UNIVERSIDADE FEDERAL DE ITAJUBÁ

PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Sentimentalista: Um framework para análise de sentimentos baseado em processamento de linguagem natural

Armando Ferraz Graça Neto

Itajubá, Agosto de 2016

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Armando Ferraz Graça Neto

Sentimentalista: Um framework para análise de sentimentos baseado em processamento de linguagem natural

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação

Área de Concentração: Hardware e Software Básico

Orientador: Prof. Dr. Laércio Augusto Baldochi Junior

Coorientadora: Profa. Dra. Isabela Neves Drumond

Agosto de 2016
Itajubá - MG

Agradecimentos

À CAPES pelo apoio financeiro que permitiu a realização deste trabalho.

Aos meus pais, Armando Ribeiro da Luz Graça (*in memorian*) e Enedina Gaspar Graça, pela educação, dedicação e carinho, e por sempre me incentivarem e acreditarem em mim. Agradeço também a primeira palavra lida, no livro O Urso e a Uva: u-r-s-o.

À minha tia, Maria Loetitia Ribeiro da Luz Graça, por todo apoio e por todos os conselhos, e por acreditar em mim.

Aos meus orientadores, Prof. Dr. Laércio Augusto Baldochi Júnior e Profa. Dra. Isabela Neves Drumond, por todo conhecimento compartilhado, lições ensinadas, pela paciência, por acreditarem e por não desistirem de mim.

Aos meus irmãos, Danilo, Mariana e Juciara, por sempre me ouvirem e por sermos os irmãos mais cheios de Graça que existem.

Aos incríveis amigos que fiz durante o mestrado, em especial Mariana, Ramon, Luíz Felizardo, Luíz Fernando e Helaine.

À minha amada esposa, Thatiane Javorka Gonçales Graça, pelo companheirismo, por acreditar em mim quando eu mesmo não acreditava, por me apoiar, ler meus resumos, me ouvir, passar madrugadas acordada enquanto eu implementava e escrevia. Esse mestrado só foi possível graças a você. *And after all, you're my wonderwall.*

Ao Sentimentalista, por todos os seus comentário positivos, negativos e as vezes neutros, durante todo esse tempo que passamos juntos.

A Deus por ter me dado tantas pessoas para agradecer.

A todos e a cada um, meu muito obrigado.

Ordem e método. - Hercule Poirot

Agatha Christie, O misterioso caso de Styles

Resumo

A área de análise de sentimentos estuda as opiniões das pessoas sobre uma determinada entidade. O principal objetivo desta área é encontrar o sentimento (positivo, negativo ou neutro) em documentos, frases e opiniões. Com o interesse crescente pela análise de sentimentos, várias empresas já possuem ferramentas desenvolvidas internamente, e muitas *startups* foram criadas para desenvolver soluções com foco na mineração de opiniões. Este trabalho apresenta um *framework* desenvolvido na linguagem Java que utiliza técnicas de processamento de linguagem natural para extrair opiniões de comentários e gerar sumários destas opiniões. O *framework* faz uso de *POSTaggers* (*Part-of-Speech Taggers*), listas de *stopwords* e bases léxicas. Para um melhor desempenho do *framework*, as bases léxicas e listas de *stopwords* foram transformadas em árvores Trie, que possuem um algoritmo de busca de complexidade $O(1)$. O módulo de extração de características disponibilizado na ferramenta permite ao usuário definir heurísticas, tornando possível avaliar e selecionar as melhores heurísticas de acordo com o foco de uso do *framework*. A ferramenta desenvolvida permite a realização das etapas da análise de sentimento: extração de características e opiniões, classificação de sentimentos e sumarização de opiniões. Foi realizado um estudo de caso para avaliar o módulo de classificação de sentimentos. Os testes demonstraram que o *framework* possui resultados comparáveis aqueles encontrados na literatura. O *framework* é independente de contexto e pode ser configurado para qualquer idioma, sendo necessário que o usuário forneça alguns arquivos de configuração, e uma lista de heurísticas de acordo com o idioma desejado.

Abstract

Sentiment Analysis is a field of study which analyses people's opinions towards entities. The aim of this analysis is to uncover the sentiment (positive, negative or neutral) expressed in documents, phrases and opinions. The increasing interest related to sentiment analysis has motivated the devopment of tools for mining opinions. This work presents a framework called Sentimentalista, which exploits natural processing language techniques in order to extract opinions from comments. The proposed solution makes use of POSTaggers (Partof-Speech Taggers), stopwords lists and lexical bases. In order to enhance the performance of the framework, the lexical bases and the stopwords lists were represented as Trie trees, as these trees present a very fast search procedure. The framework provides a feature extraction module tha allows users to define heuristics, making it possible to evaluate and select the best heuristics according to the context of use. The Sentimentalista framework supports the main tasks in sentiment analysis: feature and opinion extraction, sentiment classification and opinion summarization. The tests performed with Sentimentalista shows that it performs as good as state of the art tools available in the literature. Moreover, the framework is context independent and is able to process text in any language, as long as the user provides a configuration file and a list containing the heuristics for the target language.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Algoritmos

Glossário

POS	*Part-of-speech*
SWN	*SentiWordNet*
NLP	*Natural Language Processing*
PLN	*Processamento de linguagem natural*
PMI	*Pointwise Mutual Information*

1 Introdução

A análise da opinião ou do sentimento identificado a partir de um conteúdo influencia o comportamento das pessoas, suas escolhas e decisões. Atualmente a quantidade de informação disponível na web através de blogs, redes sociais, sites de relacionamento, comentários, entre outros tantos meios, tem sido alvo de modelos de aprendizado de máquina, na busca por conhecimento. A chamada mineração de opinião, ou análise de sentimento, é uma área de pesquisa derivada da mineração de texto, que tem como foco entender o sentimento das pessoas sobre um determinado assunto, e também identificar aspectos dos temas estudados.

A área de análise de sentimentos é hoje um campo de pesquisa extremamente ativo, uma vez que há um conjunto grande de aplicações, em diversos domínios, que se interessa pelo potencial da análise do imenso volume de dados disponível na Web. A indústria que se preocupa com a repercussão do seu produto no mercado e o consumidor que busca os comentários de um produto para decidir uma compra motivam as pesquisas na área (LIU, 2012).

A mineração de opinião, ou análise de sentimento, é uma área de estudos que lida com a recuperação de informação e conhecimento de texto utilizando técnicas de mineração de dados e processamento de linguagem natural. O objetivo da mineração de opinião é criar um modelo computacional capaz de reconhecer e expressar emoções (KHAN et al., 2009). Com base nas características de produtos que recebem opiniões de consumidores, é possível gerar uma classificação positiva ou negativa, o que motiva as empresas a investirem neste tipo de processamento. As pessoas tem curiosidade e buscam pelas opiniões apresentadas para produtos, serviços, problemas e maneiras de como aproveitar as melhores oportunidades. Estas informações estão espalhadas em fóruns, blogs, grupos de discussão e comentários, possibilitando a formação de grandes bases de dados que necessitam de modelos computacionais que possam tratar e extrair conhecimento.

De acordo com Adomavicius e Tuzhilin (2005) e Liu (2012), a investigação do senti-

mento associado pode ser realizada em diferentes níveis, com base no alvo da mineração. Na mineração de opinião focada em documentos, o documento completo é analisado e classificado como positivo ou negativo (TURNEY, 2002). Um documento pode ser um comentário em um portal, um *tweet*, uma notícia ou artigo em um blog. Na mineração de opinião focada em sentenças, a meta é definir se a sentença possui subjetividade, ou seja, apresenta um sentimento, ou é objetiva, e apresenta um fato. Porém, deve-se levar em consideração que opiniões podem estar em sentenças objetivas. Na mineração de opinião focada em aspecto, o processo de análise é mais complexo, pois não é todo o conjunto apresentado que será analisado, e sim seus aspectos, chamados neste trabalho de características. Neste caso é importante ressaltar duas abordagens: a extração das opiniões predominantes e os sentimentos positivos ou negativos sobre tais opiniões; e a medida do número de pessoas ou a porcentagem de pessoas que apresentam opiniões negativas ou positivas sobre os objetos de estudo.

Para realizar a classificação do sentimento ou identificação da polaridade, diversas abordagens foram criadas. As abordagens mais comumente empregadas são: i) baseada em recursos léxicos, também chamados de bases de conhecimento, dicionários e/ou recursos linguísticos, que utilizam bases de palavras já pontuadas ou rotuladas (PORIA et al., 2013), (BACCIANELLA; ESULI; SEBASTIANI, 2010); ii) baseada em semântica, partindo de palavras semente, e expandindo para seus antônimos e sinônimos, ou comparando com palavras próximas (TURNEY, 2002), e iii) baseada em técnicas de aprendizado de máquina que visam identificar de maneira automática os sentimentos das palavras, classificando-as.

A sumarização é a etapa final da mineração de opinião, é quando o usuário realmente tem contato com os resultados da análise realizada sobre seu tema de interesse. Esta etapa produz resumos que apresentam de maneira mais simples e prática as opiniões expressas por outras pessoas, facilitando assim a tomada de decisão.

Este trabalho apresenta um *framework* para mineração de opinião baseado em técnicas de processamento de linguagem natural com recurso léxico, que realiza todas as etapas da mineração de opinião. A principal contribuição deste trabalho é um *framework* de código aberto independente de domínio e de idioma que possui uma estrutura modular e traz comportamentos padrão para auxiliar usuários iniciantes e avançados. O *framework* também apresenta níveis de sumarização gerados a partir do processamento de comentários submetidos. Os níveis de sumarização apresentados são os seguintes: 1) comentário, 2) opinião, 3) sentimento, 4) sentença e 5) gráfico.

Para validação do modelo proposto foi utilizada uma base de dados contendo *reviews*

de filmes em inglês, tweets em inglês e tweets em português. Os testes realizados focaram nos valores de F1 e cobertura e foram comparados com os resultados obtidos por (ARAUJO et al., 2016).

Este documento está organizado da seguinte maneira:

- O Capítulo 2 apresenta a definição da análise de sentimentos, descrevendo as etapas do processo, além da descrição de alguns trabalhos na área.
- No Capítulo 3 é descrita a análise de sentimento focada em aspectos, abordagem utilizada neste trabalho.
- O Capítulo 4 apresenta o *framework* desenvolvido e as técnicas utilizadas.
- No Capítulo 5 são feitas as considerações sobre os testes realizados e os resultados obtidos.
- Ao final, no Capítulo 6, são apresentadas as principais conclusões e propostas de trabalhos futuros.

2 Análise de Sentimentos

Este capítulo apresenta um breve resumo sobre o estado da arte da Análise de Sentimentos. São abordados os primeiros trabalhos realizados sobre a área, e trabalhos que apresentam técnicas consideradas interessantes para o entendimento da área.

2.1 Definição

A análise de sentimento é uma maneira de simplificar a recuperação de dados e informações de documentos que possuem comentários sobre temas de interesse. Estas informações são opiniões e sentimentos que não são fáceis de serem tratados computacionalmente. Considere, por exemplo, que uma pessoa deseja realizar uma viagem, e busca informações sobre o destino. A grande quantidade de informações acessadas por esta pessoa contém opiniões positivas ou negativas sobre o tema. Neste contexto, a análise de sentimento busca gerar dados simplificados a partir destas informações, tornando a pesquisa do usuário um processo mais simples (TURNEY, 2002).

Na literatura, a análise de sentimento é também descrita como uma evolução da classificação de tópicos, mesmo que as técnicas utilizadas na classificação de tópicos não tenham demonstrado a mesma eficácia na análise de sentimentos (PANG; LEE; VAITHYANATHAN, 2002). Na classificação de tópicos, o objetivo é identificar o assunto tratado dentro dos documentos, - tais como esporte, saúde, política - utilizando palavras-chave para a classificação. Já a análise de sentimento tem como objetivo classificar o sentimento geral, a polaridade da opinião expressa nestes tópicos. Técnicas de aprendizado de máquina, como os modelos *Näive Bayes* e Máquinas de Vetor Suporte vem sendo empregadas para a classificação de sentimentos como em (PANG; LEE; VAITHYANATHAN, 2002).

A classificação de sentimentos e opiniões definida na análise de sentimento é uma variação dos problemas clássicos de classificação. Prabowo e Thelwall (2009) propõem uma técnica híbrida que emprega vários tipos de classificadores. Esta técnica consiste numa cadeia de classificadores, onde o comentário passa por cada classificador seguindo

a ordem estabelecida na cadeia até que seja efetivamente classificado. Em Prabowo e Thelwall (2009) também é demonstrado que a classificação dos sentimentos e opiniões pode ser diferente do usual positivo, negativo e neutro, e abranger um espectro maior, como por exemplo, muito bom, bom, ruim, muito ruim, aumentando a quantidade de classes a serem tratadas no problema e melhorando o retorno para o usuário.

Com a evolução e popularização da web e das redes sociais, o processo de busca de informação sobre produtos e serviços migrou do meio físico para o digital, é o que defende (MAHARANI, 2013). Para os autores, a análise de sentimento tem início com a identificação de certas palavras que representam opiniões e orientação semântica (positivo ou negativo).

A mineração de opinião é um dos termos sinônimos empregados para análise de sentimento. Popescu e Etzioni (2005) apresentam a mineração de opinião como um crescente campo de pesquisa, com várias abordagens sendo propostas. Os autores citam que a grande quantidade de comentários e opiniões na web, geralmente são acompanhadas de metadados, tais como estrelas, porém, estes metadados não são suficientes para expor totalmente a opinião descrita pelo usuário. Deste modo, o usuário teria que ler uma grande quantidade de documentos para chegar a uma conclusão.

A análise de sentimentos focada em aspecto é uma subárea da análise de sentimentos, que tem como objetivo simplificar a busca de opiniões dentro de comentários, porém não de maneira abrangente, mas sim específica dentro das características do tema comentado. Hu e Liu (2004) apresentam um sistema de mineração de opinião focada em aspecto composto por duas partes, a primeira realiza a extração de características e a segunda fornece a orientação semântica da característica.

Os leitores tem procurado por dados e opiniões confiáveis em suas buscas na web. Este é um dos motivos do interesse no desenvolvimento de técnicas e aplicações capazes de gerar tais informações de maneira genuína (PANG; LEE, 2008). Ao mesmo tempo que os usuários apontam resultados positivos em suas experiências pesquisando produtos na web, eles citam que a informação é confusa, incompleta, difícil de ser encontrada. Neste contexto, é notável a necessidade da criação de sistemas capazes de apresentar estas informações de maneira clara, acessível e útil. E não só para o consumidor final, as empresas e vendedores estão cada vez mais interessados no retorno gerado por sistemas automáticos.

O problema na extração de conhecimento da internet é muito desafiador, os dados armazenados são dinâmicos, uma vez que estão em constante mudança, tendo seu conteúdo adicionado e atualizado a todo momento. São dados que influenciam a tomada de decisão.

Mais de 75000 blogs são criados diariamente, acompanhados de cerca de 1.200.000 novas postagens e comentários, e 40% das pessoas confiam em opiniões, *reviews* e recomendações coletadas na internet (PANG; LEE, 2008).

A detecção automática de emoções em textos está crescendo em importância do ponto de vista de aplicações. Resumos, blogs e sites de *reviews* são usados para coletar opiniões de consumidores sobre produtos para adquirir conhecimento sobre a reputação das empresas no mercado. As companhias estão interessadas na demanda e necessidade das pessoas. As informações adquiridas na internet são sumarizadas para produzir relatórios sobre os bons e maus aspectos de um determinado produto ou serviço (PANG; LEE, 2008).

2.2 Etapas do processo de análise de sentimento

A análise de sentimentos possui três etapas: extração, classificação e sumarização. Na extração são recolhidas palavras e metadados do foco da análise seguindo regras especificas; na etapa de classificação os dados adquiridos na etapa anterior são classificados, geralmente em positivo e negativo e por último os dados são exibidos sumarizados para o usuário. Estas etapas são descritas detalhadamente a seguir.

2.2.1 Extração

Antes de apresentar o processo de extração na análise de sentimentos, é necessário entender que, como existem vários focos que podem ser alvo da análise, o processo de extração tem funções diferentes de acordo com o foco selecionado. Na análise de sentimentos focada em documentos, o processo de extração é único, e consiste na extração de palavras que respeitem uma determinada heurística. Na análise de sentimentos focada em aspectos, existem dois processos distintos de extração que utilizam as mesmas técnicas. O primeiro é a extração de características ou aspectos, e o segundo processo é o de extração de opiniões. Como ambos utilizam as mesmas abordagens, decidiu-se por descrever o processo de extração de maneira geral.

Na análise de sentimento todo o processo se inicia com a identificação de palavras ou frases que possuem uma opinião ou sentimento associado a elas, e posterior extração das mesmas. Para encontrar estas palavras e frases, diversas técnicas podem ser empregadas. Uma das técnicas mais comuns, proveniente da mineração de texto, é a técnica de bolsa-de-palavras (do inglês *bag-of-words*) (PANG; LEE, 2004) (MARTINEAU et al., 2008), cujo objetivo é criar vetores que representam a ocorrência ou frequência das palavras contidas

em um determinado texto. Por exemplo, quando as frases 1 a 4, apresentadas no Quadro 1 são submetidas à técnica de bolsa-de-palavras são criados vetores.

Quadro 1: Exemplos para entendimento da bolsa-de-palavras

Frase 1:	Eu amei este carro.
Frase 2:	Eu gostei do carro, mas não gostei do volante.
Frase 3:	Não gostei do carro, mas o preço é justo.
Frase 4:	Este carro é um lixo.

Quando uma palavra é lida, ela é adicionada na bolsa, caso já exista, não é adicionada. No caso exemplo, o seguinte vetor é formado:

$bolsa = eu; amei; este; carro; gostei; do; mas; no; volante; o; preo; ; justo; um; lixo.$

Desta forma, cada frase seria representada pelo seu vetor de ocorrência, como pode ser observado no Quadro 2:

Quadro 2: Exemplos do vetor gerado pela bolsa-de-palavras

Frase 1:	1; 1; 1; 1; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0
Frase 2:	1; 0; 0; 1; 1; 1; 1; 1; 1; 0; 0; 0; 0; 0; 0
Frase 3:	0; 0; 0; 1; 1; 1; 1; 1; 1; 0; 1; 1; 1; 0; 0
Frase 4:	0; 0; 1; 1; 0; 0; 0; 0; 0; 0; 0; 1; 0; 1; 1

Esta técnica foi empregada por Pang, Lee e Vaithyanathan (2002) e Martineau et al. (2008) na classificação focada em documentos e por Yessenov e Misailovi (2009) focada em sentenças, utilizando *tweets*.

Outra técnica muito empregada é a utilização de um *POS Tagger* (TURNEY, 2002; ELYASIR; ANBANANTHEN, 2013), uma técnica capaz de rotular as palavras de uma frase de acordo com sua classe gramatical. Esta técnica será explicada mais detalhadamente neste trabalho no Capítulo 4. Utilizando as classes gramaticais, a identificação das palavras segue heurísticas definidas pelos pesquisadores da área. A utilização do *POS Tagger* pode ser combinada com a bolsa-de-palavras, diminuindo assim o tamanho do vetor de características gerado. O uso básico desta técnica envolve a rotulação das palavras, seguida da extração das palavras de acordo com uma heurística já definida. A partir daí são criados vetores de treinamento para técnicas de aprendizado de máquina, utilizando bolsa-de-palavras, considerando a ocorrência de cada termo (PANG; LEE, 2008). Cada vetor também pode receber pesos, baseado em dicionários de palavras como a *SentiWordNet* (BACCIANELLA; ESULI; SEBASTIANI, 2010), *EmoSenticNet* (PORIA et al., 2013) e *SenticNet*(CAMBRIA; HAVASI; HUSSAIN, 2012).

Turney (2002) apresenta um modelo baseado em informação mútua, onde cada bigrama (combinação de duas palavras) dentro do documento a ser analisado tem a sua orientação semântica calculada de acordo com a proximidade entre as palavras **excelent** e **poor**, empregando a técnica *Pointwise Mutual Information* (PMI). Para extração de opiniões Turney (2002) considera que adjetivos e advérbios tem uma grande probabilidade de representar subjetividade dentro de uma sentença, sendo assim, se houver ocorrência de adjetivo e/ou advérbio em uma frase existe uma grande chance desta frase conter uma opinião, e ser analisada. O modelo considera ainda a identificação dos bigramas para aquisição de mais informação sobre o sentimento expresso na frase. As heurísticas para identificação dos bigramas estão apresentadas no Quadro 3.

Quadro 3: Regras para extração de opiniões de Turney para o idioma inglês.

Primeira Palavra	Segunda Palavra	Terceira Palavra
Adjetivo	Substantivo	Indiferente
Advérbio	Adjetivo	Não pode ser substantivo
+ Adjetivo	Adjetivo	Não pode ser substantivo
Substantivo	Adjetivo	Não pode ser substantivo
Advérbio	Verbo	Indiferente

Após a identificação das opiniões uma nota para as opiniões é gerada utilizando o PMI. O PMI é uma técnica que considera a proximidade entre duas palavras, ou seja, se a ocorrência das palavras 1 e 2 próximas for alta, e a ocorrência da palavra 1 com qualquer outra palavra for baixa, e o mesmo para a palavra 2, então as duas palavras tem uma alta correspondência, pois ocorrem quase sempre em conjunto. O contrário indica que as palavras não tem grande relevância mútua. A fórmula do PMI é apresentada na Equação 2.1.

$$PMI\left(word_1, word_2\right) = log_2\left[\frac{p\left(word_1 \& word_2\right)}{p\left(word_1\right)p\left(word_2\right)}\right] \tag{2.1}$$

Onde, $p\left(word_1 \& word_2\right)$ representa a probabilidade de coocorrência das palavras 1 e 2, $p\left(word_1\right)p\left(word_2\right)$ representa a probabilidade de independência entre elas. O PMI então demonstra estatisticamente a relação entre elas. A orientação semântica (OS) é representada pela Equação 2.2.

$$OS\left(phrase\right) = PMI\left(phrase, "excellent"\right) - PMI\left(phrase, "poor"\right) \tag{2.2}$$

(SIQUEIRA; BARROS, 2010) apresentam uma técnica de realizar a extração de características. A técnica consiste em extrair os substantivos e considerá-los como características

candidatas, e em seguida extrair os adjetivos próximos às características mais frequentes, considerando-os como opiniões. A frequência das características é dada por um valor empírico, que no trabalho citado é 3%.

2.2.1.1 Árvores Trie

Para armazenar os dados utilizados na extração das palavras diversas técnicas podem ser empregadas, como bancos de dados relacionais (MAHARANI, 2013) e a estrutura de dados *HashMap* (ARAUJO et al., 2016). Neste trabalho foi empregada uma estrutura de dados chamada de árvore Trie (FREDKIN, 1960). Esta estrutura possui complexidade de busca O(1) e trabalha com strings, estas características influenciaram na adoção das Trie pelo Sentimentalista.

A árvore Trie possui dois tipos de nó, nó de encaminhamento e nó folha. Nós de encaminhamento possuem um único caractere e um vetor de nós de encaminhamento, este vetor representa o alfabeto, e cada um de seus nós representa um caractere. Este alfabeto possui também um caractere especial chamado terminador. Este caractere indica que uma palavra (nó folha) existe no meio de um caminho, por exemplo, o caminho da palavra "ARMAR"forma a palavra "ARMA", deste modo o nó de encaminhamento que armazena a segunda letra A teria um caractere terminador que indicaria a existência da palavra "ARMA". O nó folha armazena a palavra criada pelos nós de encaminhamento.

Para utilizar esta estrutura no Sentimentalista, o nó folha foi alterado para armazenar também as notas positiva e negativa da palavra. A implementação deste trabalho utiliza o caractere $ como terminador pois este caractere sempre ocorre antes dos outros na tabela ASCII, desta maneira para verificar se existe um nó folha no caminho, é necessário verificar se a primeira posição do vetor alfabeto é um terminador ou não.

Um pequeno teste foi realizado para definir o uso das árvores Trie ao invés de *HashMap*, um dicionário inteiro foi adicionado a uma árvore Trie e a um *HashMap* e uma palavra aleatória foi procurada. A média de tempo da busca na árvore Trie foi de 0ms, já no *HashMap* foi de 10ms.

A Figura 1 demonstra graficamente como uma árvore Trie é estruturada.

2.2.2 Classificação

A etapa de classificação pode ser considerada a etapa mais emblemática da análise de sentimentos. É nesta etapa que os documentos, sentenças e opiniões recebem o rótulo

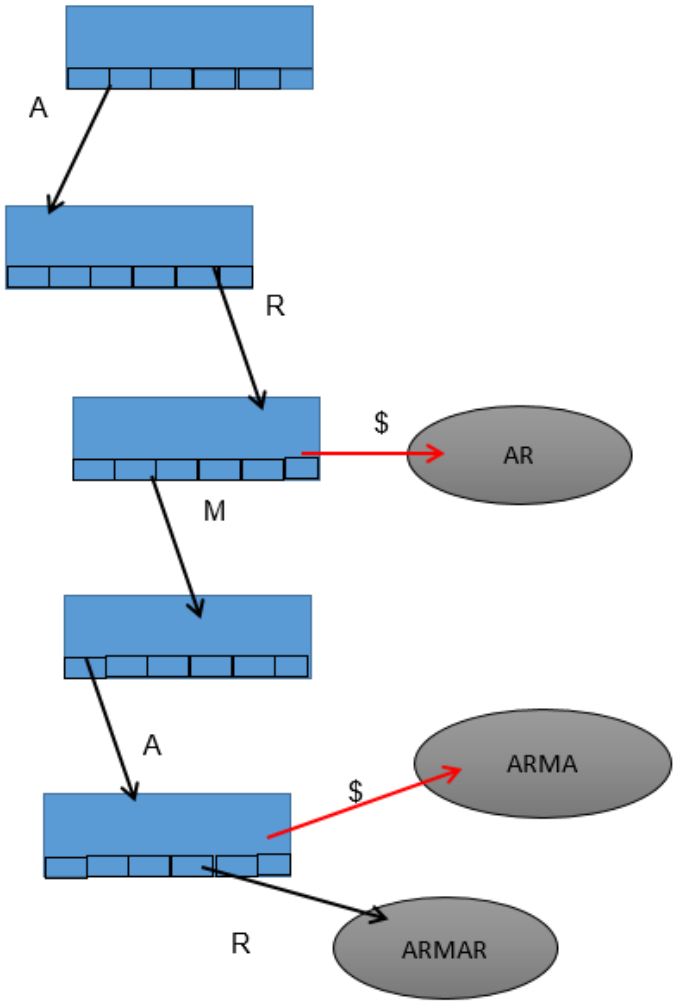


Figura 1: Árvore Trie

de positivo, negativo ou neutro (e todas as possíveis variações de rótulos). É interessante citar que o trabalho que inspirou as primeiras pesquisas na área de análise de sentimentos, visava classificar a polaridade de adjetivos (HATZIVASSILOGLOU et al., 1997), tornando a etapa de classificação a primeira etapa definida na área.

A tarefa de classificar uma opinião pode parecer simples quando realizada por um ser humano. O ser humano tem a capacidade de abstração da informação com base em conceitos chave quando lê um comentário, identificando assim o sentimento expresso no texto. Com base nisso, Pang, Lee e Vaithyanathan (2002) criaram bases de palavras-chave positivas e negativas, utilizando dois alunos de graduação como fonte. As base de palavras geradas estão apresentadas no Quadro 4.

Quadro 4: Palavras propostas pelos alunos.

	Palavras propostas
Humano 1	Positiva: dazzling, brilliant, phenomenal, excellent, fantastic Negativa: suck, terrible, awful, unwatchable, hideous
Humano 2	Positiva: gripping, mesmerizing, riveting, spectacular, cool, awesome, thrilling, badass, excellent, moving, exciting Negativa: bad, cliched, sucks, boring, stupid, slow

Após analisar as bases geradas, um terceiro aluno foi incumbido de gerar uma nova base, mas utilizando como referência as bases anteriores e testes realizados com estas mesmas bases. A base gerada pelo terceiro aluno esta apresentada no Quadro 5.

Quadro 5: Base gerada através da análise e testes na base dos alunos.

	Palavras propostas
Humano 3 + stats	Positiva:love, wonderful, best, great, superb, still, beautiful Negativa: bad, worst, stupid, waste, boring, ?, !

O foco de Pang, Lee e Vaithyanathan (2002) é demonstrar se a análise de sentimento pode ser considerada uma tarefa semelhante à classificação de tópicos ou se necessita de técnicas especificas para seu processamento e classificação. Foram executados diferentes testes, em ambos não foram consideradas técnicas de *stemming* ou *stopwords*. Quando utilizados unigramas, na ocorrência de negações foi adiciona a tag NOT_ antes de todas as palavras que apareceram entre uma palavra de negação e o próximo acento.

Utilizando unigramas considerando sua ocorrência e não a frequência como no primeiro

teste, os resultados se mostraram um pouco melhores. Com bigramas, os resultados foram piores em relação ao uso de apenas unigramas. Utilizando *POSTagger*, e considerando que apenas adjetivos representam opinião, os resultados não se mostraram muito bons utilizando 2633 adjetivos encontrados na base de comentários. E utilizando os 2633 unigramas mais frequentes, os resultados se mostraram melhores. Considerando a posição dos unigramas, os resultados se mantiveram semelhantes. A Tabela 1 mostra as técnicas utilizadas e seus resultados.

Tabela 1: Técnicas utilizadas e resultados obtidos (PANG; LEE; VAITHYANATHAN, 2002)

Características	nº de características	Frequência ou ocorrência?	Naïve Bayes	Entropia Máxima	Máquina de vetor suporte
1)Unigramas	16165	Frequência	78.7	N/A	72.8
2)Unigramas	?	Ocorrência	81.0	80.4	82.9
3)Unigramas + Bigramas	32330	Ocorrência	80.6	80.8	82.7
4)Bigramas	16165	Ocorrência	77.3	77.4	77.1
5)Unigramas + POS	16695	Ocorrência	81.5	80.4	81.9
6)Adjetivos	2633	Ocorrência	77.0	77.7	75.1

Com estes resultados, Pang, Lee e Vaithyanathan (2002) chegaram a conclusão que a análise de sentimentos é uma área de classificação mais complexa que a classificação de tópicos, e as técnicas de bolsa-de-palavras não são efetivas neste caso. Também é apresentado como conclusão que na análise de comentários, o todo não é simplesmente a soma das partes, portanto o estudo das partes pode trazer informações promissoras.

A combinação dos unigramas e/ou bigramas e/ou n-gramas encontrados na extração de opinião é utilizada como a orientação semântica do documento, sentença ou característica que estão sendo analisados (AGARWAL; XIE; VOVSHA, 2011).

Em sua pesquisa, Prabowo e Thelwall (2009) citam quatro abordagens para realizar a classificação dos sentimentos:

1. Processamento de linguagem natural. Utiliza ferramentas como *Part-Of-Speech-taggers*, *parsers* ou n-gramas. Com os dados gerados por estas ferramentas são identificados padrões, e para cada padrão é dado um sentimento, positivo ou negativo. No trabalho de Prabowo e Thelwall (2009) foi utilizado o *Montylingua parser* para gerar um conjunto de sentenças que se tornarão uma coleção de regras.

2. Aprendizado não-supervisionado. Utiliza um mecanismo de busca para descobrir o

sentimento de uma expressão, a técnica é baseada no trabalho de (TURNEY, 2002).

3. Aprendizado supervisionado. Utilizando Máquinas de Vetor Suporte (SVM) e dois algoritmos de indução, ID3 e RIPPER.

4. Classificação Híbrida. Baseado no trabalho de (KÖNING; BRILL, 2006).

As técnicas utilizadas para a classificação dos sentimentos por Prabowo e Thelwall (2009) são descritas a seguir.

1. Classificação baseada em regras. Uma regra é a associação de um antecedente e um consequente, como um *if-then*.

   (a) General Inquirer based classifier (GIBC): A regra mais simples consiste em analisar as palavras pré-classificadas presentes no *General Inquirer Lexicon*, que possui 1598 palavras positivas e 2074 negativas.

   (b) Rule-based classifier (RBC): A partir de um conjunto de documentos pré-classificados, o segundo conjunto de regras é gerado alterando os substantivos por ? e # para criar um conjunto de antecedentes, e indicando um sentimento para cada antecedente.

   (c) Statistics based classifier (SBC): Baseado no trabalho de (TURNEY, 2002).

      i. Escolha 120 palavras positivas e 120 negativas

      ii. Faça 240 consultas por antecedente.

      iii. Anote os resultados das buscas, número de páginas retornadas.

      iv. Anote os resultados de cada palavra que representa um sentimento e do antecedente.

      v. Calcule a proximidade dos antecedentes e das palavras.

   (d) Induction rule-based classifier (IRBC) Aplicação de algoritmos de indução nas regras geradas pelo RBC e SBC.

2. SVM É uma técnica de aprendizado de máquina, que calcula a maior margem possível dentre os pontos do hiperplano em que se encontram, e através dos pontos que permitem a maior margem, chamados de vetores de suporte, cria uma "linha"que separa as classes, permitindo assim a classificação. Se o problema não for linearmente separável, ela utiliza uma função kernel, e esta função aumenta o hiperplano atual, para um em que seja possível separar linearmente as classes.

3. Classificação Híbrida A classificação híbrida se da pela sequência de aplicação de diversos classificadores. A Tabela 6 apresenta as combinações de classificadores utilizada.

Quadro 6: Técnicas utilizadas e resultados obtidos por (PRABOWO; THELWALL, 2009)

RBC - GIBC
RBC - SBC
RBC - SVM
RBC - GIBC - SVM
RBC - SBC - GIBC
RBC - SBC - SVM
RBC - SBC - GIBC - SVM
RBCinduced - SBC - GIBC - SVM
RBC - SBCinduced - GIBC - SVM
RBCinduced - SBCinduced - GIBC - SVM

Eles realizaram testes utilizando os dados apresentados na tabela a seguir. O primeiro conjunto S1 de amostras pertence ao trabalho de (PANG; LEE; VAITHYANATHAN, 2002), o segundo conjunto S2 é um subconjunto do S1. O terceiro conjunto e o quarto conjuntos são proprietários. Os dados utilizados são apresentado na Tabela 2.

Tabela 2: Bases de dados utilizadas no trabalho de (PRABOWO; THELWALL, 2009).

Base de dados	Amostras	Nº de características	Nº de regras RBC	Nºde regras SBC
Filmes S1:	1000(+) 1000(-)	44140	37356	35462
Filmes S2:	100(+) 100(-)	14627	3522	2033
Produtos S3:	180(+) 180(-)	2628	969	439
MySpace S4:	110(+) 110(-)	1112	384	373

Para (MAHARANI, 2013), existem dois tipos de abordagem para a mineração de opinião, que são a abordagem léxica, que utiliza bases de palavras e dicionários para a classificação do sentimento, um exemplo destas bases é a *SentiWordNet*, e a abordagem de aprendizado de máquina, as técnicas mais utilizadas são SVM, näive Bayes, entropia máxima e *k-Nearest Neighbor*. O trabalho citado tem foco na comparação destas abordagens para mineração de opinião em *tweets* indonésios.

2.2.3 Sumarização

A etapa de sumarização de opinião é a última a ser realizada, e é onde o usuário final tem contato com os resultados gerados pela análise de sentimentos através de métricas e

resumos. A função desta etapa, é sintetizar de maneira mais compreensível e que exija o menor esforço possível de interpretação pelo leitor.

De acordo com Pang e Lee (2008), existem alguns conceitos que devem ser analisados para a geração de sumários de opiniões. Estes conceitos são apresentados a seguir.

1. Multiplicidade. Uma única opinião não é suficiente para a análise de sentimentos, logo a fonte de dados precisa conter uma quantidade de documentos/sentenças para uma análise efetiva.

2. Fator quantitativo. Sumários de opiniões devem levar em consideração a frequência das opiniões, e não apenas a ocorrência. Por exemplo, quando um usuário diz que o processador do notebook X é ótimo, e outro usuário diz que é péssimo, é totalmente diferente de 3 usuários dizerem que é ótimo, e um único dizer que é péssimo.

3. Visualização gráfica. Para Pang e Lee (2008), sumários que apresentam visualizações gráficas possuem melhores resultados do que aqueles apresentados por resumos textuais.

4. O portador da opinião. Outro ponto interessante a ser considerado é o portador da opinião. Dentro de uma mesma área, uma mesma opinião pode ter um impacto diferente, dependendo de quem expressa esta opinião. Por exemplo, um médico especialista comentando sobre um remédio X, e um leigo comentando sobre o mesmo remédio X. A opinião do médico pode ter maior valor que a do leigo.

5. Tempo. O momento em que a opinião é fornecida, também pode ser considerado. Se um produto for lançado hoje, e receber muitas *reviews* positivas, e após um mês receber várias *reviews* negativas, isto pode indicar problemas que surgem com o tempo, o que torna o produto não confiável. Seguindo o mesmo exemplo, mas o produto começa a receber *reviews* negativas anos depois, isso não quer dizer exatamente que o produto é ruim, pode ser que um novo produto tenha surgido com o tempo e seja melhor quando comparado ao produto antigo.

Outro ponto importante para a sumarização, é qual o foco da mineração esta sendo utilizado. Como já descrito, estes focos podem ser documentos, sentenças ou aspectos.

Quando é utilizado o foco em documentos, todo um documento é analisado e o retorno do processo é positivo, negativo e neutro, podendo possuir escalas dentro destes rótulos. Assim sendo, a análise de documentos considera o todo, e não as partes do que esta sendo dito.

O sumário de opinião focado em sentenças apenas informa se aquela sentença possui uma opinião ou não, e caso apresente uma opinião, se ela é positiva, negativa ou neutra, podendo apresentar escalas dentro destes rótulos.

Já no foco em aspectos, os sumários são bem mais complexos, pois vários fatores e valores podem ser levados em conta para a geração de resumos.

Por exemplo, no trabalho de Hu e Liu (2004), o sumário gerado apresenta a entidade, seus aspectos e a quantidade de opiniões positivas e negativas. Muitos outros trabalhos se basearam no modelo de sumário apresentado por estes autores.

Já Popescu e Etzioni (2005) apresentam um sumário que contém a entidade e seu aspecto (neste caso o aspecto foi dividido em várias categorias, como parte do produto, propriedade do produto, conceito, característica da parte do produto e partes e propriedades de conceitos relacionados). O grande diferencial deste sumário é que além de apresentar a frequência e se é positiva ou negativa, eles trazem a palavra opinativa, e estas opiniões são então ordenadas, apresentando no topo do sumário as mais relevantes.

Em Siqueira e Barros (2010), o sumário utiliza um gráfico de barras para representar cada aspecto da entidade, e em Liu (2010) eles apresentam um gráfico de barras para comparar os aspectos de dois produtos.

Mesmo que alguns autores defendam que sumários gráficos sejam melhores e mais compreensíveis para o usuário, a geração de sumários textuais automáticos, muito usada na área de processamento de linguagem natural, pode ser aplicada. (CARENINI; CHEUNG; PAULS, 2013) apresentam um gerador de sumários textuais automáticos, e Gamon et al. (2005) apresentam um modelo de sumário que combina gráficos e sumários textuais.

Por último, Wang, Zhu e Li (2013) apresentam um sumário que traz apenas a sentença mais representativa dos aspectos coletados das entidades. Porém, ele desconsidera o fator quantitativo dos sumários de opinião.

2.3 Problemas da área

Para Pang e Lee (2008), existem 4 problemas principais que devem ser abordados por aplicações que implementem técnicas de análise de sentimentos. Estes problemas são apresentados a seguir:

1. Caso o sistema de análise de sentimento desenvolvido for utilizado como um módulo

de outra aplicação, então o sistema deve ser capaz de determinar quando o usuário esta realmente buscando informações subjetivas, e não apenas fazendo algum outro tipo de busca. O problema é identificar o tipo de consulta utilizada, a identificação pode ser baseada em termos comuns como *review*, *reviews*, opinião ou opiniões, a aplicação também pode conter alguma maneira do usuário informar se busca dados subjetivos através de um *checkbox*, ou uma entrada de dados especifica. Assim, apenas o usuário que deseja a análise de sentimento dos dados a terá.

2. Identificar se um documento, trecho ou tópico possui informação relevante para a consulta de opiniões. Este problema pode ser contornado de maneira simples considerando como entrada do sistema apenas documentos originários de fontes altamente focadas em opiniões, como fóruns de *reviews* e *e-commerces*, com os comentários dos compradores dos produtos. Porém, se o alvo da análise forem comentários fora destes domínios, seja por considerar que informações relevantes estejam inseridas em diferentes tipos de site, ou pela busca de tópicos específicos, como politica, serviços ou pessoas, deve-se levar em consideração que o conteúdo dentro destes sites pode surgir de diferentes maneiras, estilos, gramáticas, podendo ou não conter opiniões.

3. Após adquirir os documentos com as características necessárias para a análise, é necessário que o sistema seja capaz de identificar a opinião geral expressa nestes documentos e/ou encontre as opiniões referentes a características ou aspectos específicos dos itens apresentados, ou dos tópicos debatidos. Outros cuidados são necessários, como a inclusão de citações (quote) dentro da base a ser analisada, para não utilizar a mesma informação repetidas vezes, e entender a qual entidade a informação se refere dentro da citação.

4. O último problema citado por Pang e Lee (2008), é como a informação relacionada as opiniões e sentimentos será apresentada ao usuário final. Esta apresentação precisa ser de rápida leitura e fácil entendimento. Os tópicos a seguir apresentam possíveis soluções para este problema:

   (a) Unificação da escala de apresentação. Votos que podem ter sido registrados num primeiro momento em escalas diferentes. Ex.: Um consumidor utiliza sistema de estrelas, outro classifica com letras, outro ordena com valores numéricos. O sistema transforma todos em estrelas, ou *thumbs up* e *thumbs down*.

   (b) Destaque seletivo de algumas opiniões. Opiniões como excelente, podem sozinhas alterar a maneira com que o usuário visualiza a entidade.

(c) Apresentação de pontos em que os usuários concordam, e pontos em que discordam.

(d) Identificação de comunidades de formadores de opiniões. Grupos que possuem uma forte expressão sobre determinada entidade podem alterar o pensamento dos demais.

(e) Quantificação de diferentes níveis de autoridade entre os formadores de opiniões. Segue a ideia dos críticos colunistas de jornais e revistas.

(PANG; LEE, 2008) também comentam que seria mais apropriado criar modelos de visualização dos dados de opiniões ao invés de criar modelos textuais.

2.4 Usos e Aplicações

As opiniões não tem valor caso não sejam utilizadas. A análise de sentimento pode ser aplicada de diversas maneiras para auxiliar o usuário final. Muitas aplicações foram propostas na literatura, como por exemplo em Siqueira e Barros (2010) é proposto um sistema de mineração de opinião para análise de serviços e companhias; Pang, Lee e Vaithyanathan (2002) analisam produtos, destinos de viagens e filmes; Singh et al. (2013) também visa classificar opiniões de filmes; Ofek et al. (2013) aplicam a análise de sentimento em comentários de uma comunidade online de sobreviventes de câncer e Alves et al. (2014) identificam a opinião das pessoas sobre um campeonato de futebol.

Abaixo estão algumas aplicações em que a análise de sentimentos pode ser, ou já foi empregada.

1. Websites que agregam *reviews* e opiniões (DING et al., 2008);
2. Detecção de linguagem de baixo calão ou considerada imprópria (PAZIENZA; LUNGU; TUDORACHE, 2011).
3. Suporte psicológico de sobreviventes do câncer (OFEK et al., 2013).
4. Análise de serviços e companhias (SIQUEIRA; BARROS, 2010).
5. Mineração de opinião sobre a Petrobras para predição de mercado (VILELA; MILIDIU, 2011).
6. Identificação de tópicos mais comentados sobre campeonatos esportivos (ALVES et al., 2014).

3 Análise de Sentimentos focada em aspectos

Neste capítulo são apresentados trabalhos que basearam seus esforços na área de Análise de Sentimentos focada em Aspectos. Estes trabalhos apresentam uma visão geral sobre o tema, e muitos deles inspiraram técnicas utilizadas pelo Sentimentalista.

3.1 Definição

A área de pesquisa da Análise de Sentimento focada em aspectos, também conhecida como Análise de Sentimento de características pode ser divida em 4 etapas, de acordo com (SIQUEIRA; BARROS, 2010). Estas etapas são: identificação e extração de características, identificação e extração de opiniões, classificação de sentimentos e visualização e sumarização dos resultados.

Considere a sentença a seguir, para um melhor entendimento da função de cada etapa: *I really liked the store's website*. Siqueira e Barros (2010) demonstram que opiniões e sentimentos sempre se referem a uma entidade (um objeto, um serviço, uma pessoa), seja explicitamente ou implicitamente. Esta mesma colocação é defendida por Popescu e Etzioni (2005). A etapa de extração de características foca em identificar tais entidades (ex. *the store's website*). A etapa de identificação e extração de opiniões tem como objetivo encontrar dentro da sentença, uma opinião relacionada a entidade, na sentença de exemplo a opinião encontrada é (*really liked*). A classificação tenta determinar a polaridade da opinião relacionada a entidade encontrada, a polaridade pode ser positiva, negativa ou neutra. O exemplo citado anteriormente demonstra uma opinião de polaridade positiva em relação a entidade *store's website*. A última etapa, a de sumarização e visualização, tem como objetivo apresentar de maneira simples os resultados da análise realizada nas etapas anteriores para o usuário final. Geralmente a visualização dos dados lista a quantidade de polaridades positivas e negativas de cada entidade. Alguns sistemas utilizam

gráficos para facilitar a visualização dos dados. Siqueira e Barros (2010) também afirmam que a maioria dos trabalhos na área de análise de sentimentos focam em uma ou duas destas etapas.

3.2 Extração de características

Para Siqueira e Barros (2010), a etapa de extração de características é a mais difícil de ser realizada na análise de sentimentos, e na literatura não são encontrados muitos trabalhos com foco especifico nesta etapa. Os autores afirmam ainda que os poucos trabalhos que focam na extração de características não seguem nenhuma abordagem consensual, frequentemente apresentando soluções ad-hoc. Porém pode-se afirmar que três vertentes se sobressaem: A primeira, é caracterizada pelo uso de técnicas de aprendizado de máquina para identificar termos que aparecem em opiniões positivas ou negativas. Esta abordagem é dependente de um corpus já rotulado, para servir de modelo de treino, o fato de depender de um corpus de treino torna esta abordagem dependente de contexto. A segunda abordagem se baseia em ontologias (FREITAS; VIEIRA, 2013). A outra vertente utiliza técnicas de PLN, que possui como características não precisar de um corpus e ser independente de contexto, mas sofre com uma taxa de acurácia menor (ALVES et al., 2014). Siqueira e Barros (2010) também citam que as técnicas de aprendizado de máquina e ontologias não aparentam serem adequadas no domínio de serviços, primeiro porquê o domínio não pode ser organizado em uma ontologia (suas características não são claramente pré-definidas), e não existia, na época do trabalho, um corpus que pudesse servir de base para o aprendizado de máquina. A abordagem escolhida por Siqueira e Barros (2010) e outros pesquisadores (POPESCU; ETZIONI, 2005; HU; LIU, 2004; GANAPATHIBHOTLA; LIU, 2008), baseia-se no uso de PLN.

As técnicas utilizadas na literatura para identificar e extrair características podem ser classificadas em duas categorias principais, identificação de características supervisionada e não supervisionada (HAI et al., 2014). Considerando que a mineração de opinião seja um problema estrutural de rotulagem, modelos de aprendizado supervisionado, como por exemplo máquinas de vetor suporte, tem sido utilizados para rotular características ou aspectos das entidades. Modelos supervisionados precisam ser configurados cuidadosamente para que tenham uma boa performance, dado um determinado domínio, e necessitam de um extensivo retreinamento quando aplicados a um domínio diferente. E além disto, um conjunto razoável de dados já rotulados é necessário para o aprendizado do modelo em cada domínio.

Abordagens não supervisionadas baseadas em PLN extraem características minerando padrões sintáticos e/ou morfológicos de características implícitas e explícitas nas sentenças a serem analisadas. Algumas abordagens tentam descobrir relações sintáticas entre características e opiniões ou sentimentos utilizando regras sintáticas cuidadosamente construídas ou rotulação semântica. Relações sintáticas identificadas por tais métodos ajudam a localizar características associadas a opiniões, mas também podem extrair um grande número de características inválidas graças a natureza coloquial dos textos online(HAI et al., 2014).

Abordagens estatísticas não supervisionadas utilizam os resultados de análises estatísticas de um dado corpus para entender o padrão de distribuição das opiniões referentes as características. Estas abordagens são de certa maneira resistentes a natureza coloquial dos textos online dado um corpus de tamanho suficiente. Hai et al. (2014) apontam alguns trabalhos que utilizam mineração por regra de associação (MRA) para minerar conjuntos de palavras ou grupos de palavras, que repetem-se com uma determinada frequência, estas palavras ou grupos são substantivos ou frases de substantivos. Tal abordagem apresenta alguns problemas, que são: 1) extração de características frequentes, porém inválidas e 2) características válidas e importantes são ignoradas caso possuam uma frequência abaixo da limiar estipulado.

Hu e Liu (2004) apresentam um algoritmo que utiliza PLN para extrair características. Em seu trabalho, eles definem os conceitos de característica candidata, característica frequente e característica infrequente. Características candidatas são todas as características que respeitam as heurísticas estabelecidas por eles. Características frequentes são as características que os usuários estão mais interessados sobre determinado produto, as que são mais citadas e comentadas. Porém, existem características que aparecem raramente dentro dos comentários, mas estas características podem ser interessantes para uma parcela dos usuários do produto. A estas características é dado o nome de características infrequentes. Eles propõem um método para encontrar tais características, que será explicado a seguir. Para uma melhor compreensão do método, alguns exemplos são apresentados. Estes exemplos são aplicados nas frases que se encontram no Quadro 7.

Quadro 7: Sentenças de exemplo

Sentença 1:	Red eye is very easy to correct.
Sentença 2:	The camera comes with an excellent easy to install software.
Sentença 3:	The pictures are absolutely amazing.
Sentença 4:	The software that comes with it is amazing.

As sentenças 1 e 2 compartilham a mesma opinião, *easy* (o processo de identificação de opiniões será explicado a seguir neste capítulo), mesmo que para características diferentes: *red eye* e software. Considerando que software seja uma característica frequente, isto é, ela ocorre diversas vezes no corpus, e *red eye* seja infrequente, ocorre poucas vezes, o fato de ser infrequente não significa que não seja interessante para potenciais consumidores. As sentenças 3 e 4 apresentam o mesmo problemas, mas desta vez com a opinião *amazing* e as características candidatas *pictures* e software.

Seguindo este exemplo, Hu e Liu (2004) chegaram a conclusão que os usuários utilizam as mesmas opiniões para descrever características diferentes, portanto, utilizar as opiniões encontradas para encontrar as características infrequentes é possível. Utilizando a mesma ideia de adjetivo mais próximo, são adicionadas na lista as características infrequentes que compartilhem adjetivos com as características frequentes. A heurística é apresentada a seguir:

Para cada característica candidata que não foi considerada uma característica frequente, ou seja, ocorre poucas vezes, se houver uma opinião vinculada a esta característica candidata, verifique se a opinião que a acompanha também acompanha uma característica frequente, se sim, esta característica candidata agora é uma característica infrequente.

3.3 Identificação e Extração de Opiniões

Opiniões são palavras que expressam algo positivo ou negativo sobre um determinado objeto. Hu e Liu (2004) também observam que opiniões estão próximas das palavras que representam características sobre o objeto.

Para encontrar as opiniões relacionadas as características da entidade analisada, as técnicas que utilizam PLN são as mais utilizadas (TURNEY, 2002; SIQUEIRA; BARROS, 2010; POPESCU; ETZIONI, 2005). São definidas heurísticas com base na classe gramatical das palavras, e as palavras que se adequem as regras são consideradas opiniões.

O Quadro 8 apresenta duas frases para exemplificar o processo de identificação de opiniões.

Quadro 8: Sentenças de exemplo

Sentença 1:	O tripé é muito fraco, qualquer movimento com mais força ele desaba.
Sentença 2:	Que empresa perfeita, melhor atendimento de todos.

Na primeira sentença, a característica *tripé* está próxima da opinião *fraco*. Na segunda

sentença, a característica *empresa* está próxima da opinião *perfeita*. Com base nos dados apresentados, Hu e Liu (2004) apresentam a seguinte heurística:

Para cada sentença na base de *reviews*, se ela contém algum característica frequente, extraia o adjetivo próximo. Se for encontrado um adjetivo, ele é considerado uma opinião. O adjetivo próximo refere-se a um adjetivo que modifica o substantivo/frase de substantivo que é uma característica frequente.

Eles utilizam uma técnica de *stemming*, técnicas de *stemming* visam encontrar o radical de uma palavra, estas técnicas são utilizadas para se reduzir a quantidade de palavras que necessitam ser analisadas, e *fuzzy matching* para verificar erros gramaticais.

Deve-se verificar que Hu e Liu (2004) utilizam apenas adjetivos, mas de acordo com Pang e Lee (2008), advérbios e verbos também possuem um grande potencial para expressar subjetividade, e portanto não devem ser desconsiderados.

3.4 Classificação de sentimentos

A classificação de sentimentos, também chamada de análise de sentimentos ou classificação de polaridade, não sofre alterações em seu funcionamento nos níveis de documento, sentença e aspecto. A única diferença para a classificação de sentimentos entre elas é a entrada de dados.

De maneira geral, a classificação de sentimento pode ser descrita como uma classificação binária, ou seja, classifica o elemento em duas classes distintas. No caso da análise de sentimento, estas classes são positivo e negativo. Porém, a análise de sentimento não se prende apenas as classes binárias, podendo possuir a classe neutro - o emprego da classe neutro pode ser considerado uma classificação de subjetividade - e/ou utilização de intensidade, passando a tratar positivo como muito positivo, positivo, pouco positivo, por exemplo. Na classificação focada em aspectos, a entrada do método de classificação é a opinião encontrada próxima a característica da entidade.

Os principais problemas quanto a classificação de sentimentos citados na literatura são apresentados a seguir:

1. O uso de palavras que expressam sentimento pode ser enganoso. Além disto, algumas opiniões podem ser expressas de forma objetiva, como por exemplo: ”Depois de três dias a bateria deixou de carregar ”;

2. Contexto. Muitas opiniões possuem sentido especifico dependendo da entidade estudada, por exemplo: ”O carro é muito rápido ”e ”A bateria acaba muito rápido ”, rápido tem significados diferentes dentro das duas frases;

3. Ironia. Muitos áreas são marcadas pela frequência de comentários sarcásticos e/ou irônicos, como por exemplo, política;

4. Opiniões iguais, pessoas diferentes. Se o dono da empresa X lê uma opinião negativa sobre sua empresa, ela é negativa para ele, porém o seu concorrente, dono da empresa Y, considera essa opinião positiva.

Para realizar esta classificação, dois métodos se sobressaem na literatura: classificação baseada em aprendizado de máquina e classificação baseada em recursos léxicos (dicionários).

3.4.1 Classificação baseada em aprendizado de máquina

O objetivo das técnicas de aprendizado de máquina é descobrir a qual classe pertence um elemento específico dentro de um conjunto de dados. De modo geral, as técnicas de aprendizado de máquina podem ser divididas em dois tipos: aprendizado supervisionado e aprendizado não supervisionado.

Os métodos de aprendizado supervisionado são os mais utilizados na área de análise de sentimentos. Quando trabalha-se com aprendizado supervisionado, dois passos precisam ser seguidos (NARR; HÜLFENHAUS; ALBAYRAK, 2012):

- Fornecer um conjunto de treino, com as classes já definidas para as palavras/texto, e extrair um modelo;

- Fornecer um conjunto de testes. Através do modelo gerado, o algoritmo é capaz de definir a qual classe um novo elemento pertence.

Os algoritmos mais utilizados na área de análise de sentimento são Máquinas de Vetor Suporte, Entropia Máxima e Naïve Bayes.

O conjunto de elementos que forma a base de dados de treino é composto por um conjunto de características/atributos e um rótulo. Becker e Tumitan (2013) citam os exemplos mais comuns de características/atributos:

- Palavras de sentimento: somente as palavras de sentimento são utilizadas como características. Não existe uma ordem pré-estabelecida entre a ocorrência das palavras. Cada palavra é binária dentro do vetor de características, isto é, presente ou ausente no texto;

- Termos e sua frequência: uso de unigramas, bigramas e/ou n-gramas, junto com sua frequência como peso das características;

- *Part-of-Speech*: as classes gramaticais das palavras também podem ser utilizadas em conjunto as características anteriores;

- Função sintática: as funções sintáticas das palavras podem ser empregadas com o intuito de auxiliar na definição do alvo e fonte do sentimento.

Um dos principais problemas do uso de aprendizado supervisionado para classificar a polaridade do sentimento é a necessidade de um conjunto de treino já rotulado. Este conjunto é de suma importância, e a quantidade de elementos dentro dele, e a qualidade da combinação características/rótulos influenciam enormemente o resultado do modelo gerado para a classificação. Para rotular os textos no conjunto de treino, muitas abordagens podem ser utilizadas, como estrelas, análise humana, notas fornecidas pelo escritor. Um exemplo de rotulagem: quando no domínio de produtos, as notas fornecidas pelos usuários são convertidas nas classes, por exemplo, um usuário da 5 estrelas para um produto em seu *review*, logo aquele *review* é considerado positivo. Existem domínios onde é necessário o uso de classificadores humanos para gerar estes rótulos de treino, como por exemplo, blogs de discussão política. Outro problema, também presente na classificação baseada em aprendizado de máquina, é a forte vinculação com o domínio que foi utilizado para gerar este conjunto de treino.

No aprendizado não supervisionado, o algoritmo agrupa os elementos que tenham uma grande semelhança entre si, gerando assim as classes dos sentimentos (CAVALCANTI; PRUDÊNCIO, ; CLASTER; HUNG; SHANMUGANATHAN, 2010).

3.4.2 Classificação baseada em recursos léxicos

A classificação baseada em recursos léxicos, também chamada de baseada em dicionário ou linguística, tem como foco o emprego de dicionários de palavras e/ou expressões que possuem classes pré-definidas e valores. A técnica mais utilizada com recursos léxicos é a da ocorrência de sentimento próximo a uma característica da entidade. Por exemplo,

a frase *O aroma do vinho X é adorável*, neste caso a palavra *adorável* possui um valor positivo dentro de um dicionário, logo, a característica *aroma* possui um sentimento positivo relacionado a ela. Esta técnica possui bons resultados quando empregada em sentenças ou textos pequenos, pois a proximidade com a característica é menor. Quando a análise é feita em textos maiores, as características podem estar em uma frase, e a opinião em outra, o que pode reduzir a acurácia da técnica (HU; LIU, 2004). Outros fatores a serem considerados nesta abordagem são as palavras de negação e de inversão de valores, como por exemplo *A saída de áudio não é boa* expressa um sentimento negativo, mas se não for levada em consideração a negação, pode ser que seja considerado como um sentimento positivo. Alguns recursos léxicos, como por exemplo a SentiWordNet, trazem valores para palavras de negação, deixando o tratamento de negação menos importante. Já palavras de inversão de valor como mas, porém, entretanto, geralmente indicam que a frase anterior expressa algo que terá seu valor invertido.

Um dos grandes problemas com o uso de recursos léxicos, é a escassez de dicionários em vários idiomas. Em português do Brasil, a maior base de palavras é a OpLexicon, que possui cerca de 15.000 palavras. Em comparação, a SWN possui cerca de 120.000 palavras em inglês.

Outro problema é que a maioria dos dicionários são genéricos, o que prejudica a classificação de alguns domínios, que possuem opiniões que expressam sentimentos dependendo do contexto. E a evolução da língua também deve ser notada, já que gírias e expressões populares novas surgem muito rapidamente, ao contrário dos dicionários, que demoram para serem atualizados.

3.5 Visualização e Sumarização dos resultados

Opiniões são inerentemente subjetivas, ao contrário de informações fatuais, que são objetivas. Logo, uma opinião de um único usuário geralmente não é o suficiente para se tomar uma decisão. Na grande maioria das pesquisas e aplicações, é necessário analisar e avaliar várias opiniões de diversos usuários. Criar sumários para estes resultados se mostra necessário. Liu (2012) defende que os componentes chaves de um sumário são: devem conter opiniões sobre diferentes entidades e seus aspectos, e possuir uma perspectiva quantitativa. Os dados quantitativos são importantes para apresentar ao usuário os resultados, por exemplo, é interessante para o usuário saber que 20% dos usuários acharam o filme bom, assim será bem fácil diferenciar de outro filme que 80% acham o filme bom.

Um sumário de opinião pode ser apresentado de maneira estruturada ou não estruturada, como um pequeno documento de texto. A sumarização de opinião pode ser vista como uma forma de sumarização de texto multi-documento e/ou de sumarização baseada em tópicos, que são técnicas conhecidas da PLN. A sumarização de tópico é uma abordagem que se assemelha a sumarização de um único documento pelo fato de extrair partes do texto e utilizá-las como sumários, esta tapa será melhor descrita na subseção Sumário baseado em sentença. Outra maneira seria a apresentação através de grafos utilizando as entidades como vértices e as opiniões como arestas, porém grafos muito complexos ainda não foram alcançados (PANG; LEE, 2008).

3.5.1 Sumário baseado em aspecto

A ideia mais básica e efetiva de sumário seria apresentar a quíntupla apresentada por Liu (2010), que representa uma opinião, a entidade que está sendo analisada, a característica opinada, o autor e a data em que a opinião foi escrita. Este tipo de sumário pode ser chamado de sumário baseado em aspecto, ou sumário baseado em característica (HU; LIU, 2004; SIQUEIRA; BARROS, 2010; PORIA et al., 2014).

Pode-se dizer que a sumarização baseada em aspectos possui duas características principais. A primeira é que ela captura a essência das opiniões, seus alvos (entidades e seus aspectos/características) e os sentimentos sobre elas. A segunda é que ela é quantitativa, ou seja, ela apresenta valores sobre quantos usuários forneceram opiniões positivas e ou negativas sobre aquela dada característica/aspecto.

A opção de apresentação de gráficos também esta presente na sumarização orientada a aspectos. (SIQUEIRA; BARROS, 2010) apresenta os resultados de seu sistema chamado *WhatMatter* utilizando diferentes tipos de gráficos, como visto na Figura 2. Os dados deste sumário são apresentados da seguinte maneira: No canto superior esquerdo, são apresentadas as características descobertas no corpus em um gráfico de pizza, a largura de cada fatia representa a frequência da característica. No canto superior direito, é apresentado um gráfico de pizza com a frequência de comentários positivos e negativos. Na parte inferior do gráfico, são apresentados diversos gráficos de barra, um para cada característica, apresentando a quantidade de comentários que receberam notas de 0 a 5.

Outras informações também podem ser anexadas nos sumários, de acordo com a base léxica utilizada. Por exemplo, Poria et al. (2013) apresentam a *EmoSenticNet*, uma base de palavras rotuladas de acordo com o sentimento que expressam. Os sentimentos são *joy,fear,disgust,surprise,anger* e *sadness*. A Figura 3 apresenta um exemplo de sumário

Figura 2: Sumário apresentado por (SIQUEIRA; BARROS, 2010)

Concept	Polarity	Label
Annoyed	−0.755	Anger
Birthday	+0.309	Joy
December	+0.111	Joy
Feel guilty	−0.540	Sadness
Make mistake	−0.439	Sadness
Weekend	+0.234	Joy

Figura 3: Sumário com rótulos de sentimentos apresentado por (PORIA et al., 2014)

que pode ser criado com estas informações adicionais. Estes dados podem ser muito interessantes, possibilitando uma maior gama de classes para a classificação do sentimento da opinião, e gerar dados interessantes sobre o aspecto. Consideremos que o sentimento geral sobre os freios de um determinado carro seja de medo (*fear*), a empresa responsável saberá que há algo de errado com esta característica, e poderá investir no reparo do mesmo e/ou melhorar os novos modelos.

Outro tipo de sumário interessante que pode ser apresentado, é o sumário de comparação entre entidades. Este tipo de sumário pode ser visto em (LIU, 2010). Este sumário apresenta barras para cada característica da entidade, e cada cor representa uma entidade, quanto mais acima no eixo Y, mais positivos foram os comentários sobre aquela característica, e quanto mais abaixo, mais negativos foram os comentários. A Figura 4 apresenta um sumário comparativo entre dois celulares.

Liu (2012) também discorre sobre a combinação de dados da quíntupla (entidade,

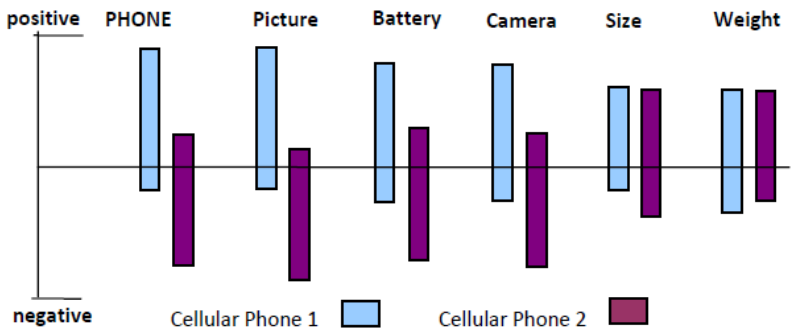


Figura 4: Sumário comparativo entre dois celulares

aspecto, sentimento, autor e data) para a geração de sumários diversos. Por exemplo, removendo-se o elemento de data, pode-se verificar os aspectos e como eles são vistos, negativamente ou positivamente, pelos usuários. E até mesmo removendo o importante elemento do sentimento, é possível apresentar as opiniões sobre os aspectos, apresentando um resultado diferente mas com capacidade de auxílio na tomada de decisão.

3.5.2 Sumário textual baseado em aspecto

Outra técnica utilizada para a geração de sumários de opiniões é a técnica baseada em textos. Esta técnica já é amplamente utilizada nas pesquisas sobre PLN, e tem como objetivo retirar informação de textos e com base nelas, apresentar um resumo compreensível do que foi recuperado (CARENINI; CHEUNG; PAULS, 2013).

Uma técnica interessante dentro da sumarização de texto, é a criação de sumários baseados em sentenças. Wang, Zhu e Li (2013) propõe um algoritmo de fatoração de matriz ponderada não-negativa para gerar sumários baseados em sentenças. A abordagem criada pode ser descrita nos seguintes passos:

- Inicialização de características relevantes: A técnica proposta é baseada no algoritmo NMF (Fatorização de Matriz não Negativa). Ao contrário deste, o algoritmo proposto não gera valores aleatórios para as variáveis U e V no algoritmo NMF onde $A = UV^t$, a decomposição da matriz U de termos-tópicos é feita selecionando as colunas de sentenças em A com a máxima contagem de cada característica, e a matriz de tópico-sentença V é consequentemente inicializada por $(U^tU)^{-1}U^tA$.
- Fatoração de Matriz não Negativa (NMF): O algoritmo NMF é realizada na matriz de termos-sentenças ponderada com a inicialização de características relevantes.

- Geração de sumário: Após a convergência do algoritmo NMF, a sentença com a maior probabilidade para cada tópico é extraída da matriz de sentença-tópicos V para formar o sumário final.

Para avaliar os sumários gerados, eles compararam o método proposto com diversos utilizados na literatura, e apresentaram os sumários de 10 produtos para 25 alunos de diversos programas de pós-graduação, e solicitaram que eles avaliassem os sumários de cada método. Os resultados adquiridos são apresentados na Figura 5, e demonstram que o método utilizado por Wang, Zhu e Li (2013) foi o mais aceito pelos usuários.

Systems	SumView	Corpernic	Centoid	Graph	NMF	LSA	Term-RS	ListAll
User 1	4.2	3	2	2.4	2	2	3.2	2.2
User 2	3.6	2.2	1.6	1.6	1.4	1	2	1.6
User 3	3.4	1.6	1.6	1.8	1.8	1.4	2.6	1.2
User 4	4	2.6	2	2.2	2	2	2.4	2
User 5	4.6	2.2	2.2	2.6	2.6	2	3	2.2
User 6	3.2	1.4	1.6	1.4	1.4	1.4	1.4	2
User 7	3	1.4	1	1.6	1.6	1	1.4	1.4
User 8	3.8	2.2	1.4	1.8	1.6	1.4	2	2
User 9	3.4	1.6	1.6	2	2	1.6	2	2.2
User 10	4.4	2.8	2.6	3	2.8	2.6	3.2	2.4
User 11	3.4	1.4	1.8	1.4	1.6	1.6	1.6	1.8
User 12	3.2	1.6	1.2	1.8	1.2	1.4	2.2	1.4
User 13	3.8	2.2	1.8	2.4	2	1.8	3	1.8
User 14	4.4	2.8	2	2.8	2.8	2	3.6	1.6
User 15	3.8	2.2	1.6	2.4	2.2	1.6	2.2	1.4
User 16	3.6	2.6	1.8	2.6	2.2	2	2.4	2
User 17	3	1.4	1.6	1.6	1.8	1.6	1.8	2.2
User 18	4.4	2.4	2	2.8	2.2	2	2.4	2
User 19	3.8	2	2	2.2	2.2	2	2.8	1.8
User 20	4	2	1.6	2.4	2.6	1.6	3	2
User 21	3.2	2	1.4	2	2	1.6	2	3.8
User 22	3.2	1.4	1.6	2.2	2.2	2	3	2
User 23	4.4	3	2.2	3	2.8	2.2	3.8	2.4
User 24	3.4	1.8	2	2	2	1.4	2	3
User 25	3.8	2.8	2	3	3	2.6	4	2.8
Average	***3.72***	***2.10***	***1.87***	***2.2***	***2.08***	***1.75***	***2.52***	***1.97***

Figura 5: Resultados adquiridos por (WANG; ZHU; LI, 2013)

4 O Framework Sentimentalista

Este capítulo apresenta o Sentimentalista, *framework* desenvolvido neste trabalho, seu funcionamento, processo de criação e definição de seus módulos, algoritmos e técnicas utilizadas.

Sentimentalista é um *framework* que oferece suporte à Análise de Sentimentos focada em aspectos. Para tanto, emprega diferentes técnicas de processamento de linguagem natural (PLN) utilizando a biblioteca OpenNLP. O Sentimentalista foi projetado para permitir que os usuários criem aplicações de análise de sentimentos que possam ser acopladas em sistemas já existentes, como fóruns, sistemas de *e-commerces*, blogs, etc.

O Sentimentalista é divido em módulos. Cada um de seus módulos é responsável por uma tarefa necessária para a realização da análise de sentimentos. Estes módulos são: i) Pré-processador, responsável por transformar os dados de entrada em dados reconhecidos pelo *framework*; ii) Extrator de Características, tem como função encontrar e retirar dos dados de entrada características seguindo heurísticas informadas pelo usuário do Sentimentalista; iii) Extrator e Classificador de Opiniões, este módulo utiliza heurísticas informadas pelo usuário do *framework* e recursos léxicos para encontrar e classificar opiniões; e iv) Apresentador do Sumário, módulo que apresenta ao usuário final da aplicação que implementa o Sentimentalista os resultados finais da análise de sentimentos realizada.

4.1 Definição de framework

Na área da Engenharia de Software, um *framework* ou arcabouço, é um esqueleto de aplicações diferentes de um mesmo domínio que permite o reuso de código. Para que diversas implementações de aplicações diferentes possam ser realizadas, o *framework* deve possuir pontos fixos (***frozen spots***) e pontos de flexibilidade (***hot spots***) (MARKIEWICZ; LUCENA, 2001).

Pontos fixos representam o núcleo do *framework*, as classes e métodos que permitem a sustentação do mesmo. Pontos fixos não são alterados pelos usuários, estando presentes em todas as aplicações originadas a partir do *framework*. Já os pontos de flexibilidade são pontos do *framework* que precisam ser implementados ou configurados pelo usuário para atender suas necessidades dentro do domínio (BOYD, 1993).

Frameworks são divididos em caixa-branca (***white box***) e caixa-preta (***black box***) de acordo com a maneira que são instanciados pelo usuário. *Frameworks* caixa-branca são instanciados pelo usuário através de herança das classes abstratas definidas no *framework*. *Frameworks* caixa-preta são instanciados através de objetos ou *scripts* de configuração. Existem também os *frameworks* caixa-cinza (***grey box***) que são um meio termo entre os caixa-branca e os caixa-preta, apresentando os dois métodos de instanciação (MARKIEWICZ; LUCENA, 2001).

Para utilizar *frameworks* caixa-branca, o usuário precisa de um profundo conhecimento da estrutura e do funcionamento interno do *framework*. Já *frameworks* caixa-preta não possuem esta exigência, demandando apenas o conhecimento dos parâmetros dos construtores dos objetos de configuração do *framework*. Isto torna *frameworks* caixa-preta mais simples de serem utilizados, porém menos flexíveis que *frameworks* caixa-branca. O Sentimentalista pode ser considerado um *framework* caixa-cinza.

O desenvolvimento de um *framework* possui três etapas principais: análise do domínio, design do *framework* e instanciação do *framework*. Na etapa de análise do domínio, são investigados casos de uso, problemas do domínio, trabalhos semelhantes e o conhecimento pessoal (ESPINOSA; HENDRICKSON; GARRETT J.H., 1999). Na etapa de design é definida a arquitetura do *framework*, as informações obtidas na primeira etapa são utilizadas para criar classes e métodos. A etapa de instanciação é realizada pelos usuários do *framework*. Nela os pontos de flexibilidade são implementados ou configurados permitindo a geração de diversas aplicaçõesas quais compartilham os pontos fixos do *framework*.

4.2 Análise do Domínio

A análise de domínio tem como objetivo gerar uma descrição estruturada do domínio, descobrindo tarefas que os usuários do domínio precisam realizar e uma lista de necessidades que precisam ser resolvidas.

Para realizar a análise de domínio foram escolhidos três casos de uso fictícios criados a partir da análise da literatura da área de análise de sentimentos focada em aspectos. A

metodologia utilizada como base para realizar a análise do domínio escolhida é chamada de Sherlock (VALERIO; SUCCI; FENAROLI, 1997). A abordagem Sherlock possui três fases distintas - caracterização do domínio, análise e modelagem dos dados e definição das estruturas reutilizáveis -, que tem início após uma análise do domínio a partir de casos de uso.

Esta seção esta estruturada da seguinte maneira: a subseção **4.1.1** apresenta casos de uso fictícios e as informações obtidas após sua análise, a subseção **4.1.2** apresenta a caracterização do domínio e o dicionário criado para representá-lo, a subseção **4.1.3** descreve os modelos gerados para o domínio e a subseção **4.1.4** detalha os pontos fixos e flexíveis do domínio.

4.2.1 Casos de Uso

Esta subseção apresenta casos de uso fictícios do domínio da análise de sentimento focada em aspectos. O *framework* foi estruturado a partir destes casos de uso e da pesquisa sobre a área de análise de sentimentos focada em aspectos descrita no **capítulo 3**.

4.2.1.1 Descobrindo características

A empresa *NewSmartPlus* está iniciando o planejamento para a criação de um novo *smartphone* de apelo popular. Para que seu novo produto tenha uma boa aceitação no mercado, eles decidiram descobrir quais são as características que mais atraem os consumidores deste tipo de produto para poder investir nas mesmas características em seu novo *smartphone*. Para descobrir quais são estas características, eles acessaram diversos sites de *reviews* de *smartphones*, coletaram estes *reviews* e analisaram todos os comentários para definir as principais características de um *smartphone*.

4.2.1.2 Opiniões sobre um produto

A construtora de carros *Fábrica de Carros* quer analisar a aceitação do público sobre um de seus modelos de veículos, o *Carro C2*, a fim de descobrir os seus pontos fracos e fortes de acordo com os seus clientes e consumidores, e assim poder melhorar seu produto e criar uma estratégia de marketing centrada em seus pontos fortes. Um grande número de *reviews* e comentários sobre o *Carro C2* foram coletados de diversos fóruns e sites. Para descobrir quais as opiniões de seus clientes, a *Fábrica de Carros* teve que analisar todos os comentários coletados.

Após ler estes comentários, eles encontraram as partes, peças e equipamentos do carro que são comentados pelos seus clientes. Os responsáveis por esta análise perceberam que muitas destas características são descritas com palavras diferentes, mesmo quando se referem à mesma parte, peça ou equipamento - como por exemplo **freio** e **breque**. Após descobrir as características, a *Fábrica de Carros* iniciou a análise para descobrir se os seus consumidores comentavam bem ou mal sobre seu produto e suas características. Eles leram novamente os comentários em busca das opiniões expressas. Sempre que encontravam uma opinião, eles decidiam se era uma opinião boa ou ruim.

Ao final deste processo, eles adquiriram informações suficientes sobre seu produto para realizar uma campanha de marketing sobre os pontos fortes de seu carro.

4.2.1.3 Sumários de opiniões

Luíz Maurício possui um blog em inglês que disponibiliza avaliações sobre hotéis ao redor do mundo. Em todas as suas postagens no blog, ele permite que os leitores postem suas próprias avaliações sobre o hotel que está sendo comentado. Seu blog é referência na área de hotéis e muitos viajantes utilizam seu site para decidir em qual hotel se hospedar.

Luíz percebeu que a maioria dos visitantes lê apenas a avaliação com mais curtidas e deixam as outras de lado, mesmo que ofereçam boas informações. Outro fato importante percebido por ele, é que cada viajante busca algo diferente quando começa a pesquisar sobre hotéis, e como eles lêem apenas um comentário, muita informação contida em seu blog fica perdida. Ele então decidiu criar e apresentar resumos dessas avaliações em cada postagem de seu blog, permitindo que os visitantes escolham as características que mais lhes interessam no hotel. Desta maneira, toda a informação acumulada em seu blog pode ser utilizada de maneira simples e rápida por seus leitores.

Luíz sabe que, geralmente, cada comentário fala sobre um ou mais aspectos de um hotel, e expressa um ponto de vista sobre este aspecto. Ele também sabe, pela sua vasta experiência, que muitos aspectos são citados com nomes diferentes, e que muitos dos aspectos comentados têm relevância para uma pequena parcela de seus leitores. Luíz pensou em diversas maneiras para solucionar seu problema, mas como sempre ele recebe novos comentários, ele precisaria analisar cada novo comentário para poder apresentar os dados de seu blog para seus leitores. Além deste problema, Luíz Maurício não tem grandes conhecimentos técnicos para implementar uma solução para esta situação.

4.2.1.4 Resultado da análise dos casos de uso

Com a análise dos casos de uso apresentados, foram extraídas informações das descrições dos casos de uso. As informações adquiridas foram:

- Todos os casos de uso focam em um alvo específico;
- Idiomas diferentes foram apresentados;
- Dois casos de uso tinham interesse na extração de opiniões enquanto um tinha interesse apenas nas características;
- Em um dos casos é necessário refazer a análise, em dois deles o processo é executado uma única vez;
- É preciso agrupar termos diferentes que se referem a um mesmo aspecto;
- Um mesmo comentário pode conter mais de um aspecto;
- Os comentários que são analisados são retirados de diversas fontes distintas;
- Comentários podem possuir opiniões positivas e/ou negativas;
- Nem sempre especialistas estão disponíveis para realizar os processos;
- O caso de uso do blog necessita de um sumário que atraia a atenção dos visitantes, os outros precisam de sumários mais técnicos e simples de serem estudados.

Estas informações serviram como base para a caracterização do domínio, que é descrita de forma detalhada na próxima subseção.

4.2.2 Caracterização do Domínio

A fase de caracterização do domínio tem por objetivo gerar um dicionário de termos que representem os objetos encontrados na análise dos casos de uso e apresentar as necessidades dos usuários em relação a estes objetos.

Na caracterização do domínio de análise de sentimentos focada em aspectos, foram encontrados alguns termos extremamente semelhantes para o usuário, mas diferentes do ponto de vista do desenvolvimento do *framework*. Estes termos receberam nomes diferentes para um melhor entendimento dos usuários do Sentimentalista.

Os termos que representam os objetos de interesse do domínio são apresentados a seguir.

- **ENTIDADE** - é o alvo da análise de sentimentos, qualquer conceito que possa ser avaliado é uma entidade em potencial. Por exemplo: um produto, pessoa, empresa, serviço, etc. As entidades se relacionam com os aspectos e com os comentários base.
- **ASPECTO** - refere-se a uma parte específica de uma entidade. Por exemplo, volante é um aspecto da entidade carro. Os aspectos se relacionam com uma entidade e com os comentários base.
- **COMENTÁRIO BASE** - é o texto escrito que expressa uma opinião sobre um aspecto de uma entidade. Os comentários se relacionam com uma entidade e com os aspectos.
- **OPINIÃO** - é uma palavra ou conjunto de palavras que expressam algo positivo ou negativo sobre um aspecto. Opiniões se relacionam com os aspectos e com os comentários base.

Com a identificação dos objetos de interesse e das tarefas executadas nos casos de uso fictícios, foram definidas as necessidades dos interessados em trabalhar no domínio. As necessidades pertencentes ao domínio analisado são descritas a seguir.

- **Diferenças de idioma** - a partir da análise dos casos de uso, nota-se a necessidade de permitir que o usuário do *framework* escolha com qual idioma irá trabalhar.
- **Encontrar aspectos da entidade** - o usuário precisa encontrar dentro dos comentários os aspectos aos quais o autor do comentário se refere.
- **Agrupar aspectos** - o usuário deve ser capaz de agrupar os aspectos encontrados utilizando seu conhecimento do domínio.
- **Definir aspectos relevantes** - o usuário deve conseguir estabelecer quais aspectos encontrados são realmente relevantes e quais não são.
- **Encontrar opiniões sobre os aspectos** - o usuário precisa descobrir quais as opiniões usadas pelo autor do comentário para se expressar sobre um aspecto.
- **Configurar o *framework*** - o *framework* deve permitir que o usuário trabalhe com diversos idiomas, e é necessário permitir que o *framework* seja adaptável e configurável para se adequar ao idioma utilizado.

- **Armazenar os dados adquiridos** - o *framework* deve permitir que o usuário salve os dados adquiridos. A Figura 6 apresenta um diagrama relacional dos dados que o Sentimentalista precisa persistir.

- **Apresentar os dados adquiridos** - o usuário precisa apresentar a seus clientes sumários dos dados obtidos através do *framework*.

- **Facilidade de uso** - o *framework* precisa trazer comportamentos padrão que sejam de fácil utilização por usuários que não tenham conhecimento avançado em desenvolvimento.

4.2.3 Análise e Modelagem dos Dados

Nesta fase da análise do domínio são definidos os modelos que representarão os termos do domínio e as tarefas que o *framework* precisa realizar, juntamente com as técnicas escolhidas para realizar tais tarefas. O dicionário de termos foi expandido para definir os objetos do *framework*.

Os modelos da análise de sentimentos focada em aspectos definidos para o desenvolvimento do Sentimentalista são apresentados a seguir.

- **PRODUTO** - é o modelo que representa uma entidade. Decidiu-se por utilizar o nome Produto, para evitar confusões com o termo entidade, que é utilizado por diversos *frameworks* na linguagem Java. Um produto se relaciona com características e com comentários.

- **COMENTÁRIO BASE** - é o comentário que ainda não foi processado pelo Sentimentalista. O comentário base é utilizado como entrada para os métodos padrão do Sentimentalista.

- **COMENTÁRIO** - é o comentário já processado pelo Sentimentalista. O comentário base precisa ser transformado em um comentário, que possui um formato aceito pelo *framework*. Um comentário se relaciona com um produto, com várias características e com várias opiniões.

- **CARACTERÍSTICA CANDIDATA** - é uma palavra ou conjunto de palavras que foram aprovadas pelas heurísticas informadas como entrada pelo usuário no módulo Extrator de Características. Para uma característica candidata ser considerada

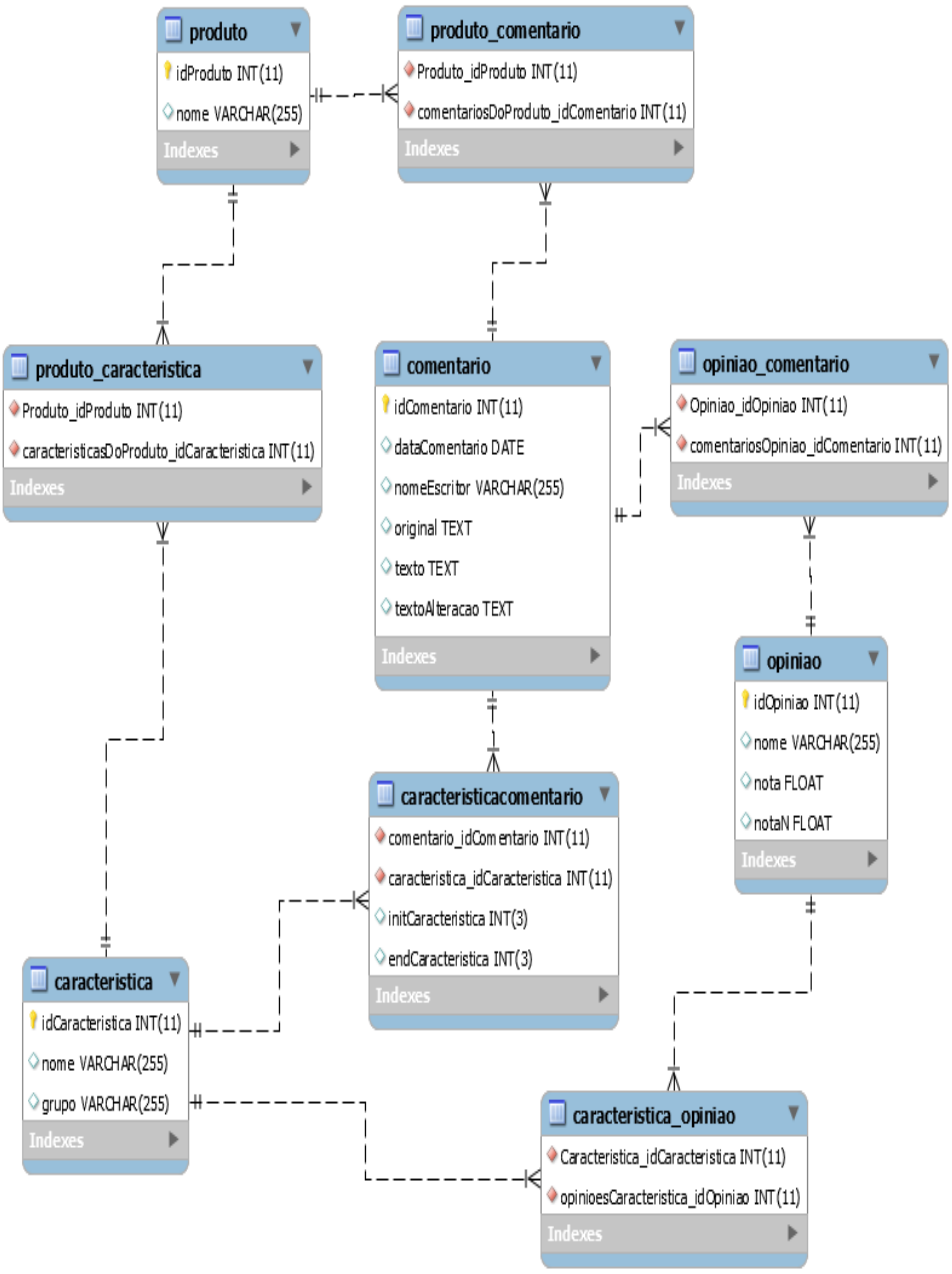


Figura 6: Diagrama relacional dos dados do Sentimentalista

uma característica, ela precisa ser aprovada por uma heurística definida pelo usuário. Características candidatas se relacionam com um produto, com vários comentários e com várias opiniões.

- **CARACTERÍSTICA** - é uma característica candidata aprovada por uma heurística definida pelo usuário.
- **OPINIÃO** - é uma palavra ou conjunto de palavras que expressam algo positivo ou negativo sobre um aspecto. Dentro do Sentimentalista, são palavras que foram aprovadas pelas heurísticas informadas pelo usuário como entrada do algoritmo de extração de opiniões. Opiniões se relacionam com vários comentários e com várias características.

Os termos a seguir representam conceitos que os usuários precisam estar familiarizados para entender como instanciar o *framework*. Eles se referem às técnicas utilizadas pelo Sentimentalista.

- **RÓTULO** - classe gramatical atribuída a uma palavra pelo *POS Tagger*. Os rótulos que podem ser atribuídos são definidos em um *Tagset*. Existem diversos *Tagsets*, o criador do arquivo de configuração do *POS Tagger* escolhe um de sua preferência. Portanto, o usuário quando escolhe o arquivo de configuração deve analisar o *Tagset* utilizado pelo *POS Tagger* escolhido.
- **TAGSET** - documento utilizado pelo arquivo de configuração do *POS Tagger* que apresenta os rótulos que representam classes gramaticais. Para os exemplos apresentados ao longo deste capítulo foi utilizado o *Penn Treebank Tagset* que se encontra no Anexo B.
- **TAG** - uma heurística fornecida pelo usuário. Uma *tag* é composta por uma ou mais classes gramaticais. Essas classes gramaticais devem ser fornecidas no formato dos rótulos atribuídos pelo *POS Tagger* escolhido pelo usuário. Para os exemplos apresentados ao longo deste capítulo foram criados apelidos para os rótulos presentes no *Penn Treebank Tagset*: todos os rótulos de substantivos são representados pela *tag* NN, todos os rótulos de adjetivos são representados pela *tag* JJ, todos os rótulos de advérbios são representados pela *tag* RB e todos os rótulos de verbos são representados pela *tag* VB.
- **RÓTULOS ESPECIAIS** - o Sentimentalista também traz dois rótulos especiais para oferecer mais opções de heurísticas ao usuário. (i) O primeiro rótulo especial é

##. Quando o usuário utiliza este rótulo em sua *tag*, ele está dizendo ao *framework*: Aceite qualquer classe gramatical, mas não extraia a palavra nesta posição. Suponha que o usuário crie a seguinte heurística: [NN,##,NN]; se o *framework* encontrar qualquer combinação de três rótulos cujo o primeiro e o terceiro rótulos sejam NN, independente do segundo rótulo, ele retornará as palavras que possuem, respectivamente, o primeiro e o terceiro rótulos. (ii) A segunda *tag* especial é ??. Quando o usuário utiliza este rótulo em sua *tag* está dizendo ao *framework*: Aceite qualquer classe gramatical e extraia a palavra nesta posição. Suponha que o usuário crie a seguinte heurística: [NN,??,NN]; se o *framework* encontrar qualquer combinação de três rótulos onde o primeiro e o terceiro rótulos são NN, independente do segundo rótulo, ele retornará as três palavras.

- **STOPWORDS** - são palavras, geralmente preposições, que se retiradas de uma sentença, esta não perde seu sentido. Não existe uma lista unânime de *stopwords*, portanto o usuário deve criar a sua própria lista de *stopwords*.

4.2.4 Definição das estruturas reutilizáveis

O objetivo desta etapa da análise de domínio é descobrir quais partes do *framework* precisam ser desenvolvidas para serem reescritas pelos usuários do *framework*, ou seja, definir os pontos de flexibilidade (**hot spots**). Verificando as informações adquiridas nas etapas anteriores decidiu-se quais partes precisam receber configurações diferentes e quais são fixas (**frozen spots**).

Os modelos do domínio descobertos na etapa Análise e Modelagem dos Dados foram considerados pontos fixos, pois eles são a estrutura da análise de sentimentos e representam o modelo de maneira geral. Esses objetos são: Característica, Comentário, Comentário Base, Opinião e Produto. As tarefas de i) preparar os comentários base, ii) extrair características, iii) extrair opiniões e iv) definir características frequentes precisam ser configuráveis para que se adequem aos diferentes domínios propostos pelos usuários, e foram consideradas pontos flexíveis.

Os dados obtidos através da utilização do *framework* precisam ser armazenados. Esta armazenagem pode utilizar técnicas e tecnologias diferentes dependendo das escolhas dos usuários e, por este motivo, o processo de armazenagem foi definido como ponto flexível. Os métodos que implementam as árvores Trie, não podem ser modificados pelo usuário, devendo o mesmo apenas instanciar e persistir as mesmas, sendo definidos como pontos fixos.

Todo *framework* precisa de comportamentos padrão, o Sentimentalista apresenta estes comportamentos para permitir que usuários iniciantes sejam capazes de realizar análise de sentimentos. Estes comportamentos são considerados como pontos flexíveis, pois podem ser instanciados e configurados por usuários avançados para personalizar o Sentimentalista.

4.3 Design do Framework

A partir da análise do domínio definiu-se a arquitetura do Sentimentalista. Esta arquitetura é apresentada na Figura 7. O primeiro passo do Sentimentalista envolve o processo de configuração do Pré-processador, preparação dos comentários base e entrega dos comentários já processados; o segundo passo consiste na preparação do extrator de características, definição de heurísticas e emissão das características candidatas; o terceiro passo é opcional, só é executado se o usuário desejar extrair opiniões, neste passo ocorrem a definição de heurísticas de opiniões, extração e classificação de opiniões, e retorno das opiniões encontradas; no quarto e último passo o usuário final tem contato com o resultado da análise de sentimentos a nível de aspecto. Nesta etapa o usuário define o limiar de frequência das características e apresenta um sumário com os dados encontrados durante a execução do Sentimentalista. Cada um dos passos é explicado detalhadamente nas subseções a seguir.

4.3.1 Pré-Processamento

Esta seção descreve as técnicas e ferramentas utilizadas pelo Sentimentalista para realizar o pré-processamento dos comentários base. Este é o primeiro passo apresentado na Figura 7. Após realizar o pré-processamento, o Sentimentalista retorna o artefato comentário no formato aceito pelas técnicas de extração de características e extração de opiniões do Sentimentalista. O pré-processamento divide o comentário base em vários comentários, os rotula e remove suas *stopwords*.

O módulo de pré-processamento recebe como entrada um artefato comentário base – opcionalmente o *framework* aceita como parâmetros de entrada o nome do autor do texto, e a data em que o texto foi escrito – e os artefatos de configuração do *Tokenizer*, do *Sentence Detector* e do *POS Tagger* e utiliza ferramentas de processamento de linguagem natural para transformar esta entrada em vários comentários.

O Quadro 9 traz como exemplo um comentário base feito por uma proprietária de um

carro (***Audi Q5 2015***).

Quadro 9: Exemplo de comentário base

Female, in my early 20's and I love my Audi Q5! I have put a few thousand miles (still under 20k) on this Audi so far and bought the Audi Care with it, brand new. I love the sound system, the navigation system, the buttons, the lights, the controls, the handling, and the traction. The performance on sport mode is phenomenal, I hit over 120mph and it handled amazingly well at such high speeds! I have drove it through rain, gravel, grass, dirt, not *yet snow.... Excellent handling! Very sleek and fancy interior, my favorite thing is the sunroof. I don't have any kids, but I would assume the back seat is average, not large or too small. I travel a lot, it goes for an average of 450 miles before needing another fill up. There is so much technology I have yet to figure out everything, but the more I learn the more I love it!

O módulo de pré-processamento do *framework* divide então este comentário em diversas sentenças, utilizando a ferramenta *Sentence Detector* da biblioteca de processamento de linguagem natural OpenNLP – esta ferramenta é configurada com o artefato de configuração do *Sentence Detector* visto na Figura 7. O Quadro 10 apresenta as sentenças resultantes deste processo.

Quadro 10: Sentenças extraídas do comentário base

Female, in my early 20's and I love my Audi Q5!
I have put a few thousand miles (still under 20k) on this Audi so far and bought the Audi Care with it, brand new.
I love the sound system, the navigation system, the buttons, the lights, the controls, the handling, and the traction.
The performance on sport mode is phenomenal, I hit over 120mph and it handled amazingly well at such high speeds!
I have drove it through rain, gravel, grass, dirt, not *yet snow.... Excellent handling!
Very sleek and fancy interior, my favorite thing is the sunroof.
I don't have any kids, but I would assume the back seat is average, not large or too small.
I travel a lot, it goes for an average of 450 miles before needing another fill up.
There is so much technology I have yet to figure out everything, but the more I learn the more I love it!

Neste ponto, o *framework* começa a trabalhar com diversos comentários base (o comentário base só será considerado como um comentário no final deste passo). Isso permite, por exemplo, enviar como entrada um grande número de comentários base, ou um texto completo. Esta etapa é executada em cada um dos comentários base adquiridos da separação do comentário base original.

O comentário base é processado então pela ferramenta *Tokenizer* que é configurada pelo artefato de configuração *Tokenizer* visto na Figura 7, que tem a função de separar as

pontuações das palavras. Isto é necessário para evitar que as características candidatas encontradas tenham seus nomes definidos como nomeDaCaracterística, ou nomeDaCaracterística. (Observe a vírgula e o ponto final anexos ao nomeDaCaracteristica). O Quadro 11 apresenta um exemplo do funcionamento da ferramenta *Tokenizer*.

Quadro 11: Funcionamento da ferramenta Tokenizer

Entrada	Female, in my early 20's and I love my Audi Q5!
Saída	Female , in my early 20 's and I love my Audi Q5 !

Em seguida, o comentário base é processado pela ferramenta *POS Tagger*, que é configurada utilizando o artefato de configuração *POS Tagger* apresentado na Figura 7. Esta ferramenta rotula cada palavra com sua respectiva classe gramatical. O Quadro 12 apresenta um exemplo do funcionamento da ferramenta *POS Tagger*.

Quadro 12: Funcionamento da ferramenta POS Tagger

Entrada	Female, in my early 20's and I love my Audi Q5!
Saida	Female_NNP ,_, in_IN my_PRP early_JJ 20_CD 's_POS and_CC I_PRP love_VBP my_PRP Audi_NNP Q5_. !_.

Após a aplicação dos rótulos pela ferramenta *POS Tagger*, o comentário base tem suas *stopwords* removidas. O processo de remoção de *stopwords* utiliza uma árvore Trie composta pela lista de *stopwords* criada pelo usuário. Após a remoção das *stopwords*, o comentário, já totalmente processado e não mais um comentário base, está pronto para a extração de características.

4.3.2 Extração de Características Candidatas

No passo 2 do *framework* o módulo Extrator de Características recebe como artefatos de entrada uma lista de comentários produzida no passo 1, um conjunto de *tags* de característica e uma árvore Trie contendo grupos de características, e retorna todas as características candidatas aprovadas pelas heurísticas representadas pelas *tags* informadas pelo usuário que se encontram nestes comentários.

O módulo Extrator de Características utiliza heurísticas definidas pelo usuário, estas heurísticas são criadas utilizando os rótulos fornecidos pelo *POS Tagger*. O módulo de extração de características aceita qualquer combinação de rótulos para compor suas *tags*.

O Sentimentalista utiliza uma solução *ad hoc* para agrupar características candidatas. A árvore Trie com o grupo da característica tem como objetivo agrupar características

candidatas formadas por palavras diferentes, mas que se referem ao mesmo aspecto da entidade. O uso das árvores Trie tem como objetivo propor uma solução de simples entendimento por parte do usuário, e que possa ser customizado com facilidade. Para criar grupos de característica com uma maior qualidade, é necessário que o usuário conheça o contexto com o qual trabalhará, ou que avalie uma primeira execução do *framework* para verificar as características encontradas e agrupá-las. O agrupamento também pode ser realizado posteriormente à extração das características candidatas.

As *tags* escolhidas pelo usuário são procuradas na ordem fornecida, em cada um dos comentários informados como parâmetro de entrada do algoritmo. Este ponto é muito importante, pois permite que o usuário escolha quais *tags* ele considera mais importantes, e colocá-las antes das que ele considera menos importantes, por isto o *framework* não executa nenhuma ordenação nas *tags*.

A Figura 8 demonstra o texto que compõe o comentário que será enviado ao algoritmo de extração de características. Considere que o usuário tenha escolhido as *tags* [NN, NN] e [NN], onde NN significa substantivo de acordo com o *Penn Treebank Tagset* que se encontra no Anexo B.

O Algoritmo 1 apresenta o procedimento para extração de características candidatas utilizado no passo 2 da arquitetura do Sentimentalista. O algoritmo é explicado detalhadamente e exemplificado a seguir.

Algoritmo 1: Extração de Características Candidatas

Entrada: comentários, tags de característica, trie de grupos

Saída: características candidatas

```
1  início
2      para cada comentário ∈ comentários faça
3          Crie um vetor texto com o conteúdo do comentário;
4          para cada tag ∈ tags de caracteristica faça
5              Crie um vetor rótulos com o conteúdo da tag;
6              Defina posição inicial é igual a 0; Defina posição final é igual a tamanho do
                vetor texto - tamanho do vetor rótulos;
7              para posição inicial até posição final faça
8                  Retire um sub-vetor do vetor texto do tamanho do vetor rótulos;
9                  Crie um vetor palavras com as palavras do sub-vetor;
10                 Crie um vetor classes gramaticais com classes gramaticais do sub-vetor;
11                 Crie um vetor posições com as posições do sub-vetor;
12                 se classes gramaticais for igual a rótulos então
13                     Crie uma característica candidata com os vetores palavras, classes
                        gramaticais e posições;
14                 Remova as palavras do conteúdo do comentário;
15                 Realize a tentativa de agrupamento da característica candidata;
16                 Adicione a característica candidata em uma lista de características
                    candidatas;
17 retorna características candidatas
```

Para cada comentário fornecido como entrada (linha 2), o algoritmo transforma o conteúdo do comentário em um vetor de Strings (linha 3). Em seguida, para cada *tag* fornecida como entrada (linha 4), o algoritmo cria um vetor de Strings com os rótulos da *tag* informada (linha 5).

As posições inicial e final são instanciadas (linha 6) e o algoritmo então começa a extrair, se for possível (linha 7), sub-vetores do vetor de Strings, os quais tem o tamanho do vetor da *tag* (linha 8). Para cada sub-vetor, 3 vetores são criados. O primeiro vetor armazena as palavras que estão no sub-vetor extraído do comentário, o segundo vetor armazena as classes gramaticais e o terceiro vetor armazena as posições extraídas do vetor original (linhas 9,10 e 11). O Quadro 13 apresenta os sub-vetores gerados pelo algoritmo a partir do exemplo da Figura 8.

O algoritmo compara então o vetor de classes gramaticas com a *tag* (linha 12), que nada mais é que um vetor de rótulos. A Figura 9 apresenta a primeira comparação

Quadro 13: Sub-vetores e seus conteúdos

Sub-vetor	Conteúdo
Palavras	[”Female”, ”,”]
Classes gramaticais	[”NNP”,”,”]
Posições	[0,1]

utilizando os sub-vetores do Quadro 13, os vetores de entrada do método de comparação seriam [”NNP”, ”,”] e [”NN”,”NN”].

A comparação então é realizada: [”NN”, ”,”] == [”NN”,”NN”] tendo *false* como retorno.

As *tags* informadas não são iguais, o algoritmo então realiza a busca nas palavras restantes do comentário. A Figura 10 resume as buscas pela característica candidata representada pela *tag* [NN,NN].

O algoritmo interrompe sua execução quando chega na última posição possível do vetor de Strings formado pelo comentário. A Equação 4.1 define se é possível realizar uma comparação na posição atual (linha 7).

$$posicaoAtual <= tamanhoTextoAlteracao - tamanhoVetorTags \tag{4.1}$$

Neste caso tem-se:

$$posicaoAtual\,(8) <= tamanhoTextoAlteracao\,(9) - tamanhoVetorTags\,(2) \Rightarrow falso$$

A última posição válida é 7. Nesta iteração, nenhuma combinação foi encontrada, portanto nenhuma característica candidata foi encontrada. O algoritmo então procura ocorrências da próxima *tag* [NN], como apresentado na Figura 11.

Utilizando a *tag* [NN] uma ocorrência é encontrada na primeira posição do comentário e o algoritmo cria uma característica candidata com as informações dos vetores gerados no início do algoritmo. O vetor que contém as palavras é utilizado para criar o nome da característica , o vetor que contém as posições é utilizado para definir as posições inicial e final da característica candidata dentro do comentário (linha 13) e para remover a característica candidata do conteúdo do comentário (linha 14). Essa remoção é feita alterando o rótulo das palavras utilizadas para criar a característica pelo rótulo <CAR>, o que impede que outra *tag* utilize uma palavra que já compõe uma outra característica.

Neste ponto o algoritmo realiza o agrupamento de características utilizando a árvore Trie de grupos de características passada como artefato de entrada (linha 15). Por exemplo, a característica *injector* e *fuel injector* compartilham o mesmo grupo na árvore. O formato de árvore Trie também foi pensado para que usuários do Sentimentalista possam compartilhar seus grupos de características.

A característica então é adicionada a uma lista de características candidatas (linha 16).

Ao final de sua execução, o algoritmo do módulo Extrator de Característica retorna uma lista de características candidatas (linha 17). Neste exemplo as características candidatas encontradas foram *female* e *Audi*.

4.3.3 Extração e Classificação de Opiniões

A extração e classificação de opiniões ocorre no terceiro passo da arquitetura do Sentimentalista, como visto na Figura 7. O algoritmo de extração de opiniões recebe como artefatos de entrada uma lista de características candidatas, que são o resultado do passo de extração de características, uma lista de *tags* de opinião, e três árvores Trie: uma árvore que contêm notas para adjetivos, uma que contêm notas para advérbios e outra que contêm notas para verbos – estas árvores são apresentadas na Figura 7 como artefato árvore Trie de palavras. A saída deste passo é a lista de características candidatas atualizadas com as opiniões encontradas.

O processamento das *tags* de opinião é semelhante ao processamento das *tags* de características, utilizando os mesmo três vetores de palavras, classes gramaticais e posições.

O módulo Extrator e Classificador de Opiniões utiliza o Algoritmo 2 para extrair e classificar opiniões. O algoritmo é explicado e exemplificado a seguir.

O algoritmo de extração de opiniões funciona da seguinte maneira: Para cada característica candidata informada no artefato lista de características candidatas (linha 2), é criado um vetor de Strings com o conteúdo do comentário vinculado à característica candidata (linha 3). Em seguida, para cada *tag* informada no artefato lista de *tags* de opinião (linha 4), cria um vetor de Strings com os rótulos da *tag* (linha 5).

O algoritmo verifica, se for possível (linha 6), nas posições anteriores da posição inicial da característica candidata se existe ocorrência da *tag* informada. Caso exista ocorrência, o algoritmo extrai a opinião e interrompe a busca por opiniões (linhas 7 à 13).

Algoritmo 2: Extração e Classificação de Opiniões

Entrada: características candidatas, tags de opinião, trie de adjetivos, trie de advérbios, trie de verbos
Saída: características candidatas atualizadas com opiniões

```
1  início
2      para cada característica candidata ∈ características candidatas faça
3          Crie um vetor texto com o comentário onde a característica candidata ocorre;
4          para cada tag ∈ tags de opinião faça
5              Crie um vetor rótulos com o conteúdo da tag;
6              se posição inicial da característica - tamanho vetor de rótulos >= 0 então
7                  Retire um sub-vetor do vetor texto do tamanho do vetor rótulos;
8                  Crie um vetor palavras com as palavras do sub-vetor;
9                  Crie um vetor classes gramaticais com as classes gramaticais do
                    sub-vetor;
10                 Crie um vetor posições com as posições do sub-vetor;
11                 se classes gramaticais for igual a rótulos então
12                     Uma opinião foi encontrada;
13                     Interrompa o laço de repetição;
14         se não foi encontrada opinião então
15             para cada tag ∈ tags de opinião faça
16                 Crie um vetor rótulos com o conteúdo da tag;
17                 se posição final da característica + tamanho vetor de rótulos < tamanho
                    do vetor texto então
18                     Retire um sub-vetor do vetor texto do tamanho do vetor rótulos;
19                     Crie um vetor palavras com as palavras do sub-vetor;
20                     Crie um vetor classes gramaticais com as classes gramaticais do
                        sub-vetor;
21                     Crie um vetor posições com as posições do sub-vetor;
22                     se classes gramaticais for igual a rótulos então
23                         Uma opinião foi encontrada;
24                         Interrompa o laço de repetição;
25         se foi encontrada opinião então
26             Crie uma opinião com os vetores palavras, classes gramaticais e posições;
27             Procure cada palavra na árvore Trie respectiva a sua classe gramatical;
28             Some as notas das palavras e atualize a opinião;
29             Remova as palavras do conteúdo do comentário;
30             Adicione a opinião em uma lista de opiniões;
31 retorna características candidatas atualizadas com opiniões
```

Se não for encontrada nenhuma ocorrência das *tags* informadas nas posições anteriores da posição inicial (linha 14), o algoritmo verifica nas posições posteriores à posição final da característica candidata se existe ocorrência da *tag* informada (linhas 15,16 e 17). Caso exista, a opinião é extraída e a busca por opiniões é interrompida (linhas 18 à 24).

Se alguma ocorrência for encontrada, cria-se uma opinião (linhas 25 e 26). As palavras contidas no vetor de palavras tornam-se a opinião e a nota de cada palavra é procurada na árvore Trie que representa a classe gramatical da palavra extraída. A nota final da opinião é a soma das notas das palavras que compõem a opinião (linhas 26, 27 e 28).

É apresentado a seguir um exemplo da extração de opiniões. O exemplo é composto de duas características candidatas apresentadas no exemplo da subseção **4.2.2** e utiliza como *tags* de entrada [JJ] (adjetivo) e [VB] (verbo).

A primeira característica candidata tem como posição inicial 0, posição final 0, o nome da característica é *female* e o comentário vinculado à característica é: Female_<CAR> ,_, early_JJ 20_CD I_PRP love_VBP Audi_<CAR> Q5_. !_.

A segunda característica candidata tem como posição inicial 6, posição final 6, o nome da característica é audi e o comentário vinculado à característica é: Female_<CAR> ,_, early_JJ 20_CD I_PRP love_VBP Audi_<CAR> Q5_. !_.

A primeira iteração tentará encontrar ocorrência da *tag* [JJ] antes da posição inicial. Antes de verificar a existência ou não o algoritmo verifica se é possível fazer uma comparação (linha 6), para isto ele utiliza a Equação 4.2.

$$posicaoInicial - tamanhoVetorTag >= 0 \tag{4.2}$$

Neste caso tem-se:

$$posicaoInicial\,(0) - tamanhoVetorTag\,(1) >= 0 \Rightarrow falso$$

Portanto a busca é interrompida e inicia-se uma busca para a próxima *tag*. A mesma verificação acontece, e o algoritmo não realiza a busca nas posições anteriores.

A busca é então realizada nas posições posteriores à posição final da característica (linha 17). O algoritmo verifica se é possível extrair a *tag* informada destas posições. A verificação nesta etapa é feita pela Equação 4.3.

$$posicaoFinal + tamanhoVetorTags < tamanhoTextoAlteracao \tag{4.3}$$

Neste caso:

$$posicaoFinal\,(0) + tamanhoVetorTags\,(1) < tamanhoTextoAlteracao\,(9) \Rightarrow verdadeiro$$

Então a busca pode ser realizada. A Figura 12 exemplifica esta busca.

A *tag* encontrada após a posição final [,] não é igual a *tag* informada [JJ] (linha 22), a próxima *tag* então é procurada. A Figura 13 exemplifica a busca pela próxima *tag*.

A *tag* encontrada após a posição final [,] não é igual a *tag* informada [VB] (linha 22), e não existem mais *tags*. O algoritmo passa para a próxima característica candidata. Primeiro o algoritmo verifica se pode realizar buscas nas posições anteriores à posição inicial utilizando a Equação 4.2 (linha 6).

$$posicaoInicial\,(6) - tamanhoVetorTag\,(1) \geq 0 \Rightarrow verdadeiro$$

A busca pode ser realizada. A Figura 14 exemplifica a busca.

A *tag* encontrada antes da posição inicial [VB] não é igual a *tag* informada [JJ] (linha 11), a próxima *tag* é procurada. Novamente o algoritmo verifica se pode realizar buscas nas posições anteriores à posição inicial utilizando a Equação 4.2.

$$posicaoInicial\,(6) - tamanhoVetorTag\,(1) \geq 0 \Rightarrow verdadeiro$$

A busca pode ser realizada. A Figura 15 exemplifica a busca.

A *tag* encontrada antes da posição inicial [VB] é igual a *tag* informada [VB], portanto interrompe-se a busca de opiniões desta característica candidata (linha 13), e tem início o processo de criação da opinião.

A palavra contida no vetor de palavras é *love*, sua classe gramatical é verbo, então ela é procurada na árvore Trie de verbos (linha 27). A palavra *love* existe na base e possui o valor positivo = 0.625 e valor negativo = -0.03125. Se existissem mais palavras elas seriam procuradas e seus valores somados (linha 28). Caso a palavra não exista, seus dois valores são 0.

Outra vantagem do uso de árvores Trie é que pode-se, com o tempo, aumentar a base ou atualizar suas notas, de maneira muito simples sem interferir no método nem no tempo de busca das palavras.

Com a opinião montada e pontuada, o algoritmo retira da(s) posição(ões) armazenada(s) no vetor de posições a(s) palavra(s) que formam a opinião, e atualiza o texto do comentário vinculado à característica candidata (linha 29), evitando assim que uma palavra que já foi usada seja extraída novamente. O Quadro 14 apresenta o texto do comentário ao final da execução do algoritmo.

Quadro 14: Texto do comentário após execução do algoritmo de extração e classificação de opiniões

Female<CAR>	,,	earlyJJ	20CD	IPRP	love<OPN>	Audi<CAR>	Q5.	!.

A opinião criada é adicionada a lista de opiniões da característica candidata à qual ela está vinculada (linha 30).

Ao final deste passo tem-se uma lista de características candidatas atualizadas com opiniões.

4.3.4 Apresentação do Sumário

A visualização, ou sumarização das opiniões, é um ponto fundamental da análise de sentimentos. É neste ponto que o usuário final terá acesso às características e suas opiniões. Dois módulos são executados no passo de Exibição do Sumário: Seletor de Características Frequentes e Exibição do Sumário, como visto na Figura 7.

4.3.4.1 Seletor de Características Frequentes

Durante todos os passos anteriores quando são citadas características, está se fazendo referência ao conceito de características candidatas. É neste passo que uma característica candidata pode ou não se tornar uma característica.

O Seletor de Características Frequentes recebe como artefatos de entrada um valor que indica o limiar de frequência das características, uma lista de características candidatas ou uma lista de características candidatas atualizadas com opiniões.

O conceito de características candidatas é importante para evitar que palavras que tenham sido aprovadas pelas *tags* de característica mas não são características sejam apresentadas ao usuário final. Por exemplo, o comentário utilizado para demonstrar

o funcionamento dos módulos do Sentimentalista refere-se a um carro (Audi Q5), e a primeira característica candidata encontrada foi *female*. É notável que *female* não é uma característica de um carro, e não faz sentido apresentá-la como tal. Porém, é uma característica candidata que passou pelas heurísticas propostas pelo usuário. A definição de características frequentes resolve este problema.

Características frequentes são as x% características mais frequentes, onde x é o artefato limiar de frequência visto na Figura 7.

As características não frequentes são características candidatas cuja frequência não ultrapassa o limiar x informado pelo usuário, porém elas representam uma característica da entidade analisada. Os usuários finais também têm o costume de utilizar as mesmas opiniões quando avaliam características diferentes, portanto considera-se características não frequentes as características candidatas que compartilham opiniões com características frequentes. Ou seja, se uma característica candidata que não foi selecionada na triagem de frequência compartilhar uma opinião com uma característica frequente, esta se torna uma característica não frequente.

O processo de definição de características frequentes e não frequentes é baseado no algoritmo proposto por (HU; LIU, 2004).

O módulo Seletor de Características Frequentes produz como saída uma lista de características.

4.3.4.2 Apresentador do Sumário

O módulo Apresentador do Sumário traz cinco níveis de sumarização que tem por finalidade seguir uma apresentação gradual de sumários, para preparar o usuário final para o último nível de sumarização. Este módulo recebe como artefato de entrada uma lista de características frequentes produzida pelo módulo Seletor de Características Frequentes.

O usuário do Sentimentalista tem liberdade para trabalhar com qualquer tecnologia para exibir seu sumário, não sendo obrigado a utilizar o módulo Apresentador do Sumário. Caso o usuário deseje criar seu próprio módulo de apresentação do sumário, ele pode se basear nos níveis implementados pelo Sentimentalista que são descritos a seguir.

4.3.4.3 1º Nível - Comentários

São apresentadas as sentenças que contenham uma característica escolhida pelo usuário. Desta maneira, o usuário sabe exatamente o que foi escrito por outros comentaristas sobre uma característica de seu interesse, porém precisa interpretá-los, o que pode demorar bastante levando em consideração o tamanho do corpus a ser analisado.

A Figura 16 apresenta o primeiro nível de sumarização proposto, com as sentenças nas quais ocorrem uma determinada característica.

4.3.4.4 2º Nível - Opinião

São apresentadas as palavras que compõe a opinião da característica selecionada pelo usuário. Com estas informações, o usuário precisa interpretar o sentido de cada opinião extraída, diminuindo assim o tempo necessário para que ele se informe do que foi dito sobre a entidade sumarizada.

A Figura 17 apresenta o segundo nível de sumarização.

4.3.4.5 3º Nível - Sentimento

Neste nível, as opiniões são apresentadas em conjunto com as notas geradas através da busca realizada nas tries que contém as notas da base *SentiwordNet*. Com estas informações, o usuário poderá identificar valores positivos e negativos das características, facilitando seu processo de interpretação das informações geradas, já que ele passa a ter valores reais associados às palavras, que são subjetivas.

A Figura 18 apresenta o 3º nível de sumarização.

4.3.4.6 4º Nível - Sentença

No 4º nível a sumarização é apresentada utilizando a técnica de sumário textual baseado em sentença. Seguindo a ideia de (WANG; ZHU; LI, 2013), foi criado um sumário que apresenta a sentença mais significativa sobre uma determinada característica. Para isto, busca-se as sentenças que possuem opiniões sobre aquela característica, e escolhe-se a que possui o maior valor independente de polaridade e frequência.

A Figura 19 apresenta o quarto nível de sumarização.

4.3.4.7 5º Nível de Sumarização

Este é o nível mais alto de sumarização no qual o usuário tem uma representação visual dos resultados. Diversos tipos de gráficos podem ser utilizados. Como gráfico base do *framework* foi escolhido o gráfico de bolhas, por ser um gráfico capaz de representar tanto o valor positivo quanto o negativo da opinião.

Neste nível é criado um gráfico de bolhas para cada uma das opiniões encontradas na criação dos níveis anteriores. A posição da bolha no gráfico é definida por seus valores positivo e negativo. A ideia por trás desta etapa é fazer com que o usuário relacione opiniões positivas com o lado superior direito do gráfico, onde a concentração de palavras positivas será maior e relacione o canto inferior esquerdo com opiniões negativas. O eixo X do gráfico representa a nota positiva, enquanto o eixo Y representa a nota negativa. O canto inferior direito contém palavras que possuem valores positivo e negativo elevados. Estes casos são opiniões formadas por mais de uma palavra, e cada uma das palavras representa sentimentos distintos. O canto superior esquerdo é preenchido por palavras com notas próximas ou iguais a 0, estas palavras possuem pouco sentimento, ou suas palavras não possuem um valor dentro da SWN (ou outra base escolhida pelo usuário). Este gráfico foi escolhido pelo formato de seus dados (posições x,y e tamanho da bolha) se enquadrarem com os dados adquiridos com os gerados pelo Sentimentalista (notas positiva, negativa e frequência da opinião).

4.3.5 Modelagem Orientada a Objetos do Framework

Nas subseções anteriores foram apresentados os módulos do Sentimentalista e sua arquitetura, vista na Figura 7. Para definir quais classes precisam ser implementadas, os casos de uso apresentados na subseção **4.1.1** foram revisitados e um diagrama de classes foi criado. O diagrama de classes é apresentado na Figura 22 e se encontra no Apêndice A.

A partir da análise das classes e da arquitetura do Sentimentalista, definiu-se a modelagem orientada a objetos do *framework*. O Sentimentalista possui 4 camadas: camada de modelo, camada de controle, camada de visão e camada de acesso a dados.

A camada de modelo pode ser divida em dois conjuntos de classes. O primeiro conjunto é composto pelas classes:

- **Caracteristica** - representa uma característica candidata ou uma característica e

possui os atributos nome, grupo, uma lista de objetos **CaracteristicaComentario** e uma lista de objetos **Opiniao**;

- **CaracteristicaComentario** - representa o relacionamento entre uma característica e um comentário e possui os atributos que indicam as posições inicial e final da características dentro do comentário;
- **Comentario** - retrata um comentário já totalmente preparado e pronto para ser processado pelos módulos extratores, e possui atributos que armazenam o conteúdo do comentário original, o conteúdo após o pré-processamento e o conteúdo após o processamento dos extratores;
- **ComentarioBase** - representa o comentário não processado que serve de entrada para o Sentimentalista;
- **Opiniao** - simboliza uma opinião expressa sobre uma característica, possui os atributos nome, nota positiva e nota negativa;
- **Produto** - representa uma entidade, possui os atributos nome, uma lista de objetos **Comentario** e uma lista de objetos **Caracteristica**.

Estas classes caracterizam os termos do domínio e as relações entre eles apresentados na subseção **4.1.3**.

O segundo conjunto de classes da camada de modelo é composto pelas classes:

- **ExtratorDeCaracteristica** - executa todas as tarefas do módulo Extrator de Característica e também as tarefas do módulo Seletor de Características Frequentes. Possui como atributos uma árvore Trie de grupos de características, uma lista de Strings que armazenam as *tags* de característica escolhidas pelo usuário e uma instância da classe **SentimentalistaUtils**;
- **ExtratorDeOpiniao** - realiza as tarefas do módulo Extrator e Classificador de Opiniões. Possui como atributos uma lista de Strings que armazenam as *tags* de opinião escolhidas pelo usuário, uma árvore Trie com notas para adjetivos, uma árvore Trie com notas para advérbios, uma árvore Trie com notas para verbos e uma instância da classe **SentimentalistaUtils**;
- **PreProcessador** - executa as regras de negócio que preparam o comentário base e o transformam em um comentário. Possui os atributos *POSTaggerME*, *Sentence-*

DetectorME, *TokenizerME* - que são os artefatos de entrada *POSTagger*, *SentenceDetector* e *Tokenizer*, e uma árvore Trie de *stopwords*;

- **SentimentalistaUtils** - é uma interface que possui métodos que customizam o Sentimentalista. Estes métodos são responsáveis por definir quais rótulos são considerados adjetivos, advérbios ou verbos no momento da busca de notas para as opiniões, também são responsáveis por definir rótulos especiais e criar apelidos para os rótulos do *Tagset* utilizado.

A camada de controle possui apenas uma classe nomeada **Sentimentalista**. Esta classe é responsável por garantir o fluxo de dados e de controle do *framework*. Para isto, ela utiliza as classes da camada de modelo e designa quando cada método deve ser executado. A classe **Sentimentalista** também é responsável por se comunicar com a camada de visão e enviar a ela os dados que serão exibidos ao usuário.

A camada de visão encerra a classe **SentimentalistaView** que utiliza o *framework Swing* para exibir telas com os níveis de sumarização para o usuário final do Sentimentalista.

A camada Dao engloba as interfaces que são responsáveis pelo armazenamento e manipulação dos dados adquiridos durante a execução dos módulos Pré-Processador, Extrator de Características e Extrator e Classificador de Opiniões. Decidiu-se por utilizar interfaces na camada Dao para permitir que o usuário do Sentimentalista escolha como deseja persistir seus dados. As interfaces são **CaracteristicaDao**, **ComentarioDao**, **OpiniaoDao** e **ProdutoDao**. O Sentimentalista traz implementações padrão destas classes utilizando o *driver* JDBC.

As definições das classes e interfaces e o diagrama de classes do Sentimentalista se encontram no Apêndice A.

4.4 Instanciação do Sentimentalista

A terceira etapa do desenvolvimento de um *framework* é a etapa de instanciação. Nesta etapa, o usuário implementa e configura o *framework* para atender a seus requisitos específicos. Esta seção apresenta um *cookbook*, cujo objetivo é exemplificar e explicar o processo de instanciação do *framework*. Para tanto serão utilizados os três casos de uso apresentados no Capítulo 4 como exemplos para a análise de domínio.

4.4.1 Cookbook

O uso de *cookbooks* é uma das técnicas mais utilizadas para exemplificar como instanciar *frameworks* (MARKIEWICZ; LUCENA, 2001). *Cookbooks* apresentam receitas para problemas comuns e padrões que o usuário do *framework* encontrará e como utilizar o *framework* nestas situações, não se preocupando em explicar o funcionamento interno do *framework*. Através das receitas fornecidas, o usuário poderá se guiar para criar sua própria aplicação utilizando o Sentimentalista.

A Figura 21 apresenta o diagrama de atividades do *framework*. Seguindo este diagrama o usuário será capaz de acompanhar o processo de instanciação.

A primeira etapa de instanciação do *framework* é uma etapa de preparação. Nesta etapa o usuário deve escolher com qual entidade deseja trabalhar. O Sentimentalista considera que o usuário está trabalhando apenas com uma única entidade, não possuindo nenhuma técnica ou ferramenta para diferenciar entidades dentro do conjunto de comentários. Após decidir sobre a entidade, o usuário deve decidir em qual idioma ele irá realizar a análise de sentimentos. O próximo passo é coletar comentários, *reviews* e textos sobre a entidade no idioma escolhido.

Em seguida, o usuário precisa escolher arquivos de configuração no idioma escolhido. Estes arquivos são: arquivo de configuração do *Tokenizer*, arquivo de configuração do *Sentence Detector* e arquivo de configuração do *POS Tagger*. Estes arquivos serão usados pelas ferramentas da OpenNLP utilizadas pelo Sentimentalista. O portal da OpenNLP fornece arquivos de configuração em alguns idiomas. A próxima tarefa que o usuário precisa realizar é decidir se deseja remover *stopwords*. Caso a resposta seja sim, ele deve montar uma lista de *stopwords* no idioma desejado, e com esta lista instanciar uma árvore Trie. Caso o usuário escolha não remover *stopwords*, ele deve instanciar uma árvore Trie vazia. Utilizando o *Tokenizer*, o *Sentence Detector*, o *POS Tagger* e a árvore Trie de *stopwords* o usuário é capaz instanciar a classe **PreProcessador**.

A segunda etapa de instanciação do *framework* foca nas interfaces que o usuário precisa implementar. Não existe uma ordem para a implementação das interfaces, podendo ser implementadas simultaneamente. O usuário deve implementar as interfaces do pacote **Dao**, que definem como os dados serão persistidos. As descrições das classes do capítulo anterior apresentam os métodos que o usuário deve implementar, e descreve o que cada um destes métodos precisa garantir e executar. A outra interface a ser implementada é a **SentimentalistaUtils**, mas para implementar seus métodos, que estão descritos no

Apêndice A, o usuário precisa estudar o *Tagset* do arquivo de configuração do *POS Tagger* escolhido.

Na terceira etapa o usuário prepara o Sentimentalista para extrair características. Para instanciar o **ExtratorDeCaracteristica**, o usuário precisa de uma árvore Trie com os grupos de características e uma lista de *tags* para representar as heurísticas que definem se um grupo de palavras é ou não uma característica, e uma instância da implementação da interface **SentimentalistaUtils**. A árvore Trie é criada a partir do conhecimento de um especialista sobre a entidade analisada. A lista de *tags* é criada a partir da análise já realizada do *Tagset* do arquivo de configuração do *POS Tagger* escolhido. Caso o usuário não deseje agrupar as características, ele precisa instanciar uma árvore Trie vazia.

Ao final da terceira etapa, o usuário precisa decidir se extrairá opiniões ou não, caso não deseje, ele pode pular para a última etapa de instanciação do *framework*, caso deseje ele vai para a etapa de instanciação do **ExtratorDeOpiniao**.

A quarta etapa é a etapa de instanciação do **ExtratorDeOpiniao**. Nesta etapa o usuário precisa escolher uma lista de palavras para criar árvores Trie com notas para as opiniões e definir as tags que irão decidir as palavras que formarão as opiniões. Não existe uma ordem para estas duas tarefas. Para criar as árvores Trie, o usuário precisa escolher uma base que contenha uma palavra, uma nota positiva e uma negativa. O **ExtratorDeOpiniao** espera como entrada três árvores Trie, uma com adjetivos, uma com advérbios e uma com verbos. O Sentimentalista apresenta a base SWN transformada em três árvores Trie, em dois idiomas, inglês e português. As *tags* de opinião seguem as regras apresentadas na seção anterior. Com as árvores criadas e as tags definidas, e uma instância da implementação da interface **SentimentalistaUtils**, o usuário pode instanciar o **ExtratorDeOpiniao**.

A última etapa tem como objetivo instanciar a classe Sentimentalista. Para executar esta etapa, as etapas 1,2 e 3 são obrigatórias, a etapa 4 é opcional. Para instanciar a classe Sentimentalista o usuário precisa de uma instância da classe Produto, representando a entidade a ser analisada, uma instância das implementações das interfaces do pacote **Dao**, uma instância de **ExtratorDeCaracteristica**, uma instância de **ExtratorDeOpiniao** se o usuário decidir por extrair opiniões, caso não seja de interesse do usuário, ele pode passar uma instância nula, uma instância de **PreProcessador** e um valor inteiro representando o limiar de frequência das características.

Após instanciar a classe Sentimentalista, o usuário precisa criar objetos **ComentarioBase** a partir dos comentários coletados na primeira etapa de instanciação do *framework* e

utilizar estes objetos como entrada para os métodos desejados da classe **Sentimentalista**.

A partir da execução dos métodos da classe **Sentimentalista**, o usuário já terá os resultados da análise de sentimentos persistidos e poderá criar visualizações para os dados, utilizando como referência os níveis de sumarização apresentados na seção anterior.

4.4.2 Casos de uso do Sentimentalista

Nesta seção, os casos de uso utilizados para a análise de domínio serão revisitados para exemplificar a instanciação do Sentimentalista.

4.4.2.1 Descobrindo características

A empresa *NewSmartPlus* está iniciando o planejamento para a criação de um novo *smartphone* de apelo popular e para isto precisam descobrir as características mais comentadas sobre *smartphones*.

Seguindo as etapas para instanciar o Sentimentalista, a primeira atividade é decidir a entidade, neste caso, *Smartphone*. O foco da empresa é o mercado norte americano e o idioma escolhido é o inglês. A empresa então precisa recolher um grande número de comentários sobre a entidade. Os arquivos de configuração escolhidos para o *Tokenizer*, *Sentence Detector* e *POS Tagger* para o idioma inglês foram adquiridos no portal da OpenNLP. Por último, é necessário montar uma árvore Trie de *stopwords*. A lista de *stopwords* foi adquirida através de uma busca pela internet. Com estes dados, a classe **PreProcessador** foi instanciada.

A próxima atividade é implementar as interfaces do pacote **Dao**. Decidiu-se por implementar as interfaces e métodos para persistir os dados utilizando a base MySQL através do driver JDBC. Para implementar a interface **SentimentalistaUtils**, foi necessário estudar o Tagset do *POS Tagger* escolhido. Com a análise do *Tagset* observou-se que os métodos com implementação padrão trabalham com o mesmo *Tagset* não sendo necessário alterá-los, e apenas os métodos obrigatórios foram implementados.

A terceira etapa é instanciar o **ExtratorDeCaracteristica**, para isto foi decidido que as características não seriam agrupadas e a árvore Trie de grupos foi passada vazia. Com a análise do **Tagset**, foram criadas as *tags* que servem de heurísticas para definir as características da entidade. A implementação da interface **SentimentalistaUtils** foi utilizada para instanciar o **ExtratorDeCaracteristica**.

A *NewSmartPlus* não tem interesse nas opiniões expressas sobre as características e pularam para a última etapa. Para instanciar a classe **Sentimentalista**, foram utilizadas as classes **PreProcessador** e **ExtratorDeCaracteristica**, as implementações do pacote **Dao**, e o valor inteiro 100, que indica que a empresa deseja conhecer todas as palavras que foram aceitas pelas *tags* definidas.

A última atividade foi transformar os comentários adquiridos pela empresa em uma lista de **ComentarioBase**, que foi passada como parâmetro de entrada do método **extrairCaracteristicas**. A partir deste ponto a *NewSmartPlus* tem acesso aos resultados da análise.

4.4.2.2 Opiniões sobre um produto

A construtora de carros *Fábrica de Carros* quer analisar a aceitação do público sobre um de seus modelos de veículos, o *Carro C2*, a entidade neste caso é o *Carro C2*. Seu público é brasileiro, portanto o idioma escolhido foi o português brasileiro.

Para instanciar a classe **PreProcessador** os arquivos de configuração do *Tokenizer* e do *Sentence Detector* escolhidos foram os oferecidos pelo portal da OpenNLP, e o arquivo de configuração do *POS Tagger* foi o Aelius *POS Tagger*. Uma árvore Trie de *stopwords* foi criada utilizando os conhecimentos de um professor de português.

A implementação das interfaces do pacote **Dao** foram feitas utilizando o driver JDBC e uma base MySQL. A implementação da interface **SentimentalistaUtils** foi realizada após a análise do *PALAVRAS Tagset* utilizado pelo *Aelius POS Tagger*. Este *Tagset* se encontra no Anexo C.

Na terceira etapa de instanciação do *framework*, a *Fábrica de Carros* decidiu agrupar as características. A árvore Trie de grupos de características foi criada utilizando o conhecimento especializado da empresa. A análise prévia do *Tagset* foi utilizada para as *tags* para representar as características.

A Fábrica de Carros quer extrair as opiniões expressas sobre as características do *Carro C2* sendo necessário realizar a quarta etapa de instanciação do Sentimentalista.

Para instanciar o **ExtratorDeOpiniao** foram definidas as *tags* que representam opiniões e foi decidido utilizar a base de palavras *SentiLexiconPT* em conjunto com a base SWN traduzida fornecida pelo Sentimentalista, e foram criadas as três árvores Tries necessárias.

Na última etapa as classes e interfaces instanciadas nas outras etapas são passadas para a classe **Sentimentalista**, em conjunto com o valor do limiar de frequência das características. A *Fábrica de Carros* quer conhecer as opiniões sobre todas as características escolhendo o valor 100 para o limiar de frequência. A última atividade a ser realizada é coletar comentários sobre o *Carro C2* e invocar o método *analisarSentimentos* passando como referência os comentários.

Após a execução do método os resultados da análise de sentimentos já estarão prontos para serem visualizados e estudados.

4.4.2.3 Sumário de Opiniões

Luíz Maurício possui um blog em inglês onde ele posta avaliações sobre hotéis ao redor do mundo. Várias entidades existem neste exemplo, portanto o Sentimentalista será instanciado diversas vezes, mas este processo é bem simples.

Neste caso não é possível definir de início uma entidade, mas pode-se definir o idioma. O blog de Luíz Maurício é escrito em inglês, então o idioma utilizado será o inglês. Os arquivos de configuração escolhidos foram os arquivos em inglês fornecidos no portal da OpenNLP. Luíz encontrou uma lista de *stopwords* fornecida por uma empresa chamada *NewSmartPlus*, e instanciou sua árvore Trie de *stopwords* com esta lista. Com estes dados ele instanciou a classe **PreProcessador**.

Luíz implementou as interfaces da camada **Dao** para se comunicarem com uma base de dados PostgreSQL, que é a base utilizada em seu blog. Ele teve que estudar o *Tagset* para implementar a interface **SentimentalistaUtils**.

Para definir sobre quais aspectos dos hotéis seus leitores comentam, o dono do blog teve que instanciar a classe **ExtratorDeCaracteristica**. Para isto, ele usou seu conhecimento para criar uma lista de características e grupos, e com esta lista ele instanciou a árvore Trie de grupos. Com a análise realizada para implementar a interface **SentimentalistaUtils**, Luíz definiu as *tags* que indicam quais palavras são características candidatas.

Luíz deseja exibir as opiniões sobre as características dos hotéis, para isso ele precisou instanciar a classe **ExtratorDeOpiniao**. Ele decidiu-se pela base SWN e instanciou as três árvores Trie necessárias e elaborou as *tags* que definem as opiniões com base no *Tagset* utilizado.

A última etapa foi instanciar a classe **Sentimentalista** utilizando as classes e interfaces instanciadas nas etapas anteriores e o valor de limiar passado foi 3, pois o dono do blog

considera que apenas as 3% características candidatas mais frequentes são características.

A diferença dos outros casos de uso, é que a classe **Sentimentalista** é instanciada com uma entidade diferente toda vez que é chamada. Como os comentários são postados frequentemente, toda vez que um novo comentário é postado é necessário chamar a classe **Sentimentalista** e chamar o método **carregarProduto**, e em seguida invocar o método **analisarSentimento** passando como parâmetro o novo comentário.

Luíz deseja apresentar aos seus leitores sumários. Para isto Luíz estudou os níveis de sumarização de opinião propostos no capítulo anterior. Ele decidiu por utilizar os níveis 1,3 e 5 de sumarização. Luíz desenvolveu páginas em PHP para exibir a sumarização de cada hotel.

4.5 Considerações finais

Neste capítulo foi apresentado o *framework* Sentimentalista. Todo o processo de projeto do *framework* foi explicado detalhadamente, desde a análise do domínio até a modelagem das classes do mesmo, permitindo que outros pesquisadores possam criar *frameworks* semelhantes em quaisquer linguagens de programação. Também foi apresentado um *cookbook* para auxiliar o processo de instanciação do Sentimentalista. As receitas apresentadas focam nas soluções mais comuns que os usuários do *framework* desenvolverão.

Ressalta-se que os comportamentos padrão do Sentimentalista foram pensados e projetados para exigir o menor esforço e conhecimento do usuário, com o objetivo de permitir que mais pessoas tenham acesso e possam trabalhar com análise de sentimentos.

Também é importante notar que usuários mais avançados e com conhecimentos mais sólidos na área de análise de sentimentos focada em aspectos podem estender o Sentimentalista, utilizando do polimorfismo para criar classes que herdam as classes **ExtratorDeCaracteristica** e **ExtratorDeOpiniao**, sobrescrevendo os métodos destas classes e passando instâncias destas novas classes para a classe **Sentimentalista**.

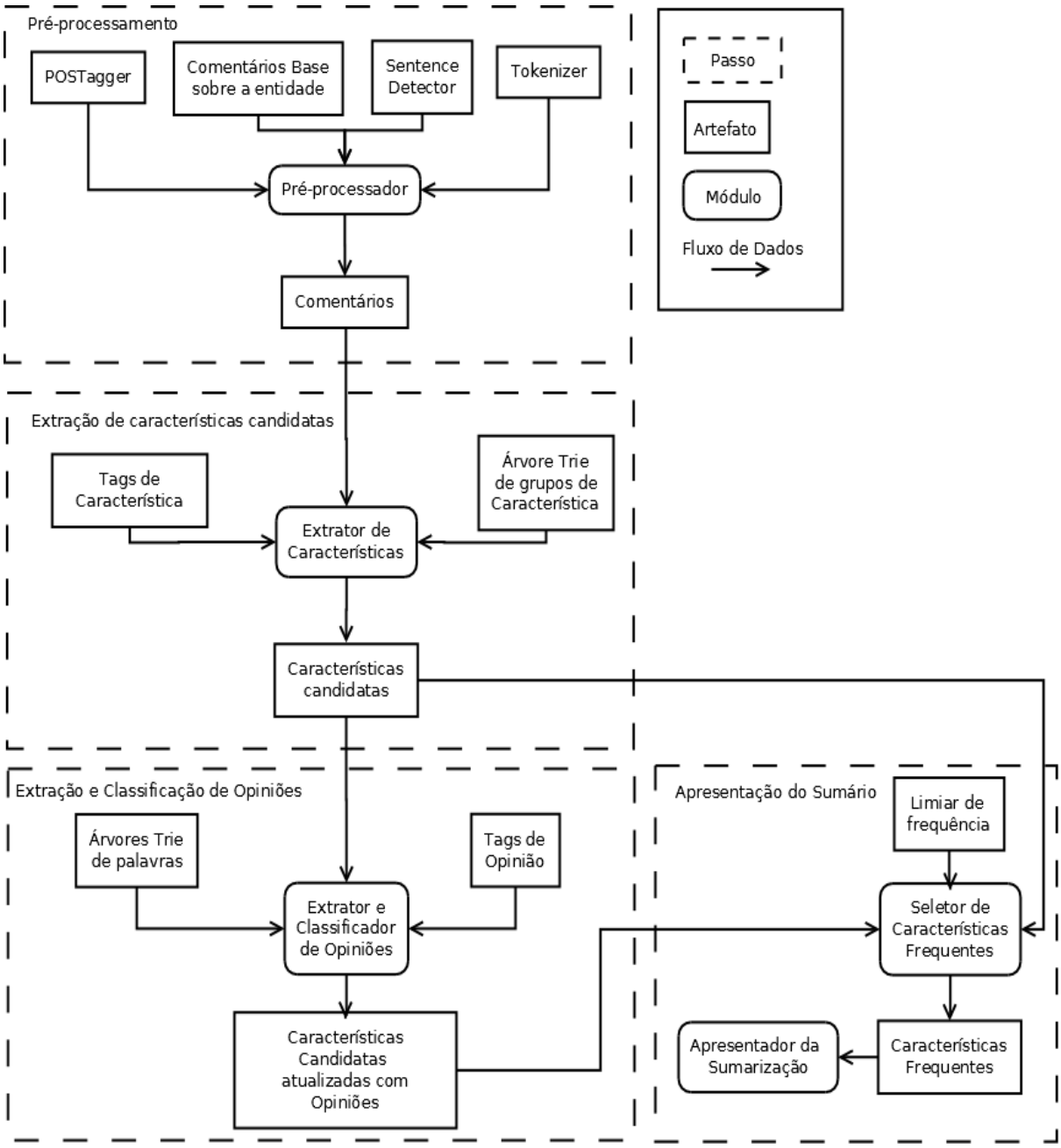


Figura 7: Arquitetura do Sentimentalista

Figura 8: Texto que compõe o comentário

Figura 9: Primeira comparação do exemplo

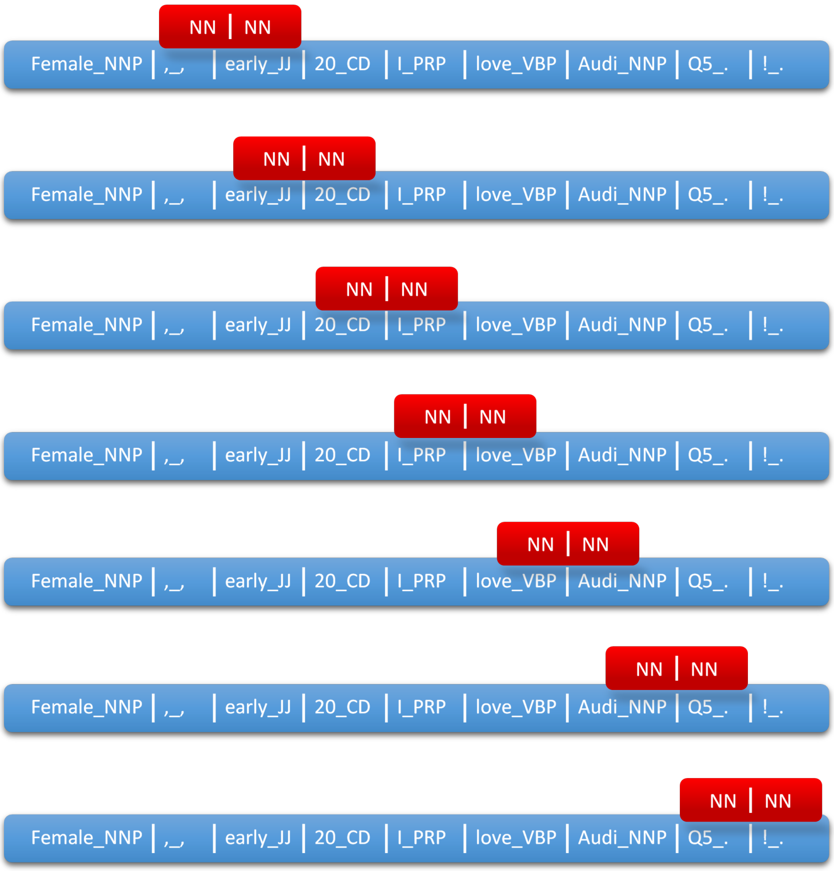


Figura 10: Iterações

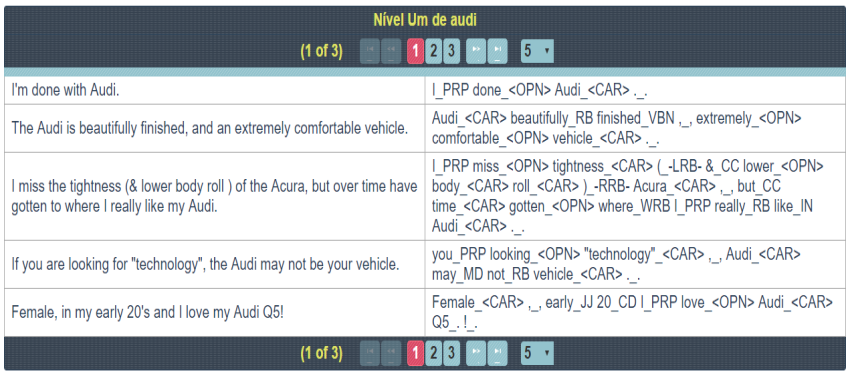
Figura 11: Iteração da *tag* [NN]

Figura 12: Procurando a *tag* [JJ]

Figura 13: Procurando a *tag* [VB]

Figura 14: Procurando a *tag* [JJ]

Figura 15: Procurando a *tag* [VB]



Nível Um de audi

(1 of 3) 1 2 3 5

I'm done with Audi.	I_PRP done_<OPN> Audi_<CAR> ._.
The Audi is beautifully finished, and an extremely comfortable vehicle.	Audi_<CAR> beautifully_RB finished_VBN ,_, extremely_<OPN> comfortable_<OPN> vehicle_<CAR> ._.
I miss the tightness (& lower body roll) of the Acura, but over time have gotten to where I really like my Audi.	I_PRP miss_<OPN> tightness_<CAR> (_-LRB- &_CC lower_<OPN> body_<CAR> roll_<CAR>)_-RRB- Acura_<CAR> ,_, but_CC time_<CAR> gotten_<OPN> where_WRB I_PRP really_RB like_IN Audi_<CAR> ._.
If you are looking for "technology", the Audi may not be your vehicle.	you_PRP looking_<OPN> "technology"_<CAR> ,_, Audi_<CAR> may_MD not_RB vehicle_<CAR> ._.
Female, in my early 20's and I love my Audi Q5!	Female_<CAR> ,_, early_JJ 20_CD I_PRP love_<OPN> Audi_<CAR> Q5_. !_.

(1 of 3) 1 2 3 5

Figura 16: Primeiro nível de sumarização da característica audi do Produto Audi Q5

Nível Dois de audi

(1 of 1) 1 5

love
not recommend driving
match
done
listening

(1 of 1) 1 5

Figura 17: Segundo nível de sumarização da característica audi do produto Audi Q5.

Nível Três de audi

(1 of 1) 1 5

love	1	0.625	-0.03125
not recommend driving	1	0.291667	-0.6
match	1	0.1	-0.05
done	1	0.0	0.0
listening	1	0.0	0.0

(1 of 1) 1 5

Figura 18: Terceiro nível de sumarização da característica audi do produto Audi Q5.

Nível quatro de audi

Female, in my early 20's and I love my Audi Q5!

Figura 19: Quarto nível de sumarização da característica audi do produto Audi Q5.

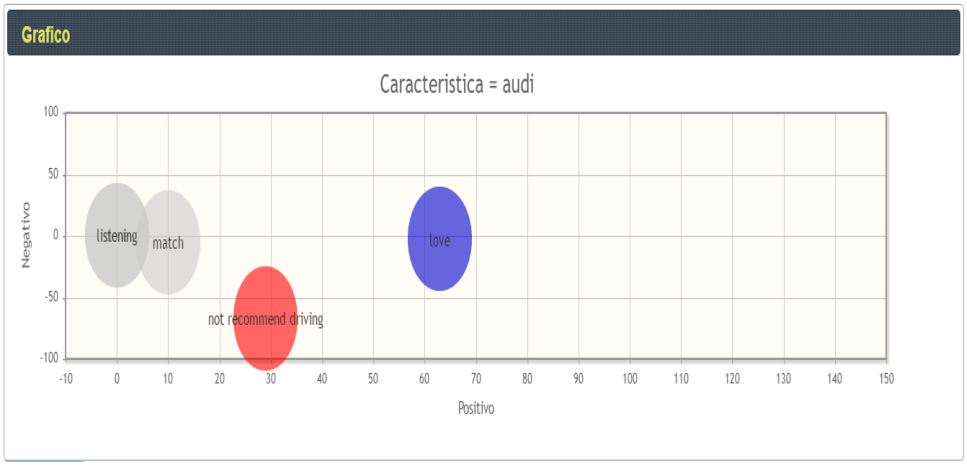


Figura 20: Quinto nível de sumarização da característica audi do produto Audi Q5.

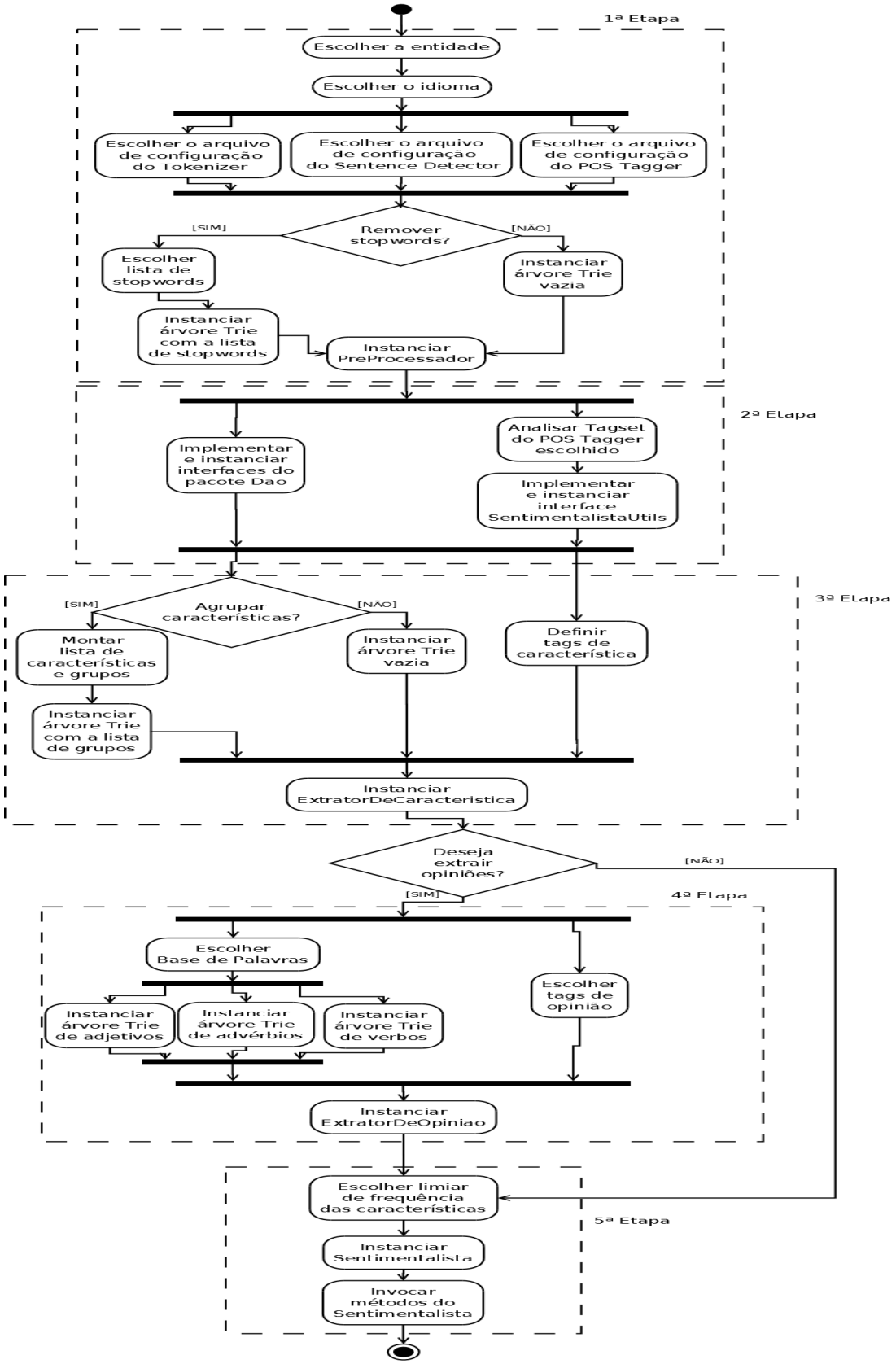


Figura 21: Diagrama de atividades do Sentimentalista

5 Experimentos e Resultados

Este capítulo apresenta os testes realizados para avaliar o método de classificação de sentimentos utilizado pelo Sentimentalista. Os testes aqui apresentados tem como objetivo comparar a classificação de sentimentos do Sentimentalista e outros métodos de classificação de sentimentos da literatura. Além disso, os testes foram executados buscando definir um conjunto de tags básico oferecido pelo framework, que auxilia usuários iniciantes na construção de uma aplicação a partir do Sentimentalista.

São descritos também os arquivos de configuração utilizados para se chegar aos resultados, permitindo que outros usuários consigam replicar os testes aqui realizados.

5.1 Metodologia

Para realizar a avaliação três bases distintas contendo comentários foram utilizadas. A primeira base é composta de *reviews* de filmes no idioma inglês, a segunda base é composta de *tweets* de assuntos diversos no idioma inglês e a terceira base é composta de *tweets* de assuntos diversos no idioma português brasileiro.

Foi implementada uma aplicação web em Java instanciando o *framework* Sentimentalista.

A aplicação de testes possui três páginas. A primeira página permite o cadastro de uma entidade que poderá ser analisada pelo Sentimentalista, esta página também apresenta uma tabela com todas as entidades já cadastradas, um botão para analisar comentários sobre a entidade, um botão para ver os resultados da análise e um campo de texto para o usuário indicar a porcentagem que indica o limiar das características frequentes.

A segunda página é a página da análise de sentimentos da entidade, ela é acessada quando o usuário clica no botão analisar da primeira página. Esta página possui uma caixa de texto onde o usuário pode inserir um ou mais comentários sobre a entidade analisada,

um campo de texto onde o usuário pode inserir *tags* que representam as heurísticas de características e um campo de texto onde o usuário pode inserir *tags* que representam as heurísticas de opiniões e um botão para dar inicio a análise.

A terceira página apresenta o resultado das análises de sentimentos, ela é acessada pelo botão ver análise na primeira página. Esta página possui uma tabela com as características que estão acima do limiar informado pelo usuário e os cinco níveis de sumarização propostos pelo Sentimentalista. Ao clicar em uma característica na primeira tabela, os níveis de sumarização são atualizados automaticamente através de uma requisição AJAX.

5.2 Instanciação do Sentimentalista

Esta seção descreve o processo de instanciação do Sentimentalista para a execução dos testes apresentados neste capítulo. A seção esta organizada seguindo o diagrama de atividades apresentado na Figura 21.

Para definir as *tags* de características e opiniões foram criados apelidos para os rótulos dos *Tagsets* utilizados. A Tabela 3 apresenta estes apelidos. Os *Tagsets* utilizados são apresentados no Anexo B e no Anexo C.

Tabela 3: Apelidos das tags

Apelido	tags
NN	NN; NNS; NNP; NNPS; PROP; SPEC; N
JJ	JJ; JJR; JJS; ADJ
VB	V; V-GER; VB; VBD; VBG; VBN; VBP; VBZ
RB	ADV; RB; RBR; RBS

5.2.1 1ª Etapa

Para a instanciação do Sentimentalista foram utilizadas 3 bases de comentários: base 1 - *reviews* de filmes em inglês; base 2 - *tweets* em inglês; base 3 - *tweets* em português. Estas bases são as entidades escolhidas para a análise.

Várias combinações de arquivos de configuração foram empregadas. Estes arquivos serão apresentados e receberão uma breve explicação e onde podem ser encontrados, para auxiliar novos usuários que desejem replicar os testes.

Os primeiros arquivos de configuração solicitados durante o processo de instanciação do Sentimentalista são os arquivos de configuração do POS Tagger. Foram utilizados 3

arquivos diferentes durante a execução dos testes: i) modelo treinado utilizando entropia máxima para o idioma inglês para as bases 1 e 2, disponível no site da OpenNLP, ii) modelo treinado utilizando entropia máxima para o idioma português para a base 3, disponível no site da OpenNLP e iii) modelo treinado utilizando entropia máxima para o idioma português brasileiro para a base 3 disponível em chamado de Aelius PosTagger [1]. Foram utilizados dois arquivos de configuração do *sentence detector*, um para o idioma inglês - para as bases 1 e 2 - e o outro para o idioma português - para a base 3-, ambos disponíveis no site da OpenNLP. Também foram utilizados dois arquivos de configuração do *tokenizer*, um para o idioma inglês - para as bases 1 e 2 - e o outro para o idioma português - para a base 3-, ambos disponíveis no site da OpenNLP.

Em todos os testes foram removidas *stopwords*. Duas árvores Trie de *stopwords* foram criadas, uma no idioma inglês para as bases 1 e 2, e outra no idioma português, para a base 3. Como não existe uma única lista de *stopwords*, as listas aqui criadas são apresentadas no Anexo A.

Após a escolha dos arquivos de configuração e das árvores Trie a classe PreProcessador foi instanciada.

5.2.2 2ª Etapa

Na 2ª Etapa do processo de instanciação apresentada na Figura 21 são instanciadas as classes do pacote DAO. Para estes testes foi utilizada a implementação padrão que utiliza JDBC para se conectar a um banco de dados MySQL. Com a análise dos *tagsets* dos arquivos de configuração dos *POSTaggers* utilizados, a interface SentimentalistaUtils foi implementada. Como os rótulos dos *tagsets* são diferentes, a interface possui apenas uma implementação para todas as bases.

5.2.3 3ª Etapa

Para a 3ª Etapa de instanciação é preciso decidir se ocorrerá agrupamento de características. Para os testes realizados decidiu-se não agrupar características, portanto a árvore Trie de grupos de características foi instanciada vazia. As *tags* de características escolhidas foram [NN,NN,NN], [NN,NN] e [NN] para todos os testes realizados em todas as bases. As *tags* levam em conta os apelidos apresentados na Tabela 3.

[1] http://aelius.sourceforge.net/manual.html

5.2.4 4ª Etapa

A 4ª Etapa vista na Figura 21 define como o Extrator de Opiniões é configurado.

A primeira configuração refere-se as bases de palavras. Três bases de palavras foram escolhidas para criar as árvores Trie utilizadas na classificação de sentimentos. Para as bases 1 e 2 foi utilizada a base SentiWordNet. Para a base 3 foram utilizadas uma base gerada a partir da tradução da SentiWordNet para o português brasileiro - a tradução foi realizada utilizando a API de tradução do Bing -, a base SentilexPt (SILVA; CARVALHO; SARMENTO, 2012) e a base OpLexicon (SOUZA; VIEIRA, 2012).

Nesta etapa também são definidas as *tags* de opinião. Para facilitar a explicação dos testes realizados, a seguir são apresentadas as combinações de *tags* de cada conjunto e o nome dado ao conjunto. As combinações levam em conta os apelidos apresentados na Tabela 3.

- JJ - O conjunto JJ apresenta apenas um adjetivo, a tag que compõe este conjunto é JJ;
- VB - O conjunto VB apresenta apenas um verbo, a tag que compõe este conjunto é VB;
- JJ-RB - O conjunto JJ-RB apresenta várias combinações de adjetivos, mas não traz nenhuma tag de advérbio, as tags que compõem este conjunto são JJ,JJ,JJ; JJ,JJ e JJ;
- VB-RB - O conjunto VB-RB apresenta várias combinações de verbos, mas não traz nenhuma tag de advérbio, as tags que compõem este conjunto são VB,VB,VB,VB; VB,VB,VB; VB,VB e VB;
- JJ+RB - O conjunto JJ+RB apresenta várias combinações de adjetivos e advérbios, as tags que compõem este conjunto são RB,JJ,JJ,JJ; JJ,JJ,JJ; RB,JJ,JJ; JJ,JJ; RB,JJ e JJ;
- VB+RB - O conjunto VB+RB apresenta várias combinações de verbos e advérbios, as tags que compõem este conjunto são VB,VB,VB,VB; VB,VB,VB; RB,VB,VB; VB,RB,VB; VB,VB,RB; VB,VB e VB.

5.2.5 5ª Etapa

Nesta etapa é instanciada a classe de controle Sentimentalista, como visto na Figura 21. Primeiramente definiu-se o limiar de frequência das características. Para facilitar a comparação com outros métodos o limiar escolhido foi 100%, ou seja, todas as características candidatas serão consideradas características. A última ação desta etapa é invocar o método que executa os processos do *framework*.

5.3 Resultados

Nesta seção são apresentadas as saídas de todas as etapas do Sentimentalista, e os resultados obtidos no módulo de extração e classificação de opiniões.

Foram selecionados 500 comentários da base 1, sendo 250 negativos e 250 positivos, 500 comentários da base 2, sendo 250 negativos e 250 positivos e 380 da base 3, sendo 190 negativos e 190 positivos. A base 3 possui menos comentários, por isto foi selecionado um menor número de comentários para os testes.

Os parâmetros de comparação utilizados foram a medida F1+, que indica a capacidade de predizer verdadeiros positivos, ou seja, classificar um sentimento como positivo quando ele realmente é positivo, a medida F1-, que indica a capacidade de predizer verdadeiros negativos, ou seja, classificar um sentimento como negativo quando ele realmente é negativo, o valor de macro F1, que indica a média do F1+ e F1-, e o valor de cobertura apresentado no trabalho de (ARAUJO et al., 2016), o valor de cobertura indica a porcentagem de comentários que foram classificados, tendo sido corretamente classificados ou não, ou seja, a porcentagem de comentários que recebeu como classe 1 ou -1, desconsiderando 0 e comentários não classificados. O valor de cobertura serve também para indicar quão confiáveis são os valores de F1, pois valores de F1 altos com uma baixa cobertura poderiam cair drasticamente se fossem considerados os outros elementos da base analisada (WANG et al., 2006).

Os resultados são apresentados seguindo a estrutura definida na Figura 7.

5.3.1 Pré-processamento

Na primeira etapa do Sentimentalista, o pré-processador foi instanciado com os arquivos de configuração e recebeu como entrada os comentários base mencionados na seção

5.2. A seguir são apresentados a base de dados utilizada, qual a combinação de arquivos de configuração e a saída do pré-processador para cada base.

- **Reviews de filmes em inglês** - os arquivos de configuração padrões no idioma inglês fornecidos pela OpenNlp foram utilizados, a árvore Trie de *stopwords* foi criada a partir da lista apresentada no Anexo A. O módulo retornou 500 comentários.
- **Tweets em inglês** - os arquivos de configuração padrões no idioma inglês fornecidos pela OpenNlp foram utilizados, a árvore Trie de *stopwords* foi criada a partir da lista apresentada no Anexo A. O módulo retornou 500 comentários.
- **Tweets em português** - os arquivos de configuração padrões no idioma português fornecidos pela OpenNlp foram utilizados, a árvore Trie de *stopwords* foi criada a partir da lista apresentada no Anexo A. O módulo retornou 380 comentários.
- **Tweets em português 2** - os arquivos de configuração padrões no idioma português fornecidos pela OpenNlp para o *Tokenizer* e para o *Sentence Detector* foram utilizados, o arquivo de configuração do *POSTagger* utilizado foi o fornecido pelo projeto Aelius POSTagger; a árvore Trie de *stopwords* foi criada a partir da lista apresentada no Anexo A. O módulo retornou 380 comentários.

5.3.2 Extração de Características

Os comentários produzidos pelo Pré-processador foram utilizados como entrada para o módulo de extração de características.

A saída do módulo para cada uma das bases testadas é descrito a seguir.

- **Reviews de filmes em inglês** - foram extraídas 1311 características candidatas únicas, considerando repetições foram extraídas 1658 características candidatas.
- **Tweets em inglês** - foram extraídas 1065 características candidatas únicas, considerando repetições foram extraídas 1401 características candidatas.
- **Tweets em português** - foram extraídas 936 características candidatas únicas, considerando repetições foram extraídas 1247 características candidatas.
- **Tweets em português 2** - foram extraídas 549 características candidatas únicas, considerando repetições foram extraídas 845 características candidatas.

É interessante notar que mesmo utilizando as mesmas *tags* de características nas bases de **Tweets em português** e **Tweets em português 2**, o fato do arquivo de configuração do *POSTagger* utilizado no pré-processamento ser diferente influi diretamente na quantidade de características candidatas extraídas. Na primeira base foram localizadas 936 características únicas, mas uma grande parte destas características são palavras que o *POS Tagger* utilizado não consegue classificar corretamente, já quando um *POS Tagger* mais especializado (*Aelius POS Tagger*) é empregado, ele classifica com uma maior exatidão os termos, encontrando menos características, mas com uma taxa de acerto maior.

5.3.3 Extração e Classificação de Opiniões

Para fins de comparação foram empregados os resultados apresentados por (ARAUJO et al., 2016). Neste trabalho os autores realizam testes empregando diferentes métodos de classificação de sentimentos, considerando vários idiomas.

Os primeiros testes foram realizados utilizando os arquivos de configuração básicos fornecidos pela OpenNLP para as três bases de comentários em conjunto com a base de palavras SentiWordNet, para as bases em inglês e a base SentiWordNet traduzida para a base em português. A Tabela 4 apresenta os resultados dos testes para base de *tweets* em inglês, a Tabela 5 apresenta os resultados dos testes para a base de *reviews* em inglês, e a Tabela 6 apresenta os resultados para a base de *tweets* em português.

Tabela 4: Tweets em Inglês

	JJ	VB	JJ + RB	VB + RB	JJ+RB + VB+RB	JJ - RB	VB - RB	JJ-RB + VB-RB
F1+	0.77	0.64	**0.79**	0.62	0.74	0.77	0.62	0.74
F1-	0.69	0.61	**0.74**	0.60	0.68	0.69	0.61	0.65
Macro F1	0.73	0.63	**0.76**	0.61	0.71	0.73	0.61	0.69
Cobertura	0.36	0.21	0.38	0.23	**0.54**	0.37	0.23	0.53

Tabela 5: Reviews em Inglês

	JJ	VB	JJ + RB	VB + RB	JJ+RB + VB+RB	JJ - RB	VB - RB	JJ-RB + VB-RB
F1+	0.65	0.64	**0.66**	0.63	0.65	0.65	0.64	0.64
F1-	0.55	0.48	**0.55**	0.49	0.53	0.54	0.46	0.52
Macro F1	0.60	0.56	**0.61**	0.56	0.59	0.60	0.55	0.58
Cobertura	0.66	0.19	0.69	0.21	**0.75**	0.67	0.20	0.72

Tabela 6: Tweets em Português

	JJ	VB	JJ + RB	VB + RB	JJ+RB + VB+RB	JJ - RB	VB - RB	JJ-RB + VB-RB
F1+	0.79	0.00	**0.79**	0.00	0.79	0.79	0.00	0.79
F1-	0.45	0.00	**0.49**	0.00	0.45	0.48	0.00	0.45
Macro F1	0.62	0.00	**0.64**	0.00	0.62	0.64	0.00	0.62
Cobertura	0.20	0.00	**0.23**	0.00	**0.23**	0.21	0.00	0.21

Analisando os resultados apresentados, é possível notar que o conjunto JJ+RB, que combina advérbios e adjetivos apresentam os maiores valores de F1+, F1- e Macro F1 nas três bases. A literatura confirma estes dados, onde adjetivos são citados e apresentados como as classes gramaticais com maior ocorrência de expressão de sentimentos sobre uma outra palavra.

Porém, o conjunto JJ+RB + VB+RB que combina adjetivos e advérbios com verbos e advérbios apresentam os maiores valores de cobertura, ou seja, quando são utilizadas estas tags uma maior quantidade de opiniões é classificada. A desvantagem em relação ao conjunto JJ+RB é uma queda nos valores de F1+, F1- e Macro F1. A diferença entre os valores dos conjuntos é pequena, tendo uma média de 0.05. A maior distinção é na primeira base, onde existe uma diferença de 0.16 entre a cobertura do conjunto JJ+RB e do conjunto JJ+RB + VB+RB.

É interessante notar que verbos ajudam a classificar mais opiniões, porém sozinhos, ou em conjunto apenas com advérbios, não apresentam bons resultados, tendo os conjuntos que utilizam verbos e verbos e adjetivos atingido o valor 0 na base em português. Isto pode ser ocasionado por uma menor quantidade de verbos pontuados dentro da base de palavras, e a tradução da SWN não ter sido muito eficiente nos verbos para a base em português.

Os maiores valores de F1+ ocorreram nas bases de *tweets* em inglês e *tweets* em português. O maior valor de F1- aconteceu na base de *tweets* em inglês. O maior valor do Macro F1 aparece na base de *tweets* em inglês. Por último, o maior valor de cobertura pertence a base de *reviews* em inglês. Estes valores demonstram que a combinação do *POS Tagger* padrão da *OpenNLP* e a base *SentiwordNet* possui uma maior chance de classificar corretamente uma opinião.

Comparando com os resultados obtidos por (ARAUJO et al., 2016) - presentes na Tabela 7 -, que também realizaram testes de classificação com a base SWN, percebe-se que a abordagem aqui apresentada possui resultados próximos. Isso se deve ao fato de que na

abordagem utilizada pelo Sentimentalista, só são recolhidas palavras que respeitam as heurísticas estipuladas pelas *tags* de opinião, e as *tags* de opinião são procuradas próximo a características.

Na abordagem empregada pelo Sentimentalista somente são consideradas as palavras que respeitam as heurísticas estipuladas pelas *tags* de opinião, que são procuradas próximo a características. Já na abordagem de (ARAUJO et al., 2016), todas as palavras da sentença são buscadas dentro da base e todos os comentários são traduzidos para o idioma inglês e submetidos aos algoritmos desenvolvidos para este idioma específico. O Sentimentalista não possui um idioma padrão, sendo mais genérico o que influi em seus resultados finais. Isto justifica a diferença nos resultados obtidos.

Tabela 7: Resultados obtidos por (ARAUJO et al., 2016)

Base	F1+	F1-	Macro F1	Cob
Tweets Português	**0.76**	**0.59**	**0.67**	**0.89**
Tweets Inglês	**0.6**	**0.67**	**0.64**	**0.92**
Reviews Inglês	**0.69**	**0.47**	**0.58**	**0.92**

A segunda bateria de testes consistiu em aumentar a árvore Trie de sentimentos utilizando as bases *SentilexPT* e *OpLexicon*, e utilizar o arquivo de configuração do *Aelius POSTagger* para analisar a base de *tweets* em português. A Tabela 8 apresenta os resultados destes testes.

Tabela 8: Tweets em Português - Aelius POSTagger e combinação de bases de palavras

	JJ	VB	JJ + RB	VB + RB	JJ+RB + VB+RB	JJ - RB	VB - RB	JJ-RB + VB-RB
F1+	0.79	**0.54**	0.79	**0.54**	0.66	0.80	**0.52**	0.64
F1-	0.57	**0.51**	0.59	**0.53**	0.56	0.60	**0.50**	0.53
Macro F1	0.68	**0.52**	0.69	**0.53**	0.61	0.70	**0.51**	0.59
Cobertura	0.18	**0.27**	0.20	**0.29**	0.41	0.18	**0.28**	0.39

Os resultados alcançados com a combinação das bases de palavras e o uso do *Aelius POSTagger* demonstram uma melhora significativa, principalmente quanto ao uso das *tags* que utilizam verbos sem adjetivos - conjuntos VB, VB + RB e VB - RB. Estes resultados demonstram que a escolha dos arquivos de configuração e das bases de dados afeta os resultados adquiridos pelo classificador de sentimentos, ou seja, a medida que o usuário adquira mais conhecimentos sobre a entidade que ele pretende analisar e sobre a área de análise de sentimento, ele pode aprimorar seus resultados obtidos através do Sentimentalista.

5.4 Considerações finais

Com os testes realizados neste capítulo, chegou-se a dois conjuntos básicos de *tags* que podem ser empregados por usuários iniciantes. Os conjuntos que obtiveram melhores resultados, considerando o equilíbrio entre as medidas F1+, F1-, Macro F1 e Cobertura, foram i) o conjunto JJ+RB + VB+RB que é composto por várias combinações de *tags* de adjetivos, advérbios e verbos, que apresenta um maior valor de cobertura com resultados menores nas medidas F1; ii) e o conjunto JJ+RB, que possui maiores valores de medida F1 com uma ligeira perda de cobertura. Os dois conjuntos possuem valores muito próximos, ficando a cargo do usuário decidir por uma cobertura maior ou uma melhor classificação das opiniões encontradas.

Os resultados obtidos também mostraram que o classificador de opiniões do Sentimentalista se encontra próximo aos classificadores apresentados na literatura, o que o torna viável para uso.

6 Conclusão

A análise de sentimento é uma área que foi alvo de diversas pesquisas e apresentou um crescimento gigantesco na última década. Os diversos trabalhos apresentaram muitas técnicas e ferramentas para auxiliar na coleta e apresentação de opiniões na web que irão auxiliar usuários no processo de decisão sobre produtos, empresas para entender os consumidores de seus produtos e melhorá-los, apresentar como a população se sente sobre um determinado candidato, etc.

A ferramenta proposta neste trabalho traz as seguintes contribuições:

- Um *framework* desenvolvido em Java, que realiza as 4 principais etapas da análise de sentimentos. As vantagens dentro de cada etapa são:
  - A etapa de extração de característica é a mais simples apresentada no trabalho. Esta etapa permite personalização através das *tags*, que representam a classe gramatical de cada palavra. A relação característica-opinião proposta permite que qualquer tipo de extração de características possa ser usado, desde que sejam fornecidas as posições inicial e final da característica extraída.
  - A etapa de extração de opiniões permite personalização através das *tags* que representam a classe gramatical de cada palavra. Com isto, qualquer combinação é possível, e estudos para avaliar qual a melhor combinação de *tags* para uma determinada língua e/ou contexto podem ser executados. Este módulo pode ser incrementado em qualquer outra ferramenta, desde que seja fornecida a posição inicial e final da opinião.
  - O módulo de classificação do sentimento apresenta o uso das árvores Trie dentro da análise de sentimento. Com isto, a busca pelas palavras dentro da base é mais rápida, já que o método de busca da árvore Trie é $O(1)$. Outra vantagem é que podem ser utilizadas diversas bases, desde que estejam no formato de árvore Trie. O *framework* também fornece todos os métodos para a geração de uma árvore Trie.

  - A etapa de sumarização também define níveis básicos que podem ser seguidos para a sumarização orientada a aspectos.

- Os resultados adquiridos demonstraram que os resultados do classificador de sentimentos do *framework* é compatível com os resultados encontrados na literatura, e como cada uma de suas etapas tem baixo acoplamento com as outras, outros pesquisadores podem escolher uma ou várias entre elas para realizar estudos na área.

- O Sentimentalista é independente de contexto e de idioma, sendo necessário apenas que seja configurado adequadamente.

À seguir são apresentadas melhorias que podem ser implementadas nos processos do *framework*. As melhorias estão relacionadas as etapas do *framework*.

- Pré-processamento

  - Implementação dos métodos de correção automática da árvore Trie, para auxiliar com o problema de palavras escritas de maneira incorreta.

- Extração de características

  - Uso de um *parser* que retorna além da classe gramatical, a função sintática das palavras. O maior problema do emprego de um *parser* é que não existe até o atual momento um *parser* funcional para muitas línguas, incluindo o português.

- Identificação e extração de opiniões

  - Criação de uma base de dados que apresente quais as melhores combinações de *tags* de acordo com o idioma desejado.

- Classificação de sentimento

  - Geração de um léxico no idioma português independente de domínio, que possua uma grande quantidade de palavras e termos.
  - Aplicação de técnicas de aprendizado de máquina nos dados gerados na etapa de extração de opiniões.

- Sumarização de opinião

– Gerar sumários textuais, como os vistos em (CARENINI; CHEUNG; PAULS, 2013), em conjunto com os níveis aqui propostos, com o objetivo de auxiliar ainda mais a tomada de decisão do usuário. E adicionar informações de frequência nestes sumários.

Referências

ADOMAVICIUS, G.; TUZHILIN, A. *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*. 2005. 734–749 p.

AGARWAL, A.; XIE, B.; VOVSHA, I. Sentiment analysis of twitter data. *Proceedings of the Association for Computational Linguistics*, p. 30–38, 2011. Disponível em: <http://dl.acm.org/citation.cfm?id=2021114>.

ALVES, A. L. F. et al. A Comparison of SVM Versus Naive-Bayes Techniques for Sentiment Analysis in Tweets : A Case Study with the 2013 FIFA Confederations Cup Categories and Subject Descriptors. p. 123–130, 2014.

ARAUJO, M. et al. An evaluation of machine translation for multilingual sentence-level sentiment analysis. In: *Proceedings of the 31st Annual ACM Symposium on Applied Computing - SAC '16*. New York, New York, USA: ACM Press, 2016. p. 1140–1145. ISBN 9781450337397. Disponível em: <http://dl.acm.org/citation.cfm?doid=2851613.2851817>.

BACCIANELLA, S.; ESULI, A.; SEBASTIANI, F. S ENTI W ORD N ET 3 . 0 : An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, v. 0, p. 2200–2204, 2010.

BECKER, K.; TUMITAN, D. Introdução à Mineração de Opiniões: Conceitos, Aplicações e Desafios. *Lectures of the 28th Brazilian Symposium on Databases*, p. 27–52, 2013.

BOYD, N. *Building object-oriented frameworks*. 1993. 1–23 p. Disponível em: <http://www.educery.com/papers/framewks/>.

CAMBRIA, E.; HAVASI, C.; HUSSAIN, a. SenticNet 2: A semantic and affective resource for opinion mining and sentiment analysis. *Proceedings of FLAIRS, Marco Island*, p. 202–207, 2012. Disponível em: <http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS12/paper/viewPDFInterstitial/4411/4794>

CARENINI, G.; CHEUNG, J. C. K.; PAULS, A. Multi-document summarization of evaluative text. *Computational Intelligence*, v. 29, n. 4, p. 545–576, 2013. ISSN 08247935.

CAVALCANTI, D. C.; PRUDÊNCIO, R. B. C. *Uma abordagem não supervisionada para classificação de opinião usando o recurso léxico SentiWordNet*. 111 p. Tese (Doutorado) — Universidade Federal de Pernambuco.

CLASTER, W. B.; HUNG, D. Q.; SHANMUGANATHAN, S. Unsupervised Artificial Neural Nets for Modeling Movie Sentiment. *2010 2nd*

International Conference on Computational Intelligence, Communication Systems and Networks, Ieee, p. 349–354, jul 2010. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5616452>.

DING, X. et al. A holistic lexicon-based approach to opinion mining. *Proceedings of the international conference on Web search and web data mining - WSDM '08*, p. 231, 2008. ISSN 12107816. Disponível em: <http://portal.acm.org/citation.cfm?doid=1341531.1341561>.

ELYASIR, A. M. H.; ANBANANTHEN, K. S. M. Opinion Mining Framework in the Education Domain. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*, v. 7, n. 4, p. 1049–1054, 2013.

ESPINOSA, O.; HENDRICKSON, C.; GARRETT J.H., J. Domain analysis: a technique to design a user-centered\nvisualization framework. *Proceedings 1999 IEEE Symposium on Information Visualization (InfoVis'99)*, 1999. ISSN 1522-404X.

FREDKIN, E. Trie Memory. *Communications of the ACM*, v. 3, n. 9, p. 490–499, 1960. ISSN 1098-6596.

FREITAS, L. A.; VIEIRA, R. Ontology based feature level opinion mining for portuguese reviews. In: *Proceedings of the 22nd International Conference on World Wide Web - WWW '13 Companion*. New York, New York, USA: ACM Press, 2013. p. 367–370. ISBN 9781450320382. Disponível em: <http://dl.acm.org/citation.cfm?doid=2487788.2487944>.

GAMON, M. et al. Pulse: Mining customer opinions from free text. *Lecture Notes in Computer Science*, v. 3646, p. 121–132, 2005. ISSN 0302-9743. Disponível em: <http://www.springerlink.com/index/94q1nrhfc8a4e8tn.pdf>.

GANAPATHIBHOTLA, M.; LIU, B. Mining opinions in comparative sentences. *Proceedings of the 22nd International Conference on Computational Linguistics - COLING '08*, v. 1, n. August, p. 241–248, 2008. Disponível em: <http://portal.acm.org/citation.cfm?doid=1599081.1599112>.

HAI, Z. et al. Identifying Features in Opinion Mining via Intrinsic and Extrinsic Domain Relevance. *IEEE Transactions on Knowledge and Data Engineering*, v. 26, n. 3, p. 623–634, mar 2014. ISSN 1041-4347. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6427744>.

HATZIVASSILOGLOU, V. et al. Predicting the semantic orientation of adjectives. *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages, p. 181, 1997. Disponível em: <http://portal.acm.org/citation.cfm?id=976909.979640>.

HU, M.; LIU, B. Mining opinion features in customer reviews. *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE (2004)*, p. 755–760, 2004. Disponível em: <http://www.aaai.org/Papers/AAAI/2004/AAAI04-119.pdf>.

KHAN, K. et al. Mining opinion from text documents: A survey. *2009 3rd IEEE International Conference on Digital Ecosystems and Technologies*, Ieee, p. 217–222, jun 2009. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5276756>.

KÖNING, A. C.; BRILL, E. Reducing the human overhead in text categorization. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, n. Figure 1, p. 598–603, 2006. Disponível em: <http://dl.acm.org/citation.cfm?id=1150474>.

LIU, B. Sentiment analysis: A multi-faceted problem. *IEEE Intelligent Systems*, v. 25, n. 3, p. 76–80, 2010. Disponível em: <http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Sentiment+Analysis+:+A+Multi-Faceted+Problem#0>.

LIU, B. Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, v. 5, n. 1, p. 1–167, 2012. ISSN 1947-4040.

MAHARANI, W. Microblogging sentiment analysis with lexical based and machine learning approaches. *2013 International Conference of Information and Communication Technology (ICoICT)*, Ieee, p. 439–443, mar 2013. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6574616>.

MARKIEWICZ, M. E.; LUCENA, C. J. P. de. Object oriented framework development. *Crossroads*, v. 7, n. 4, p. 3–9, 2001. ISSN 15284972.

MARTINEAU, J. et al. Delta TFIDF: An Improved Feature Space for Sentiment Analysis. *Proceedings of the Second International Conference on Weblogs and Social Media (ICWSM*, v. 29, n. May, p. 490–497, 2008. Disponível em: <http://ebiquity.umbc.edu/papers/select/person/Tim/Finin/>.

NARR, S.; HÜLFENHAUS, M.; ALBAYRAK, S. Language-independent Twitter sentiment analysis. *KDML workshop on knowledge discovery, data mining and machine learning*, 2012.

OFEK, N. et al. Improving Sentiment Analysis in an Online Cancer Survivor Community Using Dynamic Sentiment Lexicon. *2013 International Conference on Social Intelligence and Technology*, Ieee, p. 109–113, may 2013. Disponível em: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6545971>.

PANG, B.; LEE, L. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. 2004. ISSN 01403664. Disponível em: <http://arxiv.org/abs/cs/0409058>.

PANG, B.; LEE, L. Opinion Mining and Sentiment Analysis. *Foundations and Trends® in Information Retrieval*, v. 2, n. 1-2, p. 1–135, 2008. ISSN 1554-0669. Disponível em: <http://www.nowpublishers.com/product.aspx?product=INR&doi=1500000011>.

PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs up?: sentiment classification using machine learning techniques. *... -02 conference on Empirical methods ...*, p. 79–86, 2002. Disponível em: <http://dl.acm.org/citation.cfm?id=1118704>.

PAZIENZA, M. T.; LUNGU, I.; TUDORACHE, A. Flames recognition for opinion mining. *Economic Computation and Economic Cybernetics Studies and Research*, v. 3, 2011. ISSN 0424267X.

POPESCU, A.; ETZIONI, O. Opinion Mining: Extracting Product Feature Assessments from Reviews. 2005. Disponível em: <http://courses.cs.washington.edu/courses/cse590a/05sp/papers/opinion_mining.pdf>.

PORIA, S. et al. Enhanced senticnet with affective labels for concept-based opinion mining. *IEEE Intelligent Systems*, v. 28, n. 2, p. 31–38, 2013. ISSN 15411672.

PORIA, S. et al. Sentic Demo: A Hybrid Concept-Level Aspect-Based Sentiment Analysis Toolkit. *2014.Eswc-Conferences.Org*, n. 1, p. 1–5, 2014. Disponível em: <http://2014.eswc-conferences.org/sites/default/files/eswc2014-challenges_cl_submission_31.pdf>.

PRABOWO, R.; THELWALL, M. Sentiment analysis: A combined approach. *Journal of Informetrics*, v. 3, n. 2, p. 143–157, apr 2009. ISSN 17511577. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S1751157709000108>.

SILVA, M. J.; CARVALHO, P.; SARMENTO, L. Building a Sentiment Lexicon for Social Judgement Mining. In: *Lecture Notes in Computer Science (LNCS), International Conference on Computational Processing of the Portuguese Language (PROPOR)*. [S.l.]: Springer, 2012. p. 218–228.

SINGH, V. K. et al. Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification. In: *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*. IEEE, 2013. p. 712–717. ISBN 978-1-4673-5090-7. Disponível em: <http://iraj.in/up_proc/pdf/73-139996572180-82.pdf http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6526500>.

SIQUEIRA, H.; BARROS, F. A feature extraction process for sentiment analysis of opinions on services. *. . . of International Workshop on Web and Text . . .*, 2010. Disponível em: <http://www.labic.icmc.usp.br/wti2012/IIIWTI_camera_ready/74769.pdf>.

SOUZA, M.; VIEIRA, R. Sentiment Analysis on Twitter Data for Portuguese Language. In: *10th International Conference Computational Processing of the Portuguese Language*. [S.l.: s.n.], 2012.

TURNEY, P. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *Proceedings of the 40th annual meeting on association . . .*, n. July, p. 417–424, 2002. Disponível em: <http://dl.acm.org/citation.cfm?id=1073153>.

VALERIO, A.; SUCCI, G.; FENAROLI, M. Domain analysis and framework-based software development. *ACM SIGAPP Applied Computing Review*, v. 5, n. 2, p. 4–15, sep 1997. ISSN 15596915. Disponível em: <http://portal.acm.org/citation.cfm?doid=297075.297081>.

VILELA, P. d. C. S.; MILIDIU, R. L. *Classificação de sentimento para notícias sobre a Petrobras no mercado financeiro*. 46 p. Tese (Doutorado) — PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO - PUC-RIO, 2011.

WANG, D.; ZHU, S.; LI, T. SumView: A Web-based engine for summarizing product reviews and customer opinions. *Expert Systems with Applications*, Elsevier Ltd, v. 40, n. 1, p. 27–33, 2013. ISSN 09574174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2012.05.070>.

WANG, Z. Q. et al. An optimal SVM-based text classification algorithm. *Proceedings of the 2006 International Conference on Machine Learning and Cybernetics*, v. 2006, n. August, p. 1378–1381, 2006.

YESSENOV, K.; MISAILOVI, S. *Sentiment Analysis of Movie Review Comments*. [S.l.], 2009. 1–17 p.

APÊNDICE A – Classes do Sentimentalista

Neste apêndice são descritas as classes do Sentimentalista. A figura 22 apresenta o diagrama de classes do Sentimentalista. Como as classes estão organizadas em pacotes, elas serão apresentadas de acordo com o pacote em que estão. O primeiro pacote a ser descrito é o ***model***, onde estão as classes que representam os objetos do modelo do domínio e as classes que implementam os algoritmos, em seguida é apresentado o pacote ***dao***, que possui as interfaces que definem os métodos de acesso aos dados, o pacote ***trie***, que contém as classes que implementam a estrutura de dados árvore Trie, é apresentado em seguida, e por último o pacote ***controller***, que possui a classe Sentimentalista que traz as implementações do comportamentos padrões do *framework*.

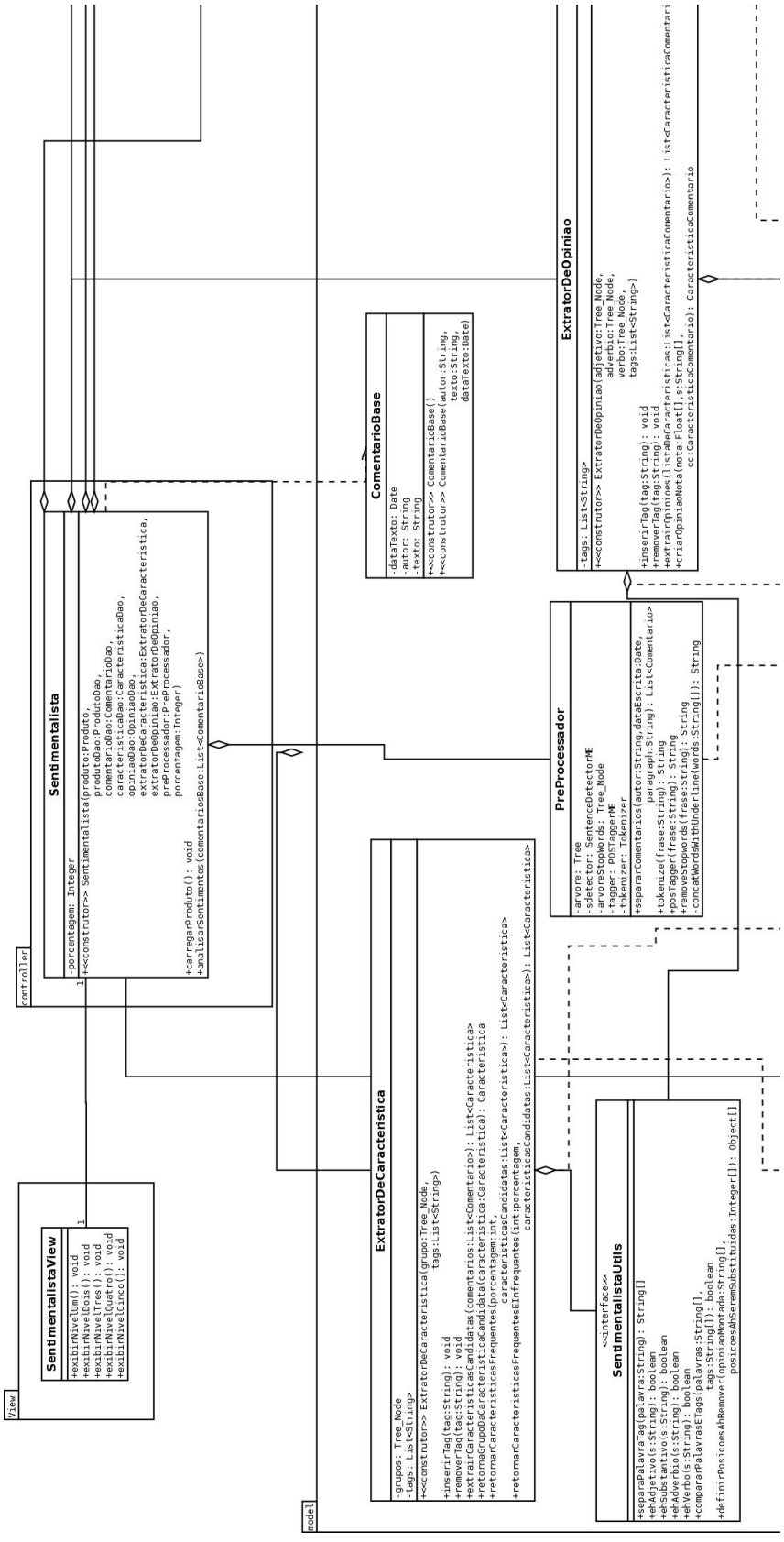
A.1 Model

A.1.1 Caracteristica

A classe Caracteristica representa as características candidatas e as características.

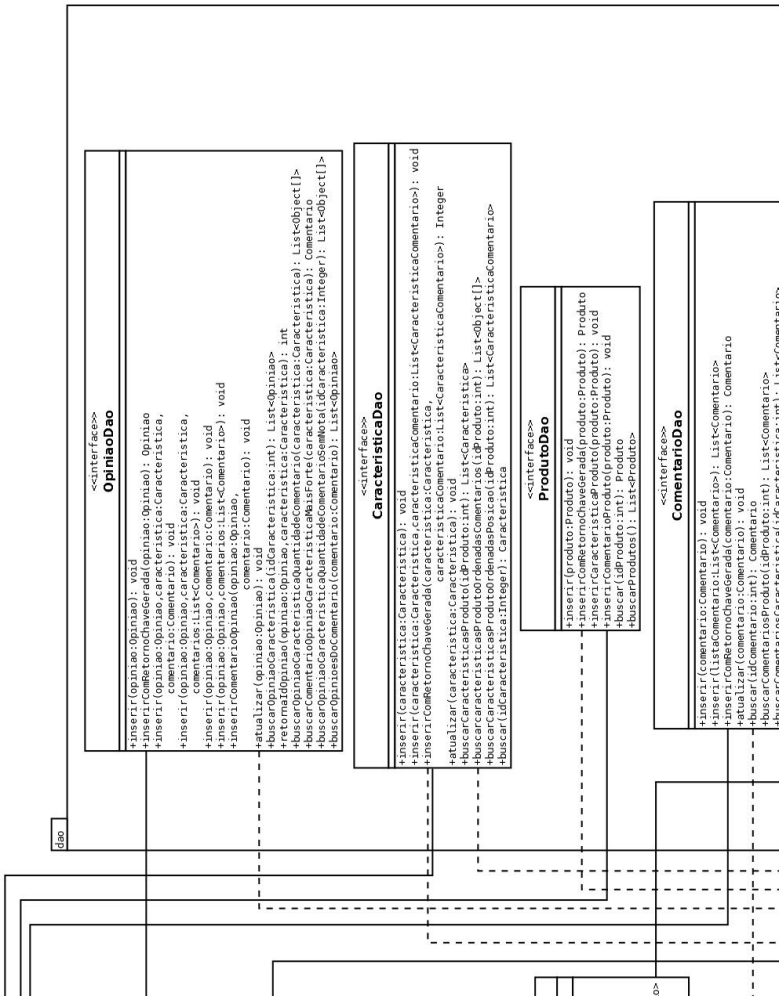
Atributos

- idCaracteristica - variável do tipo inteiro, utilizada para armazenar um código identificador único para a característica;
- nome - variável do tipo String, armazena o nome da característica;
- grupo - variável do tipo String, armazena o grupo da característica;
- comentariosCaracteristica - lista de objetos do tipo ComentarioCaracteristica, cada objeto ComentarioCaracteristica dentro desta lista indica a ocorrência desta característica dentro de um comentário;

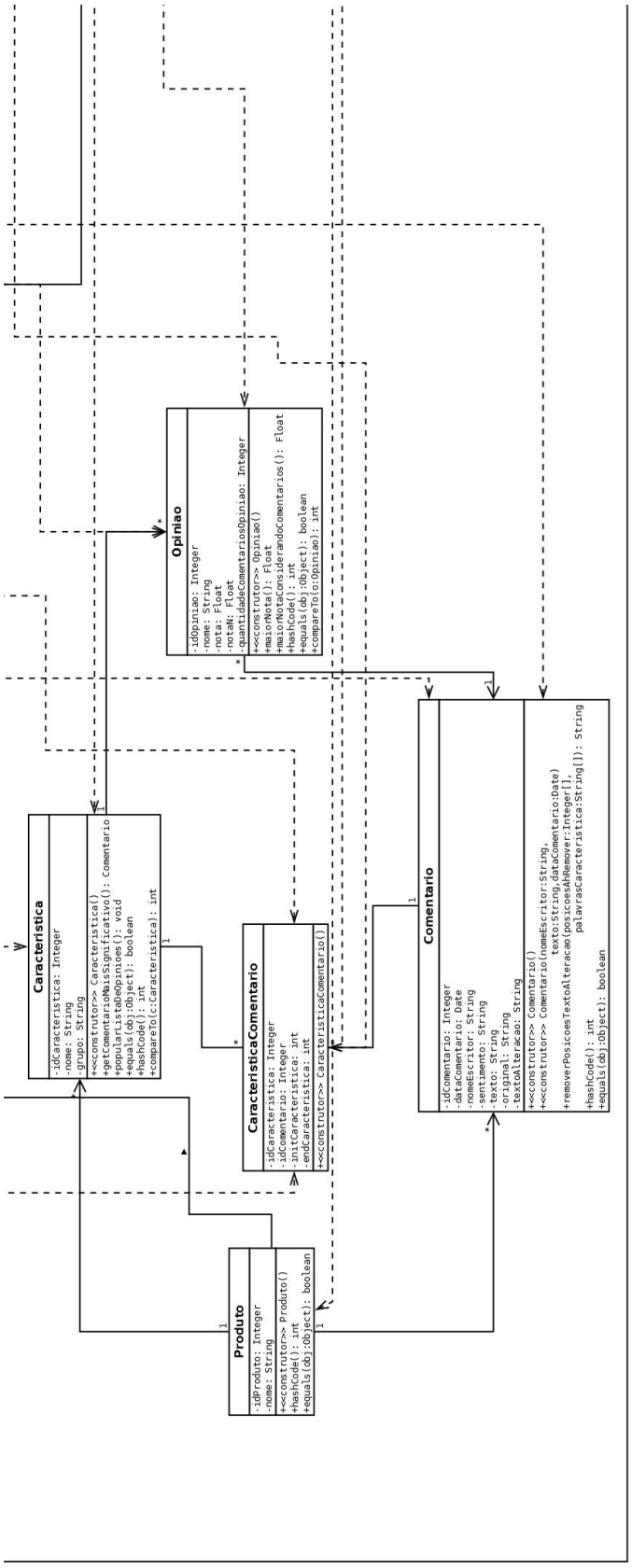


(a) Parte 1

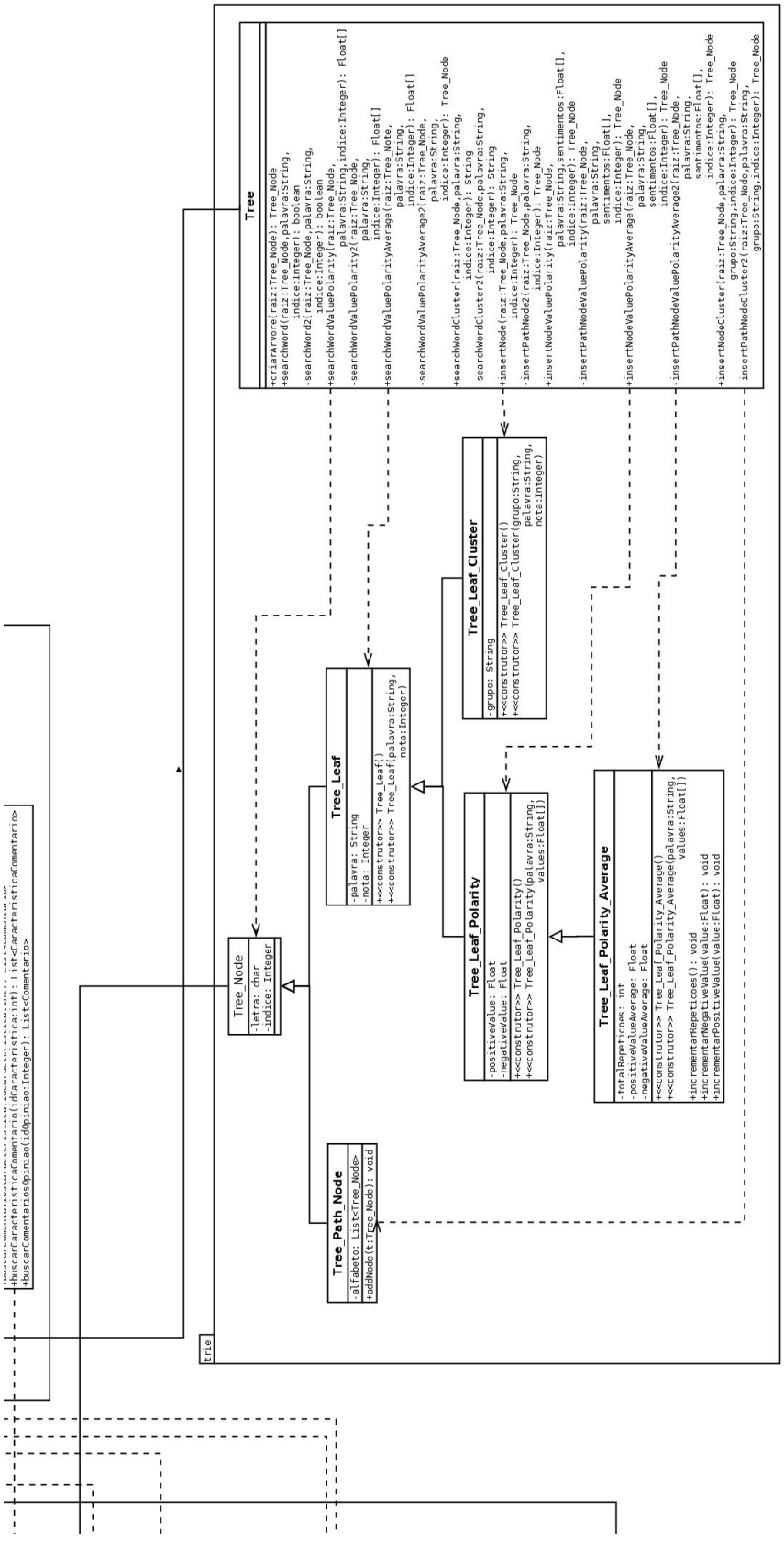
Figura 22: Diagrama de classes do Sentimentalista


dao
<<interface>>
OpiniaoDao
+inserir(opiniao:Opiniao): void
+inserirComRetornoChaveGerada(opiniao:Opiniao): Opiniao
+inserir(opiniao:Opiniao,caracteristica:Caracteristica,
comentario:Comentario): void
+inserir(opiniao:Opiniao,caracteristica:Caracteristica,
comentarios:List<Comentario>): void
+inserir(opiniao:Opiniao,comentario:Comentario): void
+inserir(opiniao:Opiniao,comentarios:List<Comentario>): void
+inserirComentarioOpiniao(opiniao:Opiniao,
comentario:Comentario): void
+atualizar(opiniao:Opiniao): void
+buscarOpiniaoCaracteristica(idCaracteristica:int): List<Opiniao>
+retornaIdOpiniao(opiniao:Opiniao,caracteristica:Caracteristica): int
+buscarOpiniaoCaracteristicaQuantidadeComentario(caracteristica:Caracteristica): List<Object[]>
+buscarComentarioOpiniaoCaracteristicaMaisForte(caracteristica:Caracteristica): Comentario
+buscarOpiniaoCaracteristicaQuantidadeComentariosEmMetade(idCaracteristica:Integer): List<Object[]>
+buscarOpinioesDoComentario(comentario:Comentario): List<Opiniao>
<<interface>>
CaracteristicaDao
+inserir(caracteristica:Caracteristica): void
+inserir(caracteristica:Caracteristica,caracteristicaComentario:List<CaracteristicaComentario>): void
+inserirComRetornoChaveGerada(caracteristica:Caracteristica,
caracteristicaComentario:List<CaracteristicaComentario>): Integer
+atualizar(caracteristica:Caracteristica): void
+buscarCaracteristicasProduto(idProduto:int): List<Caracteristica>
+buscarCaracteristicasProdutoOrdenadasComentarios(idProduto:int): List<Object[]>
+buscarCaracteristicasProdutoOrdenadasPosicao(idProduto:int): List<CaracteristicaComentario>
+buscar(idCaracteristica:Integer): Caracteristica
<<interface>>
ProdutoDao
+inserir(produto:Produto): void
+inserirComRetornoChaveGerada(produto:Produto): Produto
+inserirCaracteristicaProduto(produto:Produto): void
+inserirComentarioProduto(produto:Produto): void
+buscar(idProduto:int): Produto
+buscarProdutos(): List<Produto>
<<interface>>
ComentarioDao
+inserir(comentario:Comentario): void
+inserir(listaComentario:List<Comentario>): List<Comentario>
+inserirComRetornoChaveGerada(comentario:Comentario): Comentario
+atualizar(comentario:Comentario): void
+buscar(idComentario:int): Comentario
+buscarComentariosProduto(idProduto:int): List<Comentario>

(b) Parte 2


Opiniao
-idOpiniao: Integer
-nome: String
-nota: Float
-notaN: Float
-quantidadeComentariosOpiniao: Integer
+<<construtor>> Opiniao()
+maiorNota(): Float
+maiorNotaConsiderandoComentarios(): Float
+hashCode(): int
+equals(obj:Object): boolean
+compareTo(o:Opiniao): int
Caracteristica
-idCaracteristica: Integer
-nome: String
-grupo: String
+<<construtor>> Caracteristica()
+<<construtor>> Caracteristica()
+getComentariosCaracteristica(): Comentario
+popularListaDeOpinioes(): void
+equals(obj:Object): boolean
+hashCode(): int
+compareTo(c:Caracteristica): int
CaracteristicaComentario
-idCaracteristica: Integer
-idComentario: Integer
-initCaracteristica: int
-endCaracteristica: int
+<<construtor>> CaracteristicaComentario()
Comentario
-idComentario: Integer
-dataComentario: Date
-nomeEscritor: String
-sentimento: String
-texto: String
-original: String
-textoAlteracao: String
+<<construtor>> Comentario()
+<<construtor>> Comentario(nomeEscritor:String,
texto:String,dataComentario:Date)
+removerPosicoesTextoAlteracao(posicoesAhRemover:Integer[],
palavrasCaracteristica:String[]): String
+hashCode(): int
+equals(obj:Object): boolean
Produto
-idProduto: Integer
-nome: String
+<<construtor>> Produto()
+hashCode(): int
+equals(obj:Object): boolean

(c) Parte 3


Tree
Tree_Node
Tree_Leaf
Tree_Leaf_Cluster
Tree_Leaf_Polarity
Tree_Leaf_Polarity_Average
Tree_Path_Node
trie

(d) Parte 4

•opinioesCaracteristica - lista de objetos do tipo Opiniao, cada objeto Opiniao dentro desta lista indica uma opinião expressa sobre a característica.

Métodos

•Caracteristica() - método construtor da classe Caracteristica;

•getComentarioMaisSignificativo() - método que retorna um objeto do tipo Comentario, que é o comentário com a opinião com a maior nota (positiva ou negativa);

•popularListaDeOpinioes() - método sem retorno, utilizado para popular a variável opinioesCaracteristica com os objetos Opiniao que estão dentro dos objetos CaracteristicaComentario da variável comentariosCaracteristica;

•equals(Object obj) - recebe um objeto como parâmetro e retorna um valor booleano indicando a igualdade do objeto com a Caracteristica. Retorna verdadeiro se o objeto passado como parâmetro for uma instância de Caracteristica e as variáveis idCaracteristica dos dois forem iguais, retorna falso caso contrário;

•hashCode() - retorna um valor inteiro que representa o código hash da característica;

•compareTo(Caracteristica c) - recebe um objeto Caracteristica como parâmetro e retorna -1 se o objeto Caracteristica possuir mais comentários que o objeto de entrada, retorna 0 se o objeto Caracteristica tiver a mesma quantidade de comentários que o objeto de entrada e retorna 1 se o objeto Caracteristica possuir menos comentários que o objeto de entrada.

A.1.2 CaracteristicaComentario

A classe CaracteristicaComentario representa o relacionamento entre uma característica e um comentário.

Atributos

•idCaracteristica - variável de tipo inteiro que armazena o código identificador do objeto Caracteristica;

•idComentario - variável de tipo inteiro que armazena o código identificador do objeto Comentario;

- initCaracteristica - variável de tipo inteiro que armazena a posição inicial da característica dentro do comentário. Esta posição se refere as posições do atributo textoAlteracao encontrado na classe Comentario;
- endCaracteristica - variável de tipo inteiro que armazena a posição final da característica dentro do comentário. Esta posição se refere as posições do atributo textoAlteracao encontrado na classe Comentario;
- caracteristica - objeto do tipo Caracteristica que se relaciona com o objeto comentario do tipo Comentario;
- comentario - objeto do tipo Comentario que se relaciona com o objeto característica do tipo Caracteristica.

Métodos

- CaracteristicaComentario() - método construtor.

A.1.3 Comentario

Esta classe representa um comentário que já passou pelos métodos de pré-processamento da classe Pré-Processador.

Atributos

- idComentario - variável do tipo inteiro que armazena o código identificador do comentário;
- dataComentario - objeto do tipo java.util.Date que armazena a data que o texto foi escrito pelo autor;
- nomeEscritor - nome do autor do texto;
- original - sentença do texto original escrito pelo autor;
- texto - sentença do texto escrito pelo autor já processado pela classe PreProcessador;
- textoAlteracao - sentença do texto escrito pelo autor já processado pela classe Pre-Processador, esta variável será alterada pelas classes ExtratorDeCaracteristica e ExtratorDeOpiniao.

•caracteristicasComentario - lista de objetos CaracteristicaComentario, cada objeto na lista representa um relacionamento do comentário com uma característica.

Métodos

•Comentario(String nomeEscritor, String texto, Date dataComentario) - método construtor que recebe como parâmetros uma string contendo o nome do autor do texto, uma sentença do texto original e um objeto de java.util.Date com a data em que o comentário foi escrito;

•Comentario() - método construtor;

•removerPosicoesTextoAlteracao(Integer[] posicoesAhRemover, String[] palavrasCaracteristica) - este método recebe como parâmetros um vetor de inteiros e um vetor de strings, ele é responsável por substituir as palavras nas posições informadas no vetor de inteiros pelas palavras na mesma posição do vetor de Strings concatenada com o rótulo ?_<CAR>?. Retorna a variável textoAlteracao com as posições substituídas;

•equals(Object obj) - recebe um objeto como parâmetro e retorna um valor booleano indicando a igualdade do objeto com o Comentario. Retorna verdadeiro se o objeto passado como parâmetro for uma instância de Comentario e as variáveis idComentario dos dois forem iguais, retorna falso caso contrário;

•hashCode() - retorna um valor inteiro que representa o código hash da comentário;

A.1.4 ComentarioBase

Esta classe representa o comentário escrito originalmente pelo autor, sem nenhum processamento.

Atributos

•autor - variável do tipo string que armazena o nome do autor do texto;

•texto - variável do tipo string que armazena o texto original escrito pelo autor;

•dataTexto - objeto da classe java.util.Date que armazena a data em que o autor escreveu o texto.

Métodos A classe comentário base não possui nenhum método de interesse.

A.1.5 ExtratorDeCaracteristica

A classe ExtratorDeCaracteristica implementa os algoritmos de extração de características e de definição de características frequentes e infrequentes explicados no **Capítulo 4**. Esta classe é um ponto flexível do *framework*, sua configuração é feita através dos parâmetros informados em seu construtor.

Atributos

- grupos - instância de um objeto do tipo br.edu.unifei.sentimentalista.trie.Tree_Node, este objeto armazena a árvore Trie com os grupos de características;
- *tags* - uma lista de Strings, cada string é uma *tag* escolhida pelo usuário. O usuário deve utilizar a explicação do algoritmo de extração de características para criar suas *tags*.
- sentimentalistaUtils - instância de uma classe que implemente a interface br.edu.unifei.sentiment SentimentalistaUtils.

Métodos

- ExtratorDeCaracteristica(Tree_Node grupos, List<String> tags, SentimentalistaUtils sentimentalistaUtils) - método construtor, recebe como parâmetros de entrada uma instância de uma árvore Trie contendo os grupos de características, uma lista de Strings contendo as *tags* definidas pelo usuário e a instância de uma classe que implemente a interface SentimentalistaUtils;
- inserirTag(String tag) - método sem retorno, utilizado para inserir mais uma *tag* a lista de tags;
- removerTag(String tag) - método sem retorno, utilizado para remover uma *tag* da lista de tags, caso a mesma exista na lista;
- extrairCaracteristicasCandidatas(List<Comentario> comentarios) - principal método da classe ExtratorDeCaracteristica, este método recebe uma lista de objetos

Comentario e é responsável por encontrar as características candidatas que respeitem as *tags* definidas pelo usuário dentro dos comentários. O retorno deste método é uma lista de objetos Caracteristica;

•retornaGrupoDaCaracteristicaCandidata(Caracteristica caracteristica) - este método delega a instância da interface SentimentalistaUtils a busca pelo grupo da Caracteristica passada como parâmetro de entrada. O retorno deste método é um objeto Caracteristica;

•retornarCaracteristicasFrequentes(List<Caracteristica> caracteristicasCandidatas, int porcentagem) - este método recebe como entrada uma lista de objetos Caracteristica e um valor inteiro indicando qual a porcentagem das características são consideradas frequentes. Este método implementa o algoritmo explicado na seção yy. O retorno deste método é uma lista de objetos Caracteristica;

•retornarCaracteristicasFrequentesEInfrequentes(List<Caracteristica> caracteristicas-Candidatas, int porcentagem) - este método recebe como entrada uma lista de objetos Caracteristica e um valor inteiro indicando qual a porcentagem das características são consideradas frequentes. Após encontrar as características frequentes, este método busca as características infrequentes. Este método implementa o algoritmo explicado na seção yy. O retorno deste método é uma lista de objetos Caracteristica;

A.1.6 ExtratorDeOpiniao

A classe ExtratorDeOpiniao implementa os algoritmos de extração de opiniões e de classificação de sentimentos apresentados na seção zz. Esta classe é um ponto flexível do *framework*, sua configuração é feita através dos parâmetros informados em seu construtor.

Atributos

•adjetivo - instância de um objeto do tipo br.edu.unifei.sentimentalista.trie.Tree_Node, este objeto armazena a árvore Trie de adjetivos e notas de adjetivos;

•adverbio - instância de um objeto do tipo br.edu.unifei.sentimentalista.trie.Tree_Node, este objeto armazena a árvore Trie de advérbios e notas de advérbios;

•verbo - instância de um objeto do tipo br.edu.unifei.sentimentalista.trie.Tree_Node, este objeto armazena a árvore Trie de verbos e notas de verbos;

- *tags* - uma lista de Strings, cada string é uma *tag* escolhida pelo usuário. O usuário deve utilizar a explicação do algoritmo de extração de opiniões para criar suas *tags*.

- sentimentalistaUtils - instância de uma classe que implemente a interface br.edu.unifei.sentiment SentimentalistaUtils.

Métodos

- ExtratorDeOpiniao(Tree_Node adjetivo, Tree_Node adverbio, Tree_Node verbo, List<String> tags, SentimentalistaUtils sentimentalistaUtils) - método construtor, recebe como parâmetros de entrada uma instância de uma árvore Trie contendo os adjetivos e suas notas, uma instância de uma árvore Trie contendo os advérbios e suas notas, uma instância de uma árvore Trie contendo os verbos e suas notas, uma lista de Strings contendo as *tags* definidas pelo usuário e a instância de uma classe que implemente a interface SentimentalistaUtils;

- inserirTag(String tag) - método sem retorno, utilizado para inserir mais uma *tag* a lista de tags;

- removerTag(String tag) - método sem retorno, utilizado para remover uma *tag* da lista de tags, caso a mesma exista na lista;

- extrairOpinioes(List<CaracteristicaComentario> listaDeCaracteristicas) - principal método da classe ExtratorDeOpiniao, este método recebe uma lista de objetos CaracteristicaComentario e é responsável por encontrar as opiniões que respeitem as *tags* definidas pelo usuário dentro dos comentários. O retorno deste método é uma lista de objetos CaracteristicaComentario;

- criarOpiniaoNota(CaracteristicaComentario cc, String[] s, Float[] nota) - este método é responsável por montar a opinião utilizando os vetores de string e float passados como parâmetros de entrada e inseri-la no objeto Caracteristica do objeto CaracteristicaComentario informado como parâmetro de entrada. Este método retornar um objeto do tipo CaracteristicaComentario.

A.1.7 Opiniao

Esta classe representa o objeto do domínio de mesmo nome.

Atributos

- •idOpiniao - variável do tipo inteiro, utilizada para armazenar um código identificador único para a opinião;
- •nome - variável do tipo String, armazena o nome da opinião;
- •nota - variável do tipo Float, armazena a nota positiva da opinião;
- •notaN - variável do tipo Float, armazena a nota negativa da opinião;
- •comentariosOpiniao - lista de objetos Comentario. Cada objeto Comentario representa um comentário onde esta opinião é encontrada.
- •quantidadeComentariosOpiniao - variável do tipo inteiro utilizada para armazenar a quantidade de comentários em que a opinião se encontra.

Métodos

- •Opiniao() - método construtor;
- •MaiorNota() - retorna a maior nota da opinião, considerando o módulo da nota positiva e o módulo da nota negativa. Retorna um Float.
- •maiorNotaConsiderandoComentarios() - retorna a maior nota da opinião, considerando o módulo da nota positiva e o módulo da nota negativa multiplicado pela quantidade de comentários. Retorna um Float.
- •equals(Object obj) - recebe um objeto como parâmetro e retorna um valor booleano indicando a igualdade do objeto com a Opiniao. Retorna verdadeiro se o objeto passado como parâmetro for uma instância de Opiniao e as variáveis idOpiniao dos dois forem iguais, retorna falso caso contrário;
- •hashCode() - retorna um valor inteiro que representa o código hash da opinião.

A.1.8 PreProcessador

A classe PreProcessador é a responsável por preparar os comentários base e transformá-los em comentários aceitos pelo Sentimentalista. O PreProcessador é um ponto flexível do *framework*, sua configuração é feita pelos objetos de entrada de seu construtor. Cada um

destes objetos deve ser instanciado pelo usuário utilizando os arquivos de configuração do idioma escolhido. O site da opennlp fornece arquivos de configuração em alguns idiomas, mas outros arquivos de configuração podem ser encontrados na internet.

Atributos

- sdetector - objeto da classe opennlp.tools.sentdetect.SentenceDetectorME, deve ser instanciado com um arquivo de configuração do idioma escolhido. Este objeto é utilizado para separa os comentários base em várias sentenças;
- tagger - objeto da classe opennlp.tools.postag.POSTaggerME, deve ser instanciado com um arquivo de configuração do idioma escolhido. Este objeto é utilizado para rotular as palavras das sentenças e acordo com sua classe gramatical;
- tokenizer - objeto da classe opennlp.tools.tokenize.Tokenizer, deve ser instanciado com um arquivo de configuração do idioma escolhido. Este objeto é utilizado para separar as palavras e os acentos das sentenças;
- arvoreStopWords - objeto da classe br.edu.unifei.sentimentalista.trie.Tree_Node que armazena a árvore Trie que contém as stopwords escolhidas pelo usuário.

Métodos

- PreProcessador(SentenceDetectorME sdetector, Tree_Node arvoreStopWords, POSTaggerME tagger, Tokenizer tokenizer) - método construtor, recebe como parâmetros de entrada um objeto de configuração SentenceDetectorME, uma árvore Trie contendo as stopwords, um objeto de configuração POSTaggerME e um objeto de configuração Tokenizer;
- separarComentarios(String autor, Date dataEscrita, String paragraph) - este método utiliza o objeto sdetector para separar um comentário em sentenças, cada sentença é então processada pelo objeto tokenizer, em seguida é rotulada pelo objeto posTagger e tem as stopwords removidas. Os parâmetros de entrada são: uma string com o nome do autor, um objeto java.util.Date com a data em que o texto foi escrito, e uma string com o texto escrito. O retorno deste método é uma lista de objetos Comentario, prontos para serem utilizados pelos outros métodos do Sentimentalista;

•tokenize(String frase) - método que utiliza o objeto tokenizer para separar as palavras de uma sentença das pontuações. Retorna uma string com a frase, onde as palavras e pontuações estão separadas;

•posTagger(String frase) - método que utiliza o objeto tagger para rotular as palavras de uma sentença de acordo com suas classes gramaticais. Retorna uma string com a frase, com todas as suas palavras rotuladas;

•removeStopwords(String frase) - método que utiliza o objeto arvoreStopWords para remover as stopwords de uma sentença. Retorna uma string com a frase sem nenhuma stopword;

A.1.9 Produto

A classe produto representa a entidade que é alvo da análise.

Atributos

•idProduto - variável do tipo inteiro que armazena o código identificador do produto;

•nome - variável do tipo string que armazena o nome do produto;

•caracteristicasDoProduto - lista de objetos Caracteristica;

•comentariosDoProduto - lista de objetos Comentario.

Métodos

•Produto() - método construtor;

•equals(Object obj) - recebe um objeto como parâmetro e retorna um valor booleano indicando a igualdade do objeto com o Produto. Retorna verdadeiro se o objeto passado como parâmetro for uma instância de Produto e as variáveis idProduto dos dois forem iguais, retorna falso caso contrário;

•hashCode() - retorna um valor inteiro que representa o código hash do produto;

A.1.10 SentimentalistaUtils

Esta interface traz diversos métodos que são utilizados nas classes ExtratorDeCaracteristica, ExtratorDeOpiniao e PreProcessador. Ela apresenta alguns métodos padrões (*default methods* do Java 8), que podem ser utilizados diretamente pelo usuário sem a necessidade de implementação, e métodos que precisam obrigatoriamente serem implementados pelo usuário.

Atributos Esta interface não possui atributos.

Métodos

- separaPalavraTag(String palavra) - este método recebe uma palavra rotulada e retorna uma vetor de string com a palavra na primeira posição e o rótulo na segunda posição. Este é um método padrão;
- retornaLabelGenerica(String tag) - este método recebe um rótulo e deve retornar um rótulo genérico para ele, ou o próprio rótulo. Ex. O usuário considera que os rótulos NN e NNP são a mesma coisa para a sua aplicação, portanto este método deve retornar o mesmo rótulo para os dois. Este método também pode ser usado para criar ?apelidos? para os rótulos, tornando mais fácil seu entendimento. Por exemplo, o usuário quer informar os rótulos como ?substantivo? e ?adjetivo? quando cria suas tags, mas o Tagset do POS Tagger escolhido trabalha apenas com NN e JJ, este método será responsável por esta conversão. Este método precisa ser implementado pelo usuário;
- ehAdjetivo(String s) - este método recebe um rótulo e retorna um booleano positivo se o rótulo for um adjetivo. O usuário precisa especificar quais as *tags* são adjetivos de acordo com o Tagset de seu POS Tagger. Este método precisa ser implementado pelo usuário;
- ehAdverbio(String s) - este método recebe um rótulo e retorna um booleano positivo se o rótulo for um advérbio. O usuário precisa especificar quais as *tags* são advérbio de acordo com o Tagset de seu POS Tagger. Este método precisa ser implementado pelo usuário;
- ehSubstantivo(String s) - este método recebe um rótulo e retorna um booleano positivo se o rótulo for um substantivo. O usuário precisa especificar quais as *tags*

são substantivo de acordo com o Tagset de seu POS Tagger. Este método precisa ser implementado pelo usuário;

•ehVerbo(String s) - este método recebe um rótulo e retorna um booleano positivo se o rótulo for um verbo. O usuário precisa especificar quais as *tags* são verbo de acordo com o Tagset de seu POS Tagger. Este método precisa ser implementado pelo usuário;

•compararPalavrasETags(String[] tags, String[] palavras) - este método recebe como parâmetros de entrada uma *tag* informada pelo usuário e os rótulos de um grupo de palavras, e deve retornar true se forem iguais. Neste método o usuário também pode definir regras para rótulos especiais que ele deseje utilizar.

Este método é padrão, e em sua implementação padrão ele traz os rótulos especiais "##"e "??";

•definirPosicoesAhRemover(Integer[] posicoesAhSeremSubstituidas, String[] opiniaoMontada, String[] palavrasAhSeremComparadas) - este método recebe como parâmetros de entrada um vetor de inteiros com as posições das palavras encontradas, as palavras encontradas que respeitam as *tags* informadas pelo usuário, e *tag* definida pelo usuário que gerou os dois parâmetros anteriores. Este método é utilizado para remover alguma palavra das palavras que foram encontradas na busca de *tags* dos métodos de extração, por exemplo, a implementação padrão deste método remove das palavras encontradas as que foram selecionadas pelo rótulo especial "##".

Este método retorna um vetor de Object, cuja primeira posição indica quais as posições serão realmente removidas, e a segunda posição indica como ficou o conjunto de palavras que serão utilizadas para montar uma característica ou opinião.

Este método é padrão;

•retornarGrupoDaPalavra(String palavra, Tree_Node raiz) - este método recebe uma palavra, um objeto Tree_Node que contém a árvore Trie com os grupos das características e retorna o grupo da palavra, ou caso ela não tenha grupo, retorna a própria palavra.

A.2 Dao

A.2.1 CaracteristicaDao

Interface que define os métodos de acesso a dados da classe Caracteristica.

Métodos

- inserir(Caracteristica caracteristica) - Este método deve persistir um objeto Caracteristica;
- inserir(Caracteristica caracteristica, List<CaracteristicaComentario> caracteristicaComentario) - Este método deve persistir um objeto Caracteristica e persistir uma lista de objetos CaracteristicaComentario com o id da Caracteristica persistida;
- inserirComRetornoChaveGerada(Caracteristica caracteristica, List<CaracteristicaComentario> caracteristicaComentario) - Este método deve persistir um objeto Caracteristica e persistir uma lista de objetos CaracteristicaComentario com o id da Caracteristica persistida e retornar o id criado;
- atualizar(Caracteristica caracteristica) - este método deve atualizar um objeto Caracteristica;
- buscarCaracteristicasProduto(int idProduto) - este método deve retornar todas as características de um produto;
- buscarCaracteristicasProdutoOrdenadasComentarios(int idProduto) - este método deve retornar todas as características de um produto, ordenadas pela quantidade de comentários em ordem decrescente. Retorna uma lista de vetores de Object, onde cada vetor tem como primeira posição uma característica, e segundo valor a quantidade de comentários;
- List<CaracteristicaComentario> buscarCaracteristicasProdutoOrdenadasPosicao(int idProduto) - retorna todas as características de um produto, ordenadas pela posição inicial das características nos comentários. Retorna uma lista de CaracteristicaComentario;
- Caracteristica buscar(Integer idCaracteristica) - retorna um objeto Caracteristica que tenha seu código identificador igual ao parâmetro de entrada;

A.2.2 ComentarioDao

Interface que define os métodos de acesso a dados da classe Comentario.

Métodos

- inserir(Comentario comentario) - este método deve persistir um objeto Comentario;
- inserir(List<Comentario> listaComentario) - este método deve persistir diversos objetos Comentario, e retornar uma lista com os objetos Comentario recebidos como parâmetro atualizados com os códigos gerados;
- inserirComRetornoChaveGerada(Comentario comentario) - este método deve persistir um objeto Comentario e retornar o objeto Comentario atualizado com o código gerado;
- atualizar(Comentario comentario) - este método deve atualizar um objeto Comentario;
- buscar(int idComentario) - este método deve retornar um objeto Comentario cujo código seja igual o parâmetro de entrada;
- buscarComentariosProduto(int idProduto); - este método deve retornar todos os comentários de um produto;
- buscarComentariosCaracteristica(int idCaracteristica) - este método deve retornar todos os comentários de uma característica;
- buscarCaracteristicaComentario(int idCaracteristica) - este método deve retornar uma lista de CaracteristicaComentario que representam o relacionamento da característica e um comentário;
- buscarComentariosOpiniao(Integer idOpiniao) - este método deve retornar todos os comentários em que a opinião passada por parâmetro ocorre.

A.2.3 OpiniaoDao

Interface que define os métodos de acesso a dados da classe Opiniao.

Métodos

- inserir(Opiniao opiniao) - este método deve persistir um objeto Opiniao;
- inserirComRetornoChaveGerada(Opiniao opiniao) - este método deve persistir um objeto Opiniao e retornar o objeto com o código gerado;
- inserir(Opiniao opiniao, Caracteristica caracteristica, Comentario comentario) - este método deve persistir um objeto Opiniao e persistir os relacionamentos entre este objeto e o objeto Caracteristica e com o objeto Comentario;
- inserir(Opiniao opiniao, Caracteristica caracteristica, List<Comentario> comentarios) - este método deve persistir um objeto Opiniao e persistir os relacionamentos entre este objeto e o objeto Caracteristica e com o todos os objetos Comentario passados como parâmetro;
- inserir(Opiniao opiniao, Comentario comentario) - este método deve persistir um objeto Opiniao e persistir o relacionamento entre este objeto e o objeto Comentario;
- inserir(Opiniao opiniao, List<Comentario> comentarios) - este método deve persistir um objeto Opiniao e persistir o relacionamento entre este objeto e todos os objetos Comentario passados como parâmetro;
- inserirComentarioOpiniao(Opiniao opiniao, Comentario comentario) - este método deve persistir o relacionamento o objeto Opiniao e o objeto Comentario;
- atualizar(Opiniao opiniao) - este método deve atualizar o objeto Opiniao;
- buscarOpiniaoCaracteristica(int idCaracteristica) - este método deve retornar todas as opiniões de uma característica;
- retornaIdOpiniao(Opiniao opiniao, Caracteristica caracteristica) - este método recebe um objeto Opiniao e um objeto Caracteristica, e deve retornar, caso exista, o código de uma opinião de mesmo nome que pertença a característica assada como parâmetro, se não existir tal opinião, retorna null;
- buscarOpiniaoCaracteristicaQuantidadeComentario(Caracteristica caracteristica) - este método deve retorna todas as opiniões de uma característica e a quantidade de comentários em que esta opinião ocorre;
- buscarComentarioOpiniaoCaracteristicaMaisForte(Caracteristica caracteristica) - este método deve retornar o comentário mais importante de uma característica;

•buscarOpinioesDoComentario(Comentario comentario) - este método deve retornar todas as opiniões que ocorrem dentro de um comentário.

A.2.4 ProdutoDao

Interface que define os métodos de acesso a dados da classe Produto.

Métodos

•inserir(Produto produto) - este método deve persistir um objeto Produto;

•inserirComRetornoChaveGerada(Produto produto) - este método deve persistir um objeto Produto e retornar o objeto Produto com o código gerado;

•inserirCaracteristicaProduto(Produto produto) - este método deve persistir o relacionamento entre um produto e suas características;

•inserirComentarioProduto(Produto produto) - este método deve persistir o relacionamento entre um produto e seus comentários;

•buscar(int idProduto) - este método deve retornar um produto cujo código identificador seja igual ao parâmetro informado;

•buscarProdutos() - este método deve retornar todos os produtos;

A.3 Trie

A.3.1 Tree

Esta classe implementa os métodos de inserção e busca das árvores Trie.

Atributos Esta classe não possui atributos.

Métodos

•criarArvore(Tree_Node raiz) - este método recebe um objeto Tree_Node e retorna um objeto Tree_Path_Node instanciado com todas as variáveis inseridas e pronto para ser utilizado como a raiz de uma árvore Trie.

•searchWord(Tree_Node raiz, String palavra) - este método recebe uma árvore Trie e uma palavra e chama o método privado searchWord2 e retorna um booleano verdadeiro caso a palavra exista na árvore;

•searchWord2 (Tree_Node raiz, String palavra, Integer indice) - este método recebe uma árvore Trie, uma palavra e um índice, e busca recursivamente a palavra dentro da árvore, retorna um booleano verdadeiro caso a palavra exista na árvore, e um booleano falso caso não exista.

•searchWordValuePolarity(Tree_Node raiz, String palavra) - este método recebe uma árvore Trie e uma palavra e chama o método privado searchWordValuePolarity2 e retorna um vetor de Float, onde a primeira posição do vetor é uma nota positiva, e a segunda uma nota negativa. Caso a palavra não exista retorna 0 nas duas posições;

•searchWordValuePolarity2(Tree_Node raiz, String palavra, Integer indice) - este método recebe uma árvore Trie, uma palavra e um índice, e busca recursivamente a palavra dentro da árvore, retorna um vetor de Float, onde a primeira posição do vetor é uma nota positiva, e a segunda uma nota negativa. Caso a palavra não exista retorna 0 nas duas posições;

•searchWordValuePolarityAverage(Tree_Node raiz, String palavra) - este método recebe uma árvore Trie e uma palavra e chama o método privado searchWordValuePolarityAverage2 e retorna um vetor de Float, onde a primeira posição do vetor é uma nota positiva, e a segunda uma nota negativa. Caso a palavra não exista retorna 0 nas duas posições;

•searchWordValuePolarityAverage2(Tree_Node raiz, String palavra, Integer indice) - este método recebe uma árvore Trie, uma palavra e um índice, e busca recursivamente a palavra dentro da árvore, retorna um vetor de Float, onde a primeira posição do vetor é uma nota positiva, e a segunda uma nota negativa. Caso a palavra não exista retorna 0 nas duas posições;

•searchWordCluster(Tree_Node raiz, String palavra) - este método recebe uma árvore Trie e uma palavra e chama o método privado searchWordCluster2 e retorna um booleano verdadeiro caso a palavra exista na árvore;

•searchWordCluster2 (Tree_Node raiz, String palavra, Integer indice) - este método recebe uma árvore Trie, uma palavra e um índice, e busca recursivamente a palavra dentro da árvore, retorna um booleano verdadeiro caso a palavra exista na árvore, e um booleano falso caso não exista.

•insertNode(Tree_Node raiz, String palavra) - este método recebe um objeto Tree_Node e uma palavra, este método chama o insertPathNode2 e retorna o objeto Tree_Node com a palavra inserida;

•insertPathNode2(Tree_Node raiz, String palavra, Integer indice) - este método recebe um objeto Tree_Node, uma palavra e um indice, e insere a palavra na árvore de forma recursiva e retorna a árvore com a palavra inserida;

•insertNodeValuePolarity(Tree_Node raiz, String palavra, Float[] sentimentos) - este método recebe um objeto Tree_Node, uma palavra e um vetor de Float com as notas positiva e negativa da palavra e invoca método privado insertPathNodeValuePolarity e retorna o objeto Tree_Node com a palavra inserida;

•insertPathNodeValuePolarity(Tree_Node raiz, String palavra, Float[] sentimentos, Integer indice) - este método recebe um objeto Tree_Node, uma palavra, um vetor de Float com as notas positiva e negativa, e um indice, e insere a palavra na árvore de forma recursiva e retorna a árvore com a palavra inserida;

•insertNodeValuePolarityAverage(Tree_Node raiz, String palavra, Float[] sentimentos) - este método recebe um objeto Tree_Node, uma palavra e um vetor de Float com as notas positiva e negativa da palavra, este método chama o insertPathNodeValuePolarity e retorna o objeto Tree_Node com a palavra inserida;

•insertPathNodeValuePolarityAverage2(Tree_Node raiz, String palavra, Float[] sentimentos, Integer indice) - este método recebe um objeto Tree_Node, uma palavra, um vetor de Float com as notas positiva e negativa, e um índice, e insere a palavra na árvore de forma recursiva e retorna a árvore com a palavra inserida. Se a palavra for inserida várias vezes, este método calcula a média das notas da palavra;

•insertNodeCluster(Tree_Node raiz, String palavra, String grupo) - este método recebe um objeto Tree_Node, uma palavra e uma string que representa o grupo da característica e chama o método privado insertPathNodeCluster2 e retorna a árvore com o grupo e com a palavra inseridos;

•insertPathNodeCluster2(Tree_Node raiz, String palavra, String grupo, Integer indice) - - este método recebe um objeto Tree_Node, uma palavra, uma string que representa o grupo da característica e um índice, e insere de forma recursiva a palavra na árvore, e na folha da árvore armazena o grupo da característica. O método retorna a árvore com o grupo e com a palavra inseridos;

A.4 Tree_Leaf

Esta classe representa a folha de uma árvore Trie, é uma subclasse de Tree_Node.

Atributos

- palavra - variável do tipo String que indica a palavra da folha;
- nota - variável do tipo inteiro que armazena uma nota para a palavra;

Métodos

- Tree_Leaf() - método construtor;
- Tree_Leaf(String palavra, Integer nota) - método construtor que recebe como parâmetros de entrada uma string que contém a palavra e uma nota;

A.5 Tree_Leaf_Cluster

Esta classe representa a folha de uma árvore Trie que possui um grupo para uma palavra, é uma subclasse de Tree_Leaf.

Atributos

- grupo - variável do tipo String que indica o grupo da palavra;

Métodos

- Tree_Leaf_Cluster() - método construtor;
- Tree_Leaf_Cluster(String grupo, String palavra) - método construtor que recebe como parâmetros de entrada uma string que contém o grupo da palavra e uma string que contém a palavra;

A.6 Tree_Leaf_Polarity

Esta classe representa a folha de uma árvore Trie que possui as notas positiva e negativa de uma palavra, é uma subclasse de Tree_Leaf.

Atributos

- positiveValue - variável do tipo Float que armazena a nota positiva da palavra;
- negativeValue - variável do tipo Float que armazena a nota negativa da palavra.

Métodos

- Tree_Leaf_Polarity() - método construtor;
- Tree_Leaf_Polarity(String palavra, Float[] notas) - método construtor que recebe como parâmetros de entrada uma string que contém o grupo da palavra e um vetor de duas posições de Float com as notas positiva e negativa da palavra, respectivamente.

A.7 Tree_Leaf_Polarity_Average

Esta classe representa a folha de uma árvore Trie que possui as notas médias positiva e negativa de uma palavra e a quantidade de vezes que esta palavra foi inserida na árvore, é uma subclasse de Tree_Leaf_Polarity. Esta classe foi criada especificamente para resolver o problema da base SentiWordNet possuir notas deferentes para a mesma palavra, com esta classe podemos utilizar a média das notas de uma mesma palavra.

Atributos

- positiveValueAverage - variável do tipo Float que armazena a média da nota positiva da palavra;
- negativeValueAverage - variável do tipo Float que armazena a média da nota negativa da palavra;
- totalRepeticoes - variável do tipo inteiro que armazena a quantidade de vezes em que a palavra foi inserida na árvore.

Métodos

- Tree_Leaf_Polarity_Average() - método construtor;

- Tree_Leaf_Polarity_Average(String palavra, Float[] notas) - método construtor que recebe como parâmetros de entrada uma string que contém o grupo da palavra e um vetor de duas posições de Float com as notas positiva e negativa da palavra, respectivamente;
- incrementarRepeticoes() - soma um na variável totalRepeticoes;
- incrementarNegativeValue(Float value) - adiciona o valor do parâmetro na variável negativeValue e recalcula o valor da variável negativevalueAverage;
- incrementarPositiveValue(Float value) - adiciona o valor do parâmetro na variável negativeValue e recalcula o valor da variável negativevalueAverage;

A.8 Tree_Node

Classe abstrata que representa um nó de uma árvore Trie.

Atributos

- letra - variável do tipo char que indica a letra da folha;
- indice - variável do tipo inteiro que armazena o índice do nó;

Métodos

- Tree_Node() - método construtor.

A.9 Tree_Path_Node

Esta classe representa um nó de encaminhamento de uma árvore Trie.

Atributos

- alfabeto - lista de objetos Tree_Node que representam um alfabeto.

Métodos

- Tree_Path_Node() - método construtor.
- Tree_Path_Node(List<Tree_Node> alfabeto) - método construtor, recebe como parâmetro uma lista de objetos Tree_Node;
- addNode(Tree_Node t) - método responsável por inserir um novo nó no alfabeto e mantê-lo organizado.

A.10 Controller

A.10.1 Sentimentalista

A classe Sentimentalista traz os comportamentos padrões do *framework*. Instanciando esta classe o usuário já estará apto para trabalhar com análise de sentimentos focada em aspectos.

Atributos

- produto - uma instância de Produto;
- produtoDao - uma instância de ProdutoDao;
- comentarioDao - uma instância de ComentarioDao;
- caracteristicaDao - uma instância de CaracteristicaDao;
- opiniaoDao - uma instância de OpiniaoDao;
- extratorDeCaracteristica - uma instância de ExtratorDeCaracteristica;
- extratorDeOpiniao - uma instância de ExtratorDeOpiniao;
- preProcessador - uma instância de PreProcessador;
- porcentagem - uma variável do tipo inteiro que define o limiar das características frequentes;

Métodos

- Sentimentalista(Produto produto, ProdutoDao produtoDao, ComentarioDao comentarioDao, CaracteristicaDao caracteristicaDao, OpiniaoDao opiniaoDao, ExtratorDeCaracteristica extratorDeCaracteristica, ExtratorDeOpiniao extratorDeOpiniao, PreProcessador preProcessador, Integer porcentagem) - método construtor que recebe como parâmetros de entrada uma instância de Produto, uma instância de ProdutoDao, uma instância de ComentarioDao, uma instância de CaracteristicaDao, uma instancia de OpiniaoDao, uma instância de ExtratorDeCaracteristica, uma instância de ExtratorDeOpiniao, uma instância de PreProcessador e um valor inteiro indicando o limiar das características frequentes;

- carregarProduto() - método responsável por carregar as características, comentários e opiniões relacionados ao produto.;

- analisarSentimentos(List<ComentarioBase> comentariosBase) - método responsável por realizar todo o processo de análise de sentimentos focada em aspectos. Recebe como parâmetro de entrada uma lista de ComentarioBase.

ANEXO A – Stopwords

A.1 Stopwords para o idioma inglês

a	below	’ll	i
about	between	’ ll	i’d
above	both	’s	i ’d
after	by	’ s	i’ d
again	could	here	i ’ d
against	did	here’s	i’ll
all	do	here’ s	i ’ll
am	does	here ’s	i’ ll
an	doing	here ’ s	i ’ ll
and	during	hers	i’m
any	each	herself	i ’m
are	few	him	i’ m
as	for	himself	i ’ m
at	from	his	’m
be	further	how	i’ve
because	had	how’s	i ’ve
been	has	how’ s	i’ ve
before	have	how ’s	i ’ ve
being	having	how ’ s	if

in	our	there' s	too
into	ours	there ' s	under
is	ourselves	these	until
it	out	they	up
it's	over	they'd	was
it 's	own	they 'd	were
it' s	same	they' d	what
it ' s	should	they ' d	what's
its	so	they'll	what 's
itself	some	they 'll	what' s
let's	such	they' ll	what ' s
let 's	than	they ' ll	when
let' s	that	they're	when's
let ' s	that's	they 're	when 's
me	that 's	they' re	when' s
most	that' s	they ' re	when ' s
my	that ' s	're	where
myself	the	they've	where's
of	their	they 've	where 's
off	theirs	they' ve	where' s
on	them	they ' ve	where ' s
once	themselves	've	which
only	then	this	while
or	there	those	who
other	there's	through	who's
ought	there 's	to	who 's

who' s	why's	with	yourself
who ' s	why 's	would	yourselves
whom	why' s	your	
why	why ' s	yours	

A.2 Stopwords para o idioma português

a	cujos	dos	fez
ainda	da	durante	faz
além	das	e	fazer
ambas	de	é	feito
ambos	dela	ela	fazendo
antes	dele	elas	ha
ao	deles	ele	há
aonde	demais	eles	isso
aos	depois	em	isto
após	desde	entao	logo
aquele	desta	então	mas
aqueles	deste	entre	mediante
as	dispõe	essa	mesma
assim	dispoe	essas	mesmas
com	dispoem	esse	mesmo
como	dispõem	esses	mesmos
contra	diversa	esta	na
cuja	diversas	estas	nas
cujas	diversos	este	nas
cujo	do	estes	nem

nesse	pois	quando	tambem
neste	por	quanto que	teu
no	porque	quem	teus
nos	portanto	quer	toda
o	propia	se	todas
os	própia	seja	todo
ou	própias	sendo	todos
outra	propias	seu	tua
outras	proprio	seus	tuas
outro	próprio	sob	tudo
outros	propios	sobre	um
pelas	próprios	sua	uma
pelo	quais	suas	umas
pelos	qual	tal	uns
perante	qualquer	também	

ANEXO B – Penn Treebank Tagset

Quadro 15: Tags e classes gramaticais do Penn Treeban Tagset

CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NNP	Proper noun, singular
NNPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PRP	Personal pronoun
PRP$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP$	Possessive wh-pronoun
WRB	Wh-adverb

ANEXO C – PALAVRAS Tagset

Quadro 16: Tags e classes gramaticais do PALAVRAS Tagset

N	Nouns
PROP	Proper nouns (names)
SPEC	Specifiers (defined as non-inflecting pronouns, that can't be used as prenominals): e.g. certain indefinite pronouns, nominal quantifiers, nominal relatives
DET	Determiners (defined as inflecting pronouns, that can be used as prenominals): – e.g. articles, attributive quantifiers
PERS	Personal pronouns (defined as person-inflecting pronouns)
ADJ	Adjectives (including ordinals, excluding participles which are tagged V PCP)
ADV	Adverbs (both 'primary' adverbs and derived adverbs ending in -mente)
V	Verbs (full verbs, auxiliaries)
NUM	Numerals (cardinals)
PRP	Preposition
KS	Subordinating conjunctions
KC	Coordinationg conjunctions
IN	Interjections
EC	Hyphen-separated prefix ("elemento composto", category being phased out)