34	236,58 ± 0,53	165,99 ± 1,20
Happiness Index	2.678,43 ± 0,84	1.375,51 ± 3,56	734,98 ± 0,77	413,78 ± 1,83
MPQA	*	*	60.326,22 ± 44,58	34.185,56 ± 50,23
NRC Hashtag	*	*	*	63.289,90 ± 27,20
Opinion Lexicon	2.604,00 ± 1,45	1.306,91 ± 1,70	699,77 ± 1,62	413,15 ± 2,02
PANAS-t	1.398,45 ± 3,43	737,77 ± 2,40	406,43 ± 2,05	249,90 ± 0,43
SANN	–	–	–	–
SASA	42.889,82 ± 60,28	21.201,11 ± 2,27	11.019,07 ± 2,05	**5.640,31 ± 10,09**
SenticNet	3.797,39 ± 49,59	1.943,74 ± 4,66	1.025,51 ± 2,42	557,62 ± 2,73
Sentiment140 Lexicon	*	*	*	63.159,32 ± 66,44
SentiStrength	7.912,61 ± 15,37	3.993,28 ± 7,08	2.070,55 ± 1,67	1.091,10 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	*	53.464,76 ± 128,82	19.255,23 ± 117,91	13.762,51 ± 12,11
Spark Naive Bayes	4.675,18 ± 39,23	**1.971,88 ± 7,18**	1.059,78 ± 13,90	693,16 ± 4,52
Spark Random Forest	9.399,59 ± 33,94	4.446,13 ± 23,92	**1.256,79 ± 5,28**	1.356,20 ± 5,89
Databricks Stanford	*	*	*	*

Gráfico 5.3 – *SizeUp* da execução dos métodos integrados ao BigFeel para o *dataset Books*.



O Gráfico 5.3 apresenta os resultados de *SizeUp* dos métodos que executaram sem problemas. Os resultados observados para o *dataset Books* se resumem a (Gráfico 5.3; Tabela 5.3):

- 2/8 *Books* - melhor: Emoticons; pior: NRC *Hashtag*.
- 4/8 *Books* - melhor: Emoticons; pior: NRC *Hashtag*.
- 8/8 *Books* - melhor: Emoticons; pior: NRC *Hashtag*.

Observa-se pelo Gráfico 5.3 que todos os métodos tiveram um *SizeUp* melhor do que o esperado. O resultado bom do método Emoticons está relacionado ao fato de que existem poucas ocorrências de caracteres indicando *emoticons* dentre as revisões analisadas, o que torna seu processamento mais rápido e pouco dependente do tamanho da entrada.

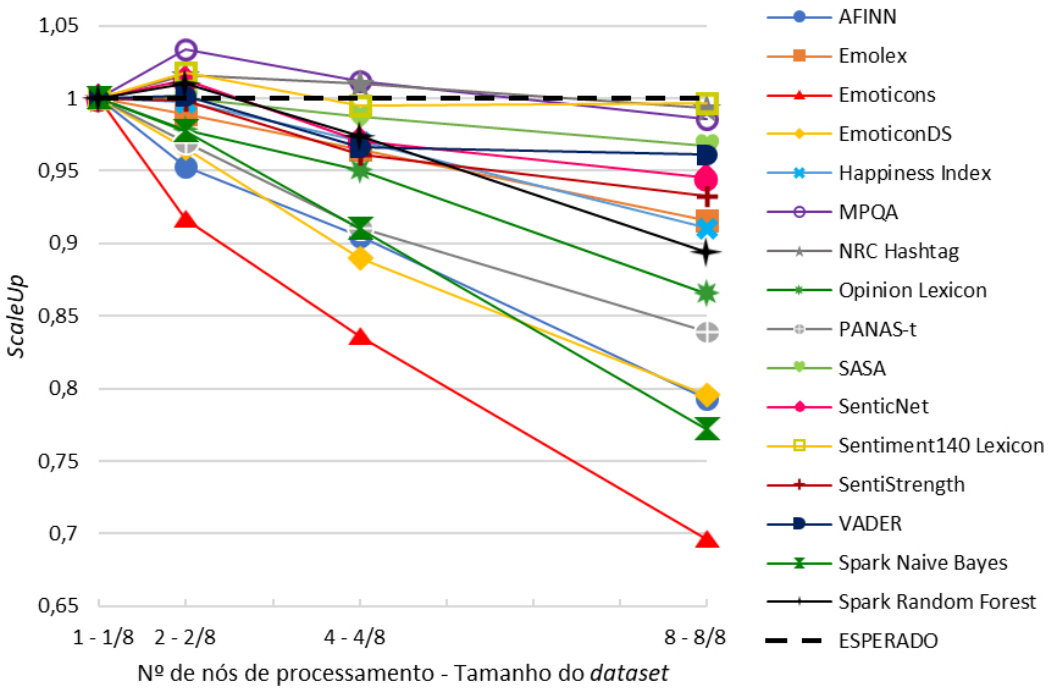
A Tabela 5.3 detalha os tempos médios de execução usados para o cálculo do *SizeUp* dos experimentos com o *dataset Books*:

Tabela 5.3 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *SizeUp* - *dataset Books*.

Método	1/8 *Books*	2/8 *Books*	4/8 *Books*	8/8 *Books*
AFINN	86,68 ± 1,38	100,84 ± 0,59	131,78 ± 0,23	188,73 ± 1,66
Emolex	136,00 ± 1,48	172,81 ± 0,56	249,25 ± 0,43	401,31 ± 0,50
Emoticons	71,64 ± 1,17	**77,39 ± 0,41**	**101,09 ± 0,54**	**130,39 ± 1,19**
EmoticonDS	81,79 ± 0,78	98,44 ± 0,22	121,00 ± 0,54	166,00 ± 1,20
Happiness Index	129,77 ± 0,70	169,03 ± 0,21	250,33 ± 1,14	413,78 ± 1,83
MPQA	7.057,70 ± 19,56	10.719,50 ± 10,89	18.145,78 ± 81,28	34.185,56 ± 64,00
NRC Hashtag	12.010,24 ± 83,38	19.909,76 ± 116,30	34.678,47 ± 134,55	63.289,90 ± 97,87
Opinion Lexicon	124,07 ± 0,62	162,39 ± 0,34	242,61 ± 2,80	413,15 ± 2,02
PANAS-t	88,33 ± 0,90	116,75 ± 0,27	164,78 ± 3,22	249,90 ± 0,43
SANN	69,65 ± 0,70	–	–	–
SASA	1.187,90 ± 7,08	1.819,19 ± 5,01	3.012,30 ± 3,59	5.640,31 ± 10,09
SenticNet	160,13 ± 0,32	220,55 ± 0,29	333,025 ± 11,41	557,62 ± 2,73
Sentiment140 Lexicon	12.178,83 ± 10,63	19.698,61 ± 89,32	33.888,80 ± 90,90	63.159,32 ± 69,78
SentiStrength	268,06 ± 1,64	385,91 ± 2,30	609,20 ± 4,29	1.091,10 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	2.768,62 ± 59,26	4.461,59 ± 51,27	7.534,62 ± 26,67	13.762,51 ± 12,11
Spark Naive Bayes	290,91 ± 1,73	337,94 ± 4,20	445,23 ± 5,01	693,16 ± 2,93
Spark Random Forest	437,51 ± 3,12	553,35 ± 4,97	797,76 ± 5,34	1.356,20 ± 5,02
Databricks Stanford	*	*	*	*

O Gráfico 5.4 apresenta os resultados obtidos de *ScaleUp* para o *dataset Books*.

Gráfico 5.4 – *ScaleUp* da execução dos métodos integrados ao BigFeel para o *dataset Books*.



Os resultados observados de *ScaleUp* para o *dataset Books* se resumem a (Gráfico 5.4; Tabela 5.4):

- 2 nós, 2/8 *Books* - melhor: MPQA; pior: Emoticons.
- 4 nós, 4/8 *Books* - melhor: MPQA; pior: Emoticons.
- 8 nós, 8/8 *Books* - melhor: Sentiment140 *Lexicon*; pior: Emoticons.

Entre quatro e oito nós de processamento é possível observar pelo Gráfico 5.4 que todos os métodos apresentaram perda no *ScaleUp*. Na execução com oito nós em todo o *dataset Books*, uma inversão de melhor *ScaleUp* ocorreu entre os métodos MPQA e Sentiment140 *Lexicon*, ficando o último desses como o melhor. O pior resultado de *ScaleUp* foi obtido pelo método Emoticons em todas as execuções.

Mais detalhes sobre o tempo de processamento obtido nos experimentos de *ScaleUp* para o *dataset Books* podem ser observados na Tabela 5.4:

Tabela 5.4 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *ScaleUp - dataset Books.*

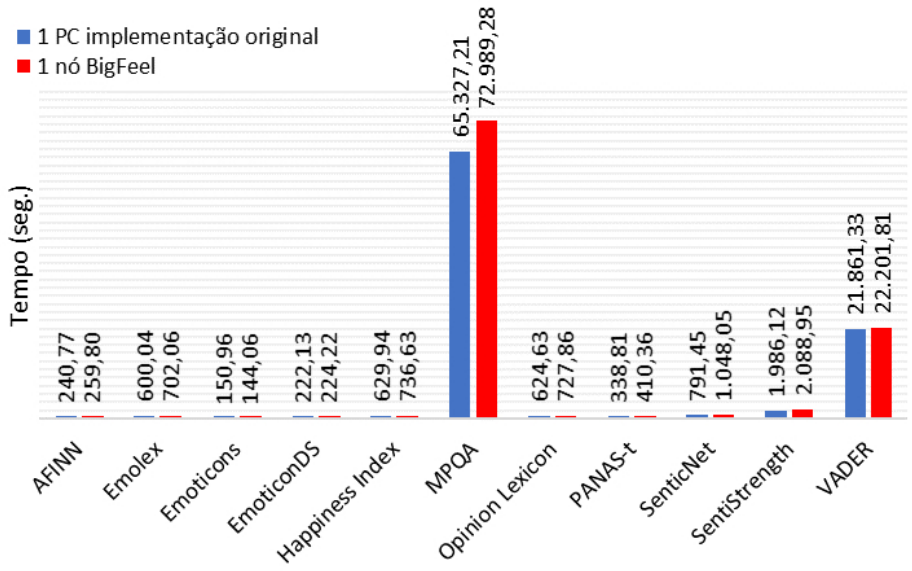
Método	1 nó - 1/8 *Books*	2 nós - 2/8 *Books*	4 nós - 4/8 *Books*	8 nós - 8/8 *Books*
AFINN	149,70 ± 0,76	157,12 ± 1,85	165,40 ± 0,77	188,73 ± 1,66
Emolex	367,56 ± 0,41	371,72 ± 0,47	380,93 ± 1,05	401,31 ± 0,50
Emoticons	90,80 ± 0,30	99,06 ± 0,09	108,60 ± 0,57	130,39 ± 1,19
EmoticonDS	132,05 ± 0,54	136,80 ± 1,15	148,41 ± 1,48	165,99 ± 1,20
Happiness Index	376,91 ± 0,53	378,01 ± 1,00	388,22 ± 1,12	413,78 ± 1,83
MPQA	33.690,56 ± 86,69	**32.600,35 ± 30,58**	**33.298,48 ± 147,60**	34.185,56 ± 64,00
NRC Hashtag	62.888,46 ± 108,91	61.912,59 ± 105,62	62.235,24 ± 87,38	63.289,90 ± 119,97
Opinion Lexicon	357,45 ± 0,72	365,34 ± 0,63	376,15 ± 2,49	413,15 ± 2,02
PANAS-t	209,63 ± 0,17	216,26 ± 0,61	230,22 ± 1,81	249,90 ± 0,43
SANN	–	–	–	–
SASA	5.454,27 ± 9,24	5.453,59 ± 3,54	5.526,25 ± 3,42	5.640,31 ± 10,09
SenticNet	526,93 ± 3,54	520,47 ± 0,64	542,70 ± 2,00	557,62 ± 2,73
Sentiment140 Lexicon	62.937,90 ± 114,16	61.862,57 ± 6,87	63.269,65 ± 107,01	**63.159,32 ± 97,76**
SentiStrength	1.017,84 ± 2,06	1.019,49 ± 1,18	1.058,50 ± 4,01	1.091,10 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	13.225,65 ± 14,41	13.206,17 ± 3,25	13.691,81 ± 59,04	13.762,51 ± 12,11
Spark Naive Bayes	534,81 ± 2,09	547,27 ± 47,44	587,52 ± 4,10	693,16 ± 2,65
Spark Random Forest	1.212,92 ± 2,19	1.200,90 ± 3,87	1.245,84 ± 3,40	1.356,20 ± 4,54
Databricks Stanford	*	*	*	*

Para os experimentos de *ScaleUp*, a Tabela 5.4 mostra que os métodos SANN e Umigon tiveram problemas em sua execução, e os métodos SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford não executaram em tempo hábil.

5.1.2 Dataset Electronics

O Gráfico 5.5 apresenta a comparação entre o tempo médio de execução dos experimentos no *dataset Electronics* em implementação original e o tempo de execução pelo BigFeel em apenas um nó.

Gráfico 5.5 – Tempos de execução de um PC na implementação original e em um nó no *cluster* para o *dataset Electronics*.



Com exceção do método MPQA, os resultados apontam uma pequena diferença no tempo de execução dos métodos, tanto em implementação original não distribuída, quanto em um único nó de processamento BigFeel (Gráfico 5.5). Esse resultado demonstra que o custo computacional de tempo não é consideravelmente maior para se executar o método de análise de sentimentos pelo BigFeel, ainda que existindo a sobrecarga de um nó mestre gerenciando a execução de um nó escravo.

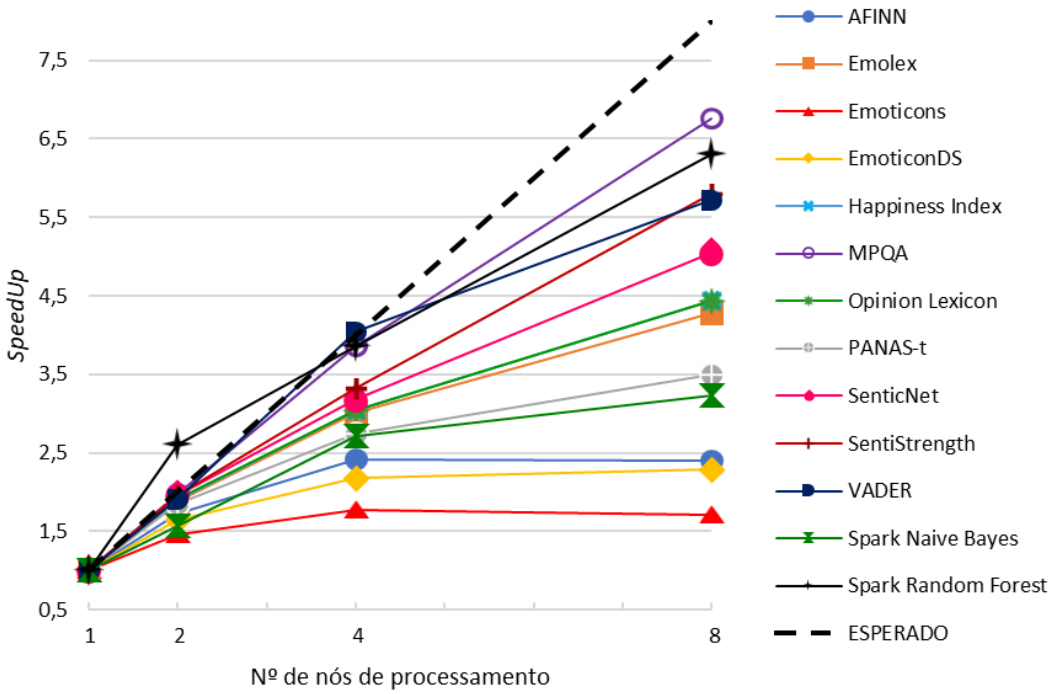
Diferentemente do resultado obtido em comparação semelhante no *dataset Books*, no caso do *dataset Electronics* apenas a execução do método Emoticons resultou em média poucos segundos mais rápida no BigFeel do que em sua implementação original não distribuída, conforme pode ser observado na Tabela 5.5. Para o método EmoticonDS, ocorre um empate estatístico.

Tabela 5.5 – Tempo médio de execução e o respectivo intervalo de confiança em segundos na implementação original e em um nó BigFeel - *dataset Electronics*.

Método	1 PC (impl. original)	1 nó BigFeel
AFINN	**240,77 ± 1,29**	259,80 ± 0,88
Emolex	**600,03 ± 14,74**	702,06 ± 1,27
Emoticons	150,96 ± 1,34	**144,06 ± 0,62**
EmoticonDS	222,13 ± 2,69	224,22 ± 0,51
Happiness Index	**629,94 ± 1,42**	736,62 ± 0,84
MPQA	**65.327,21 ± 60,33**	72.989,28 ± 103,33
NRC Hashtag	*	*
Opinion Lexicon	**624,63 ± 1,42**	727,86 ± 1,45
PANAS-t	**338,80 ± 1,21**	410,35 ± 3,43
SANN	–	–
SASA	–	–
SenticNet	**791,45 ± 0,99**	1.048,05 ± 49,59
Sentiment140 Lexicon	*	*
SentiStrength	**1.986,12 ± 2,14**	2.088,95 ± 15,37
SentiWordNet	*	*
SO-CAL	*	*
Stanford Recursive Deep Model	*	*
Umigon	–	–
VADER	**21.861,33 ± 72,35**	22.201,81 ± 77,00
Spark Naive Bayes	ND	1.070,13 ± 4,00
Spark Random Forest	ND	3.322,34 ± 7,18
Databricks Stanford	ND	*

Os demais experimentos realizados no *dataset Electronics* seguiram as métricas distribuídas de *SpeedUp*, *SizeUp* e *ScaleUp*, definidas na seção anterior.

Gráfico 5.6 – *SpeedUp* da execução dos métodos integrados para o *dataset Electronics*.



Verifica-se no Gráfico 5.6 que os métodos VADER e MPQA apresentaram uma proximidade ao valor esperado de *SpeedUp* até quatro nós de processamento, decaindo um pouco tal resultado para o caso de oito nós. Acima do esperado observa-se apenas o método Spark Random Forest para dois nós, o qual também acompanha o comportamento em *SpeedUp* dos outros métodos para o caso de oito nós de processamento, decaindo um pouco em relação ao valor esperado. Os resultados observados de *SpeedUp* para o *dataset Electronics* se resumem a (Gráfico 5.6; Tabela 5.6):

- 2 nós - melhor: Spark Random Forest; pior: Emoticons.
- 4 nós - melhor: VADER; pior: Emoticons.
- 8 nós - melhor: MPQA; pior: Emoticons.

Todos os métodos apresentaram crescentes taxas de *SpeedUp* até quatro nós, porém os métodos AFINN e Emoticons atingiram o limite de ganho em *SpeedUp* entre quatro e oito nós, mostrando-se pior para oito nós de processamento. Isso demonstra um limite superior existente de ganho em tempo de processamento em função do custo computacional de divisão do trabalho.

A Tabela 5.6 apresenta o resultado em tempo médio da execução de todos os experimentos no *dataset Electronics*:

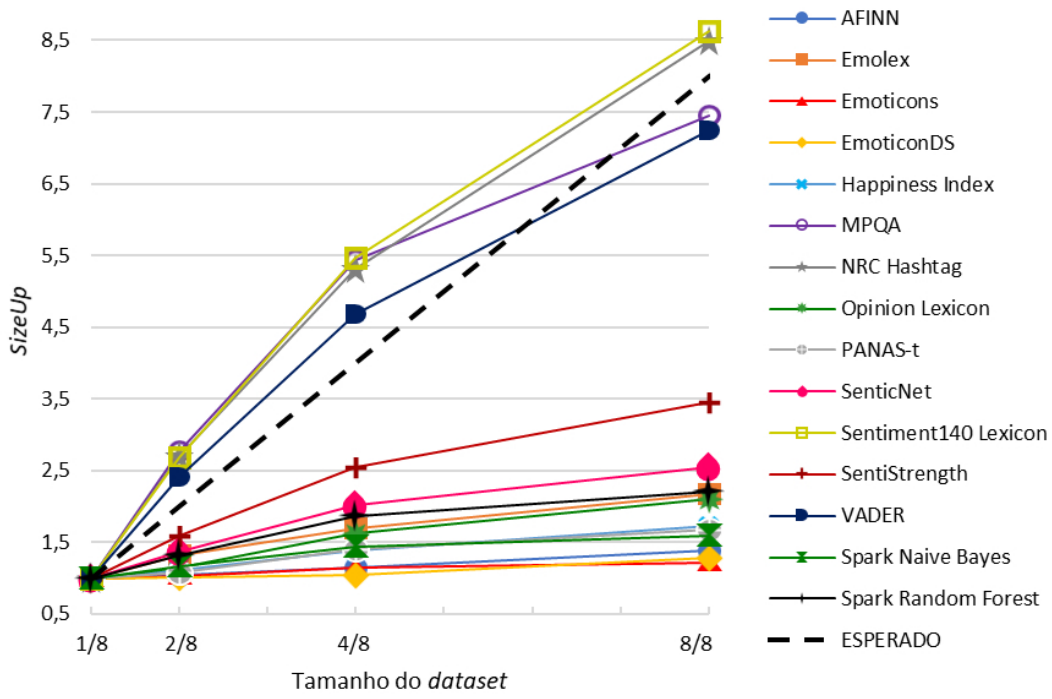
Tabela 5.6 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *SpeedUp - dataset Electronics.*

Método	1 nó BigFeel	2 nós BigFeel	4 nós BigFeel	8 nós BigFeel
AFINN	259,80 ± 0,88	150,12 ± 0,99	107,48 ± 0,98	108,53 ± 1,66
Emolex	702,06 ± 1,27	372,46 ± 15,40	233,26 ± 1,44	164,09 ± 0,50
Emoticons	144,06 ± 0,62	98,98 ± 1,04	81,02 ± 0,34	83,96 ± 1,19
EmoticonDS	224,28 ± 0,51	135,74 ± 1,34	102,90 ± 0,53	97,66 ± 1,20
Happiness Index	736,63 ± 0,84	388,54 ± 3,56	242,04 ± 0,77	166,24 ± 1,82
MPQA	72.989,28 ± 103,33	36.907,75 ± 76,47	18.889,81 ± 64,36	**10.799,11 ± 64,00**
NRC Hashtag	*	52.552,84 ± 93,51	27.599,62 ± 72,33	16.865,03 ± 52,81
Opinion Lexicon	727,86 ± 1,45	380,47 ± 1,70	239,31 ± 1,62	163,81 ± 2,02
PANAS-t	410,35 ± 3,43	221,60 ± 2,40	149,98 ± 2,05	117,46 ± 0,43
SANN	–	–	–	–
SASA	–	–	–	–
SenticNet	1.048,05 ± 49,59	533,34 ± 4,66	328,62 ± 2,42	207,37 ± 2,73
Sentiment140 Lexicon	*	52.051,81 ± 86,76	28.662,71 ± 105,38	16.855,46 ± 81,00
SentiStrength	2.088,95 ± 15,37	1.062,04 ± 7,08	628,53 ± 1,67	360,48 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	22.201,81 ± 77,00	11.632,28 ± 79,39	**5.491,76 ± 29,88**	3.876,17 ± 12,11
Spark Naive Bayes	1.070,13 ± 4,00	683,85 ± 3,13	394,62 ± 8,91	330,61 ± 5,46
Spark Random Forest	3.322,34 ± 7,18	**1.277,43 ± 3,41**	862,92 ± 4,97	527,64 ± 2,26
Databricks Stanford	*	*	*	*

Observa-se pela Tabela 5.6 que os métodos SANN, SASA e Umigon não completaram sua execução, sendo portanto removidos do Gráfico 5.6. Métodos que ultrapassaram o tempo limite convencionado de 24 horas de execução são: NRC *Hashtag*, Sentiment140 *Lexicon*, SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford.

O Gráfico 5.7 apresenta os resultados de *SizeUp* dos métodos que executaram sem problemas. Vários métodos apresentaram *SizeUp* pior que o esperado, como: MPQA, NRC *Hashtag*, Sentiment140 *Lexicon* e VADER, porém MPQA e VADER apresentaram uma normalização próxima ao valor esperado para 8/8 *Electronics*.

Gráfico 5.7 – *SizeUp* da execução dos métodos integrados ao BigFeel para o *dataset Electronics*.



Os resultados observados de *SizeUp* para o *dataset Electronics* se resumem a (Gráfico 5.7; Tabela 5.7):

- 2/8 *Electronics* - melhor: EmoticonDS; pior: MPQA.
- 4/8 *Electronics* - melhor: EmoticonDS; pior: Sentiment140 *Lexicon*.
- 8/8 *Electronics* - melhor: Emoticons; pior: Sentiment140 *Lexicon*.

Observa-se que para os métodos que apresentam *SizeUp* melhor que o esperado, os resultados demonstram baixo impacto no processamento em função da variação do tamanho de entrada, principalmente após a execução em 4/8 *Electronics*.

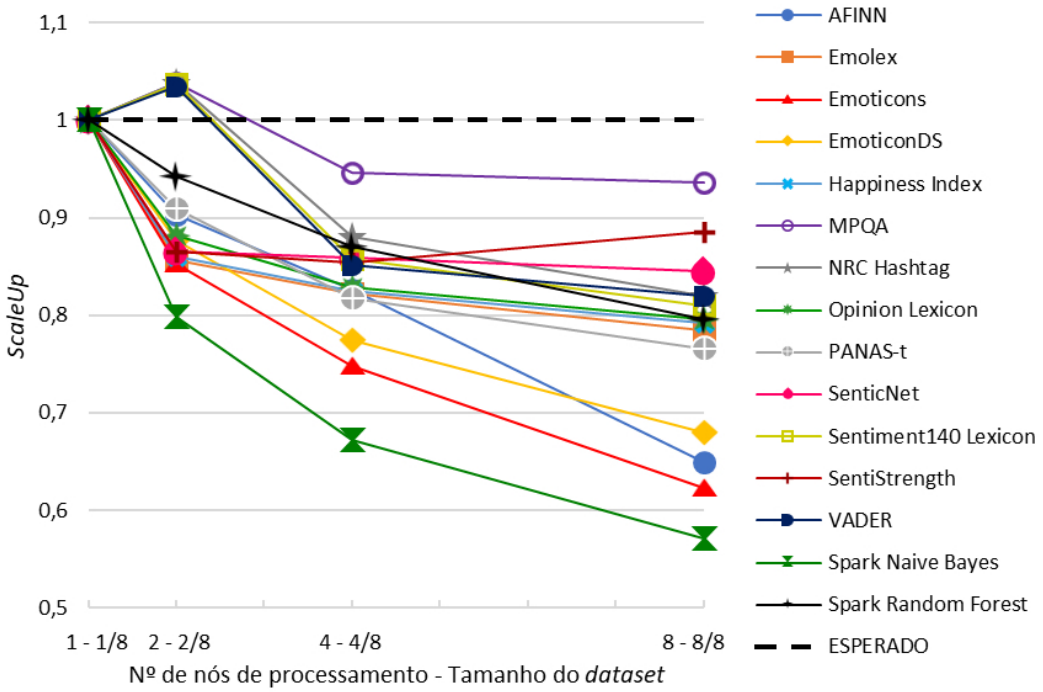
A Tabela 5.7 detalha os resultados em tempo médio usados no cálculo do *SizeUp* nos experimentos com o *dataset Electronics*:

Tabela 5.7 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *SizeUp* - *dataset Electronics*.

Método	1/8 *Electronics*	2/8 *Electronics*	4/8 *Electronics*	8/8 *Electronics*
AFINN	77,89 ± 0,48	81,26 ± 0,07	89,67 ± 0,02	108,53 ± 1,66
Emolex	75,52 ± 0,28	99,75 ± 0,06	127,87 ± 0,12	164,09 ± 0,50
Emoticons	68,89 ± 0,12	71,16 ± 0,01	78,61 ± 0,03	**83,96 ± 1,19**
EmoticonDS	76,13 ± 0,15	**77,09 ± 0,10**	**80,07 ± 0,05**	97,66 ± 1,20
Happiness Index	96,58 ± 0,24	107,81 ± 0,15	133,82 ± 0,04	166,24 ± 1,82
MPQA	1.449,39 ± 10,61	4.017,72 ± 4,49	7.875,98 ± 5,90	10.799,11 ± 64,00
NRC Hashtag	1.986,91 ± 9,69	5.359,37 ± 5,40	10.573,29 ± 12,95	16.865,03 ± 92,31
Opinion Lexicon	77,63 ± 0,31	89,22 ± 0,08	126,34 ± 0,37	163,81 ± 2,02
PANAS-t	70,06 ± 0,07	75,90 ± 0,14	98,14 ± 0,09	117,457 ± 0,43
SANN	71,08 ± 0,15	73,46 ± 0,03	–	–
SASA	276,68 ± 0,25	666,14 ± 1,96	–	–
SenticNet	81,55 ± 0,09	111,05 ± 0,06	164,43 ± 0,30	207,37 ± 2,73
Sentiment140 Lexicon	1.955,18 ± 5,29	5.261,31 ± 4,22	10.701,04 ± 2,24	16.855,46 ± 55,07
SentiStrength	104,58 ± 0,25	166,44 ± 0,79	266,40 ± 0,48	360,48 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	534,93 ± 3,26	1.291,53 ± 2,82	2.500,32 ± 10,30	3.876,17 ± 12,11
Spark Naive Bayes	207,79 ± 1,17	240,20 ± 0,57	297,33 ± 3,12	330,61 ± 2,24
Spark Random Forest	239,70 ± 1,68	314,87 ± 1,56	445,06 ± 2,85	527,64 ± 2,22
Databricks Stanford	*	*	*	*

O Gráfico 5.8 apresenta os resultados de *ScaleUp* para o *dataset Electronics*.

Gráfico 5.8 – *ScaleUp* da execução dos métodos integrados ao BigFeel para o *dataset Electronics*.



Conforme apresentado no Gráfico 5.8, os métodos MPQA, NRC *Hashtag*, Sentiment140 *Lexicon* e VADER apresentaram um melhor resultado em *ScaleUp* que o esperado para dois nós - 2/8 *Electronics*. A seguir, no experimento usando quatro nós de processamento em 4/8 *Electronics*, os mesmos métodos seguiram a tendência dos demais, apresentando valores abaixo do esperado. Com exceção do método SentiStrength em oito nós - 8/8 *Electronics*, todos os métodos apresentaram queda no *ScaleUp* após dois nós - 2/8 *Electronics*.

Em suma, os resultados observados de *ScaleUp* para o *dataset Electronics* se resumem a (Gráfico 5.8; Tabela 5.8):

- 2 nós, 2/8 *Electronics* - melhor: NRC *Hashtag*; pior: Spark Naive Bayes.
- 4 nós, 4/8 *Electronics* - melhor: MPQA; pior: Spark Naive Bayes.
- 8 nós, 8/8 *Electronics* - melhor: MPQA; pior: Spark Naive Bayes.

Mais detalhes sobre o tempo médio de processamento usado no cálculo de *ScaleUp* para o *dataset Electronics* podem ser observados na Tabela 5.8:

Tabela 5.8 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *ScaleUp* - *dataset Electronics.*

Método	1 nó - 1/8 *Electronics*	2 nós - 2/8 *Electronics*	4 nós - 4/8 *Electronics*	8 nós - 8/8 *Electronics*
AFINN	70,41 ± 0,21	77,94 ± 0,36	85,27 ± 0,30	108,53 ± 1,66
Emolex	128,66 ± 0,36	150,26 ± 0,09	156,49 ± 0,19	164,09 ± 0,50
Emoticons	52,26 ± 0,04	61,25 ± 0,18	69,92 ± 0,15	83,96 ± 1,19
EmoticonDS	66,39 ± 0,37	75,74 ± 0,22	85,68 ± 0,31	97,66 ± 1,20
Happiness Index	131,65 ± 0,51	153,04 ± 0,29	159,60 ± 0,55	166,24 ± 1,83
MPQA	10.109,06 ± 3,24	9.741,43 ± 8,12	**10.688,51 ± 3,80**	**10.799,11 ± 64,00**
NRC Hashtag	13.823,86 ± 18,92	**13.312,76 ± 7,34**	15.712,29 ± 3,99	16.865,03 ± 73,00
Opinion Lexicon	130,27 ± 0,38	147,77 ± 0,33	157,11 ± 0,45	163,81 ± 2,02
PANAS-t	89,94 ± 0,17	98,86 ± 0,22	110,10 ± 0,37	117,46 ± 0,43
SANN	63,08 ± 0,06	72,63 ± 0,06	–	–
SASA	1.600,16 ± 6,42	1.830,88 ± 4,46	–	–
SenticNet	175,18 ± 0,22	202,43 ± 0,43	204,04 ± 0,50	207,37 ± 2,73
Sentiment140 Lexicon	13.647,36 ± 5,00	13.153,17 ± 3,73	15.915,98 ± 4,48	16.855,46 ± 88,91
SentiStrength	319,40 ± 0,34	369,10 ± 0,35	373,85 ± 1,50	360,48 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	3.177,89 ± 1,40	3.071,49 ± 1,14	3.731,90 ± 2,61	3.876,17 ± 12,11
Spark Naive Bayes	188,74 ± 0,43	236,30 ± 1,35	280,64 ± 1,28	330,61 ± 5,31
Spark Random Forest	419,71 ± 2,18	445,30 ± 4,79	482,48 ± 2,88	527,64 ± 3,00
Databricks Stanford	*	*	*	*

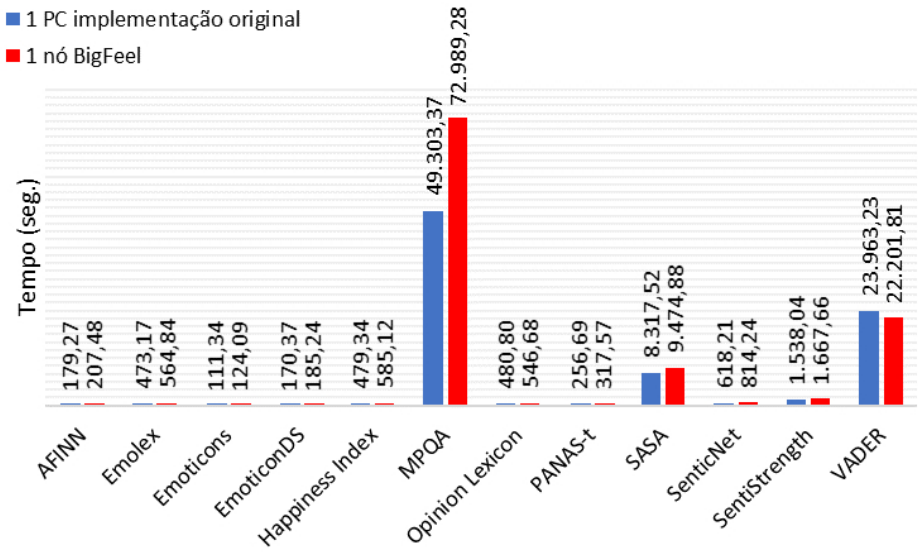
Para os experimentos de *ScaleUp*, a Tabela 5.8 apresenta que apenas os métodos Umigon e algumas execuções dos métodos SANN e SASA tiveram problemas em sua execução.

Os métodos SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford, somados aos métodos SANN, SASA e Umigon foram retirados do Gráfico 5.8.

5.1.3 Dataset Movies_TV

O Gráfico 5.9 apresenta a comparação entre o tempo médio de execução dos experimentos no *dataset Movies_TV* em implementação original e o tempo de execução pelo BigFeel em apenas um nó.

Gráfico 5.9 – Tempos de execução em um PC na implementação original e em um nó BigFeel para o *dataset Movies_TV*.



Diferentemente do *dataset Books* analisado anteriormente, para o menor *dataset* usado nos experimentos, o *Movies_TV*, observa-se pelo Gráfico 5.9 e pela Tabela 5.9 que os resultados obtidos pelo BigFeel em tempo médio de execução apresentam um tempo muito acima da implementação original não distribuída, comportamento esperado no *dataset Movies_TV*, pois a execução de pequenos *datasets* em um *cluster* tem a tendência de sofrer uma sobrecarga do trabalho de comunicação para a distribuição dos algoritmos e dados pelo *cluster*, custo esse que pode até mesmo sobrepor os ganhos da distribuição.

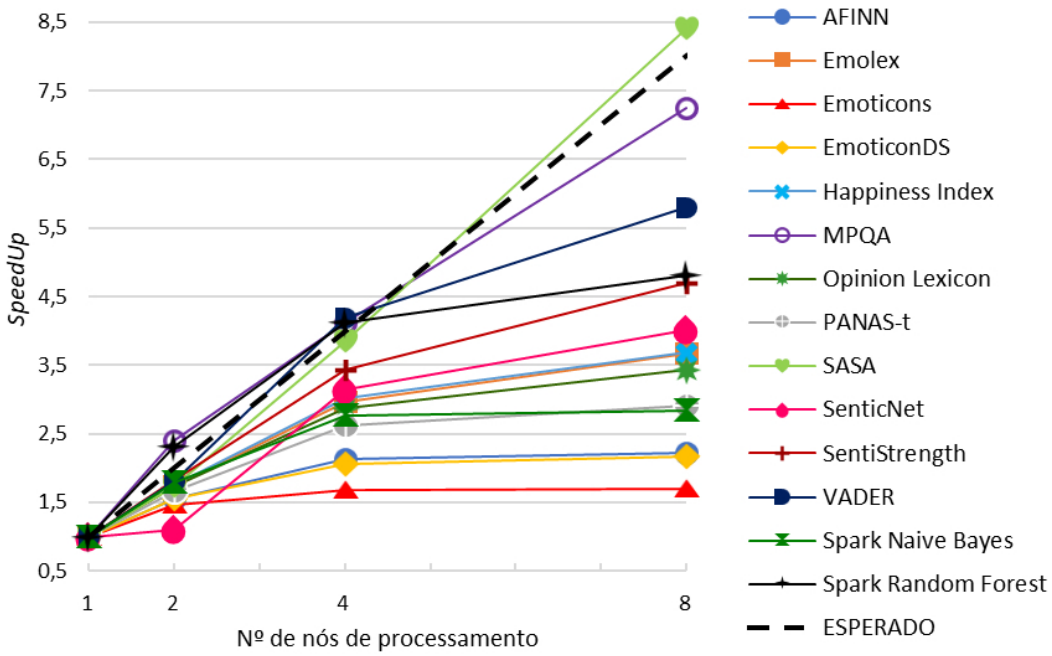
Apenas o método VADER obteve um tempo médio de processamento inferior em execução do BigFeel, tendo portanto todos os outros métodos obtido melhores tempos em implementação original não distribuída. Mais detalhes podem ser observados na Tabela 5.9.

Tabela 5.9 – Tempo médio de execução e o respectivo intervalo de confiança em segundos na implementação original e em um nó BigFeel - *dataset Movies_TV*.

Método	1 PC (impl. original)	1 nó BigFeel
AFINN	**179,27 ± 1,05**	207,48 ± 0,88
Emolex	**473,17 ± 1,52**	564,84 ± 1,27
Emoticons	**111,34 ± 1,29**	124,08 ± 0,62
EmoticonDS	**170,36 ± 1,31**	185,24 ± 0,51
Happiness Index	**479,34 ± 2,44**	585,12 ± 0,84
MPQA	**49.303,37 ± 15,01**	72.989,28 ± 96,81
NRC Hashtag	*	*
Opinion Lexicon	**480,80 ± 0,84**	546,68 ± 1,44
PANAS-t	**256,69 ± 0,85**	317,57 ± 3,43
SANN	74,17 ± 0,10	–
SASA	**8.317,51 ± 117,26**	9.474,88 ± 14,50
SenticNet	**618,21 ± 1,64**	814,24 ± 49,59
Sentiment140 Lexicon	*	*
SentiStrength	**1.538,04 ± 37,10**	1.667,66 ± 15,37
SentiWordNet	*	*
SO-CAL	*	*
Stanford Recursive Deep Model	*	*
Umigon	–	–
VADER	23.963,23 ± 54,84	**22.201,81 ± 14,99**
Spark Naive Bayes	ND	898,82 ± 5,71
Spark Random Forest	ND	2.505,07 ± 7,71
Databricks Stanford	ND	*

Os demais experimentos realizados no *dataset Movies_TV* seguiram as métricas distribuídas de *SpeedUp*, *SizeUp* e *ScaleUp*.

Gráfico 5.10 – *SpeedUp* da execução dos métodos integrados para o *dataset Movies_TV*.



No Gráfico 5.10, verifica-se que os métodos MPQA e SASA apresentaram uma proximidade ao valor esperado de *SpeedUp* em todos os casos do experimento, de um a oito nós de processamento, tendo o MPQA apresentado melhor resultado até quatro nós e a seguir declinado um pouco seu *SpeedUp* em oito nós, ficando atrás apenas do método SASA como melhor no caso de oito nós. Acima do esperado, apenas ficaram os métodos Spark Random Forest e MPQA para dois e quatro nós, além do método VADER na execução em quatro nós. Como pior *SpeedUp* geral observa-se o método Emoticons, quase invariável após dois nós de processamento, o que implica baixa variação no tempo de processamento em função do número de nós de processamento.

Os resultados observados de *SpeedUp* para o *dataset Movies_TV* se resumem a (Gráfico 5.10; Tabela 5.10):

- 2 nós - melhor: MPQA; pior: SenticNet.
- 4 nós - melhor: VADER; pior: Emoticons.
- 8 nós - melhor: SASA; pior: Emoticons.

Todos os métodos apresentaram crescente *SpeedUp* em todas as variações do número de nós de processamento, ainda que em poucas situações acima do valor esperado. Observa-se

pelo Gráfico 5.10 que entre quatro e oito nós o *SpeedUp* dos métodos Emoticons, EmoticonDS, AFINN e Spark Naive Bayes tiveram um singelo crescimento, o que indica que o aumento do número de nós de processamento trouxe pouco ganho em tempo de execução para o caso da execução com oito nós.

A Tabela 5.10 apresenta o resultado em tempo médio da execução de todos os experimentos no *dataset Movies_TV*:

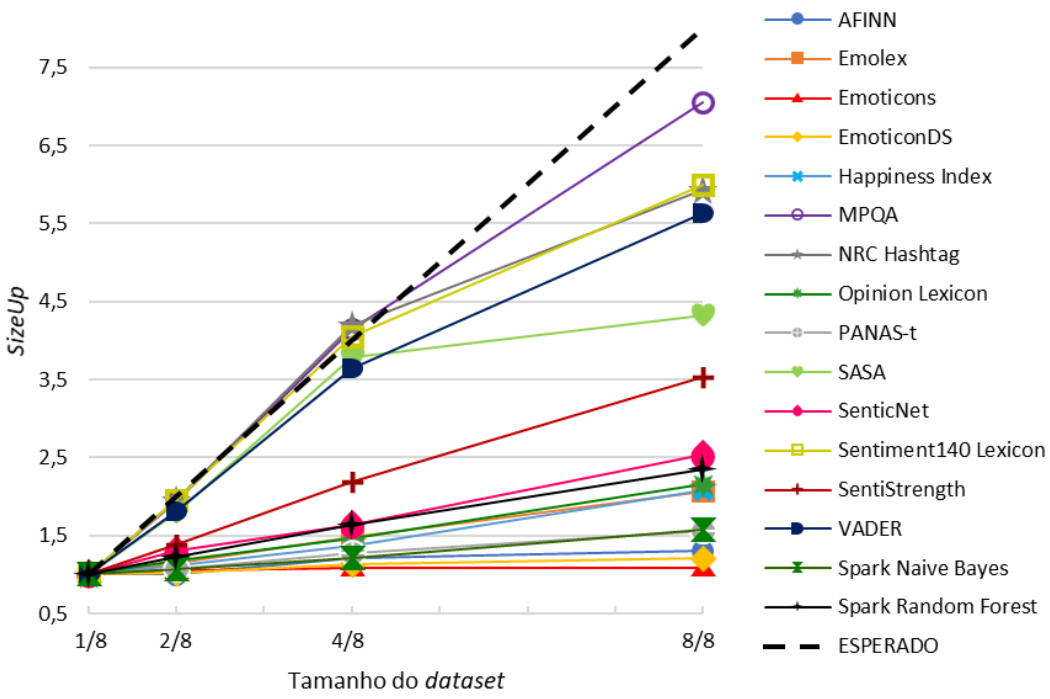
Tabela 5.10 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *SpeedUp - dataset Movies_TV.*

Método	1 nó BigFeel	2 nós BigFeel	4 nós BigFeel	8 nós BigFeel
AFINN	207,48 ± 0,88	133,00 ± 0,99	97,17 ± 0,98	93,55 ± 1,66
Emolex	564,84 ± 1,27	325,11 ± 15,40	190,88 ± 1,44	154,30 ± 0,50
Emoticons	124,08 ± 0,62	84,52 ± 1,03	74,26 ± 0,34	72,76 ± 1,19
EmoticonDS	185,24 ± 0,51	119,66 ± 1,34	89,90 ± 0,53	85,50 ± 1,20
Happiness Index	585,12 ± 0,84	334,43 ± 3,56	194,03 ± 0,77	158,53 ± 1,83
MPQA	72.989,28 ± 147,51	**30.418,54 ± 89,36**	17.713,36 ± 148,19	10.070,71 ± 64,00
NRC Hashtag	*	56.991,69 ± 97,45	25.513,12 ± 119,62	15.727,48 ± 80,89
Opinion Lexicon	546,68 ± 1,45	314,58 ± 1,70	190,42 ± 1,62	159,48 ± 2,02
PANAS-t	317,57 ± 3,43	190,97 ± 2,40	121,62 ± 2,05	109,31 ± 0,43
SANN	–	–	–	–
SASA	9.474,88 ± 14,50	5.460,32 ± 2,27	2.463,10 ± 2,05	**1.128,26 ± 10,09**
SenticNet	814,24 ± 49,59	744,26 ± 4,66	259,11 ± 2,42	203,22 ± 2,73
Sentiment140 Lexicon	*	56.855,05 ± 154,65	26.963,62 ± 132,37	15.718,56 ± 50,04
SentiStrength	1.667,66 ± 15,37	917,34 ± 7,08	486,90 ± 1,67	355,02 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	22.201,81 ± 76,58	12.348,39 ± 159,61	**5.302,32 ± 109,73**	3.821,77 ± 12,11
Spark Naive Bayes	898,82 ± 5,71	502,44 ± 4,54	325,63 ± 9,29	316,20 ± 2,49
Spark Random Forest	2.505,07 ± 7,71	1.084,86 ± 3,95	608,62 ± 3,77	521,16 ± 4,75
Databricks Stanford	*	*	*	*

Observa-se pela Tabela 5.10 que os métodos SANN e Umigon não completaram sua execução, sendo portanto removidos do Gráfico 5.10. Devido ao tempo de execução, também foram removidos os métodos NRC Hashtag, Sentiment140 *Lexicon*, SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford.

O Gráfico 5.11 apresenta os resultados de *SizeUp* dos métodos que executaram sem nenhum erro. Como se observa, todos os métodos apresentaram *SizeUp* superior ao esperado após 4/8 do *dataset Movies_TV*.

Gráfico 5.11 – *SizeUp* da execução dos métodos integrados ao BigFeel para o *dataset Movies_TV*.



Os resultados observados de *SizeUp* para o *dataset Movies_TV* se resumem a (Gráfico 5.11; Tabela 5.11):

- 2/8 *Movies_TV* - melhor: AFINN; pior: Sentiment140 *Lexicon*.
- 4/8 *Movies_TV* - melhor: Emoticons; pior: NRC *Hashtag*.
- 8/8 *Movies_TV* - melhor: Emoticons; pior: MPQA.

Observa-se pelo Gráfico 5.11 que todos os métodos ficaram próximos ao valor esperado de *SizeUp* ou melhores que ele, sendo que apenas MPQA, NRC *Hashtag* e Sentiment140 *Lexicon* para a execução em 4/8 *Movies_TV* se mostraram piores que o esperado, recuperando-se portanto na execução seguinte em 8/8 *Movies_TV*.

A Tabela 5.11 detalha os resultados em tempo médio usados no cálculo de *SizeUp* com o *dataset Movies_TV*:

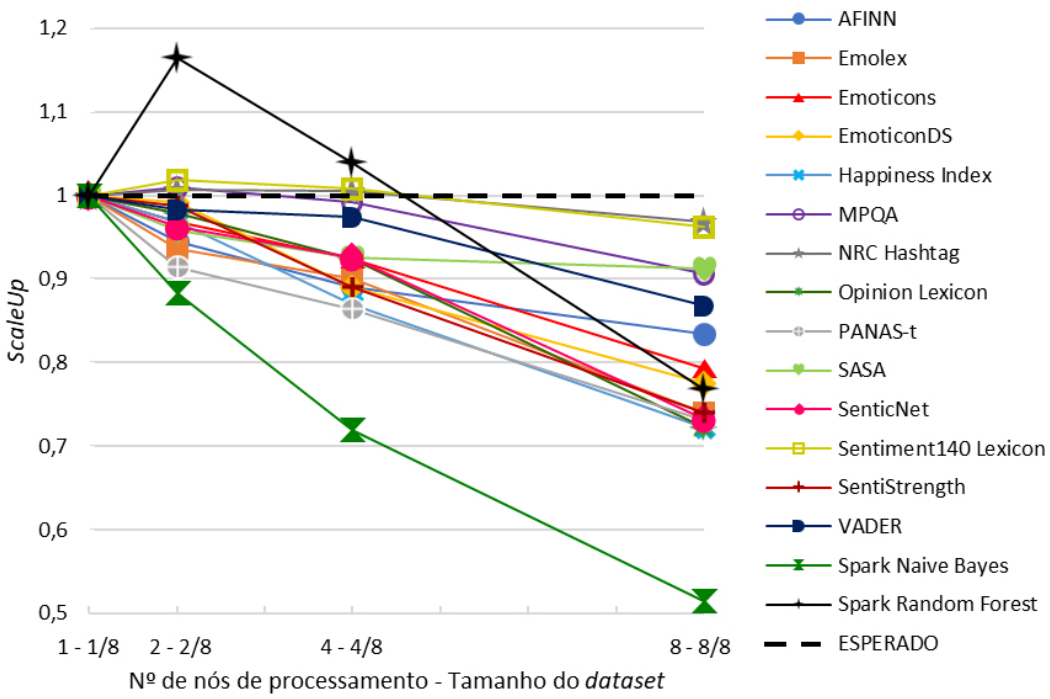
Tabela 5.11 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *SizeUp - dataset Movies_TV.*

Método	1/8 *Movies_TV*	2/8 *Movies_TV*	4/8 *Movies_TV*	8/8 *Movies_TV*
AFINN	71,47 ± 0,68	**72,06 ± 0,46**	86,06 ± 0,14	93,55 ± 1,66
Emolex	74,63 ± 0,27	86,30 ± 0,15	110,24 ± 0,64	154,3 ± 0,50
Emoticons	67,47 ± 0,32	71,44 ± 0,33	**72,88 ± 0,44**	**72,76 ± 1,19**
EmoticonsDS	70,12 ± 0,20	71,32 ± 0,43	79,15 ± 0,74	85,50 ± 1,20
Happiness Index	75,96 ± 0,21	85,10 ± 0,15	104,56 ± 0,37	158,53 ± 1,83
MPQA	1.427,34 ± 0,75	2.760,17 ± 1,79	5.909,90 ± 2,71	10.070,71 ± 64,01
NRC Hashtag	2.651,99 ± 0,19	5.165,00 ± 3,66	11.108,90 ± 1,62	15.727,48 ± 87,42
Opinion Lexicon	73,67 ± 0,11	86,49 ± 0,33	108,43 ± 0,69	159,48 ± 2,02
PANAS-t	70,17 ± 0,08	75,12 ± 0,22	89,74 ± 0,82	109,31 ± 0,43
SANN	68,00 ± 0,14	70,79 ± 0,30	73,30 ± 0,44	–
SASA	261,22 ± 0,11	469,26 ± 1,38	988,67 ± 1,28	1.128,26 ± 10,10
SenticNet	79,93 ± 0,06	104,71 ± 0,26	130,94 ± 1,28	203,22 ± 2,73
Sentiment140 Lexicon	2.620,70 ± 0,60	5.105,53 ± 1,79	10.630,62 ± 3,76	15.718,56 ± 155,58
SentiStrength	100,32 ± 0,21	138,49 ± 1,29	219,34 ± 0,87	355,02 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	678,10 ± 0,91	1.224,82 ± 3,54	2.466,50 ± 1,65	3.821,77 ± 12,11
Spark Classification by Naive Bayes	201,34 ± 1,37	216,33 ± 2,03	243,29 ± 4,21	316,20 ± 3,48
Spark Classification by Random Forest	221,97 ± 1,26	273,82 ± 1,37	363,08 ± 2,57	521,16 ± 3,49
Databricks Stanford	*	*	*	*

Foram retirados do Gráfico 5.11 os métodos SANN e Umigon por terem falhado em um ou mais casos, além dos métodos SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford, por terem tomado mais que o limite de tempo convencionado para o experimento.

O Gráfico 5.12 apresenta os resultados obtido pelo experimento de *ScaleUp* dos métodos integrados ao BigFeel. Observa-se que o único método que se sobressaiu aos valores esperados em *ScaleUp* foi Spark Random Forest para os experimentos de dois e quatro nós de processamento e, 2/8 e 4/8 do *dataset*, respectivamente, demonstrando ser o melhor dos resultados obtidos até quatro nós de processamento e 4/8 do *dataset*. Os métodos MPQA, NRC *Hashtag* e Sentiment140 *Lexicon* ficaram próximos ao valor esperado até a execução em quatro nós - 4/8 *Movies_TV*. Para a última execução do *dataset*, oito nós e 8/8 *Movies_TV*, perderam um pouco em *ScaleUp*.

Gráfico 5.12 – *ScaleUp* da execução dos métodos integrados ao BigFeel para o *dataset Movies_TV*.



Os resultados observados de *ScaleUp* para o *dataset Movies_TV* se resumem a (Gráfico 5.12; Tabela 5.12):

- 2 nós, 2/8 *Movies_TV* - melhor: Spark Random Forest; pior: Spark Naive Bayes.
- 4 nós, 4/8 *Movies_TV* - melhor: Spark Random Forest; pior: Spark Naive Bayes.
- 8 nós, 8/8 *Movies_TV* - melhor: NRC *Hashtag*; pior: Spark Naive Bayes.

Os detalhes sobre o tempo médio de processamento usado no cálculo de *ScaleUp* para o *dataset Movies_TV* podem ser observados na Tabela 5.12:

Tabela 5.12 – Tempo médio de execução e o respectivo intervalo de confiança em segundos para *ScaleUp - dataset Movies_TV.*

Método	1 nó - 1/8 *Movies_TV*	2 nós - 2/8 *Movies_TV*	4 nós - 4/8 *Movies_TV*	8 nós - 8/8 *Movies_TV*
AFINN	78,08 ± 0,08	82,66 ± 0,39	87,71 ± 0,43	93,55 ± 1,66
Emolex	114,29 ± 0,34	122,10 ± 0,18	126,95 ± 0,30	154,30 ± 0,50
Emoticons	57,67 ± 0,24	59,57 ± 0,20	62,41 ± 0,22	72,76 ± 1,19
EmoticonDS	66,30 ± 0,35	66,85 ± 0,30	74,51 ± 0,12	85,50 ± 1,20
Happiness Index	114,55 ± 0,37	118,35 ± 0,12	131,74 ± 0,27	158,53 ± 1,83
MPQA	9.126,05 ± 13,33	9.036,26 ± 10,47	9.199,14 ± 8,28	10.070,71 ± 64,01
NRC Hashtag	15.228,25 ± 31,96	15.130,42 ± 13,37	15.153,32 ± 10,47	**15.727,48 ± 62,89**
Opinion Lexicon	115,13 ± 0,14	117,66 ± 0,24	124,66 ± 0,18	159,48 ± 2,02
PANAS-t	79,94 ± 0,34	87,53 ± 0,11	92,50 ± 0,33	109,31 ± 0,43
SANN	51,27 ± 0,23	53,34 ± 0,19	57,76 ± 0,19	–
SASA	1.028,69 ± 2,54	1.072,96 ± 0,84	1.111,95 ± 5,82	1.128,26 ± 10,10
SenticNet	148,77 ± 0,26	154,67 ± 0,48	160,75 ± 0,25	203,22 ± 2,73
Sentiment140 Lexicon	15.122,59 ± 13,18	14.851,23 ± 7,57	15.002,46 ± 31,17	15.718,56 ± 82,31
SentiStrength	262,72 ± 0,38	266,21 ± 0,77	295,07 ± 1,00	355,02 ± 1,31
SentiWordNet	*	*	*	*
SO-CAL	*	*	*	*
Stanford Recursive Deep Model	*	*	*	*
Umigon	–	–	–	–
VADER	3.320,18 ± 1,29	3.378,26 ± 1,55	3.405,51 ± 1,04	3.821,77 ± 12,11
Spark Naive Bayes	162,71 ± 2,63	184,37 ± 3,43	226,10 ± 2,84	316,20 ± 1,23
Spark Random Forest	400,53 ± 3,10	**344,14 ± 4,24**	**385,40 ± 6,20**	521,16 ± 1,38
Databricks Stanford	*	*	*	*

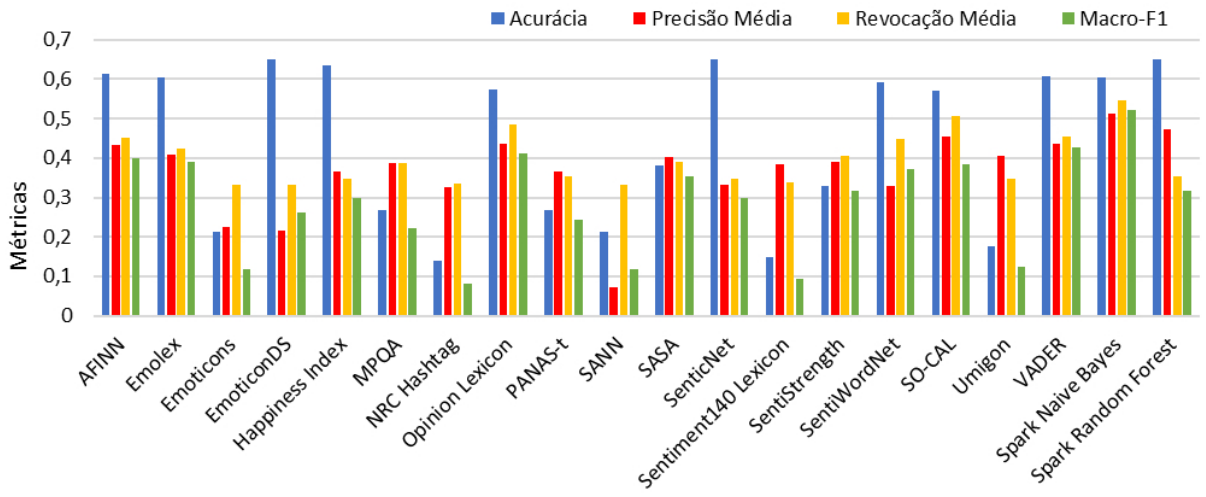
Para os experimentos de *ScaleUp*, a Tabela 5.12 apresenta que apenas os métodos SANN e Umigon tiveram problemas em algumas de suas execuções. Os métodos SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford, somados aos métodos SANN e Umigon foram retirados do Gráfico 5.12.

5.2 Eficácia

Para testar a eficácia dos métodos implementados no Spark comparando-os com os outros métodos integrados ao BigFeel, foram analisados os resultados obtidos pelas métricas de eficácia usando o *dataset* IMDB, o qual possui um gabarito (resposta) para avaliação. Os demais *datasets* usados nos experimentos de eficiência não possuem um gabarito.

Os resultados apresentados nesta seção representam a média aritmética das 10 execuções de cada experimento, como se pode observar no Gráfico 5.13 e na Tabela 5.13:

Gráfico 5.13 – Acurácia, precisão média, revocação média e macro-$F1$ sob o *dataset* IMDB.



Foi destacado em negrito na Tabela 5.13 o método cujo valor tenha se sobressaído como o melhor dentre as execuções de cada método, observado o intervalo de confiança de 95%.

Tabela 5.13 – Resultados da média e o respectivo intervalo de confiança em segundos das métricas de eficácia sob o *dataset* IMDB.

Método	Acurácia	Precisão	Revocação	Macro-F1
AFINN	0,6126 ± 0,0055	0,4324 ± 0,0098	0,4516 ± 0,0059	0,3979 ± 0,0056
Emolex	0,6045 ± 0,0042	0,4087 ± 0,0002	0,4241 ± 0,0002	0,3917 ± 0,0001
Emoticons	0,2132 ± 0,0029	0,2266 ± 0,0033	0,3334 ± 0,0000	0,1172 ± 0,0000
EmoticonDS	**0,6514 ± 0,0018**	0,2171 ± 0,0000	0,3333 ± 0,0000	0,2630 ± 0,0000
Happiness Index	0,6358 ± 0,0020	0,3654 ± 0,0012	0,3464 ± 0,0001	0,2996 ± 0,0001
MPQA	0,2666 ± 0,0024	0,3866 ± 0,0001	0,3877 ± 0,0001	0,2233 ± 0,0001
NRC Hashtag	0,1384 ± 0,0021	0,3265 ± 0,0005	0,3344 ± 0,0000	0,0828 ± 0,0000
Opinion Lexicon	0,5737 ± 0,0054	0,4366 ± 0,0001	0,4849 ± 0,0001	0,4116 ± 0,0001
PANAS-t	0,2690 ± 0,0022	0,3670 ± 0,0001	0,3533 ± 0,0001	0,2430 ± 0,0001
SANN	0,2130 ± 0,0030	0,0710 ± 0,0000	0,3333 ± 0,0000	0,1171 ± 0,0000
SASA	0,3816 ± 0,0032	0,4018 ± 0,0001	0,3900 ± 0,0001	0,3522 ± 0,0001
SenticNet	0,6401 ± 0,0026	0,3319 ± 0,0003	0,3486 ± 0,0001	0,2976 ± 0,0001
Sentiment140 Lexicon	0,1489 ± 0,0018	0,3842 ± 0,0024	0,3390 ± 0,0000	0,0945 ± 0,0000
SentiStrength	0,3287 ± 0,0036	0,3893 ± 0,0001	0,4057 ± 0,0002	0,3170 ± 0,0001
SentiWordNet	0,5925 ± 0,0033	0,3283 ± 0,0001	0,4477 ± 0,0001	0,3710 ± 0,0001
SO-CAL	0,5717 ± 0,0045	0,4533 ± 0,0003	0,5058 ± 0,0001	0,3856 ± 0,0001
Stanford Recursive Deep Model	–	–	–	–
Umigon	0,1755 ± 0,0034	0,4062 ± 0,0009	0,3484 ± 0,0001	0,1235 ± 0,0001
VADER	0,6062 ± 0,0039	0,4359 ± 0,0002	0,4558 ± 0,0002	0,4261 ± 0,0002
Spark Naive Bayes	0,6038 ± 0,0051	**0,5121 ± 0,0001**	**0,5455 ± 0,0002**	**0,5208 ± 0,0002**
Spark Random Forest	**0,6499 ± 0,0027**	0,4727 ± 0,0004	0,3547 ± 0,0001	0,3158 ± 0,0001
Databricks Stanford	–	–	–	–
MÉDIA GLOBAL	0,4439	0,3677	0,3962	0,2876

Nota-se que a melhor acurácia obtida fica estatisticamente empatada entre os métodos EmoticonDS e Spark Random Forest, tendo intervalos de confiança sobrepostos (Gráfico 5.13;

Tabela 5.13). Como pior acurácia, é apontado o método NRC *Hashtag*, o que pode ser justificado pela sua composição de um *lexicon* de *hashtags* usado quase que exclusivamente para *tweets*, o que não é um conteúdo usual em revisões de produções cinematográficas no IMDB.

A melhor precisão e revocação e, por consequência, a métrica $F1$, obtidas nos experimentos de eficácia é apontada no método Spark Naive Bayes. A pior precisão é atribuída ao método SANN e a pior revocação ao empate dos métodos Emoticons, EmoticonDS e SANN. Assim, sendo a métrica $F1$ um valor harmônico entre a precisão e revocação, o melhor valor apontado para a macro-$F1$ também é atribuído à implementação do Spark Naive Bayes. Como pior resultado pela macro-$F1$, é apresentado o método NRC *Hashtag*.

Uma justificativa para a superioridade dos resultados das métricas para o método supervisionado Spark Naive Bayes é o fato de seu treinamento, afinal o resultado poderia se apresentar completamente diferente dependendo do domínio do *dataset* de entrada. Como o modelo supervisionado usado nesse método foi treinado e testado exclusivamente sob o domínio de cinema e televisão, ou seja, o mesmo domínio, isso implica um bom resultado em eficácia.

Para tentar resolver os problemas de execução dos métodos que aparecem com "–" na Tabela 5.13, foram feitas tentativas in loco, em computadores com maior desempenho (processador mais rápido e mais memória RAM) e também no *cluster*. Em nenhuma das tentativas foi obtido êxito. Demonstrado pela Tabela 5.13, os métodos que não obtiveram sucesso na execução foram removidos do Gráfico 5.13, apenas para uma melhor visualização da informação.

5.3 Considerações Sobre os Resultados Obtidos

Destacam-se na avaliação da eficiência alguns resultados, como no caso de um *dataset* pequeno demais para garantir um resultado eficiente na distribuição em um *cluster*. Para o maior dos *datasets* (*Books*), a comparação entre um PC em implementação original não distribuída e um nó BigFeel demonstrou uma maior eficiência do BigFeel em tempo de execução para vários métodos, o que de fato não se mostrou semelhante no caso de *datasets* menores, como são os casos de *Electronics* e *Movies_TV*. Essa comparação, apresentada pelos Gráficos 5.1, 5.5 e 5.9, é corroborada pela análise das Tabelas 5.1, 5.5 e 5.9, as quais apresentam todos os resultados obtidos.

Comparando-se os experimentos de *SpeedUp* nos três *datasets*, conclui-se que o ganho da distribuição da execução de métodos de análise de sentimentos em um *cluster* está proximamente ligado ao tamanho dos dados a se processar, demonstrado pela pequena diferença dos

resultados apresentados entre os *datasets Electronics* e *Movies_TV*. Para pequenos volumes de dados, o tempo de execução da análise de sentimentos com os métodos integrados ao BigFeel pode não se mostrar tão vantajoso, afinal o custo da distribuição do algoritmo e dos dados supera o ganho obtido pela paralelização do trabalho de processamento.

O Quadro 5.1 sumariza a análise da viabilidade dos métodos para grandes volumes de dados.

Quadro 5.1 – Sumário da análise de viabilidade de execução dos métodos de análise de sentimentos para grandes volumes de dados.

Método	Viável	Viável, com ressalvas	Inviável
AFINN	X		
Emolex	X		
Emoticons	X		
EmoticonDS	X		
Happiness Index	X		
MPQA	X		
NRC Hashtag	X		
Opinion Lexicon	X		
PANAS-t	X		
SANN		X	
SASA		X	
SenticNet	X		
Sentiment140 Lexicon	X		
SentiStrength	X		
SentiWordNet			X
SO-CAL			X
Stanford Recursive Deep Model			X
Umigon		X	
VADER	X		
Spark Classification by Naive Bayes	X		
Spark Classification by Random Forest	X		
Databricks Stanford			X

No Quadro 5.1 observa-se que grande parte dos métodos analisados é viável à execução em grandes volumes de dados. Como esperado, as implementações realizadas em Spark dos classificadores Naive Bayes e Random Forest também demonstraram ganhos relevantes em sua distribuição (Gráfico 5.2). Com exceção de alguns poucos métodos que, apesar de viáveis, apresentaram falhas pontuais, causadas por erros de estouro de pilha de memória, erro por uso excessivo do *Garbage Collector*, erro na manipulação de caracteres especiais, falha da própria implementação original ou integração na forma distribuída ineficaz, SANN, SASA e Umigon

apresentaram alguns resultados com comportamento semelhante aos métodos que executaram com sucesso.

Mesmo após distribuídos, alguns métodos não ofereceram ganhos relevantes em tempo de execução ou, ainda, não executaram em tempo hábil. São eles: SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford.

Embora o volume de dados a se processar não deva ser tomado como ponto exclusivo para se decidir pela distribuição ou não de uma solução, o experimento realizado neste estudo demonstra bons resultados para maiores volumes de dados, o que não implica necessariamente que, no caso de pequenos volumes de dados, não seja viável a mesma solução, devendo portanto ser analisadas métricas como o *SpeedUp* e o *ScaleUp* para a tomada de tal decisão, afinal a quantidade de processamento pode ser grande mesmo para um pequeno volume de dados.

Ainda que alguns métodos tenham ultrapassado o limite convencionado de 24 horas de execução em algumas configurações do experimento, para alguns casos, como a distribuição em oito nós, em comparação a quatro nós, a ocorrência de *SpeedUp* positivo oferece um indicador de que afirmativamente há ganho na distribuição de tais métodos. São exemplos disso os métodos: MPQA, NRC Hashtag, Sentiment140 *Lexicon*, SentiWordNet, SO-CAL, Stanford Recursive Deep Model, VADER e Databricks Stanford

O ambiente BigFeel apresenta em seu estado de desenvolvimento corrente uma escalabilidade progressiva em processamento de grandes volumes de dados, comprovada pelo *SpeedUp*, *SizeUp* e para alguns casos do *ScaleUp* do *dataset Books*. Em contrapartida, os experimentos com *datasets* menores se mostraram escaláveis, porém com resultados não tão eficientes, como percebe-se de alguns casos do *SizeUp* e *ScaleUp* para *Electronics* e *Movies_TV*.

Os experimentos de eficácia demonstram que métodos supervisionados implementados no Spark oferecem boa acurácia, precisão e macro-$F1$, o que de fato é resultado do domínio de treinamento e testes ser o mesmo. A comparação dos resultados obtidos pelos métodos em Spark com a média global (Tabela 5.13) de todos os métodos os coloca em destaque na métrica de acurácia e um pouco acima do valor médio obtido com relação à métrica macro-$F1$.

O método SANN não completou a execução em alguns experimentos. Umigon não executou de forma bem sucedida em nenhum dos experimentos de eficiência. Tais métodos foram mantidos na apresentação dos resultados devido ao fato de que são factíveis na execução de pequenas instâncias de entrada, como por exemplo os pequenos *datasets* oferecidos por Gonçalves et al. (2013).

6 ESTUDO DE CASO

Esta seção apresenta um estudo de caso que consiste em um experimento usando as ferramentas oferecidas pelo BigFeel para a criação de uma forma automatizada de identificação de sugestões de inovação por análise textual de revisões de produtos e/ou serviços.

A opinião das pessoas sobre um determinado item é essencial para o processo de inovação e contínua recriação de ofertas de produtos. Tais ideias estão presentes abundantemente em redes sociais na *web*, como fóruns, redes de relacionamento, grupos, *blogs*, dentre outros. Grandes empresas buscam meios mais eficazes de se manterem conectadas ao público consumidor, usando de estratégias de Pesquisa, Desenvolvimento e Inovação (PD&I) para se manterem competitivas no mercado (CHRISTENSEN et al., 2017a).

O problema deste estudo de caso se baseia na seguinte questão: usando de técnicas de processamento textual, processamento de linguagem natural e aprendizagem de máquina, é possível se identificar sugestões de inovação em meio a milhares ou até mesmo milhões de revisões de produtos? A hipótese adotada é que usando de propriedades sintáticas presentes no texto, usando um *lexicon* de termos associados à inovação e também de técnicas envolvidas na área de aprendizagem de máquina, é possível se obter bons resultados na identificação de sugestões de inovação.

A literatura na área de identificação de sugestões de inovação oferece algumas ideias sobre as abordagens para efetuar tal processo. Tuarob e Tucker (2015) apresentam uma abordagem para encontrar consumidores inovadores (*lead users*), dos quais as ideias podem sugerir inovações em produtos. Os autores propõem uma metodologia de mineração de dados baseada em três passos principais: 1) são extraídas as características de produtos pela sua documentação oficial e, também, são extraídas as características discutidas pelos consumidores em mídias sociais; 2) os dois conjuntos de características são comparados e, pela disjunção desses dois conjuntos, são extraídas as características candidatas a sugestão de inovação (*latent features*); 3) os usuários e suas sugestões são ordenados por um esquema de pontuação da inovação, denominado *iScore*. Os resultados de um estudo de caso conduzido pelos autores demonstram o uso de mídias sociais como potenciais fontes de informação para o desenvolvimento de novos produtos e também a viabilidade da criação de sistemas automatizados para a identificação de consumidores com ideias inovadoras.

Apesar do trabalho de Gangi, Wasko e Hooker (2010) não apresentar um método automatizado para a identificação de sugestões de inovação, seu estudo dos desafios para se melhorar

o uso de comunidades de usuários para a inovação (do inglês *User Innovation Community*) no processo de PD&I de grandes empresas, apresenta alguns exemplos de comentários de usuário a se observar, como: "***Would love** the **ability** to have a **clean** Vista install. No AOL software, no earthlink software, no Google software - just a **clean**, original OS*". As palavras destacadas em negrito denotam um aspecto característico da sugestão de inovação, o que pode ser explorado por métodos de PLN.

Outra abordagem seguida por Ulloa et al. (2016) é a criação de um *framework* para a coleta e pré-processamento textual de revisões ou comentários de usuários em mídias sociais orientados a um produto ou uma classe de produtos de uma empresa. Tais dados têm informações extraídas por processos de reconhecimento de entidades nomeadas e desambiguação de entidades. A seguir, é utilizado um método de análise de sentimentos baseado em *lexicon* para sumarizar a opinião pública sobre produtos e seus aspectos. Os autores proveem resultados para a empresa, que podem auxiliar na tomada de decisão da inovação em seus produtos baseada na opinião pública.

Christensen et al. (2017b) apresentam uma abordagem usando de um classificador SVM para identificação de sugestões de inovação em um *dataset* de revisões extraído de uma rede social na *web*, formada por consumidores de brinquedos LEGO[1]. Em um trabalho mais recente, Christensen et al. (2018) discutem a qualidade obtida por abordagens automatizadas de detecção de sugestão de inovação em produtos. Seus resultados indicam que as soluções existentes na literatura são suficientemente auxiliadoras no processo de inovação em produtos, porém ainda apresentam um grande nicho para a pesquisa.

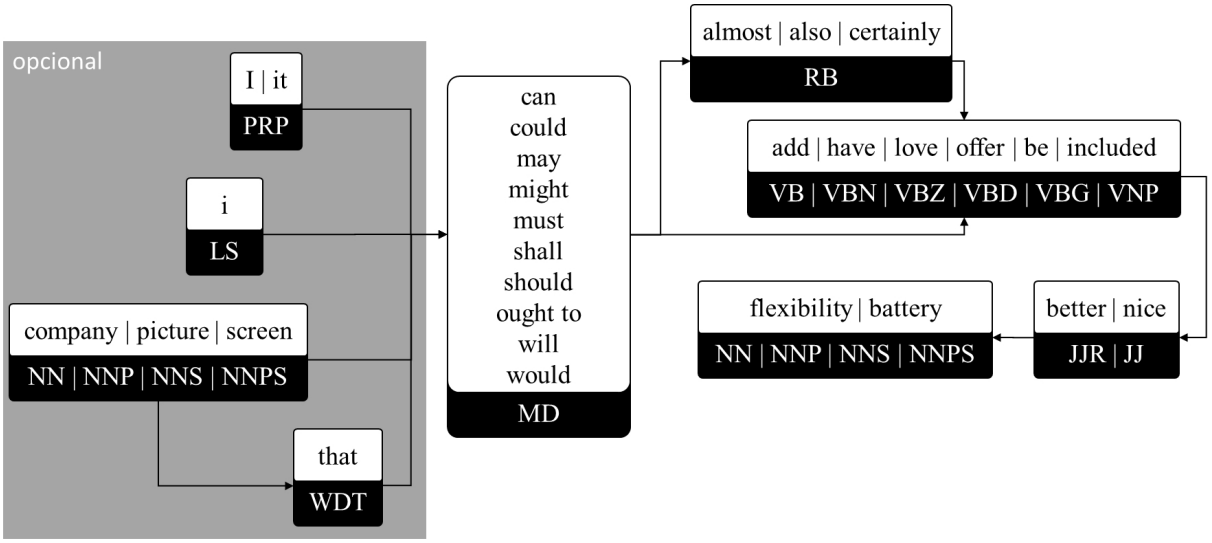
6.1 Medologia

Para a escolha do *dataset* a ser usado neste estudo de caso, foi realizada uma busca pela literatura de fontes reais rotuladas para o uso na identificação de inovação, constituídas de revisões de produtos ou serviços, porém não foram encontradas opções viáveis. Logo, optou-se pela construção de um *dataset* criado pelos próprios pesquisadores contendo 1.500 revisões sobre equipamentos eletrônicos da Amazon (MCAULEY; HE, 2016), rotuladas com as classes inovação e não-inovação, resultando 190 ocorrências de sugestão de inovação. O *dataset* construído é composto pelas sentenças na língua inglesa, idioma esse definido como alvo da pesquisa e dos experimentos.

[1] *International LEGO Users Group Network*: https://www.lugnet.com/

Após a construção do *dataset*, foi realizada uma análise de seu conteúdo para a identificação de termos e padrões sintáticos associados à sugestão de inovação. Foram selecionados termos para a construção do *lexicon* e identificados padrões sintáticos através de suas *tags* POS, como indicadores de sugestão de inovação. Dentre os termos identificados, destacam-se alguns como "*lack*", "*suggest*", "*expect*", "*perhaps*", "*hope*", "*miss*", "*prefer*" e "*want*" (termos em sua forma natural). Na análise dos padrões sintáticos, observou-se grande ocorrência de sugestão de inovação na presença de verbos modais, como "*would*", "*should*", "*might*", "*could*" e "*must*". O resultado dessa análise produziu o padrão sintático definido na Figura 6.1 e o *lexicon* demonstrado no Quadro 6.2.

Figura 6.1 – Padrão sintático indicativo de sugestões de inovação.



A Figura 6.1 apresenta que o bloco principal indicativo da presença de sugestão de inovação está ao centro do padrão sintático proposto, na presença dos verbos modais. Para cada bloco, a parte branca demonstra exemplos de termos associados à atribuição sintática na frase, identificados pelas *tags* POS (MARCUS; SANTORINI; MARCINKIEWICZ, 1993) (Quadro 6.1) nos blocos em preto. Observando as setas direcionais da figura, são formadas as frases identificadas durante a atribuição de rótulos no *dataset*, como o exemplo: "*It could certainly have better battery*". O uso do padrão sintático nesse exemplo se baseia na busca da ocorrência de sentenças com as *tags* PRP-MD-VB-JJR-NN, respectivamente: pronome - verbo modal - verbo - adjetivo - substantivo.

Para a construção do padrão sintático apresentado na Figura 6.1, alguns elementos foram ignorados, como por exemplo os termos "*of*", "*a*" e "*to*", criando-se assim uma *blacklist* de *tags* POS a se ignorar durante a análise de uma sentença. A *blacklist* final é formada pelas

tags POS: “RB”, “RBS”, “RBR”, “TO”, “DT”, “IN” e “WDT”. O lado esquerdo da Figura 6.1 apresenta que os elementos destacados dentro do painel cinza são opcionais, podendo uma sentença indicativa de sugestão de inovação começar a partir do verbo modal, ao centro, como por exemplo: “*could offer more flexibility*”.

Quadro 6.1 – Lista alfabética das *tags* POS usadas no projeto Penn Treebank.

Número	*Tag*	Descrição
1.	CC	*Coordinating conjunction*
2.	CD	*Cardinal number*
3.	DT	*Determiner*
4.	EX	*Existential there*
5.	FW	*Foreign word*
6.	IN	*Preposition or subordinating conjunction*
7.	JJ	*Adjective*
8.	JJR	*Adjective, comparative*
9.	JJS	*Adjective, superlative*
10.	LS	*List item marker*
11.	MD	*Modal*
12.	NN	*Noun, singular or mass*
13.	NNS	*Noun, plural*
14.	NNP	*Proper noun, singular*
15.	NNPS	*Proper noun, plural*
16.	PDT	*Predeterminer*
17.	POS	*Possessive ending*
18.	PRP	*Personal pronoun*
19.	PRP$	*Possessive pronoun*
20.	RB	*Adverb*
21.	RBR	*Adverb, comparative*
22.	RBS	*Adverb, superlative*
23.	RP	*Particle*
24.	SYM	*Symbol*
25.	TO	*to*
26.	UH	*Interjection*
27.	VB	*Verb, base form*
28.	VBD	*Verb, past tense*
29.	VBG	*Verb, gerund or present participle*
30.	VBN	*Verb, past participle*
31.	VBP	*Verb, non-3rd person singular present*
32.	VBZ	*Verb, 3rd person singular present*
33.	WDT	*Wh-determiner*
34.	WP	*Wh-pronoun*
35.	WP$	*Possessive wh-pronoun*
36.	WRB	*Wh-adverb*

Fonte: adaptado de Marcus, Santorini e Marcinkiewicz (1993).

Inicialmente foram incluídos no *lexicon* os verbos modais da língua inglesa. Contudo, como os mesmos já seriam utilizados pelo padrão sintático (Figura 6.1) na identificação de sugestões de inovação, eles foram removidos do *lexicon* final.

Os termos constantes no *lexicon* criado foram posteriormente adaptados pelo uso do processo de extração do *lemma*, restando portanto apenas alguns termos, conforme apresenta o Quadro 6.2.

Quadro 6.2 – Formação do *Lexicon* indicativo de sugestões de inovação.

Termos identificados	Termos no *lexicon* final
lack, lacks, lacking, lacked	lack
wish, wished, wishing	wish
suggest, suggestion, suggested	suggest
prefer, preferred	prefer
expect, expecting, expected	expect
need to, needs to, needed to	need
maybe	maybe
rather	rather
perhaps	perhaps
hope, hopefully, hoping	hope
fortunately, unfortunately	fortunate
want, wanted, wanting	want
miss, missed, missing	miss

Como método de aprendizagem de máquina, foi convencionado o uso do classificador binário SVM, o qual é fundamentado na teoria da aprendizagem estatística. Esse classificador se baseia no princípio de separação ótima entre duas classes, estabelecendo um limite de decisão usando um subconjunto dos dados de treinamento, conhecido como vetores de suporte, buscando assim um hiperplano aprendido a partir dos dados de treinamento que divide o espaço em duas regiões, uma que comporta as instâncias pertencentes a uma classe C_1 e outra que contém todas as instâncias de outra classe C_2 (BAEZA-YATES; RIBEIRO-NETO, 2011). O hiperplano aprendido então é utilizado como referência para a classificação de novas instâncias.

Como o estudo de caso proposto se baseia na classificação entre as duas classes, inovação e não-inovação, o modelo do classificador SVM pôde ser parametrizado (por processo de *tuning*) e treinado utilizando vetores de suporte construídos com base nos extratores de *features Term Frequency - Inverse Document Frequency* (TF-IDF) e Word2Vector, oferecidos pelo Spark. Posteriormente foi utilizado o seletor de *features* Qui-Quadrado para selecionar as melhores *features* do vetor final de suporte, usado no SVM.

Para avaliação do experimento de identificação de sugestões de inovação definido para este estudo de caso, foram usadas as métricas de revocação, precisão e $F1$, conforme definidas na Seção 4.2 deste trabalho.

6.2 Condução do Experimento

Esta seção aborda as configurações e os detalhes envolvidos na condução do experimento deste estudo de caso.

Como o experimento envolve a avaliação experimental de variados parâmetros e técnicas para a identificação de sugestões de inovação, nas etapas onde a execução envolveu a composição de duas ou mais técnicas, foi feito o uso de operadores lógicos OR e AND, como por exemplo: na execução nomeada de "Padrão sintático OR *Lexicon*", se algum dos resultados das técnicas apontar a sentença como sugestiva de inovação, então o resultado final será predito como inovação. No caso "Padrão sintático AND *Lexicon*", se algum dos resultados das técnicas apontar a sentença como sugestiva de não-inovação, então o resultado final será predito como não-inovação.

Da biblioteca MLLib do Spark, foram utilizados extratores, um transformador e um seletor de *features* para o pré-processamento dos dados de entrada. A construção do modelo de pré-processamento/classificação seguiu a estrutura de *Pipeline* do Spark, executada em cada conjunto de treino na seguinte ordem de passos: 1) leitura e conversão de cada conjunto de treino em *DataFrames*; 2) processo de extração do *lemma* de cada termo das sentenças analisadas, usando-se o método de Stanford CoreNLP adaptado pela Databricks (MENG, 2016); 3) cálculo do *Term Frequency* (TF), para criação de vetores de frequência dos termos presentes na sentença; 4) cálculo do *Inverse Document Frequency* (IDF). Usando-se o TF, foi calculado o TF-IDF, que intuitivamente reduz o peso de termos que aparecem frequentemente em todo o conjunto, tornando a aparição de tal termo menos significante; 5) conversão dos termos das sentenças em vetores numéricos, por meio do extrator Word2Vector, responsável por capturar a semântica das palavras, reduzindo a dimensionalidade do vetor de *features*; 6) junção dos vetores resultantes das etapas de IDF e Word2Vector, utilizando-se o transformador de *features Vector Assembler*; 7) uso do seletor de *features* Qui-Quadrado para a seleção das *n* mais significantes *features* que discernem as sentenças indicativas de sugestão de inovação, das que não são indicativas; 8) treino do modelo SVM com as *features* selecionadas e os parâmetros calibrados (SPARK, 2018c).

Para se verificar a hipótese deste estudo de caso, foi definido o seguinte procedimento para a execução do experimento de identificação de sugestões em inovação, usando do *dataset* de revisões de produtos eletrônicos com rótulos de indicação de inovações, previamente definido:

1. Divisão aleatória do *dataset* em cinco partições.

   Esse primeiro passo foi realizado por convenção em cinco partes de forma aleatória e garantindo a estratificação das partições, a qual garante que cada partição tenha aproximadamente o mesmo número de instâncias de uma determinada classe, sendo nessa divisão aproximadamente 38 sentenças indicativas de sugestão de inovação em um total de aproximadamente 300 sentenças por partição.

2. Balanceamento das partições de treinamento.

   Convencionou-se o balanceamento das partições de treinamento, mantendo-se todas as revisões com indicação de sugestão de inovação e adicionando-se aleatoriamente um número igual de revisões que não possuem indicação de inovação. Isso foi convencionado de modo a não tornar o modelo de classificação tendencioso ao subconjunto maior, no caso, o subconjunto de não indicação de inovação. As partições usadas neste estudo de caso podem ser obtidas no repositório git do projeto BigFeel.

3. Calibragem de parâmetros do SVM.

   Para a parametrização do classificador SVM, foi realizado um processo de *tuning* nos conjuntos de treinamento, usando-se de um *grid* de parâmetros avaliados sob validação cruzada em cinco partições (*folds*).

4. Validação cruzada.

   Foi utilizada a técnica de validação cruzada para medir experimentalmente os resultados. Tendo sido particionado o *dataset* em cinco partições estratificadas e balanceadas, os experimentos foram executados cinco vezes, cada uma dessas usando quatro partições para treinamento e uma partição distinta para teste.

5. Execução usando-se apenas o padrão sintático.

   a) Padrão sintático (completo): execução da identificação de sugestões de inovação utilizando o padrão sintático completo definido na Figura 6.1. Essa execução abordou o uso de todas as combinações de *tags* POS possíveis, identificadas no padrão sintático proposto. Isso implica na ideia de que uma sentença deve ter um dos caminhos completos possíveis entre um elemento inicial da esquerda dos verbos modais (opcional) até um elemento final dos nós à direita dos verbos modais para ser considerada como indicativa de sugestão de inovação.

b) Padrão sintático (leve): como alternativa à abordagem anterior, definiu-se um padrão sintático menos rigoroso, o qual permite a ocorrência de caminhos incompletos, não necessitando assim de um caminhamento completo entre os nós iniciais e finais. Os caminhos incompletos são sentenças cujas *tags* POS são formadas por três ou mais termos, necessariamente incluindo-se nesses um verbo modal.

c) Padrão sintático (analítico): execução da identificação de sugestões de inovação por uma segunda alternativa, uma versão obtida analisando-se um a um os casos do *dataset* rotulado de sugestões de inovação. Nessa alternativa, buscou-se a criação do padrão sintático baseada nos elementos sintáticos ocorridos nas sentenças das 190 revisões rotuladas como inovação, não fazendo-se assim todas as combinações possíveis entre os elementos sintáticos encontrados, o que ocorre no caso do padrão sintático completo.

6. Execução usando-se apenas o *lexicon*.

   a) *Lexicon*, incluindo os verbos modais.

   b) *Lexicon*, sem verbos modais, uma vez que os mesmos foram convencionados como os principais componentes da outra abordagem do experimento: o padrão sintático.

7. Combinação dos resultados do padrão sintático e do *lexicon*.

   a) Pelo operador lógico OR.

   b) Pelo operador lógico AND.

8. Execução usando-se o classificador SVM.
   Execução dos cinco modelos treinados com os conjuntos de treino. O resultado apresentado é a média da execução nos cinco conjuntos de teste.

9. Combinação dos resultados do classificador SVM e do padrão sintático.

   a) Pelo operador lógico OR.

   b) Pelo operador lógico AND.

10. Combinação dos resultados do classificador SVM e do *lexicon*.

a) Pelo operador lógico OR.

b) Pelo operador lógico AND.

11. Combinação dos resultados do classificador SVM, do padrão sintático e do *lexicon*.

    a) Pelo operador lógico OR.

    b) Pelo operador lógico AND.

    c) Utilizando-se, para o teste no SVM, apenas do subconjunto apontado como inovação na execução do Padrão sintático OR *Lexicon*. A escolha da combinação utilizando o operador OR entre o Padrão sintático e o *Lexicon* está baseada no melhor resultado de revocação obtido no experimento.

12. Execução da configuração experimental que levou ao melhor resultado nas execuções anteriores, no *dataset Books* (MCAULEY; HE, 2016), afim de se verificar o tempo de execução do método para grandes volumes de dados. O *dataset* rotulado de sugestões de inovação foi balanceado para o treinamento do SVM. É apresentado o valor de tempo de processamento em média de cinco execuções, juntamente com o intervalo de confiança obtido.

6.3 Resultados

Nesta subseção, são apresentados os resultados das etapas de execução definidas como metodologia do experimento deste estudo de caso. Os resultados apresentados são a média das cinco execuções, junto ao teste estatístico de intervalo de confiança de 95%. Salienta-se que a comparação sobre os resultados observa os valores das médias de cada métrica analisada. A numeração presente na primeira coluna da Tabela 6.1 possui os mesmos identificadores usados na seção anterior para orientação sobre os passos do experimento.

Tabela 6.1 – Resultados das métricas para o experimento de identificação de sugestões de inovação.

	Execução	Precisão	Revocação	Macro-$F1$
(5.a)	Padrão sintático (completo)	0,1691 ± 0,0098	0,8627 ± 0,0292	0,2760 ± 0,0144
(5.b)	Padrão sintático (leve)	0,1583 ± 0,0139	0,9468 ± 0,0203	0,2689 ± 0,0203
(5.c)	Padrão sintático (analítico)	0,1559 ± 0,0131	0,9521 ± 0,0188	0,2659 ± 0,0192
(6.a)	*Lexicon*	0,1484 ± 0,0126	**0,9947 ± 0,0062**	0,2575 ± 0,0191
(6.b)	*Lexicon* sem verbos modais	0,1807 ± 0,0153	0,9331 ± 0,0265	0,2969 ± 0,0223
(7.a)	Padrão sintático OR *Lexicon*	0,1511 ± 0,0134	**0,9922 ± 0,0099**	0,2612 ± 0,0201
(7.b)	Padrão sintático AND *Lexicon*	0,2026 ± 0,0167	0,8877 ± 0,0438	0,3170 ± 0,0247
(8)	SVM	0,2842 ± 0,0084	0,7959 ± 0,0540	**0,3653 ± 0,0268**
(9.a)	SVM OR Padrão sintático	0,1562 ± 0,0124	0,9575 ± 0,0161	0,2666 ± 0,0183
(9.b)	SVM AND Padrão sintático	**0,2802 ± 0,0252**	0,7705 ± 0,0571	**0,3519 ± 0,0415**
(10.a)	SVM OR *Lexicon*	0,1803 ± 0,0131	0,9443 ± 0,0201	0,2974 ± 0,0188
(10.b)	SVM AND *Lexicon*	**0,2871 ± 0,0265**	0,7647 ± 0,0601	**0,3534 ± 0,0446**
(11.a)	SVM OR Padrão sintático OR *Lexicon*	0,1496 ± 0,0128	**0,9947 ± 0,0062**	0,2592 ± 0,0192
(11.b)	SVM AND Padrão sintático AND *Lexicon*	**0,3032 ± 0,0210**	0,7593 ± 0,0650	**0,3515 ± 0,0506**
(11.c)	(Padrão sintático OR *Lexicon*) → SVM	**0,3184 ± 0,0134**	0,8153 ± 0,0114	**0,3987 ± 0,0257**

A Tabela 6.1 apresenta a média das métricas resultantes do experimento de identificação de sugestões de inovação, destacando-se em negrito os melhores resultados obtidos. Os vários resultados em negrito para uma mesma coluna indicam um empate estatístico entre a melhor média encontrada no experimento e o valor obtido para uma determinada execução, considerando-se seu intervalo de confiança.

Utilizando-se apenas o padrão sintático definido, a primeira execução obteve alta revocação e baixa precisão (Tabela 6.1 (5.a)). A alternativa de um padrão sintático menos rigoroso apresentou uma pequena perda em precisão, mas elevou a revocação obtida da execução anterior (Tabela 6.1 (5.b)). Foi elaborada ainda uma segunda alternativa ao padrão sintático (Tabela 6.1 (5.c)), a qual eleva um pouco mais a revocação média, porém mantendo baixa a precisão média.

Até esse ponto do experimento, foram obtidas a melhor revocação e a menor precisão pela primeira execução com o *lexicon* (Tabela 6.1 (6.a)). A remoção dos verbos modais do *lexicon* resultou uma singela melhoria na precisão, implicando aproximadamente a perda de 7% em revocação média e o ganho de 5% na macro-$F1$ (Tabela 6.1 (6.b)).

Para a combinação entre o padrão sintático e o *lexicon*, o uso do operador lógico "OR" apresentou a melhor revocação encontrada até o momento e a segunda precisão mais baixa. O operador "AND" apresentou a maior precisão encontrada até então e a segunda pior revocação (Tabela 6.1 (7.a) e (7.b)).

A primeira execução do classificador SVM apresentou baixa revocação e a mais alta precisão, obtendo como consequência a melhor macro-$F1$ até então encontrada (Tabela 6.1 (8)).

A combinação entre a execução do SVM e o padrão sintático, utilizando-se o operador lógico "OR" (Tabela 6.1 (9.a)), elevou a revocação obtida pelo SVM, mas decaiu consideravelmente sua precisão. O mesmo experimento com o operador lógico "AND" (Tabela 6.1 (9.b)) resultou em uma singela perda, tanto em precisão quanto revocação, da execução anterior.

A combinação entre a execução do SVM e o *lexicon*, utilizando-se o operador lógico "OR" (Tabela 6.1 (10.a)), resultou em alta revocação, porém baixa precisão. Com o operador lógico "AND" (Tabela 6.1 (10.b)), elevou-se a precisão ao mais alto valor até então encontrado, porém com perda na revocação.

As execuções da Tabela 6.1 (11.a) e (11.b) são a combinação entre todas as abordagens do experimento. O operador lógico "OR" obteve a segunda pior precisão de todo o experimento e a melhor revocação (empatada estatisticamente com a revocação dos experimentos (7.a) e (6.a)). Por outro lado, o operador lógico "AND" obteve a melhor precisão encontrada até então no experimento e uma revocação baixa (comparada aos melhores valores encontrados).

Experimentou-se ainda uma tentativa de se aproveitar da melhor revocação e se incrementar a precisão, usando-se o resultado do Padrão sintático OR *lexicon* (Tabela 6.1 (7.a)) e, para os casos onde foi atribuído o valor de inovação, classificar via SVM (Tabela 6.1 (11.c)). Como resultado, foi obtida a melhor precisão de todo o experimento e uma singela revocação, porém a macro-$F1$ foi a melhor obtido em todo o experimento.

Ressalta-se nos resultados apresentados para este estudo de caso, que o *lexicon* e o padrão sintático foram extraídos do próprio *dataset* no qual foram realizados os testes. Portanto, tais resultados estão maximizados, tornando provável um resultado inferior se usadas as mesmas abordagens em outro *dataset*.

Para se demonstrar a viabilidade em tempo de processamento da solução experimental proposta neste estudo de caso, foi realizada a execução da configuração do melhor resultado obtido (Tabela 6.1 (11.c)) no maior *dataset* usado neste trabalho, o *Books*. O resultado médio em tempo de processamento para cinco execuções é de aproximadamente 7.635,37 $\pm$ 14,46 segundos para se processar todas as 22,5 milhões de revisões em oito nós de processamento. Compõem esse tempo total os valores aproximados de 3.483,19 $\pm$ 11,06 segundos para a extração dos *lemmas*, 1.078,75 $\pm$ 8,75 segundos para o POS *tagger*, 222,60 $\pm$ 2,33 segundos para a execução da identificação pelo *lexicon*, 551,23 $\pm$ 5,24 segundos para a execução da identificação pelo padrão sintático e 2.362,91 $\pm$ 16,51 segundos para a execução do classificador SVM.

7 CONCLUSÕES

Neste trabalho, foi proposto e avaliado experimentalmente um ambiente que permite a experimentação variada em *Big Data* de métodos para a análise de sentimentos, processamento de linguagem natural e pré-processamento textual, denominado BigFeel. Tal ambiente foi implementado de forma distribuída para processar grandes volumes de dados.

Apesar da literatura apresentar variados métodos para a análise de sentimentos, abordando técnicas de aprendizagem de máquina, de dicionários e abordagens híbridas, existem estudos exaustivamente conduzidos para se demonstrar a eficácia de tais métodos, mas não sua eficiência. Conforme a análise da Seção 5.3, conclui-se portanto que alguns dos métodos de análise de sentimentos existentes atualmente não são eficientes para processar grandes volumes de dados, como SentiWordNet, SO-CAL, Stanford *Recursive Deep Model* e Databricks Stanford. Para os demais métodos, o uso de abordagens distribuídas, como Apache Hadoop e Spark implica grande ganho computacional em sua eficiência.

Embora o maior foco dos experimentos deste trabalho esteja na eficiência dos métodos em grandes volumes de dados, sua eficácia também foi avaliada para permitir a comparação com os métodos supervisionados implementados em Spark, disponíveis em sua biblioteca MLLib, dos quais os dois exemplos implementados neste estudo, Naive Bayes e Random Forest, obtiveram bons resultados.

O estudo de caso apresentado demonstra o potencial das abordagens de uso de padrões sintáticos e *lexicons* na identificação de sugestões de inovações em revisões de produtos ou serviços. Apesar do resultado não expressar uma boa precisão, a revocação das abordagens do padrão sintático e do *lexicon* se mostrou promissora, indicando que novos experimentos podem melhorar a precisão encontrada neste estudo de caso. A execução da solução final no *cluster* demonstra a viabilidade de se processar grandes volumes de dados, aplicando assim o BigFeel a um exemplo prático de uso: um mecanismo automatizado para identificação de sugestões de inovação em grandes volumes de dados de revisões de produtos e/ou serviços.

As principais contribuições deste trabalho se resumem a: (i) apresentação do ambiente BigFeel como um ambiente distribuído de código-fonte aberto com oferta de APIs, que integra e possibilita outras integrações de métodos de análise de sentimentos em grandes volumes de dados; (ii) apresentação de conceitos necessários à solução de problemas para a adaptação de métodos locais à execução distribuída, utilizando de uma estrutura baseada em Apache Spark e Apache Hadoop HDFS. (iii) Análise e discussão da eficiência adquirida pela avaliação dos

resultados experimentais sob os 22 métodos integrados de análise de sentimentos e comparação da eficácia dos métodos integrados em face aos métodos implementados em Spark. (iv) apresentação de um estudo de caso para a identificação automatizada de sugestões de inovação em produtos e/ou serviços como solução escalável.

Como trabalhos futuros, pretende-se realizar nova implementação dos métodos de forma nativa no Spark, para os casos onde o código-fonte é aberto. Outra proposta de continuidade é a adição de métodos específicos para outros idiomas, como o português - Brasil. Também pretende-se adicionar métodos de análise de sentimentos baseados em aspectos, além da intenção de se integrar métodos responsáveis pela coleta/mineração de variadas fontes, possibilitando ao BigFeel, por exemplo, oferecer serviços de experimentação de tempo real em ferramentas como Twitter e Facebook. Quanto ao estudo de caso, ainda há espaço para pesquisa e busca de soluções que possam incrementar de forma efetiva a precisão do resultado obtido.

REFERÊNCIAS

ALMEIDA, F. **Creating Scala fat jars for Spark on SBT with sbt-assembly plugin**. QUEIROZF.COM, 2017. Disponível em: <http://queirozf.com/entries/creating-scala-fat-jars-for-spark-on-sbt-with-sbt-assembly-plugin#spark2>. Acesso em: jun. 2018.

ASNIKA, S.; VASUDEVA, P.; SUDHINDRA, R. Restful web services. **International Journal of Advanced Information Science and Technology (IJAIST)**, v. 24, n. 24, p. 46–50, 4 2014. ISSN 2319:2682.

BACCIANELLA, S.; ESULI, A.; SEBASTIANI, F. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In: **Proceedings of the International Conference on Language Rsources and Evaluation (LREC)**). Valletta, Malta: European Language Resources Association, 2010. v. 10. ISBN 2-9517408-6-7. Disponível em: <http://nmis.isti.cnr.it/sebastiani/Publications/LREC10.pdf>. Acesso em: jun. 2018.

BACON, D. F.; KOGAN, E.; LLOYD, A.; MELNIK, S.; RAO, R.; SHUE, D.; TAYLOR, C.; HOLST, M. van der; WOODFORD, D.; BALES, N.; BRUNO, N.; COOPER, B. F.; DICKINSON, A.; FIKES, A.; FRASER, C.; GUBAREV, A.; JOSHI, M. Spanner: Becoming a sql system. In: **Proceedings of the 2017 ACM International Conference on Management of Data - SIGMOD '17**. New York, New York, USA: ACM Press, 2017. p. 331–343. ISBN 9781450341974. ISSN 07308078. Disponível em: <http://dl.acm.org/citation.cfm?doid=3035918.3056103>. Acesso em: jun. 2018.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern Information Retrieval: The Concepts and Technology Behind Search**. Harlow, England: Addison Wesley, 2011. ISBN 9780321416919.

BAKIROV, A.; ÇOĞALMIŞ, K. N.; BULUT, A. Scalable sentiment analytics. **Turkish Journal of Electrical Engineering and Computer Sciences**, TÜBİTAK, v. 24, n. 3, p. 1560–1570, 3 2016. ISSN 13000632. Disponível em: <http://online.journals.tubitak.gov.tr/openDoiPdf.htm?mKodu=elk-1311-128>. Acesso em: jun. 2018.

BALDOMINOS, A.; ALBACETE, E.; SAEZ, Y.; ISASI, P. A scalable machine learning online service for big data real-time analysis. In: **IEEE Symposium on Computational Intelligence in Big Data (CIBD)**. Orlando, Florida, USA: Institute of Electrical and Electronics Engineers, 2014. p. 1–8. ISBN 978-1-4799-4540-5. Disponível em: <http://ieeexplore.ieee.org/document/7011537/>. Acesso em: jun. 2018.

BANERVELD, M. van; LE-KHAC, N.; M., K. T. Performance evaluation of a natural language processing approach applied in white collar crime investigation. In: DANG, T. K.; THOAI, N. (Ed.). **Future Data and Security Engineering - First International Conference (FDSE 2014)**. Ho Chi Minh City: Springer International Publishing, 2014. p. 29–57. ISBN 9783319127774. Disponível em: <https://www.ebook.de/de/product/22861014/future_data_and_security_engineering.html>. Acesso em: jun. 2018.

BORSOS, D. **Deploy Spark with an Apache Cassandra cluster**. OpenCredo, 2017. Disponível em: <https://opencredo.com/deploy-spark-apache-cassandra/>. Acesso em: jun. 2018.

BORTH, D.; JI, R.; CHEN, T.; BREUEL, T.; CHANG, S.-F. Large-scale visual sentiment ontology and detectors using adjective noun pairs. In: **Proceedings of the 21st ACM international conference on Multimedia**. Barcelona, Espanha: Association for Computing Machinery (ACM), 2013. p. 223–232. ISBN 9781450324045. Disponível em: <http://dx.doi.org/10.1145/2502081.2502282>. Acesso em: jun. 2018.

BURDORF, C. **A Distributed Sentiment Analysis Development Environment**. 2016. Big Data and Analytical Data Platforms - Articles, ODBMS.org - Operational Database Management Systems, http://www.odbms.org/2016/04/a-distributed-sentiment-analysis-development-environment/. Disponível em: <http://www.odbms.org/2016/04/a-distributed-sentiment-analysis-development-environment/>. Acesso em: jun. 2018.

CAMBRIA, E. Affective computing and sentiment analysis. **IEEE Intelligent Systems**, Institute of Electrical and Electronics Engineers (IEEE), v. 31, n. 2, p. 102–107, 3 2016. Disponível em: <http://dx.doi.org/10.1109/MIS.2016.31>. Acesso em: jun. 2018.

CAMBRIA, E.; OLSHER, D.; RAJAGOPAL, D. SenticNet 3: a common and common-sense knowledge base for cognition-driven sentiment analysis. In: **Proceedings of the 28th AAAI Conference on Artificial Intelligence (AAAI)**. Québec, Canada: Association for the Advancement of Artificial Intelligence, 2014. p. 1515–1521. ISBN 9781577356783. Disponível em: <https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8479/8602>. Acesso em: jun. 2018.

CHEN, H.; SUN, M.; TU, C.; LIN, Y.; LIU, Z. Neural sentiment classification with user and product attention. In: **Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Austin, Texas, USA: Association for Computational Linguistics, 2016. p. 1650–1659. Disponível em: <http://www.thunlp.org/~chm/>. Acesso em: jun. 2018.

CHRISTENSEN, K.; LILAND, K. H.; KVAAL, K.; RISVIK, E.; BIANCOLILLO, A.; SCHOLDERER, J.; NØRSKOV, S.; NÆS, T. Mining online community data: The nature of ideas in online communities. **Food Quality and Preference**, Elsevier, v. 62, p. 246–256, 12 2017. ISSN 09503293. Disponível em: <http://dx.doi.org/10.1016/j.foodqual.2017.06.001>. Acesso em: jun. 2018.

CHRISTENSEN, K.; NØRSKOV, S.; FREDERIKSEN, L.; SCHOLDERER, J. In Search of New Product Ideas: Identifying Ideas in Online Communities by Machine Learning and Text Mining. **Creativity and Innovation Management**, Wiley, v. 26, n. 1, p. 17–30, 12 2017. ISSN 14678691. Acesso em: jun. 2018.

CHRISTENSEN, K.; SCHOLDERER, J.; HERSLETH, S. A.; NÆS, T.; KVAAL, K.; MOLLESTAD, T.; VEFLEN, N.; RISVIK, E. How good are ideas identified by an automatic idea detection system? **Creativity and Innovation Management**, Wiley, v. 27, n. 1, p. 23–31, 2 2018. ISSN 14678691. Acesso em: jun. 2018.

CLOUDERA. **Cloudera CDH components**. Cloudera, 2018. Disponível em: <https://www.cloudera.com/products/open-source/apache-hadoop/key-cdh-components.html>. Acesso em: jun. 2018.

DASGUPTA, S. S.; NATARAJAN, S.; KAIPA, K. K.; BHATTACHERJEE, S. K.; VISWANATHAN, A. Sentiment analysis of Facebook data using Hadoop based open

source technologies. In: **Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)**. Paris, França: Institute of Electrical and Electronics Engineers (IEEE), 2015. p. 1–3. ISBN 9781467382731. Disponível em: <https://ieeexplore.ieee.org/document/7344883/>. Acesso em: jun. 2018.

DATABRICKS. **Lauch cloud-optimized Apache Spark clusters in minutes**. Databricks, 2018. Disponível em: <https://databricks.com/try-databricks>. Acesso em: jun. 2018.

DODDS, P. S.; DANFORTH, C. M. Measuring the happiness of large-scale written expression: songs, blogs, and presidents. **Journal of Happiness Studies**, v. 11, n. 4, p. 441–456, 8 2010. ISSN 1389-4978. Disponível em: <http://link.springer.com/10.1007/s10902-009-9150-9>. Acesso em: jun. 2018.

EDER, L. **The 10 most annoying things about Java after using Scala.** Jaxenter, 2014. Disponível em: <https://jaxenter.com/the-10-most-annoying-things-about-java-after-using-scala-108012.html>. Acesso em: jun. 2018.

ESULI, A.; SEBASTIANI, F. SENTIWORDNET: a publicly available lexical resource for opinion mining. In: **Proceedings of the 5th Conference on Language Resources and Evaluation (LREC)**. Génova, Itália: European Language Resources Association (ELRA), 2006. p. 417–422. ISBN 2-9517408-2-4. ISSN 09255273. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.7217>. Acesso em: jun. 2018.

Expedia SEA. **With Expedia, you're already there!** YouTube, 2017. Disponível em: <https://www.youtube.com/watch?v=vHug0CEA7fI>. Acesso em: jun. 2018.

FAN, W.; BIFET, A. Mining Big Data. **ACM SIGKDD Explorations Newsletter**, Association for Computing Machinery (ACM), New York, USA, v. 14, n. 2, p. 1–5, 4 2013. ISSN 19310145. Disponível em: <https://dl.acm.org/citation.cfm?doid=2481244.2481246>. Acesso em: jun. 2018.

FELDMAN, R. Techniques and applications for sentiment analysis. **Communications of the ACM**, Association for Computing Machinery (ACM), New York, USA, v. 56, n. 4, p. 82–89, 4 2013. ISSN 00010782. Disponível em: <https://dl.acm.org/citation.cfm?doid=2436256.2436274>. Acesso em: jun. 2018.

FERRARO, N. **Using non-serializable objects in Apache Spark**. 2016. Disponível em: <https://www.nicolaferraro.me/2016/02/22/using-non-serializable-objects-in-apache-spark/>. Acesso em: jun. 2018.

GANGI, P. D.; WASKO, M.; HOOKER, R. Getting customers' ideas to work for you: Learning from dell how to succeed with online user innovation communities. **MIS Quarterly Executive**, University of Minnesota, v. 9, n. 4, p. 213–228, 12 2010. Disponível em: <https://www.researchgate.net/publication/220500534_Getting_Customers'_Ideas_to_Work_for_You_Learning_from_Dell_how_to_Succeed_with_Online_User_Innovation_Communities>. Acesso em: jun. 2018.

GAUTAM, G.; YADAV, D. Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In: **Seventh International Conference on Contemporary Computing (IC3)**. Noida, Índia: Institute of Electrical & Electronics Engineers (IEEE), 2014.

p. 437–442. ISBN 978-1-4799-5173-4. Disponível em: <http://dx.doi.org/10.1109/IC3.2014.6897213>. Acesso em: jun. 2018.

GO, A.; BHAYANI, R.; HUANG, L. Twitter sentiment classification using Distant Supervision. **Processing**, v. 150, n. 12, p. 1–6, 2009. ISSN 00370738. Disponível em: <https://www.researchgate.net/publication/228523135_Twitter_sentiment_classification_using_distant_supervision>. Acesso em: jun. 2018.

GOLE, S.; TIDKE, B. A survey of Big Data in social media using data mining techniques. In: **2015 International Conference on Advanced Computing and Communication Systems**. Coimbatore, Índia: Institute of Electrical and Electronics Engineers (IEEE), 2015. p. 1–6. ISBN 978-1-4799-5173-4. Disponível em: <http://dx.doi.org/10.1109/ICACCS.2015.7324059>. Acesso em: jun. 2018.

GONÇALVES, P.; ARAÚJO, M.; BENEVENUTO, F.; CHA, M. Comparing and combining sentiment analysis methods. In: **Proceedings of the first ACM Conference on online social networks (COSN)**. New York, New York, USA: ACM Press, 2013. p. 27–38. ISBN 9781450320849. Disponível em: <http://dl.acm.org/citation.cfm?doid=2512938.2512951>. Acesso em: jun. 2018.

GONÇALVES, P.; BENEVENUTO, F.; CHA, M. PANAS-t: A pychometric scale for measuring sentiments on twitter. **Computing Research Repository (CoRR)**, v. 8, 2013. Disponível em: <http://arxiv.org/abs/1308.1857>. Acesso em: jun. 2018.

GUPTA, B. **10 Hadoop alternatives that you should consider for Big Data**. Analytics India Magazine, 2017. Disponível em: <https://analyticsindiamag.com/10-hadoop-alternatives-consider-big-data/>. Acesso em: jun. 2018.

HADOOP. **Hadoop Homepage**. Apache Hadoop, 2018. Disponível em: <https://hadoop.apache.org>. Acesso em: jun. 2018.

HAI, M.; ZHANG, Y.; ZHANG, Y. A performance evaluation of classification algorithms for Big Data. **Procedia Computer Science**, Elsevier B.V., v. 122, p. 1100–1107, 2017. ISSN 18770509. Disponível em: <https://doi.org/10.1016/j.procs.2017.11.479>. Acesso em: jun. 2018.

HANNAK, A.; ANDERSON, E.; BARRETT, L. F.; LEHMANN, S.; MISLOVE, A.; RIEDEWALD, M. Tweetin' in the rain: exploring societal-scale effects of weather on mood. In: **Proceedings of the Sixth International AAAI Conference on Weblogs and Social Media**. Dublin, Irlanda: Association for the Advancement of Artificial Intelligence, 2012. p. 479–482. ISBN 9781577355564. Acesso em: jun. 2018.

HARRIS, D. **Because Hadoop isn't perfect: 8 ways to replace HDFS**. GIGAOM, 2012. Disponível em: <https://gigaom.com/2012/07/11/because-hadoop-isnt-perfect-8-ways-to-replace-hdfs/>. Acesso em: jun. 2018.

HOLMES, A. **Hadoop in practice**. Shelter Island, N.Y.: Manning, 2015. ISBN 1617292222 9781617292224.

HU, H. **How we optimize Apache Spark jobs**. RealeState, 2017. Disponível em: <https://rea.tech/how-we-optimize-apache-spark-apps/>. Acesso em: jun. 2018.

HU, M.; LIU, B. Mining and summarizing customer reviews. In: **Proceedings of the 2004 ACM SIGKDD International Conference on Knowledge discovery and data mining (KDD)**. Seattle, Washington, USA.: Association for Computing Machinery (ACM), 2004. p. 168–177. ISBN 1581138881. Disponível em: <http://dx.doi.org/10.1145/1014052.1014073>. Acesso em: jun. 2018.

HUTTO, C. J.; GILBERT, E. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In: **The 8th International AAAI Conference on Weblogs and Social Media**. Michigan, USA: AAAI Publications, 2014. p. 216–225. ISBN 9781577356578. Disponível em: <http:/comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf>. Acesso em: jun. 2018.

KANAVOS, A.; NODARAKIS, N.; SIOUTAS, S.; TSAKALIDIS, A.; TSOLIS, D.; TZIMAS, G. Large scale implementations for Twitter sentiment classification. **Algorithms**, MDPI AG, v. 10, n. 1, p. 1–21, 3 2017. ISSN 19994893. Disponível em: <http://www.mdpi.com/1999-4893/10/1/33>. Acesso em: jun. 2018.

KARAU, H.; KONWINSKI, A.; WENDELL, P.; ZAHARIA, M. **Learning Spark: lightning-fast Big Data analytics**. 1. ed. Sebastopol, CA, USA: O'Reilly Media, Inc., 2015. ISBN 978-1-449-35862-4.

KARAU, H.; WARREN, R. **High performance Spark**. 1. ed. Sebastopol, CA, USA: O'Reilly Media, Inc, 2017. 356 p. ISSN 0954-0083. ISBN 9781491943199.

KHAN, F. H.; BASHIR, S.; QAMAR, U. TOM: Twitter opinion mining framework using hybrid classification scheme. **Decision Support Systems**, Elsevier BV, v. 57, n. 1, p. 245–257, 1 2014. ISSN 01679236. Disponível em: <http://dx.doi.org/10.1016/j.dss.2013.09.004>. Acesso em: jun. 2018.

KIM, S.; HOVY, E. Identifying and analyzing judgment opinions. In: **Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics**. Nova York, Nova York: Association for Computational Linguistics (ACL), 2006. p. 200–207. Disponível em: <http://dx.doi.org/10.3115/1220835.1220861>. Acesso em: jun. 2018.

KOEHLER, J.; SRIVASTAVA, B. Web service composition: Current solutions and open problems. In: **The 13th International Conference on Automated Planning & Scheduling (ICAPS 2003) workshop on Planning for Web Services**. Trento, Itália: American Association for Artificial Intelligence, 2003. p. 28–35. Disponível em: <https://www.researchgate.net/publication/2905650_Web_Service_Composition_-_Current_Solutions_and_Open_Problems>. Acesso em: jun. 2018.

LASKOWSKI, J. **Mastering Spark SQL**. GitBook, 2018. Disponível em: <https://legacy.gitbook.com/book/jaceklaskowski/mastering-spark-sql/details>. Acesso em: jun. 2018.

LEVALLOIS, C. Umigon : sentiment analysis for tweets based on lexicons and heuristics. In: **Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)**. Atlanta, Georgia, USA: Association for Computational Linguistics, 2013. v. 2, n. 2, p. 414–417. ISBN 9781937284497. Disponível em: <http://www.aclweb.org/anthology/S13-2068>. Acesso em: jun. 2018.

LIN, C. D. J. **Data-Intensive Text Processing with MapReduce**. 1. ed. Morgan & Claypool Publishers, 2010. ISBN 9781608453429. Disponível em: <https://doi.org/10.2200/S00274ED1V01Y201006HLT007>. Acesso em: jun. 2018.

LIU, B. **Opinion mining and sentiment analysis**. 2. ed. Berlin, Alemanha: Springer Berlin Heidelberg, 2011. 459–526 p. ISBN 978-3-642-19460-3. Disponível em: <http://dx.doi.org/10.1007/978-3-642-19460-3_11>. Acesso em: jun. 2018.

LIU, B. Sentiment analysis and opinion mining. **Synthesis Lectures on Human Language Technologies**, Morgan & Claypool Publishers LLC, v. 5, n. 1, p. 1–167, 5 2012. ISSN 1947-4040. Disponível em: <http://dx.doi.org/10.2200/S00416ED1V01Y201204HLT016>. Acesso em: jun. 2018.

MANNING, C. D.; SURDEANU, M.; BAUER, J.; FINKEL, J.; BETHARD, S. J.; MCCLOSKY, D. The Stanford CoreNLP natural language processing toolkit. In: **Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations**. Baltimore, Maryland, USA: Association for Computational Linguistics, 2014. p. 55–60. Disponível em: <http://www.aclweb.org/anthology/P14-5010>. Acesso em: jun. 2018.

MARCUS, M. P.; SANTORINI, B.; MARCINKIEWICZ, M. A. Building a large annotated corpus of English: The Penn Treebank. **Computational Linguistics**, MIT Press - Journals, v. 19, n. 2, p. 313–330, 3 1993. ISSN 08912017. Disponível em: <http://aclweb.org/anthology/J93-2004>. Acesso em: jun. 2018.

MCAULEY, J.; HE, R. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: **Proceedings of the 25th International Conference on World Wide Web**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2016. (WWW '16), p. 507–517. ISBN 978-1-4503-4143-1. Disponível em: <https://dl.acm.org/citation.cfm?doid=2872427.2883037>. Acesso em: jun. 2018.

MEDHAT, W.; HASSAN, A.; KORASHY, H. Sentiment analysis algorithms and applications: a survey. **Ain Shams Engineering Journal**, Elsevier BV, v. 5, n. 4, p. 1093–1113, 12 2014. ISSN 20904479. Disponível em: <https://doi.org/10.1016/j.asej.2014.04.011>. Acesso em: jun. 2018.

MENG, X. **Stanford CoreNLP wrapper for Apache Spark**. GitHub, 2016. Disponível em: <https://github.com/databricks/spark-corenlp>. Acesso em: jun. 2018.

MESLEH, A. M. A. Chi square feature extraction based svms arabic language text categorization system. **Journal of Computer Science**, Science Publications, v. 3, n. 6, p. 430–435, 6 2007. ISSN 15493636. Disponível em: <http://dx.doi.org/10.3844/jcssp.2007.430.435>. Acesso em: jun. 2018.

MOHAMMAD, S.; KIRITCHENKO, S.; ZHU, X. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In: **Proceedings of the seventh international workshop on semantic evaluation exercises (SemEval)**. Ottawa, Ontario, Canada: Science Publications, 2013. p. 7. Disponível em: <https://www.researchgate.net/publication/256187019_NRC-Canada_Building_the_State-of-the-Art_in_Sentiment_Analysis_of_Tweets>. Acesso em: jun. 2018.

MOHAMMAD, S. M. #emotional tweets. In: **Proceedings of the First Joint Conference on Lexical and Computational Semantics - Volume 1: Proceedings of the Main Conference and the Shared Task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval)**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. p. 246–255. Disponível em: <http://dl.acm.org/citation.cfm?id=2387636.2387676>. Acesso em: jun. 2018.

MOHAMMAD, S. M.; TURNEY, P. D. Crowdsourcing a word-emotion association lexicon. **Computational Intelligence**, v. 29, n. 3, p. 436–465, 6 2013. ISSN 08247935. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.2012.00460.x>. Acesso em: jun. 2018.

MORITZ, P.; NISHIHARA, R.; STOICA, I.; JORDAN, M. I. SparkNet: training deep networks in Spark. In: **Proceedings of 4th International Conference on Learning Representations (ICLR2016)**. San Juan, Puerto Rico: [s.n.], 2016. p. 1–12. ISBN 1601324480. Disponível em: <http://arxiv.org/abs/1511.06051>. Acesso em: jun. 2018.

NIELSEN, F. Å. A new ANEW: evaluation of a word list for sentiment analysis in microblogs. **Computing Research Repository (CoRR)**, abs/1103.2903, 2011. Disponível em: <https://arxiv.org/abs/1103.2903>. Acesso em: jun. 2018.

NLTK. **Natual Language Toolkit**. NLTK Project, 2018. Disponível em: <http://www.nltk.org/>. Acesso em: jun. 2018.

NODARAKIS, N.; SIOUTAS, S.; TSAKALIDIS, A. K.; TZIMAS, G. Large Scale Sentiment Analysis on Twitter with Spark. In: **Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference**. Bordeaux, França: EDBT/ICDT, 2016. v. 15. ISSN 16130073. Disponível em: <http://ceur-ws.org/Vol-1558/paper41.pdf>. Acesso em: jun. 2018.

ONETO, L.; BISIO, F.; CAMBRIA, E.; ANGUITA, D. Statistical learning theory and ELM for big social data analysis. **IEEE Computational Intelligence Magazine**, Institute of Electrical and Electronics Engineers (IEEE), v. 11, n. 3, p. 45–55, 8 2016. ISSN 1556603X. Disponível em: <https://ieeexplore.ieee.org/document/7515290/>. Acesso em: jun. 2018.

OTWELL, T. **Laravel: the framework for web artisans**. Laravel, 2018. Disponível em: <http://www.laravel.com>. Acesso em: jun. 2018.

PAGE, R. Saying 'sorry': Corporate apologies posted on Twitter. **Journal of Pragmatics**, Elsevier BV, v. 62, n. 8, p. 30–45, 2 2014. ISSN 03782166. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0378216613003317>. Acesso em: jun. 2018.

PANG, B.; LEE, L. Opinion mining and sentiment analysis. **Foundations and Trends in Information Retrieval**, Now Publishers Inc., Hanover, MA, USA, v. 2, n. 1-2, p. 1–135, 1 2008. ISSN 1554-0669. Disponível em: <http://dx.doi.org/10.1561/1500000011>. Acesso em: jun. 2018.

PAPPAS, N.; POPESCU-BELIS, A. Sentiment analysis of user comments for one-class collaborative filtering over Ted talks. In: **Proceedings of the 36th International ACM Conference on Research and Development in Information Retrieval (SIGIR)**. New York, NY, USA: ACM, 2013. p. 773–776. ISBN 978-1-4503-2034-4. Disponível em: <http://doi.acm.org/10.1145/2484028.2484116>. Acesso em: jun. 2018.

PLUTCHIK, R. Chapter 1 - a general psychoevolutionary theory of emotion. In: PLUTCHIK, R.; KELLERMAN, H. (Ed.). **Theories of Emotion**. Academic Press, 1980. p. 3 – 33. ISBN 978-0-12-558701-3. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780125587013500077>. Acesso em: jun. 2018.

RAJURKAR, G. D.; GOUDAR, R. M. A speedy data uploading approach for Twitter trend and sentiment analysis using Hadoop. In: **Proceedings of the 2015 International Conference on Computing Communication Control and Automation**. Pune, Índia: Institute of Electrical and Electronics Engineers (IEEE), 2015. p. 580–584. ISBN 9781479968923. Disponível em: <https://ieeexplore.ieee.org/document/7155914/>. Acesso em: jun. 2018.

RANGANATHAN, J.; IRUDAYARAJ, A. S.; TZACHEVA, A. A. Action rules for sentiment analysis on Twitter data using Spark. In: **Proceedings of 2017 IEEE International Conference on Data Mining Workshops (ICDMW)**. New Orleans, LA, USA: IEEE, 2017. p. 51–60. ISBN 978-1-5386-3800-2. Disponível em: <http://ieeexplore.ieee.org/document/8215644/>. Acesso em: jun. 2018.

RAO, J.; SU, X. A survey of automated web service composition methods. In: **Proceedings of the First International Conference on Semantic Web Services and Web Process Composition**. Berlin, Heidelberg: Springer-Verlag, 2005. (SWSWPC'04), p. 43–54. ISBN 3-540-24328-3, 978-3-540-24328-1. Disponível em: <http://dx.doi.org/10.1007/978-3-540-30581-1_5>. Acesso em: jun. 2018.

RAVI, K.; RAVI, V. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. **Knowledge-Based Systems**, Elsevier BV, v. 89, p. 14–46, 6 2015. ISSN 09507051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2015.06.015>. Acesso em: jun. 2018.

RIBEIRO, F. N.; ARAÚJO, M.; GONÇALVES, P.; André Gonçalves, M.; BENEVENUTO, F. SentiBench - a benchmark comparison of state-of-the-practice sentiment analysis methods. **EPJ Data Science**, v. 5, n. 1, p. 23, 12 2016. ISSN 2193-1127. Disponível em: <https://epjdatascience.springeropen.com/articles/10.1140/epjds/s13688-016-0085-1>. Acesso em: jun. 2018.

RICHARDSON, L.; RUBY, S. **Restful Web Services**. 1. ed. Gravenstein Highway North, Sebastopol: O'Reilly Media, Inc., 2007. 446 p. ISBN 9780596529260.

SHEELA, L. J. A review of sentiment analysis in Twitter data using Hadoop. **International Journal of Database Theory and Application (IJDTA)**, Science and Engineering Research Support Society, v. 9, n. 1, p. 77–86, 1 2016. ISSN 20054270. Disponível em: <http://dx.doi.org/10.14257/ijdta.2016.9.1.07>. Acesso em: jun. 2018.

SHI, J.; QIU, Y.; MINHAS, U. F.; JIAO, L.; WANG, C.; REINWALD, B.; ÖZCAN, F. Clash of the titans. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 8, n. 13, p. 2110–2121, 9 2015. ISSN 21508097. Disponível em: <http://dl.acm.org/citation.cfm?doid=2831360.2831365>. Acesso em: jun. 2018.

SOCHER, R.; PERELYGIN, A.; WU, J. Recursive deep models for semantic compositionality over a sentiment treebank. In: **Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)**. Seattle, Washington, USA: Association for Computational Linguistics, 2013. p. 1631–1642. ISBN 9781937284978. ISSN 1932-6203.

Disponível em: <https://nlp.stanford.edu/~socherr/EMNLP2013_RNTN>. Acesso em: jun. 2018.

SPANN, T. **Scala vs. Java for Big Data Engineering**. DZone - Big Data Zone, 2016. Disponível em: <https://dzone.com/articles/scala-vs-java-for-big-data-engineering>. Acesso em: jun. 2018.

SPARK. **Spark Documentation: Submitting Applications**. Apache Spark, 2018. Disponível em: <https://spark.apache.org/docs/2.3.0/submitting-applications.html>. Acesso em: jun. 2018.

SPARK. **Spark Documentation: Tuning Spark - Data Locality**. Apache Spark, 2018. Disponível em: <https://spark.apache.org/docs/2.3.0/tuning.html#data-locality>. Acesso em: jun. 2018.

SPARK. **Spark Homepage**. Apache Spark, 2018. Disponível em: <https://spark.apache.org>. Acesso em: jun. 2018.

SPARKS, E. R.; TALWALKAR, A.; HAAS, D.; FRANKLIN, M. J.; JORDAN, M. I.; KRASKA, T. Automating model search for large scale machine learning. In: **Proceedings of the Sixth ACM Symposium on Cloud Computing (SoCC)**. New York, USA: ACM Press, 2015. p. 368–380. ISBN 9781450336512. Disponível em: <http://dl.acm.org/citation.cfm?doid=2806777.2806945>. Acesso em: jun. 2018.

STEVENSON, R. A.; MIKELS, J. A.; JAMES, T. W. Characterization of the affective norms for english words by discrete emotional categories. **Behavior Research Methods**, Springer Nature, v. 39, n. 4, p. 1020–1024, 11 2007. ISSN 1554-351X. Disponível em: <http://dx.doi.org/10.3758/BF03192999>. Acesso em: jun. 2018.

TABOADA, M.; BROOKE, J.; TOFILOSKI, M.; VOLL, K.; STEDE, M. Lexicon-based methods for sentiment analysis. **Computational Linguistics**, MIT Press - Journals, v. 37, n. 2, p. 267–307, 6 2011. Disponível em: <https://doi.org/10.1162%2Fcoli_a_00049>. Acesso em: jun. 2018.

TANG, D.; QIN, B.; LIU, T. Learning semantic representations of users and products for document level sentiment classification. In: **Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing**. Beijing, China: Association for Computational Linguistics, 2015. p. 1014–1023. ISBN 9781941643723. Disponível em: <http://www.aclweb.org/anthology/P15-1098>. Acesso em: jun. 2018.

THELWALL, M. The heart and soul of the web? Sentiment strength detection in the social web with SentiStrength: Collective emotions in cyberspace. In: HOLYST, J. A. (Ed.). **Cyberemotions**. Springer International Publishing, 2017. (Understanding Complex Systems), p. 119–134. ISBN 978-3-319-43639-5. Disponível em: <https://doi.org/10.1007/978-3-319-43639-5_7>. Acesso em: jun. 2018.

TSYTSARAU, M.; PALPANAS, T. Survey on mining subjective data on the web. **Data Mining and Knowledge Discovery**, Springer Nature, v. 24, n. 3, p. 478–514, 10 2011. Disponível em: <http://dx.doi.org/10.1007/s10618-011-0238-6>. Acesso em: jun. 2018.

TUAROB, S.; TUCKER, C. S. Automated Discovery of Lead Users and Latent Product Features by Mining Large Scale Social Media Networks. **Journal of Mechanical Design**, ASME International, v. 137, n. 7, p. 071402, 7 2015. ISSN 1050-0472. Disponível em: <http://mechanicaldesign.asmedigitalcollection.asme.org/article.aspx?doi=10.1115/1.4030049>. Acesso em: jun. 2018.

ULLOA, D.; SALEIRO, P.; ROSSETTI, R. J. F.; SILVA, E. R. Mining social media for open innovation in transportation systems. In: **Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)**. Rio de Janeiro, Brasil: IEEE, 2016. p. 169–174. ISBN 978-1-5090-1889-5. Disponível em: <http://ieeexplore.ieee.org/document/7795549/>. Acesso em: jun. 2018.

WANG, H.; CAN, D.; KAZEMZADEH, A.; BAR, F.; NARAYANAN, S. A system for real-time Twitter sentiment analysis of 2012 U.S. presidential election cycle. In: **Proceedings of the 2012 System Demonstrations (ACL)**. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. p. 115–120. Disponível em: <http://dl.acm.org/citation.cfm?id=2390470.2390490>. Acesso em: jun. 2018.

WHITE, T. **Hadoop: The definitive guide - storage and analysis at internet scale**. 4. ed. Beijing: O'Reilly Media, Inc., 2015. 688 p. ISBN 978-1-4919-0163-2.

WILSON, T.; HOFFMANN, P.; SOMASUNDARAN, S.; KESSLER, J.; WIEBE, J.; CHOI, Y.; CARDIE, C.; RILOFF, E.; PATWARDHAN, S. Opinionfinder: a system for subjectivity analysis. In: **Proceedings of HLT/EMNLP on Interactive Demonstrations (HLLT-Demo)**. Vancouver, Canada: Association for Computational Linguistics, 2005. p. 34–35. Disponível em: <http://dx.doi.org/10.3115/1225733.1225751>. Acesso em: jun. 2018.

WILSON, T.; WIEBE, J.; HOFFMAN, P. Recognizing contextual polarity in phrase-level sentiment analysis. In: **HLT Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing**. Vancouver, Canadá: Association for Computational Linguistics, 2005. p. 347–354. Disponível em: <https://dl.acm.org/citation.cfm?id=1220619>. Acesso em: jun. 2018.

WOLDEMARIAM, Y. Sentiment analysis in a cross-media analysis framework. In: **Proceedings of the 2016 IEEE International Conference on Big Data Analysis (ICBDA)**. Institute of Electrical and Electronics Engineers (IEEE), 2016. p. 1–5. ISBN 9781467395908. Disponível em: <http://dx.doi.org/10.1109/ICBDA.2016.7509790>. Acesso em: jun. 2018.

WRIGHT, A. **Mining the Web for Feelings, Not Facts**. New York Times, 2009. Disponível em: <http://www.nytimes.com/2009/08/24/technology/internet/24emotion.html>. Acesso em: jun. 2018.

XIN, R.; ARMBRUST, M.; LIU, D. **Introducing DataFrames in Apache Spark for large scale Data Science**. Databricks, 2015. Disponível em: <https://databricks.com/blog/2015/02/17/introducing-dataframes-in-spark-for-large-scale-data-science.html>. Acesso em: jun. 2018.