CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS
Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U nº 198, de 14/10/2016
ASSOCIAÇÃO EDUCACIONAL LUTERANA DO BRASIL

Luan Gomes de Almeida Araújo

SENTIMENTALL VERSÃO 2: Desenvolvimento de Análise de Sentimentos em Python

Palmas – TO

2017

Luan Gomes de Almeida Araújo

SENTIMENTALL VERSÃO 2: Desenvolvimento de Análise de Sentimentos em Python

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Profa. M.e Parcilene Fernandes de Brito.

Palmas – TO

2017

Luan Gomes de Almeida Araújo

SENTIMENTALL VERSÃO 2: Desenvolvimento de Análise de Sentimentos em Python

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientadora: Prof.a M.e Parcilene Fernandes de Brito.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof.a M.e Parcilene Fernandes de Brito

Orientadora

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Jackson Gomes de Souza

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. M.e Fabiano Fagundes

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2017

RESUMO

O objetivo desse trabalho é implementar uma ferramenta que realiza Análise de Sentimentos de opiniões oriundas de sites da internet escritos na língua portuguesa. A análise de sentimentos utiliza Processamento de Linguagem Natural para extrair opiniões ou sentimentos de textos subjetivos, classificando-as como positivas ou negativas. Dessa forma, nesse trabalho são abordadas algumas técnicas de processamento de linguagem natural, análise de sentimentos e geração de léxico de sentimentos. Essas técnicas serão utilizadas para o desenvolvimento da segunda versão da ferramenta SentimentALL, que realiza análise de sentimentos em aspectos. Como estudo de caso, foram utilizados os dados coletados do site de turismo TripAdvisor. A versão 2 da SentimentALL é composta de três módulos: módulo de coleta de dados, módulo de pré-processamento e módulo de análise de sentimentos. Todos eles foram desenvolvidos na linguagem Python e, como suporte, foram utilizadas as API Scrapy, NLTK e MaltParser. A coleta de dados extraiu mais de seis milhões de comentários, que foram pré-processados realizando a normalização dos dados, correção ortográfica, *PoS tagging* e identificação de possíveis expressões multipalavras. Na análise de sentimentos foi utilizada a abordagem baseada em léxico de sentimentos, com os léxicos OpLexicon, SentiLex e LIWC fazendo a polarização das opiniões. Para a identificação dos aspectos relacionados a uma opinião, as relações sintáticas entre eles foram determinadas através do uso do grafo de dependência. Ao final de todo esse processo, é realizada uma avaliação de desempenho entre as duas versões da SentimentALL.

PALAVRAS-CHAVES: Análise de Sentimentos, Processamento de Linguagem Natural, Turismo.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CoGrOO - Corretor Gramatical acoplável

DS – Double Propagation

IDE – Integrated Development Environment

HTML - Hypertext Markup Language

kNN - k-Nearest Neighbors

MWE – Multiwords Expression

NLP – Natural Language Processing

NLTK – Natural Language Toolkit

PoS – Part-of-Speech

PLN – Processamento de Linguagem Natural

PMI – Pointwise Mutual Information

SQL – Structured Query Language

SVM - Support Vector Machine

UD – Universal Dependencies

SUMÁRIO

1 INTRODUÇÃO

As mídias sociais na internet, como blogs, fóruns de discussões, redes sociais, têm ampliado a forma como as pessoas têm acesso e geram informações. Com isso, a internet tem se tornado um importante espaço para que as pessoas possam expressar opiniões sobre os mais diversos assuntos. Isso gera uma grande quantidade de dados, que podem ser analisados para extrair algum tipo de informação relevante sobre algo. Liu (2010) diz que esses dados podem ser categorizados em dois tipos principais: fatos e opiniões. Fatos são expressões objetivas sobre algo, ou seja, representam um evento concreto que não depende do julgamento de um indivíduo. Opiniões são expressões subjetivas sobre algo, podem expressar sentimentos de um determinado indivíduo.

A busca por opiniões na internet adquire uma especial relevância quando se deseja tomar uma decisão sobre algo como, por exemplo, qual produto comprar ou em qual hotel se hospedar. Mas quando o volume de dados é muito grande, encontrar a informação desejada pode ser um problema. Para resolver esse tipo de desafio, podem ser utilizadas técnicas computacionais que têm a capacidade de auxiliar na descoberta de informações. Uma dessas técnicas é a análise de sentimentos ou (mineração de opiniões), que tem como objetivo extrair os sentimentos de opiniões sobre algo utilizando métodos computacionais. A partir dessa análise, as pessoas ou organizações podem se favorecer dos resultados obtidos para os mais diversos fins.

A análise de sentimentos pode ser realizada em diferentes granularidades: em documentos, sentenças e aspectos. A análise de sentimentos baseada em aspectos, que será utilizada nesse trabalho, identifica os aspectos (substantivos) de um texto e as palavras opinativas (adjetivos); e determina qual a polaridade (positiva ou negativa) de cada aspecto a partir das palavras opinativas relacionadas a ele.

A abordagem utilizada nesse trabalho para realizar a análise de sentimentos é a baseada em léxico. Essa abordagem utiliza um léxico de sentimentos, que define se uma palavra opinativa é positiva ou negativa, possibilitando determinar a polaridade de um aspecto. Os léxicos de sentimentos utilizados foram: SentiLex-PT, LIWC e OpLexicon.

A maioria das informações presentes na internet está na forma não estruturada e em linguagem natural. Portanto, é necessária a utilização de técnicas

de processamento de linguagem natural (PLN) para extrair as informações que estão na forma não estruturada, passando-as para a forma estruturada, com isso, auxiliando a tarefa de análise de sentimentos. Para realizar o PLN foi utilizada a biblioteca NLTK do Python.

A ferramenta desenvolvida nesse trabalho é parte do projeto SentimentALL de Brito *et al.* (2015) do grupo de pesquisa Engenharia Inteligente de Dados do CEULP/ULBRA, sendo a mesma, a versão 2 da ferramenta desenvolvida originalmente por Oliveira (2015). Nesse trabalho, foram desenvolvidos três módulos: módulo de coleta de dados, módulo de pré-processamento e módulo de análise de sentimentos. Na coleta de dados obteve-se mais de seis milhões de dados extraídos. Esses dados foram pré-processados utilizando técnicas de processamento de linguagem natural. O pré-processamento consistiu em realizar a normalização dos dados, correção ortográfica, identificação da classe morfológica das palavras e identificação de expressões. A análise de sentimentos utilizou a abordagem baseada em léxico utilizando os léxicos de sentimentos para português OpLexicon, SentiLex e LIWC.

A segunda versão da ferramenta SentimentALL tem a finalidade de realizar a análise de sentimentos em um contexto geral, porém como estudo de caso foi utilizado o contexto do turismo, sendo que os dados serão coletados do site de turismo TripAdvisor. Esse contexto foi escolhido, pois, a versão anterior da ferramenta também o utilizou e, com isso, podem ser realizadas comparações entre as versões. O modo como serão realizadas essas comparações está descrito na metodologia (seção 3).

O objetivo geral desse trabalho é implementar uma ferramenta que realiza Análise de Sentimentos de opiniões oriundas de sites da internet escritos na língua portuguesa. Os objetivos mais específicos são:

- Coletar opiniões na língua portuguesa presentes em sites;
- Realizar análise de sentimentos nas opiniões coletas;
- Realizar comparações entre as duas versões da ferramenta SentimentALL.

Este trabalho foi dividido da seguinte forma: a seção 2 contém a revisão sobre processamento de linguagem natural, análise de sentimentos e trabalhos relacionados, a seção 3 apresenta a metodologia utilizada para o desenvolvimento

desse trabalho, seção 4 dispõe os resultados obtidos e por último as considerações finais (seção 5).

2 REFERENCIAL TEÓRICO

2.1 PROCESSAMENTO DE LINGUAGEM NATURAL

Processamento de Linguagem Natural (PLN), ou em inglês *Natural Language Processing (NLP)*, é uma subárea da linguística computacional. Segundo Domingues (2011, p.26) a linguística computacional é uma “área de pesquisa que se originou da interseção entre a linguística e a uma subárea da ciência da computação, a inteligência artificial”. A PLN trata do uso de técnicas computacionais para que computadores possam “entender” e gerar linguagem natural, a fim de que haja uma melhor interação entre homem e computador, ou para que o computador possa adquirir informações que estão em linguagem natural (DOMINGUES, 2011; RUSSELL; NORVIG, 2013).

Um grande volume de dados textuais é gerado na internet diariamente, sendo necessária a utilização de PLN para poder fazer um melhor uso deles. Boa parte são dados não estruturados e a minoria, estruturados. Dados estruturados são aqueles que possuem uma identificação do que eles representam e que podem ser organizados em linhas e colunas, tornando fácil a sua recuperação. Geralmente os dados estruturados são armazenados em banco de dados relacionais. Já os não estruturados, são dados que não possuem uma estrutura definida (como documentos textuais, e-mails e vídeos), são mais difíceis de serem trabalhados, logo a tarefa para recuperá-los não é feita de forma trivial como, por exemplo, em uma consulta em SQL, necessitando do uso, por exemplo, de PLN.

Linguagens naturais, que estão no formato não estruturado, são linguagens usadas por seres humanos para se comunicarem, como, por exemplo, inglês, português, chinês e francês. No entanto, essas linguagens naturais, considerando o seu tamanho e que estão em constante mudança, podem se tornar ambíguas. A ambiguidade é evidenciada quando um enunciado possui dois ou mais sentidos, o que torna difícil sua compreensão, especialmente para a máquina, pois o computador diante de enunciados ambíguos tem que decidir qual o sentindo correto, e essa decisão nem sempre produz bons resultados. Por exemplo: “Pedro entregou a Maria o seu lápis”. De quem é o lápis? De Pedro ou de Maria? Essa frase já exemplifica que mesmo em uma análise manual a identificação correta dos elementos de uma frase é uma tarefa complexa, às vezes até impossível dada a

sua estrutura. Então, um sistema que trabalhe com PLN tem que estar preparado para tentar resolver esse tipo de problema.

Para o computador tentar resolver esses problemas e outros, são realizados diferentes níveis de análise. Segundo Dale (2010, p. 4), “tradicionalmente, o trabalho em processamento de linguagem natural tende a ver o processo de análise da linguagem como sendo decomposto em uma série de etapas, espelhando as distinções linguísticas teóricas traçadas entre sintaxe, semântica e pragmática”.

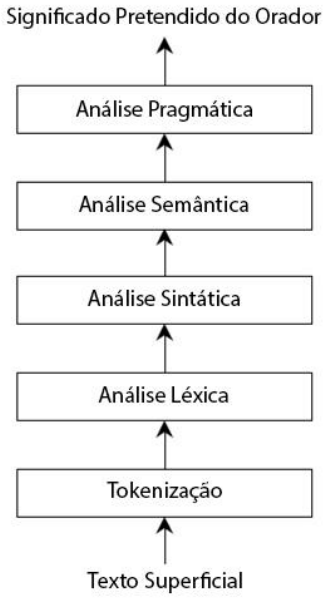
Conforme Dale (2010), a análise, em processamento de linguagem natural, inicia com a sintaxe, fornecendo uma ordem e estrutura às frases. Essa ordem e estrutura servem como entrada para a análise semântica. Isto é seguido por uma etapa de análise pragmática, em que o significado do texto em contexto é determinado.

A subseção a seguir apresenta com mais detalhes essas etapas e as etapas de tokenização e de análise léxica.

2.1.1 Etapas de Análise em PLN

Segundo Dale (2010), as análises tradicionais (sintaxe, semântica e pragmática) servem, no máximo, como um ponto de partida para o processamento de textos reais em linguagem natural. Então, quando se lida com dados reais, os estágios são decompostos como na Figura 1.

Figura 1: Os estágios de análise em PLN



Fonte: Dale (2010, tradução nossa)

Como pode ser visto na Figura 1, o processo inicia recebendo como entrada um texto, passa pelo estágio de tokenização, depois por diferentes granularidades de análise e, por fim, tem como saída o significado pretendido do orador. A seguir, esses estágios serão apresentados com mais detalhes.

2.1.1.1 Tokenização

Segundo Palmer (2010), a tokenização quebra uma sequência de caracteres de um texto em palavras (*tokens*), localizando os limites das palavras, os pontos onde uma palavra termina e outra começa. Há duas abordagens de tokenização: para idiomas delimitadas por espaços e para línguas não-segmentadas. As linguagens delimitadas por espaços são aquelas em que alguns limites entre as palavras são indicados pela inserção de espaços em branco, como por exemplo, inglês, português e espanhol. Nas linguagens não-segmentadas, as palavras são escritas em sucessão, sem indicação de limites entre as palavras, por exemplo, tailandês e chinês. Como esse trabalho trata da utilização da língua portuguesa, então a abordagem utilizada daqui por diante será a delimitada por espaço.

Palmer (2010) explica que podem existir ambiguidades no uso de sinais de pontuação que afetam a tokenização, uma vez que os sinais de pontuação podem

ter muitas funções diferentes. O Exemplo 1 apresenta uma frase que contém ambiguidade.

Exemplo 1: "*João, pai de Maria, comprou um pacote de bolacha de R$1,25 para ela.*".

No Exemplo 1, tem-se então uma ambiguidade no uso da vírgula. As duas primeiras vírgulas são usadas para delimitar o aposto e a terceira vírgula para separar a parte inteira da parte decimal de um número decimal. Na tokenização deve-se conseguir determinar quando uma pontuação faz parte ou não de um token.

2.1.1.2 Análise Léxica

A análise léxica ou morfológica é uma análise realizada a nível de palavra. As palavras que foram separadas na etapa anterior, agora são decompostas em partes (morfemas). Segundo Coppin (2013), a análise léxica "examina os modos pelos quais palavras se desmembram em componentes e como isso afeta o *status* gramatical delas". Como por exemplo, quando uma palavra tem "s" no final, costuma estar no plural. Nessa etapa é indicado em qual classe uma palavra se encaixa, se é um adjetivo, advérbio, verbo, entre outras.

Para exemplificar a análise léxica, será utilizada a palavra "infelizmente". Essa palavra pode ser decomposta em 3 partes: prefixo "in", radical "feliz" e sufixo "mente". Essas são as menores partes (morfemas) que se pode extrair dessa palavra, não podendo ser mais fragmentada. Cada um dos morfemas tem um significado gramatical, podendo ou não modificar a classe gramatical da palavra. O radical "feliz" é um adjetivo que expressa alegria. O prefixo "in" faz a negação do radical, mas não modifica a classe gramatical da palavra. O sufixo "mente" modifica o radical, transformando o adjetivo em advérbio. Com isso, a palavra "infelizmente" é da classe gramatical advérbio.

2.1.1.3 Análise Sintática

A análise sintática verifica se a estrutura gramatical de uma frase está correta. Segundo Ljunglöf e Wirén (2010), para realizar a análise sintática pode-se utilizar a gramática livre de contexto, que é representada pelo seguinte formalismo: $G = \{\Sigma, N, S, R\}$. Onde, Σ é um conjunto de símbolos terminais, palavras (Ex.:

$\{pessoa, cachorro, ganhar\}$). N é um conjunto de símbolos não terminais, podendo ser categorias de palavras, frases ou segmentos de frases (Ex.: $\{frases\ nominais, verbos, artigos\}$). S é um símbolo inicial não terminal. R é um conjunto de regras de produção. Uma regra de produção possui a seguinte forma: $A \rightarrow \alpha$, sendo A um símbolo não terminal e α símbolos não terminais e/ou terminais. A Figura 2 apresenta um exemplo de regras de produção:

Figura 2: Regras de Produção

$$FRASE \rightarrow SINTAGMANOMINAL\ SINTAGMAVERBAL$$
$$SINTAGMANOMINAL \rightarrow ARTIGO\ SUBSTANTIVO$$
$$|SUBSTANTIVO$$
$$SINTAGMAVERBAL \rightarrow VERBO\ SINTAGMANOMINAL$$
$$|VERBO$$
$$ARTIGO \rightarrow a \mid o \mid um \mid uma$$
$$SUBSTANTIVO \rightarrow João \mid Maria \mid cachorro$$
$$VERBO \rightarrow ganhou \mid deu \mid recebeu$$

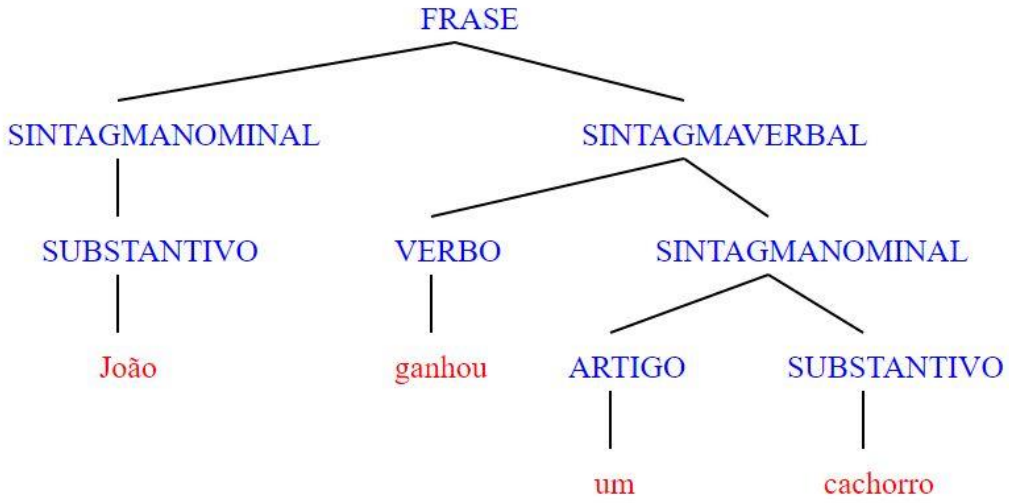
Nas regras de produção acima, os símbolos escritos todos em maiúsculo são símbolos não terminais e os demais símbolos são terminais. Partindo do símbolo inicial “FRASE”, são realizadas as derivações que, segundo Menezes (2011), “é a substituição de uma subpalavra de acordo com uma regra de produção”. Com isso, é possível verificar se uma frase está sintaticamente correta, caso ao final das derivações restarem apenas símbolos terminais. Utilizando como exemplo a seguinte frase: “João ganhou um cachorro”. Essa frase tem as derivações apresentadas na Figura 3 (FRASE é o símbolo inicial):

Figura 3: Exemplo de Derivação

$$
\begin{aligned}
FRASE &\rightarrow SINTAGMANOMINAL\ SINTAGMAVERBAL \\
&\rightarrow SUBSTANTIVO\ SINTAGMAVERBAL \\
&\rightarrow João\ SINTAGMAVERBAL \\
&\rightarrow João\ VERBO\ SINTAGMANOMINAL \\
&\rightarrow João\ ganhou\ SINTAGMANOMINAL \\
&\rightarrow João\ ganhou\ ARTIGO\ SUBSTANTIVO \\
&\rightarrow João\ ganhou\ um\ SUBSTANTIVO \\
&\rightarrow João\ ganhou\ um\ cachorro
\end{aligned}
$$

Na Figura 3, foi analisado que a frase está sintaticamente correta para essa gramática, pois ao final, através das regras de derivação, foi possível o reconhecimento da frase. Uma forma de representar uma estrutura sintática pode ser utilizando uma árvore de sintaxe, como na Figura 4, usando o exemplo anterior:

Figura 4: Árvore de Sintaxe



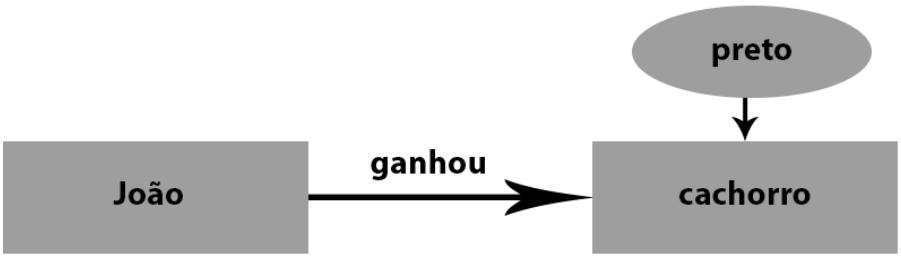
A árvore de sintaxe possui uma forma de representação que pode facilitar a visualização das derivações que ocorrem para reconhecer uma frase como sendo de uma linguagem.

Reproduzir uma gramática de uma linguagem natural utilizando as regras de produções é uma tarefa complexa, devido à grande quantidade de estruturas sintáticas possíveis.

2.1.1.4 Análise Semântica

A análise semântica trata do significado da sentença, sobre aquilo que é possível entender através de um determinado enunciado. “Análise semântica envolve a elaboração de uma representação dos objetos e ações que uma sentença esteja descrevendo, incluindo detalhes fornecidos por adjetivos, advérbios e preposições” (COPPIN, 2013, pág. 511). Para a representação dos objetos, por exemplo, pode-se utilizar uma rede semântica. Modificando a frase utilizada como exemplo na seção anterior para “João ganhou um cachorro preto”, constrói-se a rede semântica como na Figura 5:

Figura 5: Rede Semântica



Na Figura 5, os objetos “João” e “cachorro” estão sendo representados por retângulos, o atributo “preto” de “cachorro” é representado por uma elipse e o relacionamento entre os objetos é representado por uma seta rotulada com o verbo “ganhou”.

2.1.1.5 Análise Pragmática

A análise pragmática trata do significado da sentença em um contexto. Quando se considera o contexto, as ambiguidades que possam ocorrer em uma frase são eliminadas. A seguir, o Exemplo 2 apresenta duas frases para demonstrar a análise pragmática.

Exemplo 2:

(1) “Pitoco é um cachorro. ”

(2) “Ele tem três patas. ”

Conforme o Exemplo 2, a Frase 2 pode gerar ambiguidade, devido ao uso da palavra “patas”. Se cada frase for considerada individualmente, não é possível

saber qual o significado da palavra “patas”. Então, considerando que as duas frases estão no mesmo contexto, como Pitoco é um cachorro, então “patas” são os membros de Pitoco.

Dale (2010) diz que as análises semânticas e pragmáticas são menos estudadas que a análise sintática, pois, quanto mais profunda é a análise, maior é a abstração de representação, que é mais difícil de definir. Todas as etapas de análise são importantes quando deseja-se trabalhar com PLN. Apesar de que às vezes não é necessário realizar todas as análises para resolver um problema.

2.1.2 *PoS Tagging*

PoS tagging é uma tarefa que ocorre na etapa de análise léxica em PLN. Segundo Jurafsky e Martin (2000), as palavras são agrupadas em classes chamadas de *Part-of-Speech* (*PoS*), por exemplo: verbos, substantivos, adjetivos, pronomes, e assim por diante. Um conjunto de *PoS tag* se chama *tagset. PoS Tagging* é um importante passo que ocorre no pré-processamento em algumas aplicações de PLN. *PoS Tagging* envolve, dada uma sentença, a seleção da sequência mais provável de *PoS tag*s de um *tagset,* para cada palavra da sentença (ALLEN, 1995; GÜNGÖR, 2010).

Güngör (2010) diz que há duas dificuldades básicas em *PoS tagging*: palavras ambíguas e palavras desconhecidas. A ambiguidade ocorre quando uma palavra possui mais de um *PoS tag* possível, como por exemplo, “para”, que pode ser um verbo (parar) ou uma preposição. Para tentar resolver esse problema deve-se olhar o contexto, a estrutura sintática na qual a palavra está inserida. No caso do problema relacionado a palavras desconhecidas, Güngör (2010) cita duas abordagens em que esse problema pode ocorrer: quando se usam regras escritas manualmente, com isso, algumas palavras podem não ter sido consideradas, ou quando se usa sistemas estatísticos e a palavra não aparece no conjunto de treinamento.

A seguir um exemplo para ilustrar como as palavras seriam marcadas com as *POS tags*.

Exemplo 3: “*O menino não para de jogar vídeo game.* ”

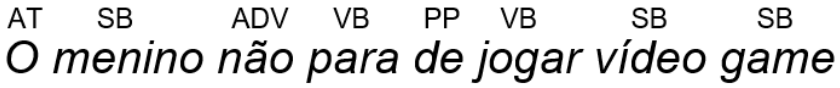
A Tabela 1 representa um *tagset* simplificado disponível para identificar em qual classe uma palavra na frase se encaixa.

Tabela 1: Exemplo de Tagset

Tag	Descrição
AT	Artigo
ADJ	Adjetivo
ADV	Advérbio
PP	Preposição
PR	Pronome
SB	Substantivo

Usando o *tagset* (Tabela 1), pode-se atribuir as *tags* em cada palavra, analisando a classe gramatical de cada palavra e a estrutura sintática da frase. O resultado fica como na Figura 6:

Figura 6: Exemplo de *PoS tagging*



O ideal é que se tenha mais granularidade nas *tags*, criando subclasses de palavras, por exemplo, os verbos poderiam ser separados em presente, passado e futuro.

A próxima seção apresenta a tarefa de identificação de expressões multipalavras, que pode ser realizada com a ajuda das palavras etiquetadas com *PoS tags*.

2.1.3 Expressões Multipalavras (MWE)

Expressões Multipalavras, ou em inglês *Multiwords Expression* (MWE), são, segundo Sag et al (2002), palavras combinadas em que o significado da expressão não pode ser obtido a partir de suas partes. Ou seja, expressões multipalavras são conjuntos de palavras que se comportam como uma unidade, tendo um valor sintático e semântico diferente daquele caso sejam consideradas as palavras individualmente. A seguir um exemplo de expressão multipalavra.

Exemplo 4: “*João comeu um cachorro quente.*”.

No Exemplo 4, “cachorro quente” é uma expressão multipalavra, pois ela se comporta como se fosse uma palavra. Se consideradas as palavras separadamente, então, seria um “cachorro” que está “quente”. Nenhuma das duas alternativas está errada, considerando o aspecto sintático e semântico, embora a probabilidade de ser uma expressão multipalavra é maior. Para ter conhecimento se um conjunto de palavras tem maior probabilidade de serem expressões ou não, pode-se utilizar a medida de associação entre palavras *Pointwise Mutual Information* (PMI). O cálculo de PMI, proposto por Church e Hanks (1990), é o seguinte.

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

Onde x e y são palavras. $P(x,y)$ é a probabilidade de ocorrência de x e y ao mesmo tempo. $P(x)$ e $P(y)$ é a probabilidade de ocorrência independentemente de x e y, respectivamente. Com isso, tem-se o grau de dependência entre duas palavras. Quanto maior o resultado, maior a dependência.

Exemplificando o uso dessa equação, tem-se na Tabela 2 dados fictícios de um determinado contexto, as palavras e sua probabilidade de ocorrência, ou seja, qual a frequência em que ela pode aparecer em um documento.

Tabela 2: Exemplo de probabilidade de ocorrência de palavras

Palavra	**Probabilidade de Ocorrência**
Cachorro	23%
Raça	20%
Quente	11%
Ração	12%
Casinha	15%
Coleira	19%

Na Tabela 3, tem-se a probabilidade de ocorrência de possíveis expressões, ou seja, foram calculadas as probabilidades da ocorrência de duas palavras juntas iniciada com a palavra “Cachorro”.

Tabela 3: Exemplo de probabilidade de ocorrência de expressões

Possíveis Expressões	Probabilidade de Ocorrência
Cachorro Raça	3%
Cachorro Quente	41%
Cachorro Ração	2%
Cachorro Casinha	7%
Cachorro Coleira	5%

Utilizando os dados da Tabela 2 e da Tabela 3, a expressão “cachorro quente”, tem a probabilidade de ocorrência de 41%, a palavra “cachorro” de 23% e “quente” de 11%. Fazendo as substituições dos valores na equação:

$$I(cachorro, quente) = \log_2 \frac{P(cachorro, quente)}{P(cachorro)P(quente)}$$

$$I(cachorro, quente) = \log_2 \frac{41\%}{23\% * 11\%}$$

$$I(cachorro, quente) \cong 4{,}02$$

O problema dessa equação está no fato de poder calcular a dependência de apenas duas variáveis. Essa equação pode ser estendida para realizar o cálculo com mais variáveis através da equação proposta por Cruys (2011).

$$SI(x_1, x_2, \dots, x_n) = \log_2 \frac{P(x_1, x_2, \dots, x_n)}{\prod_{i=1}^{n} P(x_i)}$$

A ideia dessa equação continua a mesma da anterior. A mudança está no cálculo da probabilidade de ocorrência de palavras em conjunto, em que se considera agora mais de duas palavras ($P(x_1, x_2, \dots, x_n)$) e é realizado o produto da probabilidade individual de todas as palavras ($\prod_{i=1}^{n} P(x_i)$).

O cálculo de PMI é realizado sobre um *corpus*. Nesse *corpus* é realizado o PoS tagging e depois, são identificados padrões morfológicos. Boos, Prestes e Villavicencio (2014), apresenta seis padrões morfológicos que podem ser utilizados

para identificar possíveis expressões multipalavras do tipo composto nominal (que será utilizada nesse trabalho). A Tabela 4 apresenta esses padrões, onde as letras N (Substantivo), A (Adjetivo) e P (Preposição) são PoS tags.

Tabela 4: Padrões Morfológicos

Padrão	Exemplo
N N	Nações Unidas
N A	Governo federal
N N N	Supremo Tribunal Federal
N N A	Fundo Monetário Internacional
N A A	Produto interno bruto
N P N	Casa de praia, bolsa de valores

Fonte: Boos, Prestes e Villavicencio (2014)

Identificada as possíveis expressões multipalavras utilizando esses padrões morfológicos, em seguida é aplicada o PMI. Por fim, um conjunto de palavras é considerado uma expressão multipalavra quando o valor do PMI for maior ou igual a um valor definido pelo usuário.

As etapas de análise e as técnicas de *PoS tagging*, Expressões Multipalavras ou outras, podem ser aplicadas em algumas aplicações que realizam PLN, tais como, recuperação de informação, extração de informação e análise de sentimentos. “A Análise de Sentimentos ou Mineração de Opinião é o estudo computacional de opiniões, sentimentos e emoções expressos em texto” (LIU, 2010, p. 629, tradução nossa). O objetivo básico da análise de sentimentos é definir uma polaridade para determinados aspectos presentes em um texto. Como esse trabalho trata do uso da análise de sentimentos, então, será apresentada com mais detalhes na próxima seção.

2.2 ANÁLISE DE SENTIMENTOS

“Análise de sentimentos é a prática de aplicar processamento de linguagem natural e técnicas de análise de texto para identificar e extrair informações subjetivas” (HUSSEIM, 2016, p. 1, tradução nossa). Liu (2015, p. 20, tradução nossa) define a análise de sentimentos da seguinte forma:

> Análise de Sentimentos, também chamada de mineração de opiniões, é o campo de estudo que analisa opiniões, sentimentos, avaliações, atitudes e emoções de pessoas na direção de entidades e seus atributos expressos em texto escrito. As entidades podem ser produtos, serviços, organizações, indivíduos, eventos, questões ou tópicos.

Em síntese, a análise de sentimentos tem a finalidade de identificar se a opinião ou o sentimento de um certo autor é positiva, negativa ou neutra em relação à uma determinada entidade. A tarefa principal da análise de sentimentos é extrair os componentes de uma opinião, que podem ser sistematizados de acordo com Liu (2010, 2015):

- **Uma entidade** (e) possui um conjunto de componentes e um conjunto de atributos. Cada componente pode ter seu próprio conjunto de subcomponentes e atributos. Utilizando como exemplo a frase a seguir: “A pousada é maravilhosa”. Nessa frase a entidade é “pousada”.
- **Os aspectos** (a) podem estar explícito ou implícito. Se um aspecto ou algum dos seus sinônimos aparecerem em uma sentença, então é um aspecto explícito. Se não aparece, mas o aspecto está implícito em uma frase, então esse aspecto é chamado de implícito. Como por exemplo o seguinte comentário: “o sabor da picanha estava delicioso”. Nesse comentário o aspecto explícito é sabor.
- **A polaridade da opinião ou sentimento** (s) é a orientação da opinião sobre um aspecto indicando se a opinião é positiva, negativa ou neutra.
- **O titular ou autor da opinião** (h) é a pessoa ou organização que expressa a opinião.
- **O tempo** (t) é um momento específico no tempo que a opinião foi expressa.

Uma opinião pode ser de qualquer um dos dois tipos a seguir (LIU, 2010):

Opinião direta: Uma opinião direta é uma quíntupla (e, a, s, h, t), também chamada de opinião regular. Este é o tipo de opinião mais comum e expressa um sentimento sobre uma entidade em particular ou um aspecto de uma entidade. A

seguir um exemplo com dados fictícios: “O atendimento do Hotel Girassol é péssimo”, comentário postado no dia 7/11/2016 por Maria Andrade. Nesse exemplo a entidade é Hotel Girassol, o aspecto é atendimento, a polaridade é negativa (“péssimo”), a titular é Maria Andrade e o tempo é 7/11/2016. Portanto, a quíntupla fica da seguinte maneira: (Hotel Girassol, atendimento, negativa, Maria Andrade, 7/11/2016).

Opinião Comparativa: Compara diversas entidades baseado em alguns de seus aspectos compartilhados. Ex.: “Hotel Girassol é melhor do que o Hotel Pequi”.

A análise de sentimentos pode ser realizada em três tipos de granularidade: nível de documento, de sentença e de aspecto (LIU, 2012). A análise que considera o nível de documento classifica, em um contexto geral, se a opinião expressa em um texto é positiva ou negativa. Mas a opinião contida no texto deve ser apenas sobre uma entidade. A análise realizada sobre o nível de sentença classifica as sentenças de um texto em positiva ou negativa. A análise referente ao nível de aspecto classifica as entidades e/ou seus aspectos em positivo ou negativo.

A seguir um exemplo que apresenta o resultado dos diferentes tipos de análise de um texto (um número está associado com cada frase para facilitar a referência).

Exemplo 5: “(1) Fui no feriado da semana santa. (2) A pousada é muito boa, atendimento e serviço são ótimos. (3) O restaurante é um pouco caro. (4) O lugar é lindo, boa hospedagem e excelente paisagem. ”

Assim, tem-se:

Nível de documento: Apesar de haver uma frase negativa (Frase 3), no contexto geral, o texto expressa uma opinião afirmativa sobre a pousada.

Nível de sentença: A frase 1 não é opinativa, por ser a apresentação de um fato, não de uma opinião, assim é considerada neutra. As frases 2 e 4 são positivas. A frase 3 é negativa.

Nível de aspecto: A frase 2 tem os aspectos pousada, atendimento e serviço, todos positivos. Na frase 3, o aspecto restaurante é negativo. Na frase 4, lugar, hospedagem e paisagem são positivos.

Quando uma análise que considera o nível de documento polariza uma entidade como positiva, não significa que o autor é positivo sobre todos os aspectos dessa entidade. O mesmo acontece quando a análise polariza uma entidade como

negativa. Para ter o nível de detalhe desejado, utiliza-se a análise de sentimentos baseada em aspectos. Segundo Liu (2012), esse tipo de análise tem duas tarefas principais: extrair as entidades e os seus aspectos e identifica a polaridade de cada aspecto ou entidade. As próximas seções abordarão com mais detalhes essas duas tarefas.

2.2.1 Extração de Entidades e Aspectos

Essa tarefa concentra-se na extração de entidades e aspectos no qual sentimentos ou opiniões foram expressos. Os métodos utilizados para reconhecer entidades e aspectos são diferentes, pois possuem características diferentes. Entidades são referenciadas como nomes de produtos, serviços, indivíduos, eventos e organizações. Aspectos são referenciados como atributos e componentes de entidades.

Os aspectos podem estar explícitos ou implícitos na frase. Liu (2015) cita a possibilidade de se explorar as relações sintáticas para realizar a extração de aspectos explícitos. Essa forma de extração considera que a maioria das palavras opinativas é conhecida, desse modo, ao encontrar o substantivo ou a frase nominal mais próxima a um desses termos, é possível identificar um aspecto ou uma entidade. Após a identificação do aspecto ou entidade, pode-se extrair, caso haja, outros aspectos ou entidades que estão ligadas com o primeiro aspecto extraído, através da conjunção “e”. A seguir um exemplo para ilustrar esse processo.

Exemplo 6: “*A localização e o atendimento são horríveis*”

No Exemplo 6 é identificada a palavra “horríveis” como palavra opinativa. Buscando o substantivo mais próximo, tem-se a palavra “atendimento” que passa a ser um dos aspectos extraídos. Depois verifica se há um substantivo ligado à palavra “atendimento” através de uma conjunção. No caso, há a palavra “localização” que passa a ser também um dos aspectos extraídos.

Qiu *et al.* (2009) apresentam uma abordagem baseada em regras mais generalizada para realizar a extração de aspectos e entidades, chamada de *double propagation* (DP). O algoritmo DP extrai as palavras opinativas e aspectos a partir de palavras opinativas e aspectos já conhecidos, iniciando apenas com um conjunto de palavras opinativas conhecidas. Realiza-se a propagação, utilizando as palavras opinativas conhecidas, para encontrar novas palavras opinativas e/ou

aspectos. Essas palavras opinativas e aspectos extraídos passam a compor o conjunto de palavras opinativas e aspectos conhecidos. Realiza-se a propagação novamente, porém, além das palavras opinativas, agora pode haver aspectos conhecidos, que também são utilizados para encontrar novas palavras opinativas e aspectos. A propagação se repete até não ter mais palavras opinativas e aspectos desconhecidos. A propagação realiza quatro subtarefas:

1. Extração de aspectos usando palavras opinativas conhecidas
2. Extração de aspectos utilizando aspectos conhecidos
3. Extração de palavras opinativas usando aspectos conhecidos
4. Extração de palavras opinativas usando palavras opinativas conhecidas

A Figura 7 a seguir, apresenta o pseudocódigo do algoritmo Double Propagation.

Figura 7: Pseudocódigo do algoritmo Double Propagation

Entrada: Palavras Opinativas {O}, Dados do Review R
Saída: Todos os aspectos {F}, Palavras Opinativas Expandidas {O-Expandido}
Função:
1. {O-Expandido} = {O}
2. $\{F_i\} = \emptyset, \{O_i\} = \emptyset$
3. para cada sentença em R
4. se aspectos extraídos não estão em {F}
5. Extraia aspectos $\{F_i\}$ usando palavras opinativas em {O-Expandido}
6. fimse
7. se palavras opinativas extraídas não estão em {O-Expandido}
8. Extraia novas palavras opinativas $\{O_i\}$ usando palavras opinativas em {O-Expandido}
9. fimse
10. fimpara
11. $\{F\} = \{F\} + \{F_i\}, \{O\text{-}Expandido\} = \{O\text{-}Expandido\} + \{O_i\}$
12. para cada sentença em R
13. se aspectos extraídos não estão em {F}
14. Extraia aspectos {F′} usando aspectos em $\{F_i\}$
15. fimse
16. se palavras opinativas extraídas não estão em {O-Expandido}
17. Extraia novas palavras opinativas {O′} usando aspectos em $\{F_i\}$
18. fimse
19. fimpara
20. $\{F_i\} = \{F_i\} + \{F'\}, \{O_i\} = \{O_i\} + \{O'\}$
21. $\{F\} = \{F\} + \{F'\}, \{O\text{-}Expandido\} = \{O\text{-}Expandido\} + \{O'\}$
22. Repita 2 até tamanho($\{F_i\}$) = 0, tamanho($\{O_i\}$) = 0

Fonte: Qiu *et al.* (2011, tradução nossa)

Para que o algoritmo reconheça qual palavra deve extrair utilizando as palavras conhecidas são utilizadas relações de dependência. Há duas formas de

dependência entre palavras, dependência direta e indireta. Qiu *et al* (2011, pág. 13) as define da seguinte maneira:

Dependência direta: “indica que uma palavra depende de outra palavra sem quaisquer palavras em seu caminho de dependência (i.e., diretamente) ou ambas dependem de uma terceira palavra diretamente”. Exemplo: “localização e atendimento” e “ótimo atendimento”.

Dependência indireta: “indica que uma palavra depende de outra palavra através de algumas palavras adicionais (i.e., indiretamente) ou ambos dependem de uma terceira palavra através de palavras adicionais”. Exemplo: “O Hotel Unique é o melhor hotel de São Paulo”. Tendo “melhor” como uma palavra opinativa conhecida, pode-se extrair “Hotel Unique” como aspecto, através da palavra “hotel”. Pois, “melhor” tem uma relação de dependência com hotel (“melhor” modifica “hotel”) e “Hotel Unique” tem uma relação de dependência com “hotel” (“Hotel Unique” é o sujeito de “hotel”).

Para extrair aspectos que estão implícitos no texto é necessário realizar um mapeamento das expressões (normalmente, advérbios e adjetivos) que indicam a proximidade de um determinado aspecto (LIU, 2015; SU et al, 2008). Com isso, é possível reconhecer quando uma dessas expressões estiverem no texto e a quais aspectos implícitos elas estão relacionadas. O Exemplo 7 apresenta uma frase que contém um aspecto implícito.

Exemplo 7: “No restaurante Braseiro, a comida é deliciosa”.

No Exemplo 7, o aspecto “sabor” da entidade “comida” não está citado explicitamente na frase, mas é possível identificá-lo através do mapeamento, pois a palavra opinativa “deliciosa” está relacionada ao “sabor da comida”.

Realizada a tarefa de extração de entidades e aspectos, tem-se como resultado um conjunto de aspectos relacionados com suas palavras opinativas. Esse resultado será utilizado como entrada para a tarefa de identificação da polaridade, que será apresentado na seção a seguir.

2.2.2 Identificação da Polaridade

Essa tarefa determina se as opiniões sobre os diferentes aspectos são positivas, negativas ou neutras. Existem duas formas de realizar essa tarefa,

através da abordagem de aprendizado supervisionado e da abordagem não supervisionada baseado em léxico.

2.2.2.1 Abordagem de Aprendizado Supervisionado

A abordagem de aprendizado supervisionado utiliza algoritmos de aprendizado de máquina para a classificação dos aspectos em positivo ou negativo. “O campo da aprendizagem de máquina está preocupado com a questão de como construir programas de computadores que melhoram automaticamente com a experiência” (MITCHELL, 1997, p. 3, tradução nossa). Programas de computadores que aprendem com experiências, segundo Facelli et al. (2011), fazem um processo de inferência chamado indução, utilizando exemplos para obter conclusões genéricas.

Na aprendizagem supervisionada, o computador “observa alguns exemplos de pares de entrada e saída, e aprende uma função que faz o mapeamento da entrada para a saída” (RUSSELL; NORVIG, 2013, p. 607). Por exemplo, para um computador aprender a classificar frases em positivas ou negativas, pode-se utilizar um algoritmo de classificação. Um algoritmo de classificação determina, dado um conjunto de classes categóricas pré-definidas, a qual dessas classes um item específico pertence (GOEBEL; GRUENWALD, 1999). Logo, seria informado para o algoritmo de classificação uma série de frases como entrada e qual a saída desejada para cada frase, se a frase é positiva ou negativa. Com isso, dada uma frase qualquer, o classificador conseguiria classificar, com uma certa porcentagem de erro, a frase como positiva ou negativa, com base no que ele aprendeu.

O processo de aprendizagem é composto de dois passos: treinamento e teste (BISHOP, 2007). Antes de iniciar, deve-se ter um conjunto de dados, em que cada elemento deve ter a marcação de qual classe pertence. Uma determinada porcentagem desses dados será utilizada no treinamento e a restante será utilizada no teste. O treinamento é a tarefa de informar para o classificador qual dado é de determinada classe. A tarefa de teste é realizada para medir o desempenho do classificador ao classificar os dados de teste.

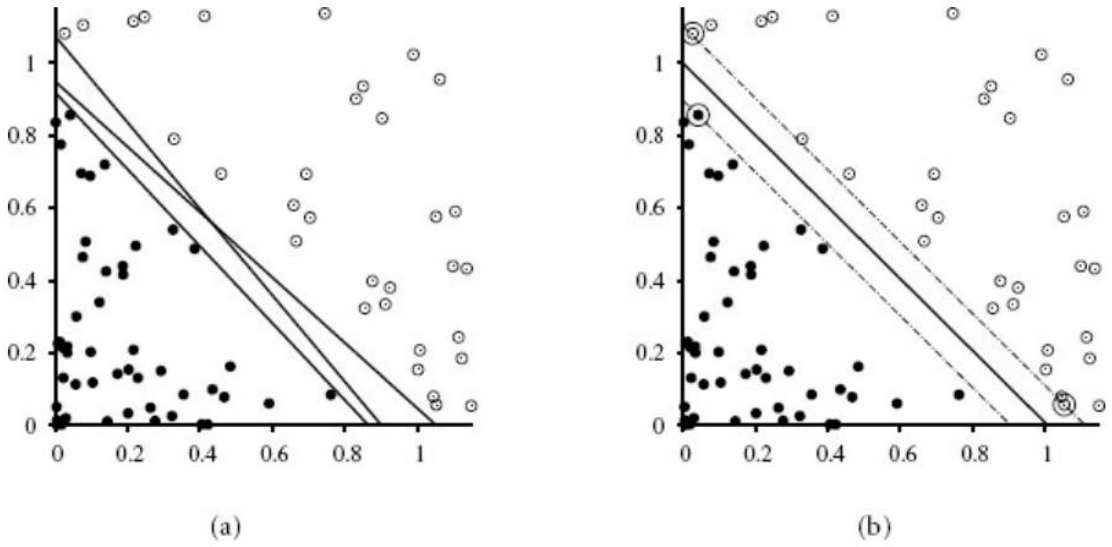
O conjunto de dados utilizado para teste e treinamento é formado por *features*. No caso, na análise que considera o nível de documento e o nível de sentença, as *features* são as palavras opinativas, pois são elas que indicam se o

autor falou positivamente ou negativamente sobre a entidade. No entanto, na análise sobre o nível de aspecto, além de considerar as palavras opinativas como *features*, deve-se considerar quais são as entidades ou os aspectos alvos dessas palavras. Por exemplo, dispõe-se de 5000 comentários, em que para cada comentário há uma marcação se é positivo ou negativo. Usa-se um certo quantitativo desses comentários para o treinamento, como por exemplo 75%. Então, dos 5000 comentários, são separados 3750, e em cada um deles é realizada a extração das palavras opinativas que serão informadas para o classificador. Após essa etapa, os 1250 comentários restantes serão usados para testar o classificador e, assim, avaliar o nível de acerto.

Existem diversos algoritmos de aprendizado supervisionado de máquina, como SVM (*Support Vector Machine*), Classificador Naive Bayes, kNN (*k-Nearest Neighbors*), dentre outros. Porém, SVM tem superado outros algoritmos e obtido melhores resultados na classificação (HAHN; OSTENFORF, 2008; FACELI et al., 2011; KALAIVANI; SHUNMUGANATHAN, 2013)

Máquina de Vetor de Suporte, ou em inglês *Support Vector Machine* (SVM) podem ser lineares e não lineares. SVMs lineares resolvem problemas que são linearmente separáveis. Como, por exemplo, considerando um conjunto de treinamento com dados de duas classes, classe A e classe B, é possível traçar uma linha (separador), em que de um lado ficam os dados da classe A e do outro os dados da classe B. A função do SVM é traçar essa linha que separa os dados, para quando for informado um elemento, o SVM classifique-o de acordo com o lado adequado. Existem diversos separadores que podem resolver o mesmo problema, como é possível ver na Figura 8-a.

Figura 8: SVM Linear

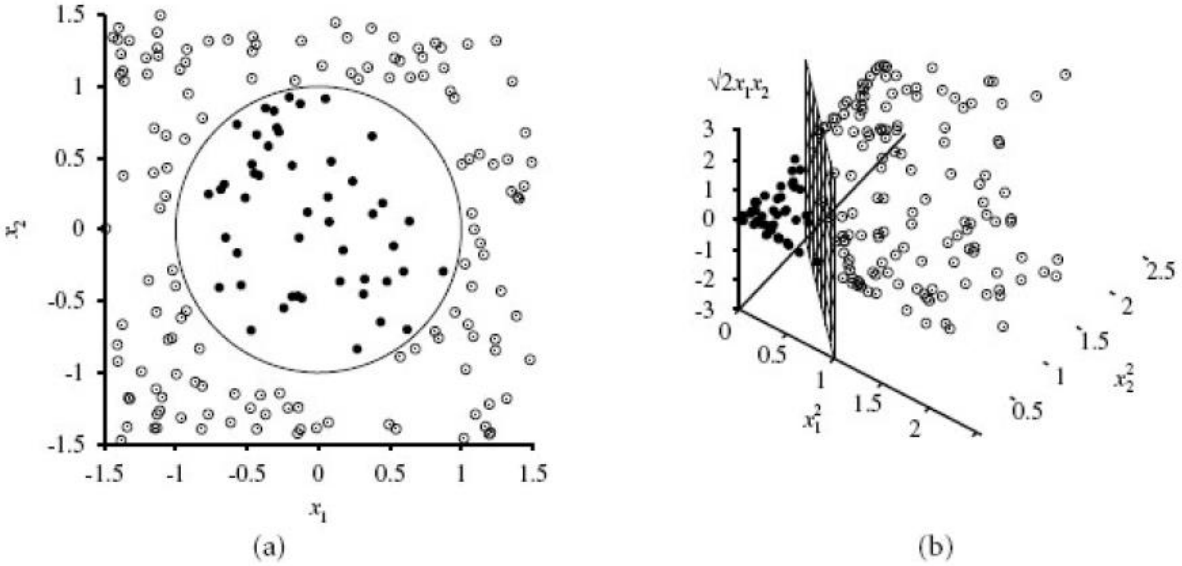


Fonte: RUSSEL e NORVIG (2013)

Na Figura 8-a, há duas classes (pontos preto e pontos brancos) e 3 separadores. Para encontrar o separador que generaliza melhor, utilizam-se os vetores de suporte. Vetores de suporte são aqueles que se encontram mais perto da margem entre as classes. Eles são utilizados para encontrar o separador. Portanto, levando em consideração a posição dos vetores de suporte, calcula-se a posição do separador, que deverá ter a mesma distância entre os vetores de suporte das duas classes. Na Figura 8-b, os vetores de suporte são pontos circulados e que possuem uma linha tracejada (margem) passando por eles. No ponto médio entre as margens há o separador.

Há também a possibilidade de usar os SVMs para dados não linearmente separáveis, como na Figura 9, mapeando-se o conjunto de dados de seu espaço original para um novo espaço de tamanho maior, denominado espaço de características (HEARST *et al.*, 1998, *apud* LORENA; CARVALHO, 2007). Segundo Russel e NORVIG (2013, p. 650), “se os dados forem mapeados em um espaço de dimensão suficientemente alta, eles serão quase sempre linearmente separáveis”. Quando os dados estiverem mapeados no espaço de características, então é possível utilizar o SVM linear para separar os dados linearmente. A Figura 9 apresenta um exemplo.

Figura 9: SVM não Linear



Fonte: RUSSEL e NORVIG (2013)

Na Figura 9-a, há duas classes (pontos pretos e pontos brancos) separados por um círculo. Da forma com os dados estão, não é possível traçar uma linha que separe as duas classes. Por isso, os dados deixam a segunda dimensão e são mapeados para a terceira dimensão (Figura 9-b), sendo possível a separação linear.

Liu (2015) cita algumas vantagens e desvantagens na utilização da abordagem de aprendizado supervisionado. A vantagem dessa abordagem é que o algoritmo pode aprender automaticamente todos os tipos de *features*. Como desvantagem, o fato de que os dados de treinamento devem ser rotulados manualmente para cada domínio. O classificador é dependente do domínio, não funcionando em outro domínio para o qual ele não foi treinado. O classificador tem dificuldade de aprender algo que não ocorre com frequência.

2.2.2.2 Abordagem baseada em léxico

A abordagem baseada em léxico presume que há um léxico de sentimentos contendo um conjunto de palavras opinativas com suas respectivas polaridades. Existem alguns léxicos para português prontos, como por exemplo, SentiLex-PT (SILVA; CARVALHO; SARMENTO, 2012) e LIWC.

Existem inúmeras formas de calcular a polaridade utilizando essa abordagem. Liu (2015) cita uma delas, que consiste de quatro passos. Para exemplificar esses passos será utilizado a seguinte frase:

Exemplo 8: "*A comida não estava boa, mas o atendimento foi rápido*"

O primeiro passo é marcar as expressões de sentimentos com +1 ou -1, para as palavras opinativas positivas e negativas, respectivamente. O léxico de sentimentos informa se uma palavra opinativa é positiva ou negativa. A frase do Exemplo 8 tem a seguinte marcação: "*A comida não estava* ***boa*** [+1]*, mas o atendimento foi rápido*". Com "boa" é um sentimento positivo, então ela foi marcada como tal. Porém, apesar de que nesse contexto a palavra "rápido" inferir um sentimento, não é possível saber se é positivo ou negativo, por isso ela não será marcada ainda.

O segundo passo é encontrar palavras na frase que podem mudar a polaridade das palavras opinativas. Por exemplo, as palavras: não, nunca, nenhum, entre outras. Quando alguma dessas palavras for encontrada, a polaridade da palavra opinativa que ela está relacionada é invertida. No Exemplo 8, foi encontrada a palavra "não" negando a palavra "boa", então a frase fica da seguinte forma: "*A comida não estava* ***boa*** [-1]*, mas o atendimento foi rápido*". Existem algumas expressões que indicam contrariedade entre frases/orações, por exemplo: mas, exceto que, entretanto, entre outras.

No terceiro passo, procura-se por essas palavras. Quando elas são encontradas, se a polaridade de uma palavra opinativa estiver positiva em uma frase/oração, na outra frase/oração será marcada como negativa, e vice-versa. No exemplo 8, existe a palavra "mas" que indica contrariedade entre as orações "*A comida não estava boa*" e "*o atendimento foi rápido*". Como "boa" está negativa, então a palavra "rápido" é marcada como positiva. Com isso, a palavra "rápido" por não ser expressão opinativa, não estará no léxico de sentimentos, não sendo polarizado no primeiro passo, porém, mesmo assim ainda é possível inferir que "rápido" é positivo para "atendimento". A frase fica da seguinte forma: "*A comida não estava* ***boa*** [-1]*, mas o atendimento foi rápido* [+1]".

No quarto passo, realiza-se o relacionamento entre a palavra opinativa e o aspecto, porém esse passo já foi realizado anteriormente na fase de extração de

entidades e aspectos. Então, a saída final (aspecto – palavra opinativa - polaridade) do Exemplo 8 é a seguinte: comida – boa – negativo e atendimento – rápido – positivo.

A abordagem baseada em léxico resolve alguns problemas que ocorrem na abordagem de aprendizado supervisionado, tais como (LIU, 2015): é independente de domínio; não é necessário marcar manualmente os dados; e é flexível (pode ser facilmente estendido e melhorado). Liu (2015, p. 206, tradução nossa) diz que essa abordagem tem como desvantagem a necessidade de um “investimento pesado em tempo e esforço para construir a base de conhecimento inicial de léxicos” e que apesar de ser independente de domínio, ainda é necessário tratar algumas particularidades de determinados domínios.

2.2.3 Geração de Léxicos de Sentimentos

Existem três abordagens para gerar os léxicos de sentimentos: a abordagem manual, a abordagem baseada em dicionário e abordagem baseada em *corpus*. A abordagem manual exige muito esforço e tempo, por isso, não é muito utilizada isoladamente. Essa abordagem é utilizada em conjunto com outras abordagens, que são automatizadas, para correção de erros que venham a ocorrer.

A abordagem baseada em dicionário é um processo automatizado para extrair palavras de dicionários online (LIU, 2010). Inicialmente, é construída uma lista de palavras opinativas polarizadas, chamadas de sementes. Essas palavras são pesquisadas no dicionário e são extraídos seus sinônimos e antônimos, que serão adicionadas às sementes. As novas palavras coletadas terão a mesma polaridade da semente, caso seja sinônimo, e polaridade diferente, se for antônimo. O procedimento se repete até que não haja palavras a ser visitadas. Ao final, verificam-se os erros manualmente para efetivar a correção.

A abordagem baseada em *corpus* é um método automatizado que depende de padrões sintáticos e de uma lista de sementes para encontrar palavras opinativas em um *corpus* (LIU, 2012). Hatzivassiloglou e McKeown (1997) descrevem um método para realizar essa tarefa. Esse método demonstra que a partir de conjunções é possível polarizar um adjetivo. Um exemplo é a conjunção “e”. Quando dois adjetivos estão ligados através dessa conjunção, os autores dizem que existe uma grande probabilidade deles terem a mesma polaridade. Os autores

utilizaram o *corpus* da *Wall Street Journal* com 21 milhões de palavras, que foram etiquetadas com *PoS tags*. Os adjetivos e conjunções são extraídos e é aplicado o processo de aprendizado de máquina para determinar se os adjetivos ligados pela conjunção possuem ou não a mesma polaridade. Com o resultado da etapa anterior, um algoritmo de agrupamento separa os adjetivos de polaridades diferentes. No final, a frequência média em cada grupo é calculada e o grupo que tiver a maior frequência é marcado como positivo e o menor como negativo. A seguir, três frases serão usadas para exemplificar esse processo.

Exemplo 9:
(1) *“O hotel é confortável e aconchegante”*
(2) *“O café da manhã é bom, mas é caro”*
(3) *“O clima estava bom e agradável”*

No exemplo 9, com a extração dos adjetivos e da conjunção de cada sentença, tem-se o seguinte resultado: (1) palavras opinativas “confortável” e “aconchegante” com conjunção “e”; (2) palavras opinativas “bom” e “caro” com conjunção “mas”; e (3) palavras opinativas “bom” e “agradável” com conjunção “e”. É aplicado modelo preditivo que utiliza as propriedades linguísticas das conjunções para predizer se os adjetivos relacionados através da conjunção têm a mesma polaridade ou não. Com isso, obtem-se o seguinte resultado: (1) confortável possui a mesma polaridade de aconchegante; (2) bom tem polaridade diferente de caro; e (3) bom e agradável possui a mesma polaridade. Um algoritmo de agrupamento agrupa os adjetivos de mesma polaridade. Um grupo possui os adjetivos: confortável, aconchegante, bom e agradável. O outro tem o adjetivo: caro. Calcula-se a frequência média das palavras de cada grupo, o primeiro grupo tem frequência média de 1,25 e o segundo de 1. Então o primeiro grupo é rotulado como positivo e o segundo como negativo.

2.3 TRABALHOS RELACIONADOS

A seguir serão apresentados dois exemplos que realizam análise de sentimentos usando Python. Um exemplo utiliza aprendizado de máquina e o outro, léxicos de sentimentos.

Bromberg (2013) demonstra quais os procedimentos que realizou para fazer uma análise de sentimentos da sentença com aprendizado de máquina. Foram utilizados dois *corpora* para treinar e testar o classificador. Cada *corpus* é composto de mais de cinco mil frases sobre *reviews* de filmes, um possui frases positivas e o outro, frases negativas. O classificador utilizado foi o *Naive Bayes*. Foram usados diversos mecanismos de seleção de *features* para que posteriormente fosse feita a avaliação do classificador para cada forma de seleção. Foram criados 6 métodos de seleção de *features*: em um método foram selecionadas todas as palavras do *corpus* e nos outros métodos foram selecionadas as 10, 100, 1000, 10000 e 15000 melhores palavras do *corpus*. As melhores palavras são aquelas que possuem maior ganho de informação. Ganho de informação "mede quão bem um dado atributo separa os exemplos de treinamento de acordo com a classificação alvo" (KOERICH, 2005). Para fazer o cálculo do ganho de informação foi utilizado o teste do qui-quadrado (x^2).

$$x^2 = \Sigma(\frac{(Observado - Esperado)^2}{Esperado})$$

Para cada palavra calcula-se o qui-quadrado. Uma palavra pode estar no *corpus* positivo e negativo. O valor observado é a frequência que uma palavra aparece no *corpus* de uma polaridade. Na primeira iteração do somatório, o valor observado é calculado a partir do *corpus* positivo, e na segunda, do *corpus* negativo. Valor esperado é a frequência que uma palavra aparece nos dois *corpora*.

Antes de iniciar o treinamento do classificador, os dados são tratados. Primeiro é realizada a leitura dos dois arquivos que contêm os dados. Os arquivos são divididos em sentenças e para cada sentença são selecionadas as palavras de acordo com um determinado mecanismo de seleção. As palavras selecionadas são adicionadas à estrutura de dados do tipo dicionário e etiquetadas com a *tag* "pos", se for positivo ou "neg", se for negativo. Por fim, três quartos dos dados serão usados no treinamento e um quarto serão usados para teste.

O classificador é treinado e feito o mapeamento entre o resultado do teste e o resultado esperado, para realizar a avaliação. Os critérios de avaliação foram: *accuracy*, *precision* e *recall*. Analisando os resultados obtidos para os diferentes mecanismos de seleção, percebe-se que das 10 para as 10000 melhores palavras,

houve um aumento substancial nos valores dos critérios de avaliação. Porém das 10000 para as 15000 esse aumento foi quase imperceptível. O autor conclui dizendo que aumentar a quantidade de *features* não significa que vai ter melhores resultados, mas que a seleção das *features* tem que ser feita de forma inteligente para ter melhores resultados.

Outro trabalho, realizado por Alba (2012), traz um exemplo de análise de sentimentos do documento como um todo utilizando léxicos de sentimentos. O algoritmo faz a pontuação de um documento a partir da soma das pontuações de cada sentença. Uma pontuação positiva ou negativa significa que o documento é predominante positivo ou negativo, respectivamente. Esse é um exemplo básico de análise em que o léxico possui apenas cinco adjetivos negativos e quatro positivos. Esse exemplo conta também com duas palavras que invertem a polaridade dos adjetivos (“não” e “falta”), três palavras que aumentam (“muito”, “extremamente” e “demasiado”) e duas que diminuem (“pouco” e “apenas”) a intensidade dos adjetivos.

O autor inicia informando um texto, que contém uma opinião predominantemente negativa sobre um restaurante. O texto é pré-processado, separado em palavras e cada palavra é etiquetada com *PoS tag*. É feita uma busca pelas palavras no texto que estão no léxico de sentimentos e, quando uma palavra é encontrada, ela é etiquetada com sua respectiva polaridade. O algoritmo etiqueta também as palavras que invertem a polaridade, aumentam e diminuem a intensidade dos adjetivos. Ao final, é realizada a contagem dos pontos em cada frase com base nas etiquetas das palavras. A pontuação final é definida pela soma da pontuação de cada frase.

A abordagem baseada em léxico utilizado por Alba (2012) é a mesma que será utilizada nesse trabalho. O seu artigo contém algumas etapas básicas para realização da análise de sentimentos que se assemelham bastante aos que serão realizados nesse projeto, tendo grande relevância para a construção da segunda versão da ferramenta SentimentALL. Apesar da abordagem utilizada por Bromberg (2013) ser diferente, há contribuições importantes, como por exemplo, formas de realizar uma avaliação da ferramenta de análise de sentimentos e como, modificando algumas variáveis da ferramenta, é possível mudar consideravelmente

os resultados obtidos. Os dois artigos também são interessantes para familiarização do modo como a NLTK funciona juntamente com o Python.

3 METODOLOGIA

Este trabalho é uma pesquisa aplicada a fins práticos, no qual se objetiva desenvolver uma ferramenta em Python para análise de sentimentos de avaliações de destinos turísticos disponíveis na internet.

3.1 MATERIAIS

Para a criação do referencial teórico, foram utilizados artigos científicos, livros, teses, dissertações, TCCs e sites da internet.

A Figura 10 apresenta os materiais usados na coleta e no armazenamento dos dados.

Figura 10: Materiais usados na coleta de dados

O módulo de coleta de dados foi desenvolvido baseado nos módulos desenvolvidos em Python nos estágios dos alunos do CEULP/ULBRA, Pedro Henrique Gomes Camargo e Kevin Martins Araújo. Esse módulo, desenvolvido usando a biblioteca Scrapy, substituiu o import.io, usado na primeira versão do SentimentALL. O import.io[1] possui algumas falhas, que foram resolvidas utilizando esse módulo, por exemplo: acessar URL já visitadas, não poder pausar a execução e caso haja perda de internet, a execução deve ser reiniciada. Os dados coletados são avaliações sobre destinos turísticos brasileiros (atrações, hotéis e restaurantes) escritos em língua portuguesa extraídos do site TripAdvisor[2].

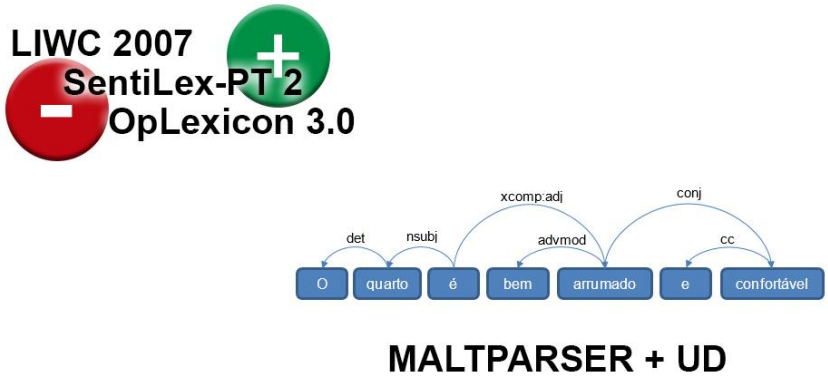
[1] https://www.import.io
[2] https://www.tripadvisor.com.br

No armazenamento dos dados foi utilizado o banco dados Microsoft SQL Server[3] 2014. Nesse banco conterá todos os dados resultantes dos módulos de coleta, pré-processamento e análise de sentimentos.

Para o módulo de pré-processamento foi utilizado o módulo de *PoS tag* criado especificamente para a ferramenta SentimentALL. Esse módulo foi desenvolvido em Python, usando a biblioteca NLTK, no estágio do aluno da CEULP/ULBRA, Matheus Rodrigues Leal. O módulo de *PoS tag* contém a normalização, porém, a ferramenta realiza a normalização e o *PoS tagging* em momentos distintos, então foi necessária a divisão desse módulo em dois: o módulo de normalização e o módulo de *PoS tag*. Outras alterações foram realizadas, por exemplo, na normalização foram adicionadas nova expressões regulares e o processo de *PoS tag* foi dividido em: treinamento do classificador e *PoS tagging*.

A Figura 11 apresenta os materiais usados na análise de sentimentos.

Figura 11: Materiais usados na análise de sentimentos



Na análise de sentimentos foram utilizados os seguintes léxicos de sentimentos para o português: SentiLex-PT[4] versão 2, LIWC[5] 2007 e OpLexicon[6] v3.0.

- SentiLex-PT versão 2 (SILVA; CARVALHO; SARMENTO, 2012) possui 7014 lemas e 82347 formas flexionadas. O SentiLex-PT

[3] http://www.microsoft.com/SQL
[4] http://dmir.inesc-id.pt/project/SentiLex-PT_02
[5] http://143.107.183.175:21380/portlex/index.php/pt/projetos/liwc
[6] http://ontolp.inf.pucrs.br/Recursos/downloads-OpLexicon.php

versão 2 pode ser obtido através de uma solicitação para o e-mail: dmir-resources@inesc-id.pt. Esse léxico de sentimentos inclui em cada linha, informações sobre: lema, a flexão em gênero e número do lema, categoria gramatical, polaridade e como foi atribuída a polaridade.

- LIWC 2007 (BALAGE FILHO; PARDO; ALUÍSIO, 2013) possui mais de 127 mil palavras e, para cada palavra, é atribuída uma ou mais categorias. Das sessenta e quatro categorias, apenas duas são relevantes para esse trabalho: posemo (emoções positivas) e negemo (emoções negativas). O LIWC 2007 é composto por mais de 12 mil palavras categorizadas como posemo e mais de 15 mil como negemo.
- OpLexicon v3.0 (SOUZA *et al*, 2012) é composto por mais de 32 mil palavras, em que 14 mil são negativas, 9 mil neutras e 9 positivas. Esse léxico possui 4 categorias de palavras: verbos, adjetivos, hashtag e emoticons.

Para a identificação dos aspectos e opiniões, foi necessário conhecer as relações sintáticas entre as palavras de uma sentença. Para isso, foi utilizada a MaltParser[7], uma ferramenta JAVA em que é possível criar um modelo que realiza a análise de dependência de uma frase. Para o treinamento e teste do modelo foi utilizado o UD[8] (*Universal Dependencies*), que está na sua segunda versão. O UD contém *corpora* anotados sintaticamente de quase cem idiomas, inclusive para o português brasileiro. O corpus para o português possui um pouco mais de nove mil sentenças para treinamento e mil para teste. Os corpora seguem o formato CoNLL-U. A Figura 12 apresenta um exemplo de um arquivo nesse formato.

[7] http://www.maltparser.org
[8] http://universaldependencies.org

Figura 12: Exemplo arquivo no formato CoNLL-U

```
1   Para            _   ADP     ADP     _   3   case    _   _
2   o               _   DET     DET     _   3   det     _   _
3   Planejamento    _   PROPN   PNOUN   _   9   nmod    _   SpaceAfter=No
4   ,               _   PUNCT   .       _   3   punct   _   _
5   as              _   DET     DET     _   6   det _   _
6   áreas           _   NOUN    NOUN    _   9   nsubj   _   _
7   prioritárias    _   ADJ     ADJ     _   6   amod    _   _
8   serão           _   VERB    VERB    _   9   cop     _   _
9   educação        _   NOUN    NOUN    _   0   root    _   SpaceAfter=No
10  ,               _   PUNCT   .       _   11  punct   _   _
11  saúde           _   NOUN    NOUN    _   9   conj    _   _
12  e               _   CCONJ   CONJ    _   13  cc      _   _
13  segurança       _   NOUN    NOUN    _   9   conj    _   _
14  pública         _   ADJ     ADJ     _   13  amod    _   SpaceAfter=No
15  .               _   PUNCT   .       _   9   punct   _   _
```

Nesse formato, cada linha representa um *token*, em que as informações principais são: a posição do *token* na sentença (coluna 1), o *token* (coluna 2), a *PoS tag* (coluna 5), qual *token* ele tem relação (coluna 7) e qual o tipo de relação (coluna 8).

As diferenças da primeira versão com a segunda são: mudança de linguagem de programação JAVA para Python, substituição do import.io pelo Scrapy e substituição do CogrOO[9] (Corretor Gramatical acoplável) pela NLTK[10] (*Natural Language ToolKit*).

A Figura 13 apresenta os materiais usados durante todo o desenvolvimento da ferramenta.

Figura 13: Materiais usados em todo o projeto

[9] http://cogroo.sourceforge.net
[10] http://www.nltk.org

Conforme a Figura 13, para o desenvolvimento da ferramenta foi utilizada a linguagem Python, codificando através da *Integrated Development Environment* (IDE) PyCharm[11] da Jet Brains. A IDE PyCharm permite a codificação seja mais rápida com o auto completar, possui uma ferramenta de depuração de código, o console interativo do python é integrado à IDE, e possui outras ferramentas que ajudaram no processo de desenvolvimento desse trabalho.

Foi utilizada a biblioteca NLTK para processamento de linguagem natural. Essa biblioteca contém por volta de 35 módulos, cada um com diversos submódulos, que realizam múltiplas tarefas de PLN, como tokenização, *PoS tagging,* tem módulo para análise de sentimentos, e entre outros.

3.2 BASE DE DADOS

No início de fevereiro de 2017, o TripAdvisor continha aproximadamente 9 milhões de avaliações escritos em português de 5376 cidades brasileiras. Como o tempo é limitado, então foram selecionados apenas as cem cidades (APÊNDICE B) mais comentadas para a extração, resultando em 6 milhões e 300 mil avaliações extraídas entre o início de fevereiro e o final de março. O quantitativo de avaliações extraídas representa 71% das avaliações totais.

Os dados extraídos foram os seguintes.

- Nome do objeto
- Código do objeto
- Tipo do objeto (Atração, Hotel ou Restaurante)
- Nome da cidade
- Código da cidade
- Nome do estado
- Nome do país
- Nome do autor
- Cidade do autor
- Nível de colaboração autor

[11] https://www.jetbrains.com/pycharm

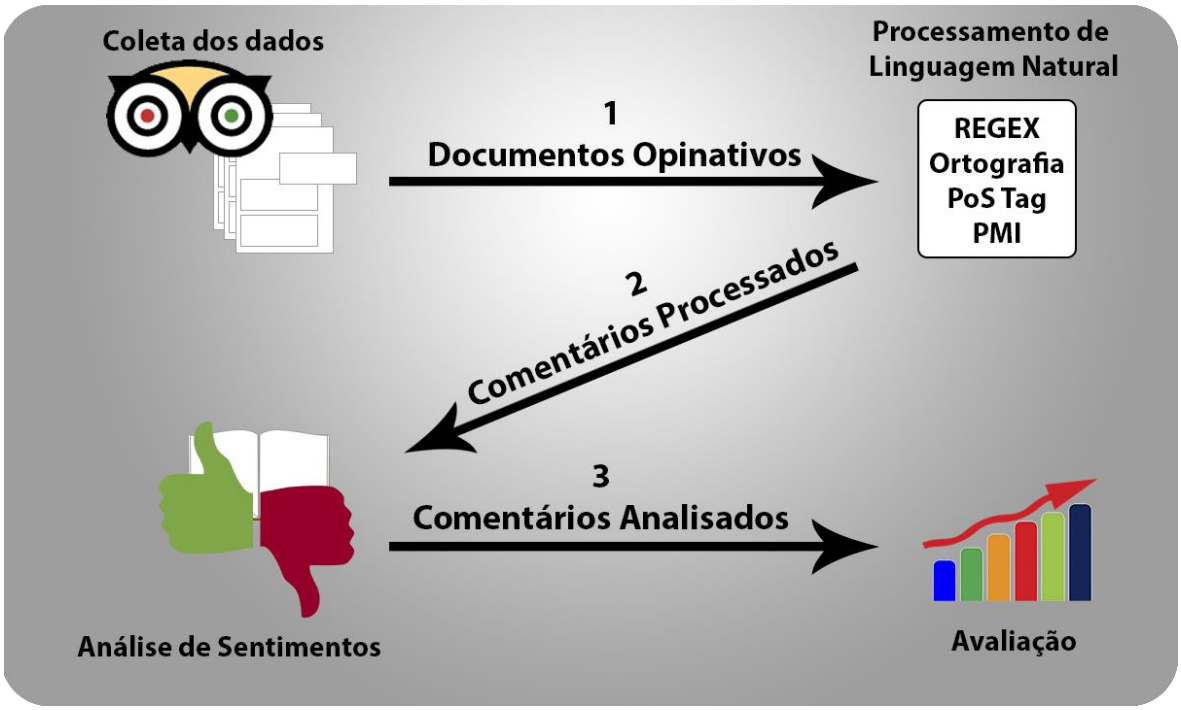
- Ano de cadastro do autor no site
- Código do autor
- Título
- Nota (de 1 a 5 dada pelo autor)
- Data de postagem
- Comentário
- Código da avaliação
- Útil (quantidade de leitores que votaram como útil a avaliação)

Resumindo, foram extraídos dados do autor da avaliação, da cidade/destino, do objeto e da avaliação.

3.3 PROCEDIMENTOS

Os procedimentos foram sistematizados em: coleta de dados, processamento de linguagem natural, análise de sentimentos e avaliação. Durante esse processo foram realizadas reuniões com a responsável pelo projeto da SentimentALL, prof.a M.e Parcilene Fernandes de Brito, para o entendimento da ferramenta. A Figura 14 ilustra as etapas que foram definidas para o desenvolvimento da ferramenta.

Figura 14: Procedimentos



Conforme demonstrado na Figura 14, para o desenvolvimento desse trabalho, inicialmente foi realizada a etapa de coleta de dados. Nessa etapa, foi necessário o entendimento do contexto dos dados (TripAdvisor) para o desenvolvimento do módulo de Coleta de Dados. Para realizar a coleta de dados, foi analisado como as páginas HTML (Hypertext Markup Language) do TripAdvisor estão estruturadas, para que possa ser realizado a extração apenas das informações (avaliações) desejadas. A coleta dos dados foi realizada extraindo as avaliações de atrações, hotéis e restaurante de cada destino turístico brasileiro presente no site TripAdvisor. Essas avaliações comporão a base de dados opinativos (1) extraídos para posteriormente serem processadas e analisadas.

A segunda etapa é a de pré-processamento, em que foi realizada o processamento de linguagem natural. Para realização do PLN, desenvolveu-se o módulo de pré-processamento. Nele foi feita a normalização do texto de entrada (usando expressões regulares) e a correção ortográfica, utilizando o módulo *PoS tagging* desenvolvido no Estágio do aluno Matheus Rodrigues Leal, e implementado o algoritmo de PMI para a definição de expressões compostas. Esse módulo teve como entrada os dados coletados e como saída (2) os dados normalizados, corrigidos ortograficamente, etiquetados e as possíveis expressões multipalavras identificadas.

A próxima etapa é a de análise de sentimentos, em que foi desenvolvido o módulo de análise de sentimentos. Esse módulo recebe os dados processados na etapa anterior e o léxico de sentimentos. A partir desses dados, faz a identificação da palavra opinativa e de sua polaridade (utilizando o SentiLex-PT, LIWC e OpLexicon) e identificação dos aspectos. Esse módulo tem como saída (3), para cada comentário analisado, uma lista, em que cada item contém uma opinião presente no comentário, a polaridade da opinião e o aspecto relacionado.

Ao final realizou-se a avaliação da ferramenta. Nessa avaliação foi feita a análise de sentimentos utilizando a versão 1 da SentimentALL, a versão 2 e uma análise manual. Foram fornecidos os mesmos cem comentários como entrada para cada análise de sentimentos realizada, sendo que os comentários foram selecionados aleatoriamente. A análise manual e a análise da versão 1 foram realizadas em Leal, Oliveira e Brito (2017). Os resultados da análise manual, da versão 1 e da versão 2 são inseridos em uma planilha. A partir disso, é realizada

uma análise quantitativa e uma qualitativa entre as ferramentas. Na primeira análise são comparados apenas os aspectos e na segunda são consideradas também a opinião e a polaridade. Com a equação F-Measure foi possível medir o desempenho geral de cada ferramenta e, com isso, foi possível saber qual versão teve o melhor desempenho.

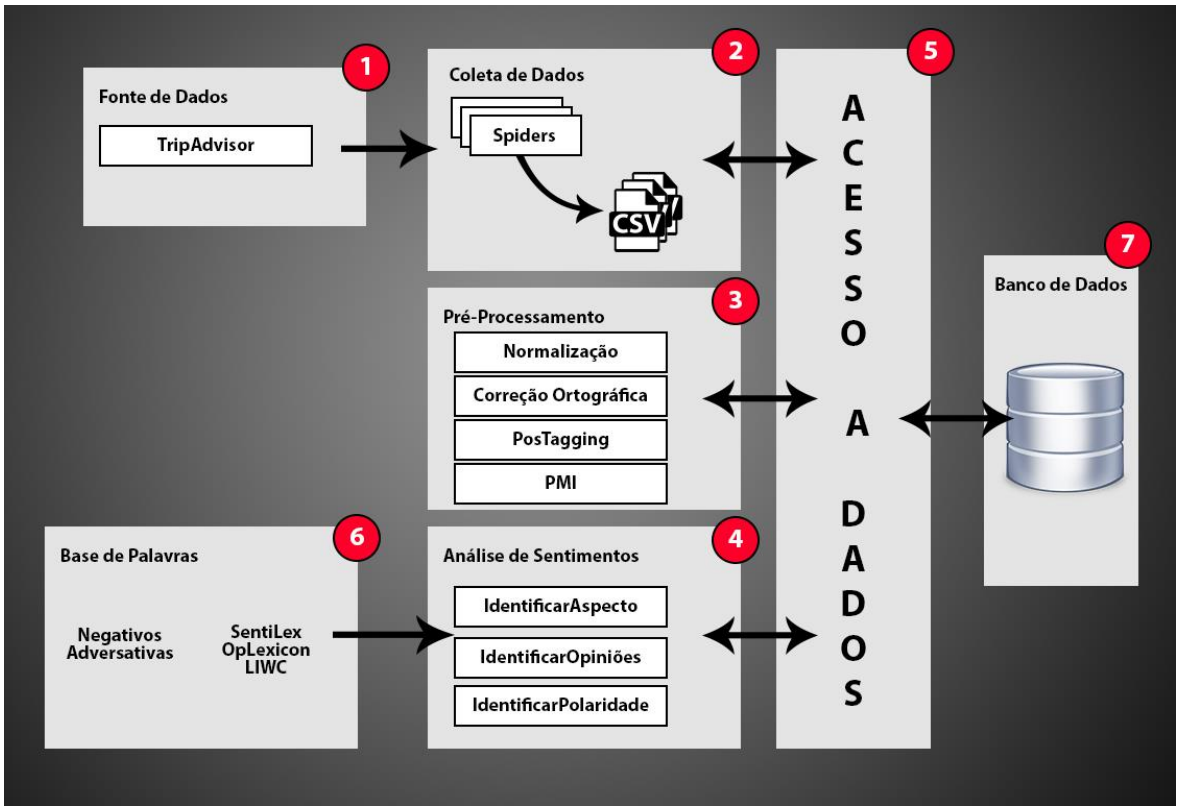
4 RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados obtidos no desenvolvimento desse trabalho, bem como as discussões referentes aos resultados. Inicialmente será apresentada uma visão geral da ferramenta. Posteriormente serão apresentados, com mais detalhes, cada módulo da ferramenta na seguinte ordem: coleta de dados, pré-processamento e análise de sentimentos.

4.1 VISÃO GERAL

Nesta seção será apresentada uma visão geral da ferramenta, demonstrando todos os seus componentes através da arquitetura da SentimentALL versão 2. A Figura 15 apresenta a arquitetura da ferramenta.

Figura 15: Arquitetura da SentimentALL versão 2



A seguir será descrito cada componente da arquitetura da Figura 15:

- **Fonte de Dados** (Figura 15-1): mostra a origem dos dados utilizados nesse trabalho, no caso, o site de Turismo TripAdvisor. O TripAdvisor contém milhões de avaliações sobre atrações, hotéis e restaurantes de diferentes partes do mundo. As avaliações realizadas pelos usuários são compostas de diversas informações, como um título, a

data de postagem, uma nota, um comentário, dentre outras. A informação mais importante dentre estas, são os comentários, pois, a partir deles é possível realizar a análise de sentimentos.

- **Coleta de Dados** (Figura 15-2): tem como objetivo extrair os dados do TripAdvisor utilizando uma *spider*, que será apresentada com mais detalhes na próxima seção. Foi criada uma *spider* diferente para cada tipo de objeto (atrações, hotéis e restaurantes), pois as estruturas das páginas são distintas. O arquivo CSV representa a saída do módulo de coleta de dados, sendo que cada *spider* gera um arquivo diferente. Contém todas as informações referentes às avaliações dos usuários, como os dados do usuário, do destino e a avaliação em si. Depois, é lido cada arquivo CSV, realizada a limpeza dos dados e os mesmos são carregados para a base de dados.
- **Pré-Processamento** (Figura 15-3): realiza a divisão dos comentários em sentenças, no mesmo passo que os normaliza e corrige a ortografia. Depois, para cada sentença, realiza a etiquetagem com PoS tag de cada token da sentença. De posse dos tokens etiquetados, então é realizado o cálculo do PMI para encontrar possíveis expressões multipalavras.
- **Análise de Sentimentos** (Figura 15-4): o módulo central da ferramenta, responsável pela identificação dos aspectos e das palavras opinativas que os caracteriza. Para aumentar a precisão dos resultados foram utilizados uma base de palavras. Posteriormente é identificada a polaridade das palavras opinativas.
- **Acesso a dados** (Figura 15-5): toda e qualquer operação de leitura e escrita na base de dados realizada pelos outros módulos deve primeiro passar pela camada de acesso à dados. Essa camada contêm todos os modelos de dados necessários para se ter acesso aos dados do banco de dados.
- **Base de Palavras** (Figura 15-6): contém listas de palavras que são necessárias para a realização da análise de sentimentos. As listas contemplam palavras negativas, adversativas, *stopwords* e delimitadores, cada uma separada em arquivos distintos. Contém

também os léxicos de sentimentos, dentre eles estão presentes o SentiLex-Pt, o OpLexicon e o LIWC.

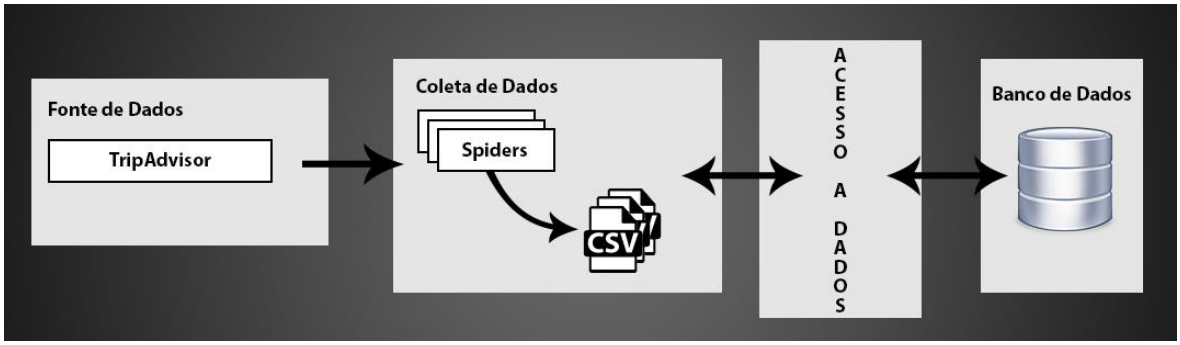
- **Base de Dados** (Figura 15-7): responsável pelo armazenamento dos dados. Essa base de dados contém todos os dados originais, bem como os dados resultantes da etapa de pré-processamento e de análise de sentimentos

4.2 COLETA DOS DADOS

O módulo para coleta de dados do site TripAdvisor foi desenvolvido utilizando um *framework* para Python chamado Scrapy. O Scrapy permite a criação de *spiders*, que são programas que navegam por páginas e extraem dados presentes nelas. A coleta de dados foi dividida em dois momentos, sendo que para cada um, foram criadas três *spiders*. Uma *spider* para cada tipo de objeto. No primeiro momento, as *spiders* criadas fazem uma espécie de sondagem, coletando a quantidade de avaliações para cada objeto de todas as cidades do Brasil. No segundo momento, foi realizado um filtro, selecionando para extração dos dados as cem cidades com a maior quantidade de avaliações.

A Figura 16 apresenta a parte da arquitetura relacionada ao módulo de coleta de dados.

Figura 16: Coleta de Dados - Arquitetura



Conforme a Figura 16, o módulo de coleta de dados tem com fonte o site TripAdvisor. Dessa fonte são extraídos os dados de cada tipo de objeto utilizando as *spiders* e gerando um arquivo CSV para cada coleta realizada. O arquivo CSV contém todas as informações referentes às avaliações dos usuários, como os dados do usuário, do destino e a avaliação em si. Depois, é lido cada arquivo CSV, realizada a limpeza dos dados e os mesmos são carregados para a base de dados.

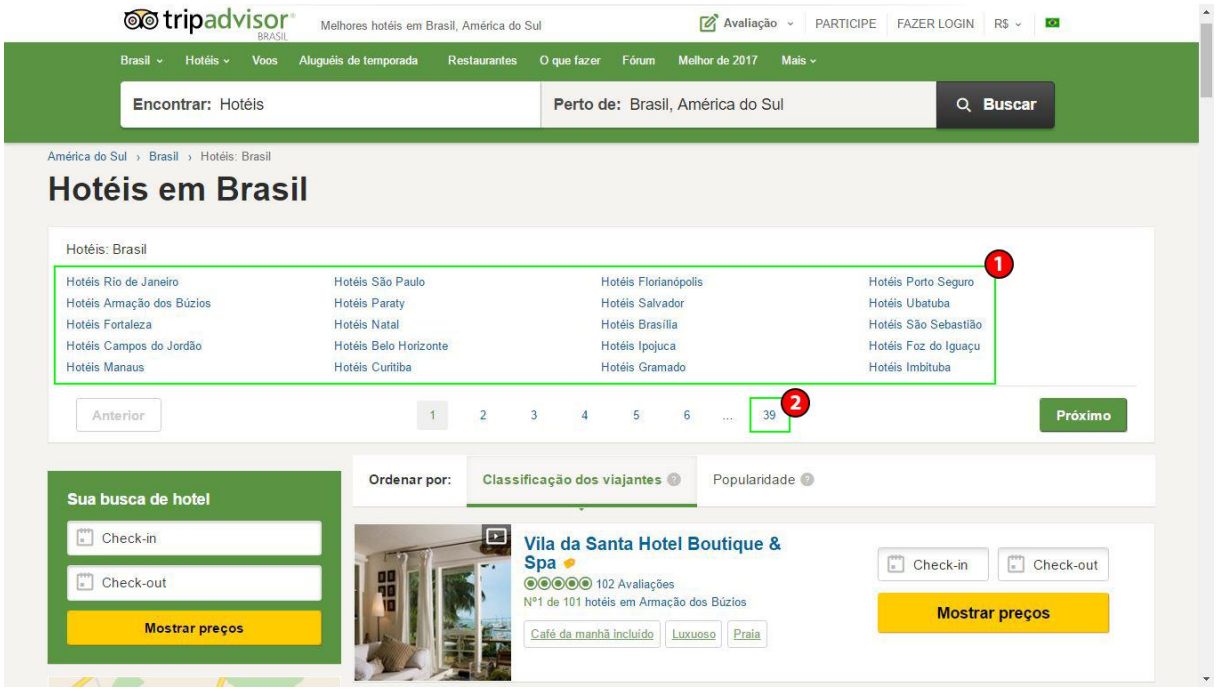
A seguir será apresentado com mais detalhes o processo de extração da quantidade de avaliações e a extração das avaliações.

4.2.1 *Spiders* para extração da quantidade de avaliações

Nessa seção será demonstrado, como exemplo, o funcionamento da *spider* que realiza a extração da quantidade de avaliações de hotéis. Apesar das *spiders* de hotéis, atrações e restaurantes não serem iguais, elas se assemelham bastante.

Inicialmente deve ser definido o *link* da página inicial de onde a *spider* partirá e navegará até encontrar a página do objeto. Os *links* iniciais definidos são os seguintes: atrações (https://www.tripadvisor.com.br/Attractions-g294280-Activities-Brazil.html), hotéis (https://www.tripadvisor.com.br/Hotels-g294280-Brazil-Hotels.html) e restaurantes (https://www.tripadvisor.com.br/Restaurants-g294280-Brazil.html). A Figura 17 mostra a página inicial da *spider* de hotéis, contendo a lista de cidades.

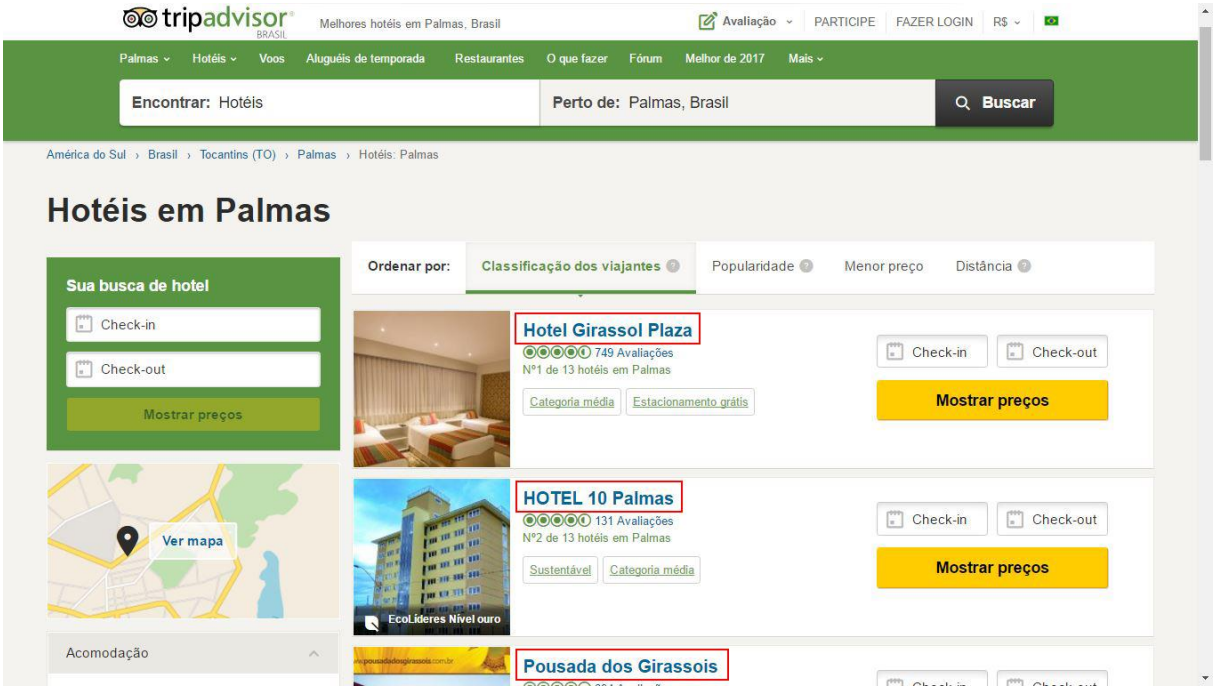
Figura 17: Página Inicial com a lista de cidades



Como é possível observar na Figura 17, a página inicial dos hotéis é composta por diversos *links* (Figura 17-1) que dão acesso a lista de hotéis de cada cidade. Então, a *spider* percorre toda a página e extrai e acessa o *link* de cada cidade. Como há uma paginação na página inicial que levará às outras listas de

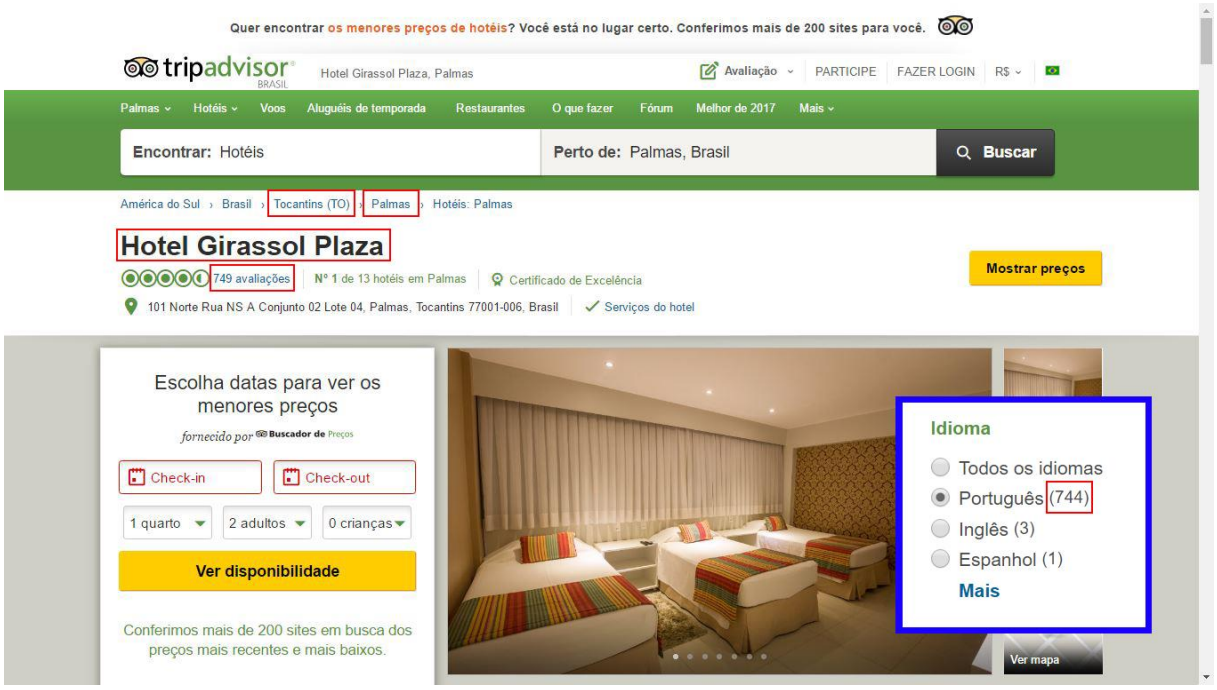
cidades, dessa forma a *spider* extrai a quantidade de páginas (Figura 17-2) e monta os *links* de cada página da paginação. Como em cada página há vinte cidades, então para montar os *links* deve ser incrementado de vinte em vinte o valor de “{X}” no *link*: (https://www.tripadvisor.com.br/Hotels-g294280-oa{X}). A partir desses novos *links* é possível extrair novas cidades. Acessando os *links* de cada cidade, extraídos anteriormente, chega-se, por exemplo, na página apresentada na Figura 18.

Figura 18: Página com a lista de hotéis



A Figura 18 apresenta um exemplo da página da cidade contendo a lista de hotéis. Análogo ao processo anterior, na página da cidade, os *links* (destacados em vermelho na Figura 18) extraídos e acessados são os dos hotéis. Caso haja paginação, o processo é parecido com o anterior, só que o valor “{X}” no link (https://www.tripadvisor.com.br/Hotels-{CÓDIGOCIDADE}-oa{X}) deve ser incrementado de trinta em trinta e “{CÓDIGOCIDADE}” deve ser substituído pelo código da cidade (encontrado na URL da página atual). Por último, a página de cada hotel é acessada. A Figura 19 apresenta a página de um hotel.

Figura 19: Página do Hotel



O objetivo desse *spider* é chegar nessa página (Figura 19), em que serão extraídos os dados e serão salvos em um arquivo CSV. Os dados extraídos (destacados em vermelho na Figura 19) são: cidade, estado, nome do objeto, quantidade de avaliações e quantidade de avaliações em português. Além desses, o código da cidade e o código do objeto, que podem ser encontrados na URL da página, também foram extraídos.

A seção a seguir apresentará a outra etapa da coleta, que trata do uso do resultado (arquivo CSV) dessa etapa para realizar a filtragem das cidades.

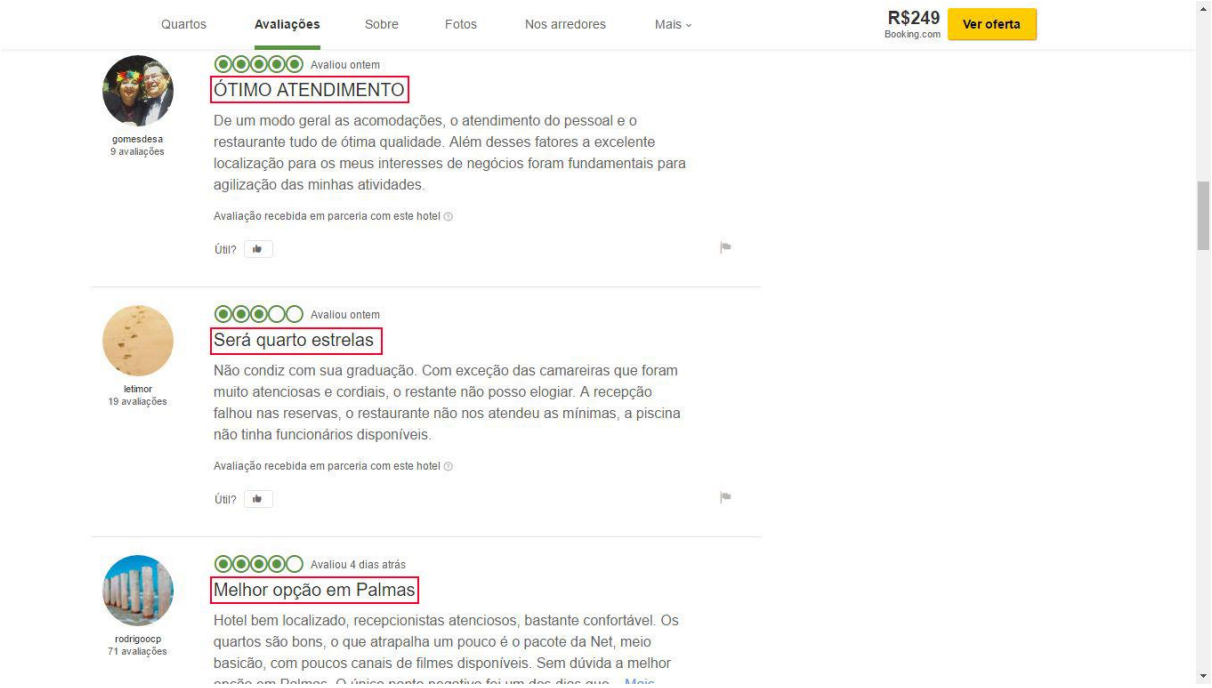
4.2.2 *Spiders* para extração de avaliações

Da mesma forma que na seção anterior, para explicar o funcionamento da *spider* para extração de avaliações será apresentada como exemplo a de hotéis. Na construção da *spider*, primeiro devem ser definidos os *links* iniciais. Para isso foi criado a classe `FiltraCidades`. Essa classe possui o método `montar_links`, que tem a função de montar os *links* de cada objeto de um determinado conjunto de cidades (filtrado pelo usuário) e retornar os *links* para que estes possam ser usados pela *spider* como *links* iniciais. O método recebe como parâmetro os códigos das cidades que serão extraídas, o arquivo CSV (extraído na etapa

anterior) e o tipo do objeto. A partir desses parâmetros é possível montar os *links* fazendo uma varredura nos arquivos CSV, buscando por todos os objetos das cidades selecionadas e preenchendo os links abaixo com o código da cidade “{CÓDIGOCIDADE}” e o código do objeto “{CÓDIGOOBJETO}”. A seguir os links para atrações, hotéis e restaurantes, respectivamente:
https://www.tripadvisor.com.br/Attraction_Review-{CÓDIGOCIDADE}-{CÓDIGOOBJETO}
https://www.tripadvisor.com.br/Hotels_Review-{CÓDIGOCIDADE}-{CÓDIGOOBJETO}
https://www.tripadvisor.com.br/Restaurant_Review-{CÓDIGOCIDADE}-{CÓDIGOOBJETO}

Com todos os *links* montados, a extração é iniciada a partir da página do hotel (a última página acessada pela primeira *spider*). Ao rolar para baixo a página do hotel (Figura 19), uma prévia das avaliações é exibida, conforme apresentado na Figura 20.

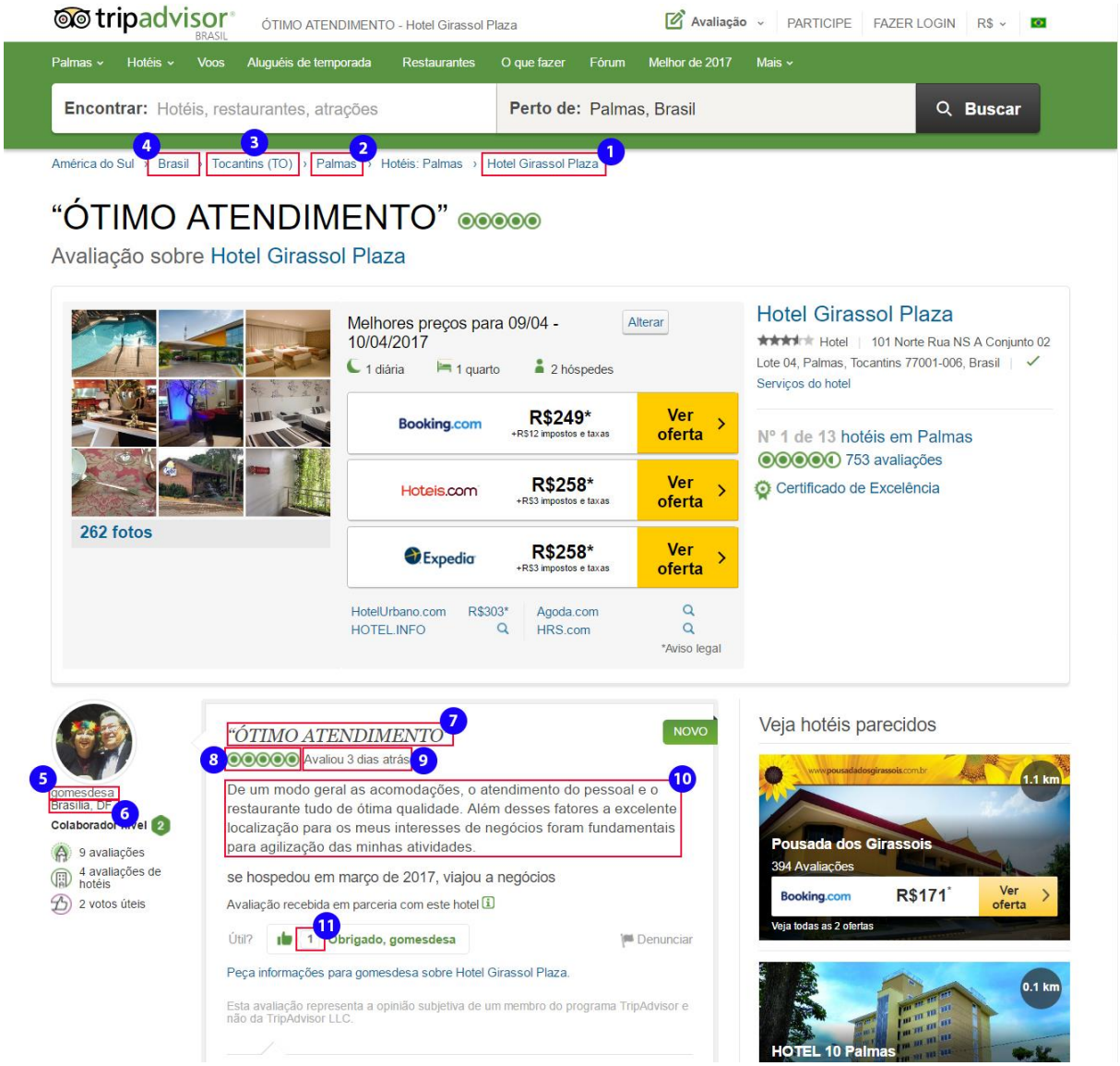
Figura 20: Página do Hotel (Avaliações)



A Figura 20 apresenta um exemplo da página do hotel na seção das avaliações. Na página do hotel, os *links* (destacados em vermelho na Figura 20) extraídos e acessados são das avaliações. Caso haja paginação, o valor “{X}” no *link* (https://www.tripadvisor.com.br/Hotel_Review-{CÓDIGOCIDADE}-{CÓDIGOOBJETO}-or{X}) deve ser incrementado de cinco em cinco, “{CÓDIGOCIDADE}” deve ser substituído pelo código da cidade e

“{CÓDIGOOBJETO}” pelo código do objeto (os dois código são encontrados na URL da página atual). Por último, a página de cada avaliação é acessada. A Figura 21 apresenta a página de uma avaliação.

Figura 21: Página da Avaliação

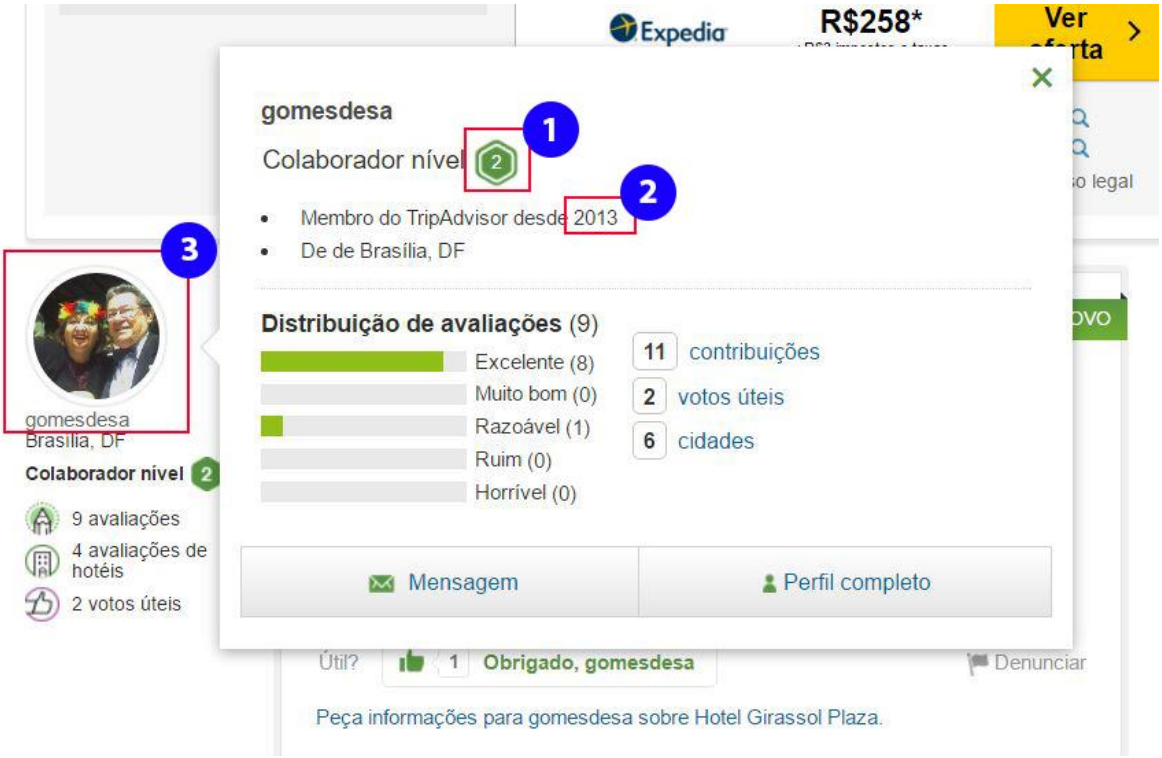


Na Figura 21 é apresentada a página final onde os dados serão extraídos e salvos em um arquivo CSV. Os dados (destacados em vermelho na Figura 21 e na Figura 22) são:

- Objeto (Figura 21-1)
- Código do objeto (URL)
- Tipo de objeto (Atração, Hotel ou Restaurante)

- Cidade (Figura 21-2)
- Código da cidade (URL)
- Estado (Figura 21-3)
- País (Figura 21-4)
- Nome do autor (Figura 21-5)
- Cidade do autor (Figura 21-6)
- Nível do autor (Figura 22-1)
- Ano de cadastro do autor no site (Figura 22-2)
- Código do autor (Figura 22-3, `id` da `div` destacada)
- Título (Figura 21-7)
- Nota (Figura 21-8)
- Data de postagem (Figura 21-9)
- Comentário (Figura 21-10)
- Código da avaliação (URL)
- Útil (Figura 21-11)

Figura 22: Frame com informações do autor



Quando se posiciona o mouse sobre a foto do autor, um frame aparece (Figura 22), fornecendo dados (nível do autor e o ano de cadastro no site) que serão

coletados. Porém, para que a *spider* possa acessar essas informações, deverá realizar uma requisição HTTP. O modelo da URL para requisição é o seguinte: (https://www.tripadvisor.com.br/MemberOverlay?uid={CÓDIGOAUTOR}). O código do autor "{CÓDIGOAUTOR}" pode ser encontrado no `id` da `div` destacada na Figura 22-3.

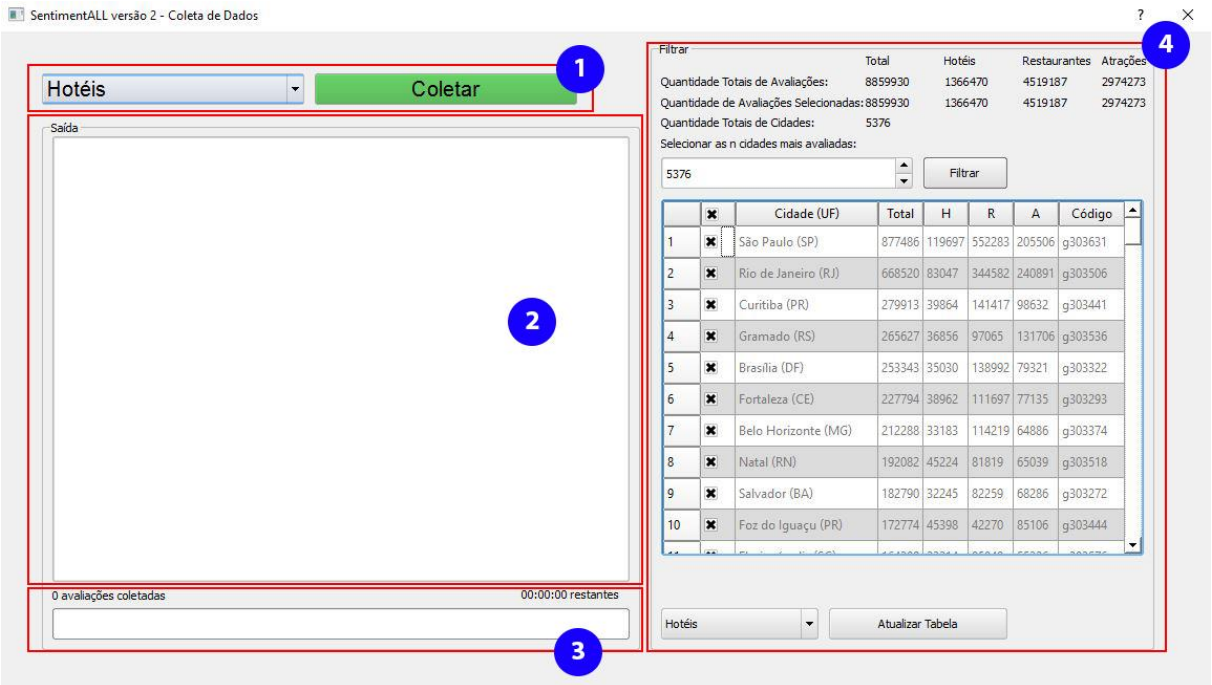
A coleta está concluída, mas falta a etapa de transformação e carga para o banco de dados. Porém, antes dessa etapa, será apresentada a interface gráfica do módulo de coleta de dados.

4.2.3 Interface gráfica do módulo de coleta de dados

O desenvolvimento dessa interface gráfica veio com o intuito de facilitar a coleta de dados do site TripAdvisor, o que tornou possível selecionar as cidades desejada sem precisar alterar o código-fonte. Essa interface facilita, também, na questão de poder estimar o tempo necessário para a finalização de uma coleta, com isso, caso esse módulo seja utilizado em algum projeto, esse tempo pode ser uma métrica útil para estimar o prazo de conclusão do projeto.

A seguir será apresentada a tela criada para a interface com o usuário e a explicação do que ela está apresentando. A tela (Figura 23) foi dividida em partes para uma melhor explicação.

Figura 23: Interface Gráfica (Módulo de Coleta)

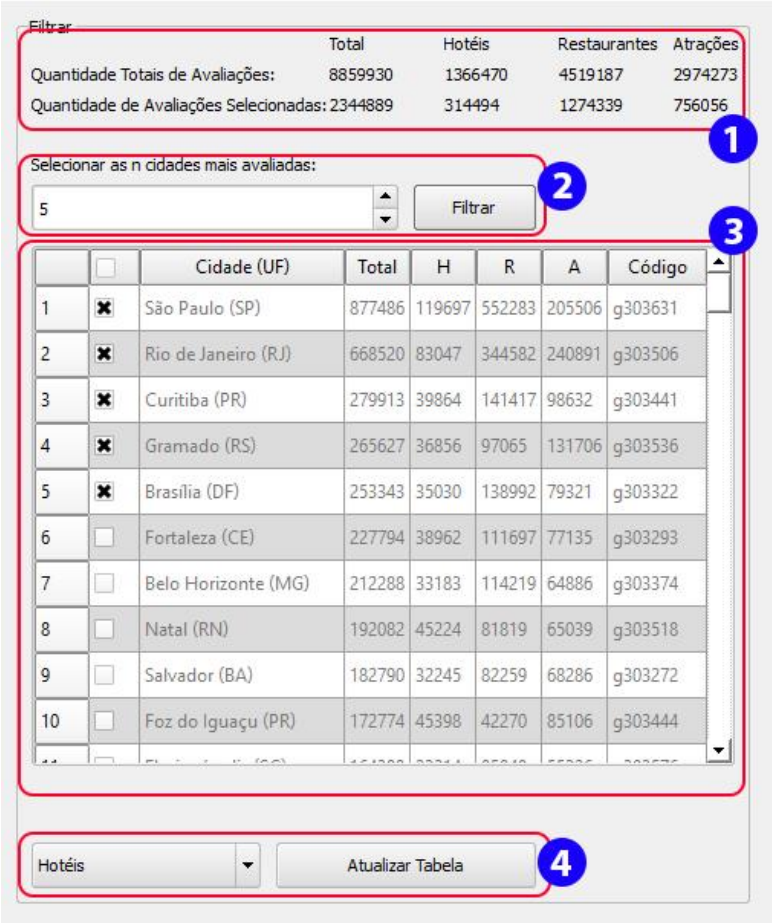


Na parte superior da tela (Figura 23-1) há um *combo box* e um botão. O *combo box* é utilizado para selecionar o tipo de objeto que será coletado. O botão “Coletar”, ao ser clicado, abre uma janela para que seja informado o nome do arquivo CSV de saída, depois inicia a execução da coleta e o *label* do botão muda para “Parar coleta”, que ao ser clicado interromperá a coleta.

A seguir, há uma área de texto (Figura 23-2) que mostra o log fornecido pelo próprio Scrapy. Abaixo da caixa de texto, tem uma barra de *status* (Figura 23-3) contendo a quantidade de avaliações coletadas, um tempo estimado de término da coleta e uma barra de progresso.

No lado direito da tela (Figura 23-4) tem um painel com algumas informações sobre os dados. Nesta tela também é possível selecionar as cidades para realizar a filtragem. Para uma melhor visualização, a Figura 24 mostra a ampliação da Figura 23 parte 4.

Figura 24: Interface Gráfica (Painel do Módulo de Coleta)

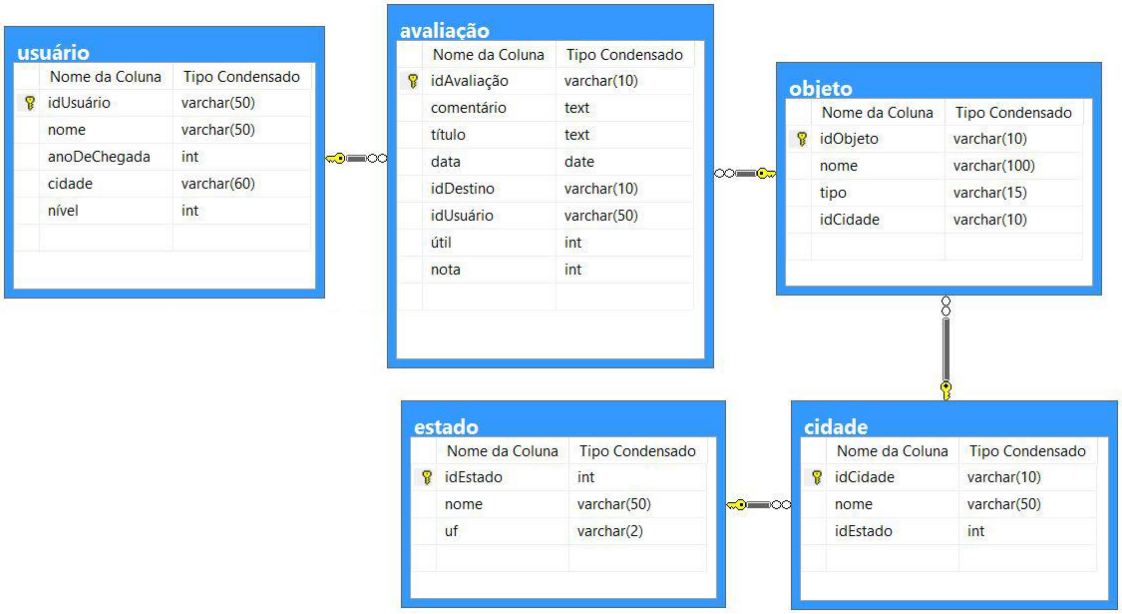


As informações contidas no painel da Figura 24 foram construídas a partir do arquivo CSV de saída da *spider* que extraiu a quantidade de avaliações. Na Figura 24-1 são informadas a quantidade de avaliações e a quantidade de avaliações selecionadas para os tipos de objetos e para o total. Abaixo (Figura 24-2), há uma caixa onde é possível digitar um valor numérico “n” que determina quantas cidades mais avaliadas serão selecionadas para coleta. Na tabela (Figura 24-3), são apresentadas as cidades e para cada uma delas há um *checkbox*, o nome da cidade, a quantidade de avaliações de Hotéis “H”, Restaurantes “R” e Atrações “A”, a quantidade total de avaliações e o código da cidade. Quando o usuário pressiona o botão de coleta (Figura 23-1), os códigos das cidades selecionadas são usados para montar os *links* iniciais, conforme demonstrado na seção anterior. Na Figura 24-4, o botão permite atualizar a tabela executando a *spider* que extrai a quantidade de avaliações. A *spider* executada é selecionada com base no item selecionado no *combo box*.

4.2.4 Transformação e Carga

As *spiders* foram desenvolvidas para realizar todas as transformações necessárias, deixando os arquivos CSV de saída prontos para serem carregados no banco de dados. Foram criadas no banco as tabelas apresentadas no modelo relacional da Figura 25 (modelo completo no APÊNDICE A), para armazenar os dados da coleta.

Figura 25: Modelo relacional parcial – Coleta de Dados

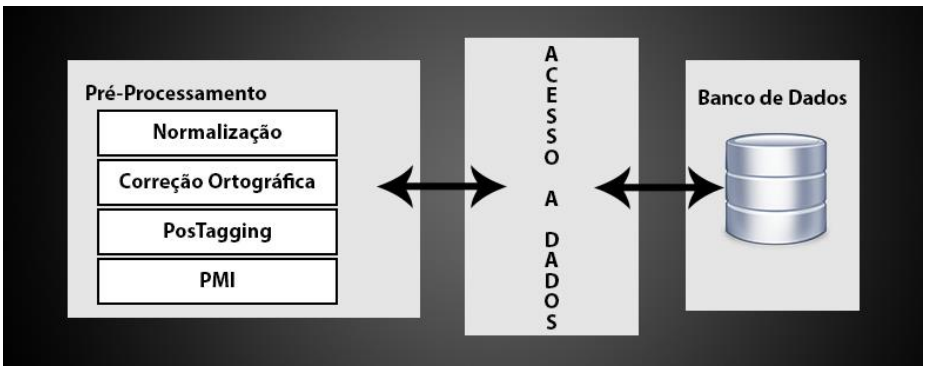


Conforme o modelo apresentado na Figura 25, para armazenar os dados extraídos foram criadas as seguintes tabelas: usuário, avaliação, objeto, cidade e estado. Para cada linha do arquivo, antes de inseri-la no banco, verifica-se o usuário, o objeto, a cidade e o estado já estão armazenados nele. Esse procedimento é necessário para evitar redundâncias nos dados. Para que não fosse realizada uma consulta no banco de dados toda vez que uma avaliação fosse inserida, foi utilizada uma lista para cada tabela, contendo o código dos itens que já foram inseridos no banco de dados. Com isso, quando for realizada a carga de uma avaliação, para cada um dos tipos de dados que podem estar redundantes (usuário, objeto, cidade e estado), verifica-se se o seu código está na lista correspondente. Se estiver, não o insere no banco, caso contrário, os dados são inseridos e um novo código é adicionado na lista correspondente.

4.3 PRÉ-PROCESSAMENTO

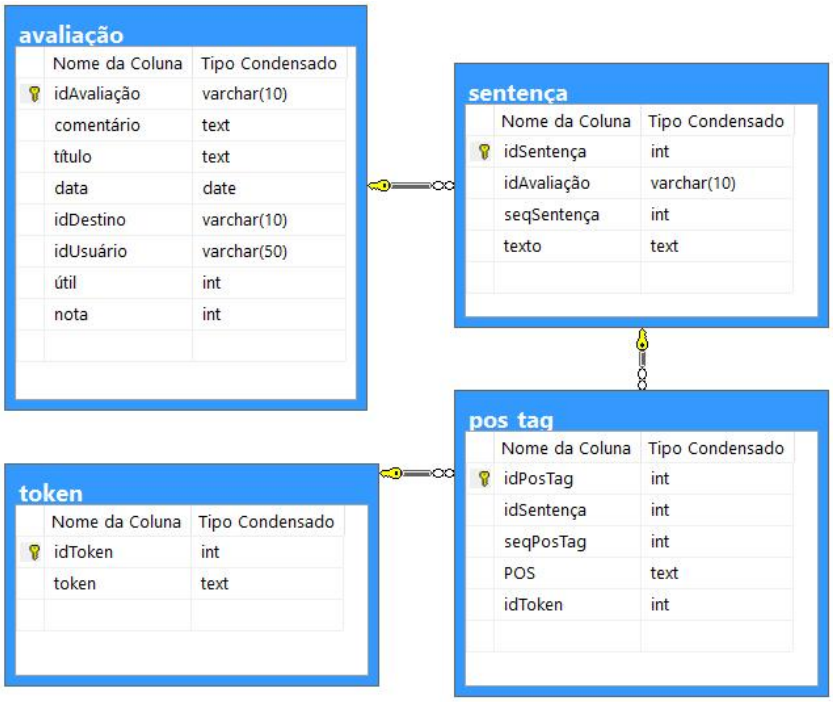
O módulo de pré-processamento é composto de quatro etapas: normalização, correção ortográfica, *PoS tagging* e identificação de expressões multipalavras. Conforme apresentada na parte da arquitetura relacionada ao módulo de pré-processamento (Figura 26).

Figura 26: Pré-processamento - Arquitetura



Para o armazenamento dos resultados do processamento dessas etapas foram criadas no banco de dados as tabelas sentença, pos_tag e token, conforme modelo apresentado na Figura 27.

Figura 27: Modelo relacional parcial – Pré-Processamento



A Figura 27 apresenta a estrutura criada para armazenar os dados do pré-processamento. O pré-processamento inicia na normalização e correção ortográfica. Essas etapas processam o comentário da avaliação. O resultado são os comentários normalizados, corrigidos ortograficamente e divididos em sentenças. As sentenças são salvas na tabela sentença e a coluna seqSentença identifica a ordem da sentença no comentário.

A próxima etapa é a de *PoS tagging*, que é aplicada em cada sentença. Nessa etapa, as sentenças são etiquetadas morfologicamente e divididas em

tokens (palavras, números e pontuações). A etiqueta de cada *token* é salva na tabela pos_tag, com a coluna seqPosTag identificando a ordem do *token* na sentença e a coluna idToken fazendo referência ao registro do *token* na tabela token.

A última etapa é a de identificação de expressões multipalavras, que utiliza os dados da tabela sentença e pos_tag para a identificação de possíveis expressões.

As etapas de pré-processamento serão apresentadas com mais detalhes nas seções a seguir.

4.3.1 Normalização

A normalização textual é uma etapa do pré-processamento que realiza a padronização dos dados, removendo ou substituindo caracteres ou palavras. A identificação dos itens a serem normalizados é realizada através da utilização de expressões regulares. A seguir um exemplo de normalização textual.

Tabela 5: Exemplo de normalização textual

Texto de Entrada	Texto Normalizado
A entrada estava R$ 20 por cabeça quando fui em junho/12. O preço é caro mas vale a pena. As cachoeiras são d+, deliciosas p/ se curtir. Muito legal, adooooorei !!! :)	A entrada estava valor por cabeça quando fui em data. O preço é caro mas gostei. As cachoeiras são demais, deliciosas para se curtir. Muito legal, adorei! Feliz

A Tabela 5 apresenta um exemplo de entrada dessa etapa e sua saída, o texto normalizado. As expressões, termos ou símbolos que estiverem destacados no texto de entrada são normalizados e suas respectivas saídas também aparecem em destaque no texto normalizado. A seguir alguns dados que foram substituídos ou removidos no processo de normalização:

- Remoção de links e e-mails. Ex: remove “www.google.com”
- Remoção de itens duplicados (letras, emoticons e pontuação). Ex: remove a letra “e” repetida de “ameeeei”, resultando em “amei”

- Substituição de datas e horas. Ex: substitui “01/04/2017” pela palavra “data”
- Substituição de valores monetários. Ex: substitui “R$ 300,00” pela palavra “valor”
- Correção ortográfica de algumas palavras mais comuns (ótimo, excelente, bom, incrível, e entre outras). Ex: substitui “encrevel” por “incrível”
- Correção de algumas abreviações (para, com, não, também, demais, e entre outras). Ex: substitui “p/” por “para”
- Substituição de risos. Ex: substitui “rsrsrs” por “sorridente”
- Substituição de emoticons (feliz, tristeza e neutro). Ex: substitui “:)” por “feliz”

A lista acima apresenta apenas alguns tipos de itens. A lista completa, com sua respectiva expressão regular, pode ser encontrada no APÊNDICE C. A maioria das expressões regulares foi criada por Oliveira (2015) para a versão 1 da SentimentALL. Porém, para a versão 2 algumas dessas expressões foram alteradas e, também, novas expressões foram adicionadas. As expressões adicionadas para remover datas com meses por extenso, para tratar algumas abreviações (de com, para, também, etc.), remoção de caracteres etc. As alterações nas expressões foram sutis, como, por exemplo, no grupo de expressões que deveria adicionar espaços antes e depois de delimitadores “[](){}”, duas expressões não estavam funcionando como deveria, pois em uma estava faltando o delimitador “(“ e na outra o “)”. O APÊNDICE C apresenta todas as expressões alteradas, adicionadas e que permaneceram iguais.

4.3.2 Correção Ortográfica

Essa etapa tem a função de corrigir a grafia das palavras. O algoritmo utilizado foi baseado em Norvig (2016), com algumas adequações necessárias para o funcionamento dele na língua portuguesa.

O algoritmo precisa de uma lista de palavras conhecidas e a frequência dessas palavras com base em um *corpus*. Mais adiante isso será explicando com mais detalhes. Em síntese, o algoritmo funciona da seguinte forma: ele recebe a

palavra; se ela estiver na lista de palavras conhecidas, não precisará ser corrigida; caso contrário, diversas transformações na palavra são realizadas para gerar palavras candidatas; a palavra candidata conhecida que tiver a maior frequência será a palavra corrigida; se o algoritmo não gerou nenhuma palavra candidata conhecida, a palavra continua com sua forma original.

A primeira transformação realizada foi adicionada ao algoritmo de Norvig (2016). Ela está relacionada aos erros ortográficos mais comuns, em que algumas letras têm mais chance de serem trocadas por outras semelhantes (foneticamente) ou pela falta de acentuação. Por exemplo, a palavra “cafe”, em que o autor deveria ter substituído “e” por “é”. Com isso, deve-se agrupar letras que têm mais probabilidade de serem trocadas. Os agrupamentos considerados nesse trabalho foram os seguintes.

- cç
- aáâã
- eéê
- ií
- oóôõ
- uúü

Caso uma palavra tenha alguma letra que está em algum dos agrupamentos, cada letra do grupo substitui a letra original, gerando uma palavra candidata. No exemplo anterior, a palavra “cafe” gera as seguintes palavras candidatas (considerando apenas o grupo ‘eéê’): “cafe”, “café” e “cafê”. Quando mais de um grupo é encontrado, são realizadas combinações, como apresentadas a seguir (considerando o grupo “eéê” e “aá”): “cafe”, “café”, “cafê”, “cáfe”, “cáfé” e “cáfê”.

Outros agrupamentos podem ser feitos para aumentar a precisão, por exemplo: ‘mn’, ‘sz’, ‘ie’ e ‘jg’. Porém, a quantidade de palavras candidatas geradas pode crescer exponencialmente quando um novo agrupamento for adicionado, e isso afeta o tempo de execução do algoritmo. Por isso, foram definidos apenas seis agrupamentos.

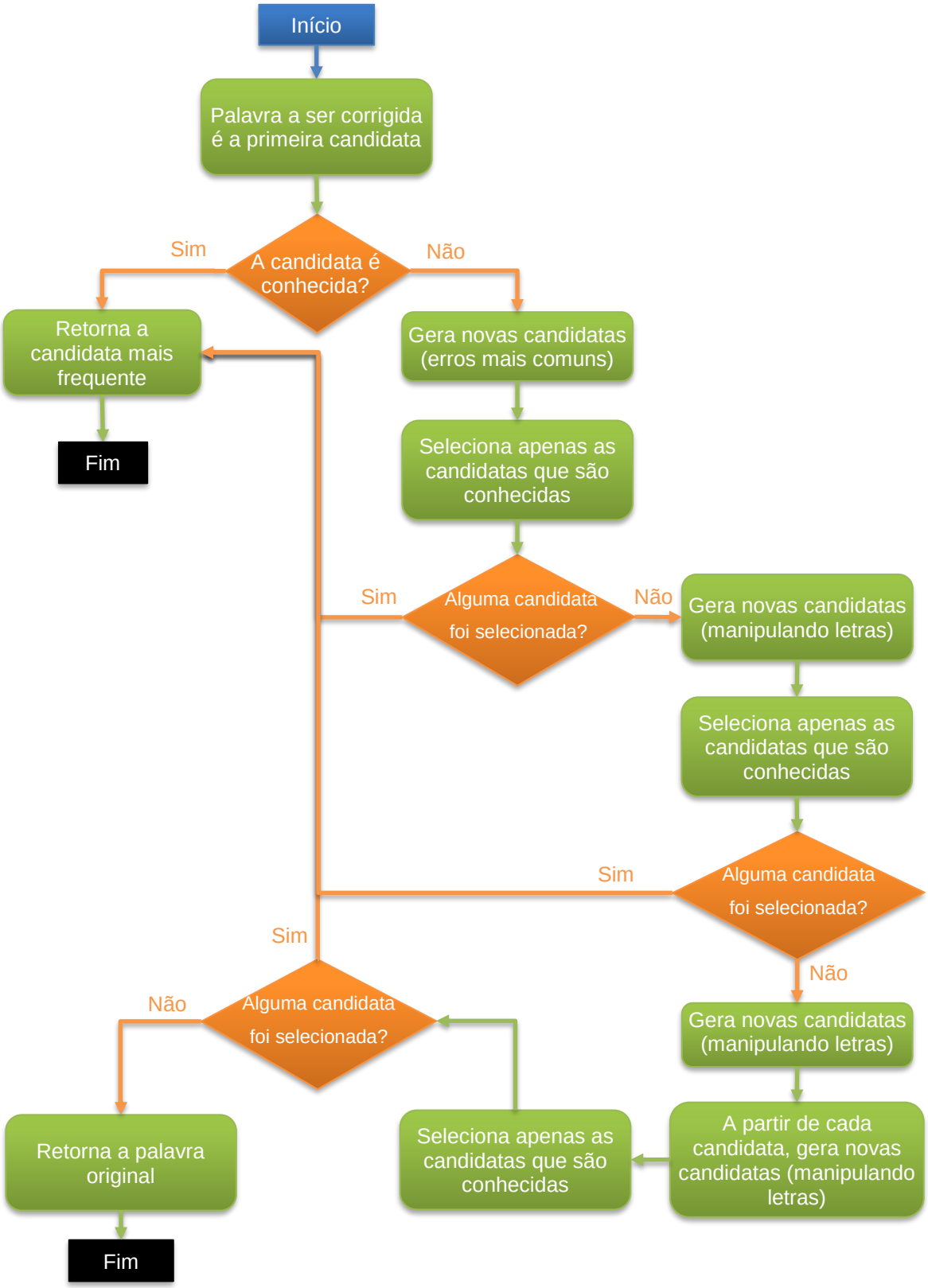
Na segunda transformação são realizadas algumas manipulações nas letras. As manipulações são de eliminação, adição, transposição e substituição. A letras utilizadas na adição e substituição estão no conjunto de letras “aáâãbcçdeéêfghiíjklmnoóôõpqrstuúüvwxyz”. A seguir é apresentado um exemplo para cada uma dessas manipulações.

- Eliminação – Ex.: “cafré”, removendo o “r”, fica “café”
- Adição – Ex.: “fejão”, adicionando o “i”, fica “feijão”
- Transposição – Ex.: “arrzo”, trocando as “z” e “o” de lugar, fica “arroz”
- Substituição – Ex.: “carme”, substituindo “m” por “n”, fica “carne”

Na terceira transformação é realizado o procedimento anterior novamente e, para cada palavra gerada, são realizadas novas manipulações. Ou seja, é realizada a manipulação das palavras manipuladas. Por exemplo, a primeira manipulação da palavra “café” geraria (dentre outras) a palavra “chafé” (Adição). Na segunda manipulação geraria (dentre outras) a palavra “chalé” (Substituição).

Após cada transformação é gerado um conjunto de palavras candidatas. Antes de passar para a próxima transformação, primeiro é verificado se foi gerada uma palavra conhecida. Se foi, não serão necessárias as outras transformações. A seguir o fluxograma do algoritmo de correção ortográfica.

Figura 28: Fluxograma do algoritmo de correção ortográfica



Fonte: Norvig (2016)

Conforme apresentado na

Figura 28, o algoritmo necessita da lista de palavras conhecidas praticamente o tempo todo e a precisão do algoritmo depende da quantidade de palavras que ela tem e que estejam corretas. Norvig (2016) utilizou apenas um *corpus* e, a partir dele, criou uma lista de palavras e calculou suas frequências. Porém, nesse trabalho, além do uso do *corpus*, para aumentar a quantidade de palavras conhecidas, foi utilizada uma lista com quase um milhão de palavras encontradas no site do Project Natura no link a seguir: http://natura.di.uminho.pt/download/sources/Dictionaries/wordlists/LATEST/.

Como o corpus de Norvig (2016) está em inglês, foi necessário criar um novo. Para isso, foram desenvolvidas duas *spiders* para coleta de notícias nos sites de BBC Brasil (http://www.bbc.com/portuguese) e TNH1 (http://www.tnh1.com.br). Foram escolhidos sites de notícias, pois eles têm muitos conteúdos em texto e têm uma menor probabilidade de ter erros ortográficos do que uma rede social, por exemplo. Ao final, foram adicionadas à lista de palavras, novas palavras dos sites de notícias e adicionada a frequência de cada palavra no *corpus*.

Como a conclusão do desenvolvimento da lista de palavras, o algoritmo foi testado para avaliar a sua eficiência. Primeiro foram informadas para o algoritmo cem palavras corretas e depois cem palavras incorretas. Com as palavras corretas, o algoritmo teve 100% de acerto, ou seja, ele não tentou corrigir algo que já estava correto. A seguir, a Tabela 6 apresenta uma amostra das palavras corrigidas corretamente:

Tabela 6: Amostra de palavras corrigidas corretamente

Palavra incorreta	Palavra corrigida
pimentaoes	pimentões
yougurtes	iogurtes
belissimna	belíssima
Utização	utilização
pagagens	bagagens
Polpitica	política

Com as palavras incorretas, o algoritmo teve 88% de acerto, porém em muitas destas palavras observou-se uma complexidade na realização da correção, devido à semelhança apresentada com outras palavras, por exemplo, “fazrem” foi corrigida para “fazem”, mas a palavra correta é “fazerem”. A seguir, a Tabela 7 apresenta uma amostra das palavras corrigidas incorretamente:

Tabela 7: Amostra de palavras corrigidas incorretamente

Palavra incorreta	Palavra corrigida	Palavra correta
frotografica	fotografia	fotográfica
hoetal	total	hotel
tanaho	tanto	tamanho
contrartio	contrario	contrário
utl	uti	útil
aomentos	momentos	aumentos
municipai	municipal	municipais
fazrem	fazem	fazerem
localizicao	localizicao	localização
indlcrivel	indlcrivel	incrível
paãzinho	pauzinho	pãozinho
confortavavel	confortavavel	confortável

Ao final dessa etapa, tem-se os comentários tokenizados em sentenças normalizadas e corrigidas ortograficamente. Na seção a seguir será apresentada a etapa de a PoS Tagging, em que as sentenças serão etiquetadas morfologicamente.

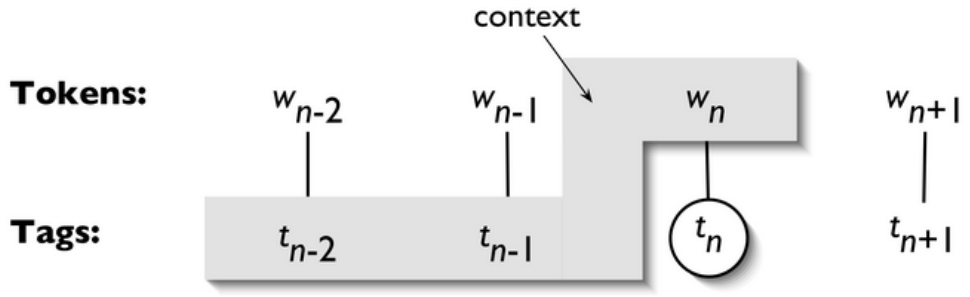
4.3.3 *PoS Tagging*

Para realizar o *PoS tagging* foi necessário utilizar o *corpus* Mac-Morpho. Esse *corpus*, disponibilizado na NLTK, possui mais de um milhão de palavras etiquetadas com 26 tipos de *PoS tags*. O conjunto de *PoS tags* possíveis encontra-se no ANEXO A.

A NLTK possui um módulo que realiza o *PoS Tagging*. Com ele é possível treinar classificadores, chamados de *taggers*. Para isso, foi utilizado o Mac-Morpho

para treinar oito *taggers*. Os *taggers* tem a seguinte nomenclatura: *n-gram tagger*. Onde n representa a quantidade de *tokens* que formam o contexto de um *token* que será etiquetado. O contexto de um *token* é ele próprio mais as n – 1 *tags* de tokens anteriores. A Figura 29 apresenta um exemplo do contexto de um *3-gram tagger*, também chamado de *trigram tagger*.

Figura 29: Contexto do *Tagger*

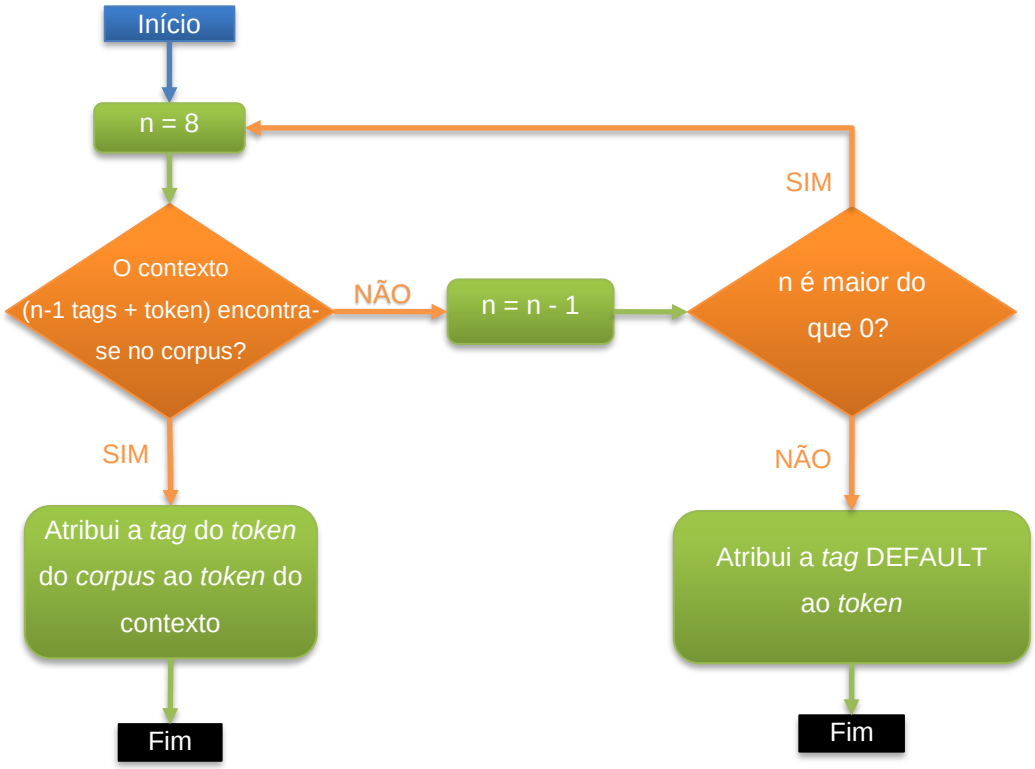


Fonte: Bird, Klein e Loper (2009)

A área sombreada em cinza na Figura 29 representa o contexto do token w_n. Para o *3-gram tagger* determinar a *tag* de w_n, considera-se o token w_n e as duas *tags* (t_{n-2} e t_{n-1}) dos *tokens* anteriores. O *3-gram tagger* busca pelo padrão, caso encontre determina a tag t_n de w_n.

Além dos oitos *taggers* (*8-gram tagger*, *7-gram tagger*, ...) treinados, há também um *tagger* padrão. A classificação de um *token* utilizando esses *taggers* funciona da seguinte forma:

Figura 30: Fluxograma do *PoS Tagging*



Fonte: NLTK (2016)

Conforme apresentado na Figura 30, o *PoS Tagging* funciona basicamente como explicado a seguir. Inicia-se utilizando o *8-gram tagger*, se no *corpus* possuir o mesmo padrão apresentado no contexto do token, ou seja, a mesma sequência de setes *tags* mais o token. Então, a mesma *tag* do *token* do *corpus* é atribuída ao *token* do contexto. Se não possuir o padrão, utiliza-se o 7-gram tagger e o processo se repete. Se nenhum dos *n-gram taggers* conseguir classificar o token, então é usado o *tagger* padrão, etiquetando o *token* como a tag DEFAULT.

O procedimento de *PoS tagging* utilizando o corpus Mac-Morpho possui um problema ao etiquetar as contrações e combinações de preposições (de, a, em e por) com as palavras apresentadas na Tabela 8.

Tabela 8: Tabela de palavras que podem ser contraídas

Tipo	Palavras	Exemplo
Artigo Definido	O, os, a e as	Do, à, pela
Artigo Indefinido	Um, uns, uma e umas	Dum, num
Pronome Pessoal	Ele, eles, ela e elas	Nele, delas

Pronome Demonstrativo (1ª pessoa)	Este, estes, esta, estas e isto	Deste, nisto
Pronome Demonstrativo (2ª pessoa)	Esse, esses, essa, essas e isso	Nesse, nessa
Pronome Demonstrativo (3ª pessoa)	Aquele, aqueles, aquela, aquelas e aquilo	Daquele, naquilo
Pronome Indefinido	Outro, outra, outros e outras	Noutro, doutro
Advérbio	Aqui, aí, ali e além	Daqui, dali

Esse problema se deve ao fato de que no *corpus* não existem essas contrações, elas são escritas na sua forma sem contrações, por exemplo, “da” é escrito como “de a”. Porém, esse problema foi contornado atualizando o Mac-Morpho para suportar as contrações e gerando um novo *corpus*. Para isso, foi feita uma varredura em todo o Mac-Morpho buscando pelas preposições (de, a, em e por), caso a próxima palavra for alguma das palavras da Tabela 8, então é feita a contração. Quando é realizada uma contração, foi feita a união das *pos tags* com um sinal de “+”. Por exemplo, a varredura encontra a palavra “de” com a *pos tag* “PREP” a próxima palavra é “a” com a *pos tag* “ART”. A contração de “de” com “a” é “da” e a nova *pos tag* será “PREP+ART”. Com isso, é gerado um novo *corpus* e foi resolvido o problema das contrações e combinações.

Um outro problema foi identificado, em que muitas palavras foram identificadas com a *tag* DEFAULT, pois não estavam presentes no MAC-MORPHO. Portanto, foi utilizada uma lista de adjetivos e uma de verbos, disponíveis no site Linguateca. Essas listas informam a palavra e uma frequência. As palavras com frequência menor ou igual a três foram removidas, pois não estavam classificadas de forma correta. Quando os *taggers* etiquetavam uma palavra como DEFAULT, verifica-se ela está na lista de adjetivos, se estiver, define a *tag* como “ADJ”. Se não for um adjetivo, verifica-se ela está na lista de verbos, se estiver, define a *tag* como “V”. Se não for um verbo, verifica-se ela é um número usando a expressão regular “\d+([,\.]\d+)?” e etiqueta como “NUM”. Se depois de todas as verificações nenhuma condição for satisfeita, a palavras será etiquetada como substantivo (*tag* N).

A seguir, é apresentada a última etapa do pré-processamento, que utiliza o *PoS tag* para encontrar padrões para identificar possíveis expressões multipalavras.

4.3.4 Identificação de Expressões Multipalavras

Essa etapa é composta de dois passos: encontrar expressões candidatas e filtrar as expressões. O primeiro passo faz a busca nos dados procurando por padrões morfológicas que identifica expressões candidatas. Os padrões utilizados servem para identificar expressões do tipo composto nominal. A Tabela 9 apresenta esses padrões.

Tabela 9: Padrões Morfológicos

Padrão	Exemplo
N N	Nações Unidas
N A	Governo federal
N N N	Supremo Tribunal Federal
N N A	Fundo Monetário Internacional
N A A	Produto interno bruto
N P N	Casa de praia, bolsa de valores

Fonte: Boos, Prestes e Villavicencio (2014)

O algoritmo utilizado para realizar o primeiro passo é baseado no pseudocódigo usado em Christhie e Brito (2015). A mudança está apenas no fato de que em Christhie e Brito (2015) foram consideradas expressões com até quatro *tokens* e nesse trabalho foram consideradas apenas três, pois as contrações (por exemplo, “das” – “de” + “as”) não foram desmembradas. A Figura 31 apresenta o pseudocódigo da busca por expressões candidatas.

Figura 31: Pseudocódigo da pesquisa por expressões candidatas

```
para cada comentario em corpus
    para cada sentenca em comentario.sentencas
        para cada token em sentenca.tokens
            se token.getPosTag() é "n" ou "n-adj"
                tamanho = -1;
                enquanto (j = 3; j > 1 e não encontrou; j--)
                    se j é 2
                        se token.indice() + 1 >= sentenca.size()
                            sai enquanto;
                        //NN - NA
                        se (token + 1).getPosTag() não é "n" nem "adj" nem "n-adj"
                            sai enquanto;
                        encontrou = j;
                    se j é 3
                        se token.indice() + 2 >= sentenca.size()
                            continua enquanto;
                         //NPN
                        se (token + 1).getPosTag() é "prp"
                        e (token + 2).getPosTag() é "n" ou "n-adj"
                            encontrou = j;
                        //NAA
                        senao se    ((token + 1).getPosTag() é "adj" ou "n-adj")
                        e   ((token + 2).getPosTag() é "adj" ou "n-adj"))
                            encontrou = j;
                        //NNN - NNA
                        senao se    ((token + 1).getPosTag() é "n" ou "n-adj")
                        e   ((token + 2).getPosTag() é "n" ou "adj" ou "n-adj"))
                            encontrou = j;
                        senao continua enquanto;
                fim enquanto
                se tamanho > 0
                    FREQUENCIA = CalculaFrequencia(sentenca, token, tamanho);
                    PMI = CalculaPMI(sentenca, token, tamanho);
                    GravaExpressao(sentenca, token, tamanho, PMI, FREQUENCIA);
```

Fonte: Adaptação de Christhie e Brito (2015)

Conforme apresentado na Figura 31, o algoritmo inicia procurando por um substantivo nos comentários, quando encontra, verifica se ele e mais dois tokens a frente formam alguns dos padrões morfológicos. Caso forme, os três tokens passam a compor o conjunto de expressões candidatas. Se não formar, verifica se o próximo *token* é um substantivo ou adjetivo e se for, esses tokens serão uma expressão candidata. Esse processo se repete até que todas expressões candidatas sejam encontradas. Quando uma expressão candidata é encontrada, calcula-se o seu PMI, usando a equação proposta por Cruys (2011), apresentada na seção 2.1.3:

$$SI(x_1, x_2, \dots, x_n) = \log_2 \frac{P(x_1, x_2, \dots, x_n)}{\prod_{i=1}^{n} P(x_i)}$$

Foram encontradas por volta de três milhões de expressões candidatas. Utilizando essas expressões candidatas, inicia-se o segundo passo, que é a de

filtragem das expressões. A filtragem é realizada tendo como base um intervalo de valor do PMI. Para determinar esse intervalo, foram feitos testes com diversos valores de intervalos em um conjunto de cinquenta frases. Nesse conjunto foram destacadas sessenta e quatro expressões multipalavras. Para definir o melhor intervalo foi utilizado a Precision e o Recall. Para ter uma única medida de desempenho foi usado o F-Measure, que faz uma média harmônica com os valores de Precision e Recall. Essa métrica foi utilizada, pois ela tem a característica de que, caso um dos valores esteja muito baixo, a média também será baixa (Sasaki, 2007). Como tanto a *Precision* como o *Recall* são relevantes, isso é uma característica desejável. A seguir são apresentadas as equações de Precision, Recall e F-Measure, respectivamente.

$$Precision = \frac{V_p}{V_p + F_p}$$

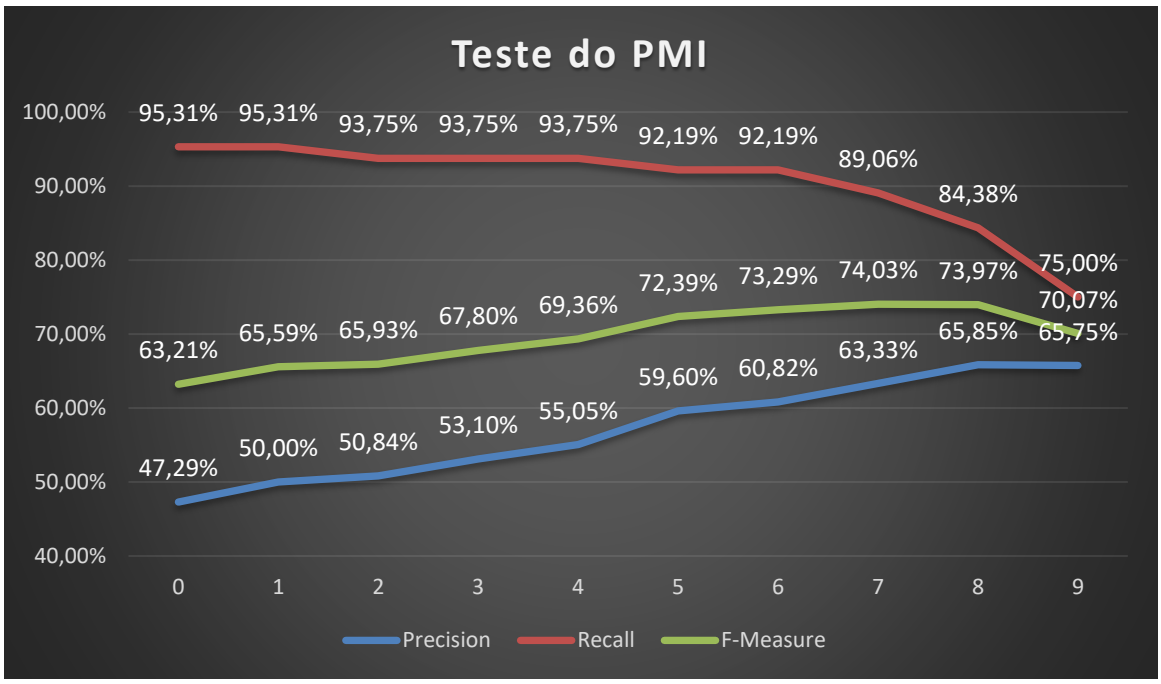
$$Recall = \frac{V_p}{V_p + F_n}$$

$$FMeasure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Onde V_p é a quantidade de expressões reconhecidas que estão corretas; F_p é quantidade de expressões reconhecidas que estão erradas; e F_n é a quantidade de expressões que deveriam ter sido reconhecidas, mas não foram. Essas equações serão explicadas com mais detalhes na seção 4.5 sobre a avaliação da ferramenta.

O teste foi feito variando o x de 0 a 9, com intervalo de 1, da seguinte equação: $x < PMI$. Com isso, são consideradas como expressões multipalavras as expressões que tiverem o PMI maior do que x. A Figura 32 apresenta o gráfico com os valores de Precision, Recall e F-Measure, para cada valor de x.

Figura 32: Resultado do teste do PMI

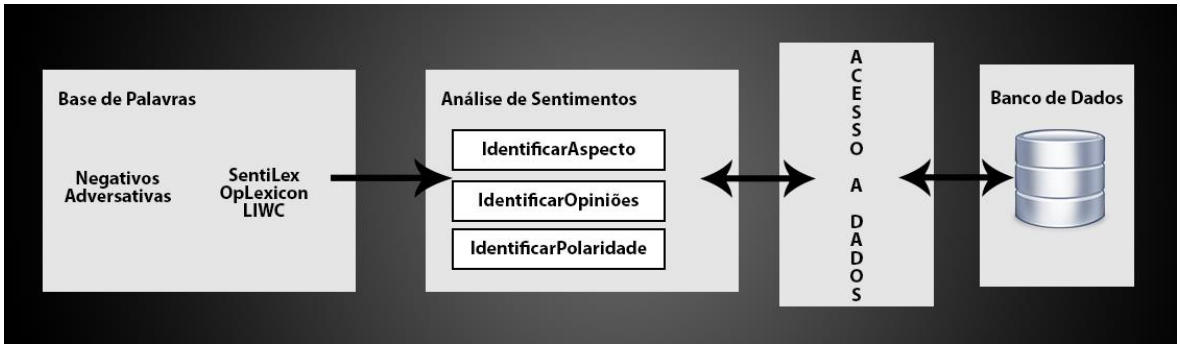


Como pode se observar na Figura 32, o valor de x em que houve a melhor F-Measure é 7. Ressalta-se ainda que, eliminando expressões com pouca frequência (menor do que 25), o resultado melhora. A Precision vai para 73,33%, o Recall para 85,94% e F-Measure, 79,14%.

4.4 ANÁLISE DE SENTIMENTOS

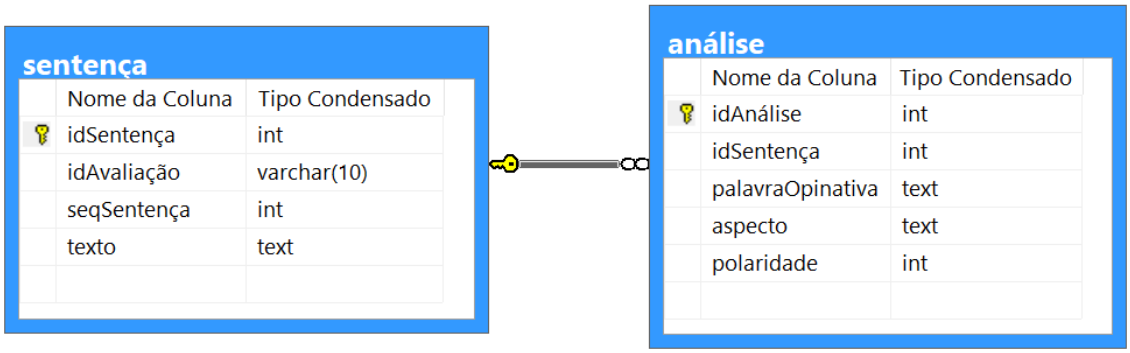
O módulo de análise de sentimentos realiza as seguintes etapas: identificação de opiniões, polaridade da opinião e polaridade dos aspectos. Conforme apresentada na parte da arquitetura relacionada ao módulo de análise de sentimentos (Figura 33).

Figura 33: Análise de Sentimentos - Arquitetura



De acordo com a Figura 33, o módulo de análise de sentimentos tem acesso à uma base de palavras, contendo palavras negativas, adversativas e três léxicos de sentimentos (SentiLex, OpLexicon e LIWC). Esse módulo possui também acesso ao banco de dados, que contém as sentenças que serão analisadas e saída do módulo será armazenada no banco de dados. Para armazenar o resultado dessas etapas foi criada a tabela análise no banco de dados, conforme apresentado no modelo da Figura 34.

Figura 34: Modelo relacional parcial - Análise de Sentimentos



A tabela análise, apresentada na Figura 34, possui uma relação com a tabela sentença, ou seja, a análise realizada é sobre uma sentença de uma avaliação. A análise contém a palavra opinativa, o aspecto e a polaridade.

Para realizar a análise de sentimentos, para cada sentença, foram identificadas as relações de dependência entre as palavras. Depois, utilizando os léxicos de sentimentos, foram identificadas as palavras opinativas e suas polaridades. Por fim, utilizando as relações de dependência, foram identificados os aspectos relacionados às palavras opinativas.

4.4.1 Análise de Dependência Sintática

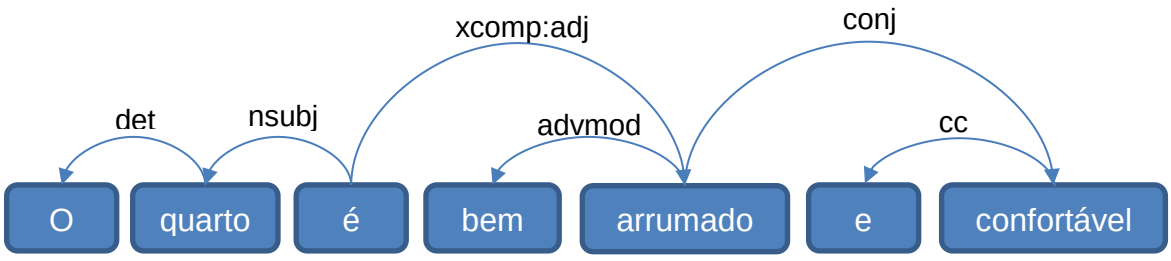
A análise de dependência sintática tem o objetivo de encontrar relações sintáticas entre duas palavras. Essas relações de dependência podem ser, por exemplo: determinante, modificador adverbial, palavras ligadas por conjunção, entre outras. O UD (*Universal Dependencies*) possui para o português algumas dezenas de relações, a Tabela 10 apresentam algumas delas. A tabela completa está no ANEXO B.

Tabela 10: Relação de Dependência do UD

Dependências	Descrição
advmod	Modificador do advérbio
amod	Modificador do Adjetivo
cc	Conjunção coordenativa
conj	Palavras conectadas por conjunção
det	Determinante
nmod	Modificador Nominal
nsubj	Sujeito Nominal
xcomp:adj	Complemento

Utilizando as dependências da Tabela 10 para exemplificar o funcionamento das relações de dependência, tem-se as relações da frase “O quarto é bem arrumado e confortável”, apresentadas na Figura 35. A relações de dependência são representadas pelo grafo de dependência.

Figura 35: Exemplo de relações de dependência - Grafo de Dependência



Na Figura 35, as setas representam as dependências entre os nós (*tokens*) e a origem da seta representa o termo dependente. Por exemplo, “quarto” é dependente de “O”, com isso, pode-se dizer que “O” é o determinante (det) de “quarto”. Com essas associações é possível encontrar outras relações, como: o sujeito (“quarto”) do verbo (“é”), palavras ligadas (“arrumado” e “confortável”) por conjunção (“e”) e o advérbio (“bem”) modificando o adjetivo (“arrumado”).

Para fazer a análise de dependência, foi treinado um modelo utilizando a ferramenta MaltParser. Porém, antes de fazer o treinamento do modelo, foi necessário adequar as *PoS tags* do UD para as *PoS tags* utilizadas nesse trabalho.

Para isso, foi utilizado o módulo de *PoS Tagging* para substituir o *PoS tags* do arquivo de treinamento.

Após a substituição das *PoS tags*, foi realizado o treinamento utilizando os dados de treinamento do UD que contém 9262 sentenças. O treinamento do modelo usando o MaltParser é feito via linha de comando. O comando para treinamento é o seguinte:

java -jar maltparser-1.9.0.jar -c <MODEL> -i <TRAIN> -a <ALG> -m learn

No comando acima, onde tem “<MODEL>” substitui-se pelo nome definido para o modelo. “<TRAIN>” deve ser preenchido com o nome do arquivo de treinamento do UD, com extensão “.conllu”. E “<ALG>”, com o nome do algoritmo para treinamento. Os nomes possíveis são listados na Tabela 11.

Tabela 11: Algoritmos para Análise Sintática do MaltParser

Nome	Algoritmo
nivreeager	Nivre arc-eager
nivrestandard	Nivre arc-standard
covnonproj	Covington non-projective
covproj	Covington projective
stackproj	Stack projective
stackeager	Stack eager
stacklazy	Stack lazy
planar	Planar eager
2planar	2-Planar eager

MaltParser implementa nove algoritmos determinísticos de análise sintática, conforme a Tabela 11. Ao final do treinamento é gerado o arquivo do modelo, com o nome definido em “<MODEL>” e com a extensão “.mco”.

Foi realizado o treinamento e teste (utilizando os dados de teste do UD) de todos os algoritmos de análise sintática implementados pelo MaltParser. No teste é realizado a análise utilizando os dados de teste, gerando um arquivo de saída no formato CoNLL-U. Para isso foi utilizado o seguinte comando para fazer o *parse* (análise sintática) dos dados:

java -jar maltparser-1.9.0.jar -c <MODEL> -i <TEST> -o <OUT> -m parse

No comando acima, “<MODEL>” deve ser substituído pelo nome do arquivo do modelo (sem a extensão). “<TEST>” preenchido pelo nome do arquivo de teste do UD, com extensão “.conllu”. “<OUT>” preenchido pelo nome do arquivo de saída (com extensão “.conllu”).

Depois, realizou-se a comparação entre o arquivo de teste e o de saída, com o objetivo de avaliar o quão bem os algoritmos estavam definindo as relações sintáticas entre os *tokens*. Para isso, foi comparado para cada *token*, se ele tinha relação com o *token* correto e se o tipo da relação estava correto. Nos testes, o algoritmo que obteve o melhor resultado foi o Stack swap-lazy. Com acerto de aproximadamente 84% na relação dos *tokens* e aproximadamente 88% de acerto no tipo de relação. Com isso, ele foi o escolhido para ser utilizado nesse trabalho.

Para fazer o *parse* das sentenças, foi criada a função `parser_malt` que permite a execução do comando de *parse*, fazendo uma chamada de linha de comando para o sistema. Como foi visto anteriormente, para fazer o *parse*, necessita-se de um arquivo de entrada no formato CoNLL-U para realizar a análise. Logo, precisa-se converter as sentenças etiquetadas para esse formato e salvá-las em um arquivo. O NLTK já possui uma função, chamada `taggedsents_to_conll`, para realizar essa conversão. O retorno da função é salvo em um arquivo temporário. É criado também um arquivo temporário que conterá a saída da análise. Com isso é montada uma *string* com o comando de parse (apresentado anteriormente) e é feita a chamada ao sistema para executá-la.

Quando termina a execução da análise é realizada a leitura do arquivo temporário de saída e para cada sentença é criado um grafo de dependência, utilizando a classe `DependencyGraph` fornecida pela NLTK. O grafo de dependência está no formato da estrutura dicionário e seus elementos são chamados de nós, em que a chave é a posição do *token* na sentença e o valor possui as informações sobre o *token* (as mesmas apresentadas no arquivo CoNLL-U). A Figura 36 apresenta um exemplo dessa estrutura usando a frase “O quarto é amplo”, porém foram mantidas, para uma melhor visualização, apenas as informações usadas nesse trabalho.

Figura 36: Saída da análise de dependência

```
{
0:{'address':0,'word':None,'tag':'TOP','head':None,'deps':{'root':[3],'null':[]},'rel':None},
1:{'address':1,'word':'O','tag':'ART','head':2,'deps':{},'rel':'det'},
2:{'address':2,'word':'quarto','tag':'N','head':3,'deps':{'det':[1]},'rel':'nsubj'},
3:{'address':3,'word':'é','tag':'V','head':0,'deps':{'nsubj':[2],'xcomp:adj':[4]},'rel':'root'},
4:{'address':4,'word':'amplo','tag':'ADJ','head':3,'deps':{},'rel':'xcomp:adj'}
}
```

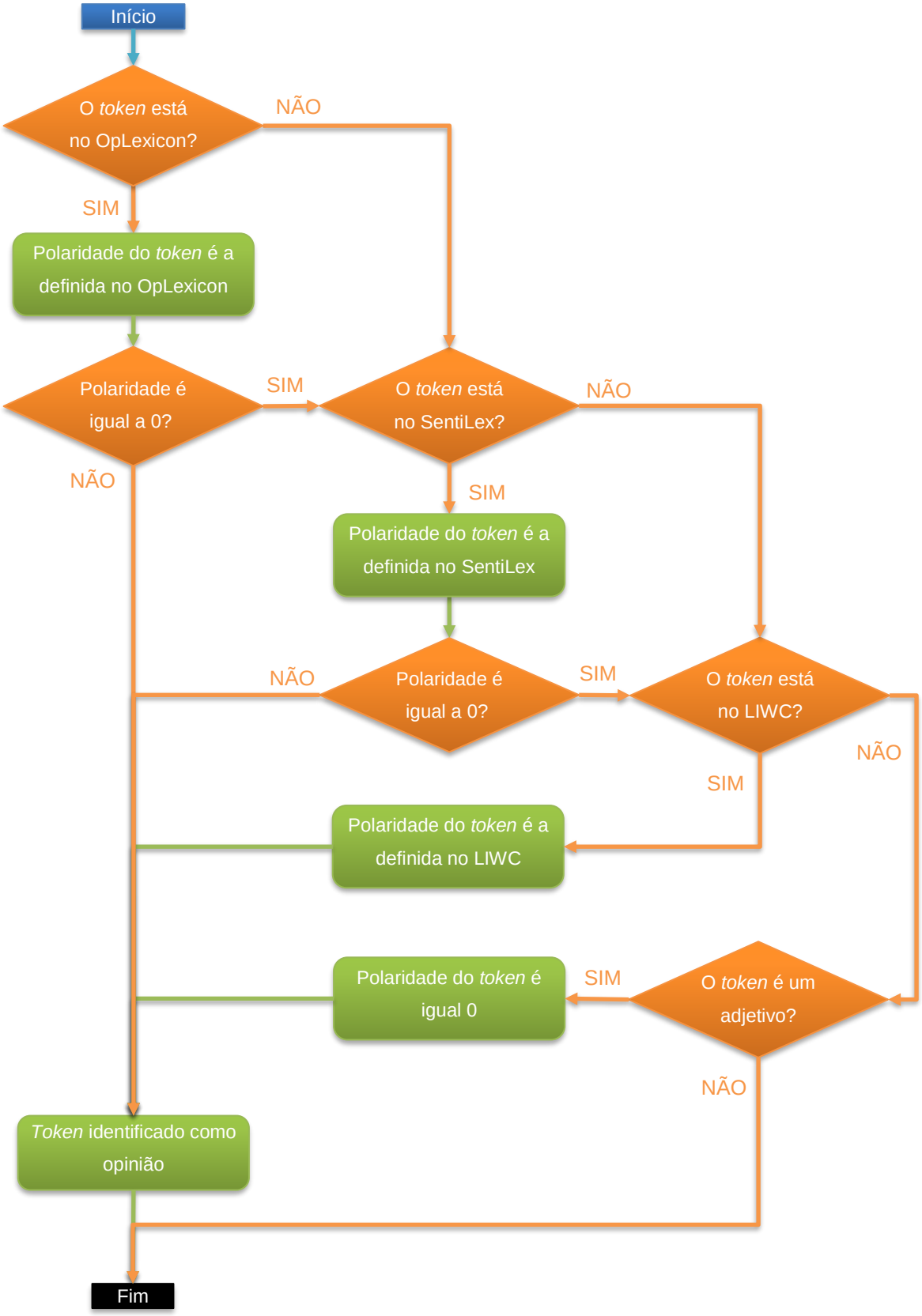
Conforme a Figura 36, “*address*” é posição do *token* na sentença; “*word*” é o *token* (“O”, “quarto”, “é” e “amplo”); “*tag”* é a *PoS tag*; “*head*” é posição do *token* dependido (“quarto” depende de “O”, logo o “*head”* de “O” é a posição de “quarto”); “*deps*” apresenta o tipo de relação e posição dos *tokens* dependentes (“quarto” depende de “O”, logo o “*deps”* de “quarto” possui o tipo de relação e a posição de “O”); e “*rel*” é o tipo relação do dependido (“quarto” é o sujeito de “é”, logo a sua relação é “nsubj”). Com isso, é possível, a partir de um *token*, descobrir a posição do seu dependente ou dependido e navegar entre nós da sentença, para fazer a identificação do aspecto de uma opinião.

4.4.2 Identificação de Opiniões e Polaridade

A identificação de opiniões e polaridade é dividida em três passos: identificar opiniões e polaridade inicial; determinar a polaridade final; e identificar lista de opiniões.

No primeiro passo, o algoritmo percorre os nós resultantes da análise de dependência em busca das opiniões. As opiniões identificadas serão as palavras presentes nos léxicos de sentimentos ou se ela é um adjetivo. A Figura 37 apresenta o fluxograma usado para identificar se um *token* é uma opinião e identifica sua polaridade.

Figura 37: Fluxograma para identificação da polaridade de um token



Conforme fluxograma apresentado na Figura 37, a ordem de verificação do *token* no léxico de sentimentos é: OpLexicon, SentiLex e LIWC. Se o *token* não for encontrado em um léxico, então será feita a verificação no próximo. Se o *token* não foi encontrada em nenhum léxico, verifica-se ele é um adjetivo e, caso seja, define a polaridade como 1 (positiva). Quando um *token* for encontrado em um léxico, a sua polaridade é definida com base no léxico. Se essa polaridade for 0, busca-se nos próximos léxicos. Se mesmo assim a polaridade não foi mudada, a polaridade será 0. Quando uma palavra for identificada como opinião, é criado uma lista contendo alguns dados sobre a opinião e seu aspecto.

- Identificação da opinião (posição da palavra na frase)
- Identificação do aspecto
- Identificação da palavra que a opinião depende (*head*)
- Tipo de dependência (*rel*)
- Tag da opinião
- Polaridade da opinião
- Opinião
- Aspecto

Cada um desses itens é representado em uma lista, pois pode haver mais de uma opinião sobre um aspecto e pode haver mais de um aspecto para uma mesma opinião. No final, a lista contendo os dados sobre a opinião e seu aspecto são adicionadas a uma lista contendo todas as opiniões identificadas na frase, que será usada nos próximos passos. Um exemplo da lista de opiniões usando a frase “A comida é saudável e deliciosa” é apresentado na Tabela 12.

Tabela 12: Exemplo de lista de opinião

ID opinião	4	6
ID aspecto		
Head	3	4
Rel	xcomp:adj	conj
Tag	ADJ	ADJ
Polaridade	1	1
Opinião	saudável	deliciosa
Aspecto		

No segundo passo, para determinar a polaridade final, percorre-se a lista de opiniões e verifica, em cada opinião, se alguma palavra negativa ou adversativa está dependendo dela. A lista de palavras negativas e adversativas (ANEXO C) é a mesma utilizada na primeira versão da SentimentALL, exceto pela adição da palavra “mal”.

O algoritmo identifica duas formas de negação de uma opinião. A primeira é quando uma opinião depende de alguma palavra negativa diretamente, como por exemplo, “não gostei”. Para isso, percorre-se a lista de opiniões e verifica se há alguma opinião dependente de um advérbio, através da dependência “advmod”. Quando encontra, confere se esse advérbio está na lista de palavras negativas. Se estiver, inverte a polaridade da opinião e adiciona o prefixo “não_” nela. A segunda forma ocorre quando um verbo depende diretamente de duas palavras (uma opinião e um advérbio), como, por exemplo, “não é bom”. Para isso, se uma opinião tiver como dependência um verbo e ele tiver uma dependência do tipo “advmod” com um advérbio e se esse advérbio for uma palavra negativa, é invertida a polaridade da opinião e adicionado o prefixo “não_” nela.

As conjunções adversativas são identificadas através do verbo que tem como complemento dois adjetivos, quando um deles possui uma palavra adversativa como dependente, por exemplo, “O quarto é pequeno, porém confortável.”. Para isso, procura-se por alguma opinião (com polaridade 0) que seja dependente de algum verbo com uma dependência do tipo “xcomp:adj”. Verifica se algum desses nós que o verbo possui dependência tem dependência com uma conjunção adversativa. Se encontrar, e se esse nó for uma opinião, multiplica por -1 a polaridade desse nó e atribui à opinião que tinha polaridade 0.

Ao final do segundo passo, todas as opiniões que estiverem com a polaridade neutra passarão a ser positiva. A polaridade foi definida como positiva, pois segundo Hatzivassiloglou e McKeown (1997), as palavras positivas são mais frequentes. Esse fato foi corroborado na primeira versão da SentimentALL, em que 83% das palavras foram detectadas como positivas.

O terceiro passo tem como objetivo unir opiniões ligadas por conjunção e remover palavras irrelevantes que foram identificadas como opinião. Primeiro foi feito a remoção das palavras irrelevantes, que são palavras que possuem relação do tipo “advmod” com um advérbio ou adjetivo. Por exemplo, “A piscina é bem

grande”, as palavras “bem” e “grande” seriam definidas como opiniões, contudo, “bem” é irrelevante para a análise, pois ela é usada apenas para intensificar a palavra “grande”, não mudando a sua polaridade.

Para unir opiniões ligadas por conjunção, busca-se por opiniões que possuem dependência do tipo “conj”. Se encontrar alguma opinião, a mesma é mesclada com as opiniões conjuntadas. Logo, o exemplo da Tabela 12, usando a frase “A comida é saudável e deliciosa”, ficaria da seguinte forma (Tabela 13).

Tabela 13: Exemplo da lista de opiniões com mesclagem

ID opinião	4, 6
ID aspecto	
Head	3, 4
Rel	xcomp:adj, conj
Tag	ADJ, ADJ
Polaridade	1, 1
Opinião	saudável, deliciosa
Aspecto	

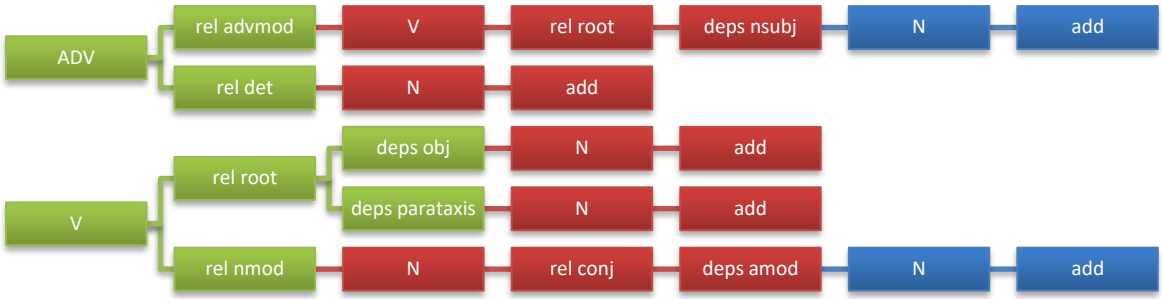
Como é possível perceber nas Tabela 12 e Tabela 13 não há dados sobre os aspectos referenciados pelas opiniões. Isso será apresentado na próxima seção, que tem o objetivo de explicar como foram realizados a identificação dos aspectos de uma opinião e o preenchimento das lacunas na lista de opiniões.

4.4.3 Identificação de Aspectos

O processo de identificação de aspectos é feito através de um percurso no nó da opinião, passando ou não por outros nós até chegar em um substantivo. Então, o que foi feito nesse trabalho foi tentar criar alguns percursos possíveis, utilizando algumas frases da base de dados para identificar as dependências presentes entre os aspectos e as opiniões.

Uma opinião tem três *tags* possíveis: adjetivo (ADJ), advérbio (ADV) e verbo (V). Com isso, foram criadas para cada *tag* um conjunto de caminhos. A Figura 38 apresenta os percursos identificados para os advérbios e os verbos.

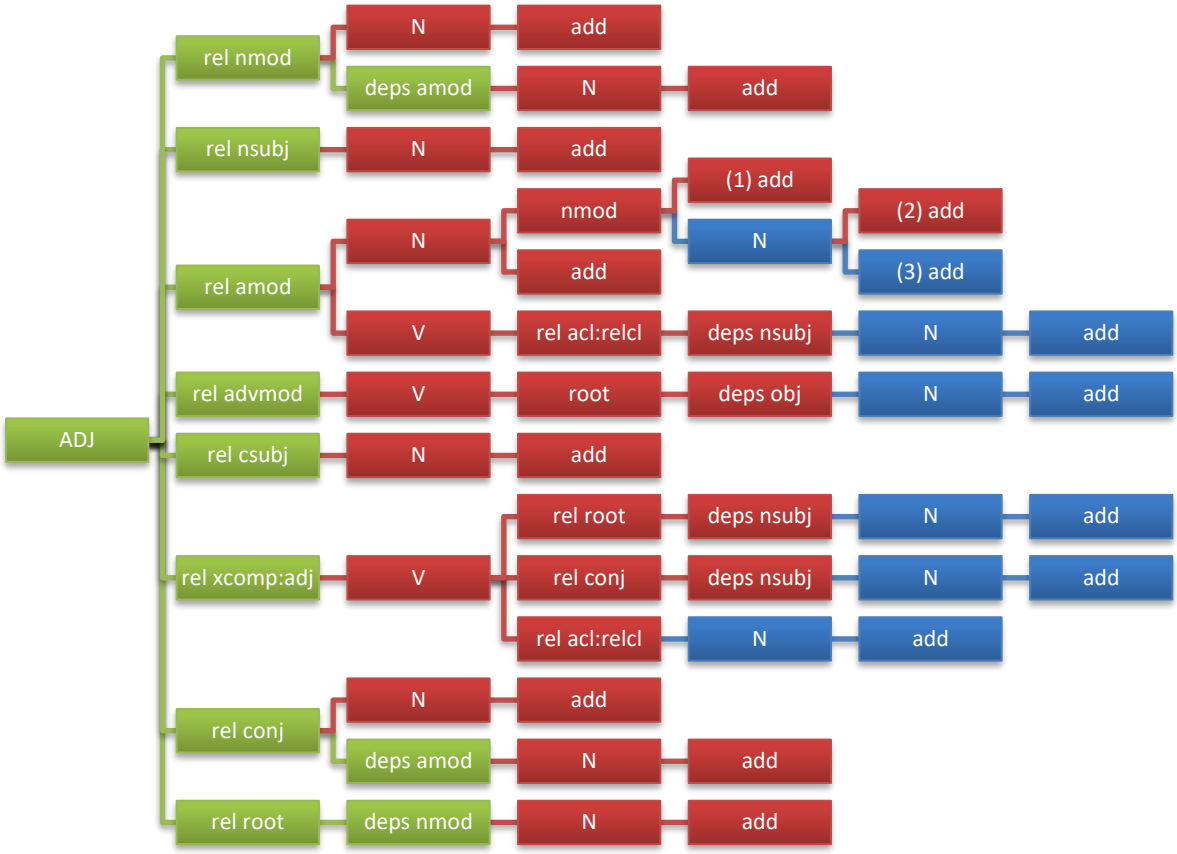
Figura 38: Percurso para identificação de aspecto (Advérbio e Verbo)



De acordo com a Figura 38, os caminhos estão representados da esquerda para a direita. Os retângulos representam os atributos (*tag*, *rel* e *deps*) de um nó e as cores identificam de qual nó. Os atributos *rel* e *deps* retornam à identificação de um nó para um determinado tipo de relação de dependência. Com isso, essa identificação é usada para acessar um novo nó. Quando, no percurso, houver uma mudança de nó, a cor irá mudar. Ao final de cada percurso, há um retângulo com o texto “add”, ele é usado para especificar, através da cor, qual o nó foi identificado como aspecto. Por exemplo, usando o primeiro percurso, se a opinião for um advérbio, verifica se o atributo *rel* é um “advmod”. Se for, acessa o nó e verifica se ele é um verbo, e assim por diante, até chegar em “add” e identificar que o nó de cor azul foi definido como o aspecto da opinião.

A Figura 39 apresenta os percursos identificados para os adjetivos.

Figura 39: Percurso para identificação de aspecto (Adjetivo)



Os percursos apresentados na Figura 39 funcionam da mesma forma que a anterior, porém com algumas exceções, que são os retângulos marcados com (1), (2) e (3). Em (1), antes de confirmar que o nó é um aspecto, primeiro faz uma verificação se o nó está exatamente antes ou após a opinião. Em (2), se tiver uma diferença de maior de quatro *tokens* entre o nó atual (azul) e o nó anterior (vermelho) então o aspecto será o nó anterior, senão (3) será o nó atual.

Quando um aspecto for identificado, então é preenchida a lacuna na lista de opiniões da sua respectiva palavra opinativa. Nesse processo descrito anteriormente, não são identificados aspectos ligados por conjunção. Logo, de forma análoga ao acontecido nas opiniões, para unir aspectos ligados por conjunção, busca-se por aspectos que possuem dependência do tipo “conj”. Se encontrar algum aspecto, mescla-se ele com os substantivos conjuntados.

4.5 AVALIAÇÃO

Esta seção objetiva comparar o desempenho das duas versões da ferramenta. Em cada versão da SentimentALL mediu-se a quantidade de acerto em relação a uma análise de sentimentos feita manualmente. Com isso, foi possível saber qual versão teve maior quantidade de acertos.

Em Leal, Oliveira e Brito (2017) foi realizada a avaliação da SentimentALL v1. A avaliação consistia de cinco etapas: análise manual, recuperação dos resultados da análise manual, organização dos resultados, criação da matriz de confusão para cada conjunto de informações e aplicação das técnicas de Precision e Recall para análise de desempenho. Tomando como base esse trabalho, foi então realizada a avaliação da SentimentALL v2

A etapa de análise manual foi realizada em cem comentários. A análise consiste em identificar os aspectos e sua polaridade em cada um dos cem comentários. Os comentários foram categorizados em simples e complexos, cada um com cinquenta. Os comentários simples são aqueles que possuem uma estrutura sintática mais comum na língua portuguesa, por exemplo, “O atendimento é excelente, porém o preço da comida é caro”. Os comentários complexos são aqueles em que há elementos implícitos ou com duplo sentidos, por exemplo, “Muito boa mesmo, considero a melhor da cidade, amei”. Os comentários utilizados nesse trabalho foram os mesmos utilizados na avaliação original (de Leal, Oliveira e Brito (2017)). Contudo, foi feita uma revisão na análise manual, por exemplo, foram identificadas algumas expressões multipalavras, como “vista da cidade”, “lagoa de Jacumã” e “camarão à parmegiana”. Também foi considerado que um aspecto pode ter mais de uma opinião, como por exemplo, “O quarto é pequeno, mas é confortável”.

Na recuperação dos resultados, a avaliação original fazia a relação do resultado da análise manual com a da SentimentALL v1 através do identificador do comentário na base de dados. Mas, como a base de dados é diferente, foi criado um programa que utiliza o módulo de análise de sentimentos da SentimentALL v2. Os comentários simples e complexos foram copiados para dois arquivos de texto (um para cada tipo de comentário). Então o programa faz a leitura dos arquivos e apresenta, para cada comentário, os aspectos, as opiniões e as polaridades.

A etapa de organização do resultado foi feita em duas planilhas, uma para comentários simples e outra para complexos. As planilhas utilizadas foram construídas em Leal, Oliveira e Brito (2017), entretanto foi adicionada a análise da segunda versão. A planilha foi dividida em três seções, uma para cada análise, como apresentado na Figura 40.

Figura 40: Disposição das análises na planilha

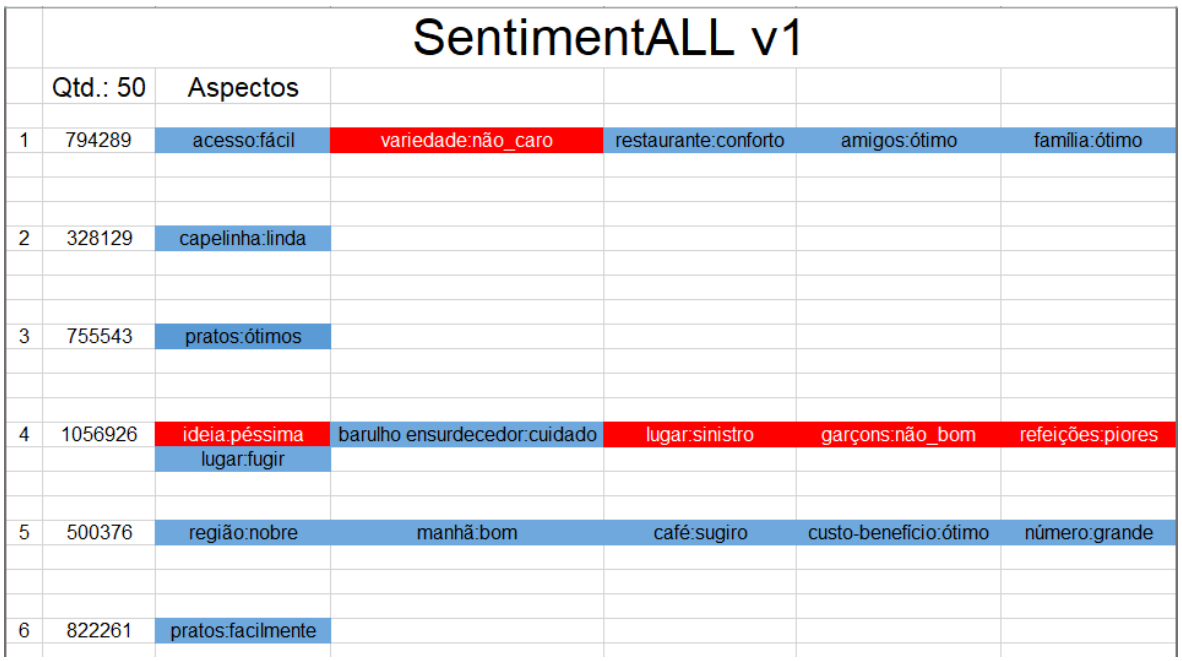
Ampliando as seções da Figura 40, tem-se as amostras apresentadas nas Figura 41 (Manual), Figura 42 (SentimentALL v1) e Figura 43 (SentimentALL v2).

Figura 41: Amostra da análise manual

Manual

	Qtd.: 50	Aspectos				
1	794289	atendimento:primeira restaurante:conforto	local:refrigerado	local:privilegiado	acesso:fácil	preço por pessoa:não_caro
2	328129	parque:arborizado	capelinha:linda			
3	755543	atendente:simpática	pratos:ótimos	acarajé:espetacular		
4	1056926	ideia:péssima lugar:sinistro	ambiente:feíssimo toalha:puída	ambiente:triste roupa:manchada	ambiente:não_cuidado roatsbeef:péssimo	barulho:ensurdecedor refeições:piores
5	500376	hotel:novo café da manhã:bom	região:nobre custo-benefício:ótimo	bairro batel:badalado check in:demorado	quartos:pequenos número de hóspedes:grande	quartos:confortáveis
6	822261	cardápio:organizado	pratos:servidos			

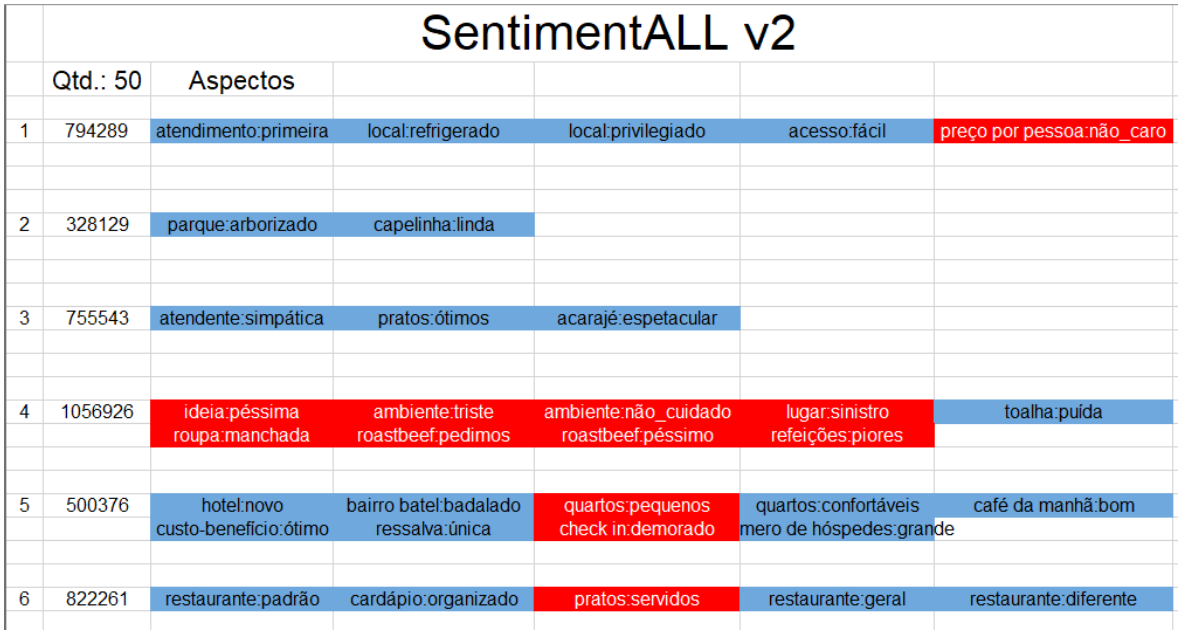
Figura 42: Amostra da análise da SentimentALL v1



SentimentALL v1

	Qtd.: 50	Aspectos				
1	794289	acesso:fácil	variedade:não_caro	restaurante:conforto	amigos:ótimo	família:ótimo
2	328129	capelinha:linda				
3	755543	pratos:ótimos				
4	1056926	ideia:péssima lugar:fugir	barulho ensurdecedor:cuidado	lugar:sinistro	garçons:não_bom	refeições:piores
5	500376	região:nobre	manhã:bom	café:sugiro	custo-benefício:ótimo	número:grande
6	822261	pratos:facilmente				

Figura 43: Amostra da análise da SentimentALL v2



SentimentALL v2

	Qtd.: 50	Aspectos				
1	794289	atendimento:primeira	local:refrigerado	local:privilegiado	acesso:fácil	preço por pessoa:não_caro
2	328129	parque:arborizado	capelinha:linda			
3	755543	atendente:simpática	pratos:ótimos	acarajé:espetacular		
4	1056926	ideia:péssima roupa:manchada	ambiente:triste roastbeef:pedimos	ambiente:não_cuidado roastbeef:péssimo	lugar:sinistro refeições:piores	toalha:puída
5	500376	hotel:novo custo-benefício:ótimo	bairro batel:badalado ressalva:única	quartos:pequenos check in:demorado	quartos:confortáveis mero de hóspedes:grande	café da manhã:bom
6	822261	restaurante:padrão	cardápio:organizado	pratos:servidos	restaurante:geral	restaurante:diferente

Como apresentado nas Figura 41, Figura 42 e Figura 43, a segunda coluna informa a identificação do comentário na base de dados da SentimentALL v1. Nas colunas à direita são informados em cada célula preenchida, um aspecto e a sua opinião, separados pelo caractere “:”. As células foram coloridas em azul e vermelho, representando, respectivamente, a polaridade positiva e negativa do aspecto.

Com essas figuras (Figura 41, Figura 42 e Figura 43), pode-se verificar os erros e acertos de cada ferramenta, por exemplo, o primeiro comentário (identificação 794289). Nesse comentário, a análise manual identificou seis aspectos (atendimento, local, local, acesso, preço por pessoa e restaurante) com polaridade positiva. A análise da SentimentALL v1 identificou quatro aspectos positivos (acesso, restaurante, amigos e família) e um negativo (variedade), sendo que apenas dois estão presentes em ambos: acesso e restaurante. Tanto as polaridades (ambas positivas) como as opiniões (fácil e conforto) desses aspectos foram as mesmas. A análise da SentimentALL v2 identificou quatro aspectos positivos (atendimento, local, local e acesso) e um negativo (preço por pessoa). Todos os aspectos identificados na SentimentALL v2 estão presentes na análise manual e as opiniões (primeira, refrigerado, privilegiado, fácil e não caro) também foram iguais. Porém, a polaridade do aspecto “preço por pessoa” foi identificado de forma errônea, “não-caro” foi identificado como negativo, daí a ocorrência do erro na polarização.

A quarta etapa faz uso também de planilhas para a construção de uma matriz de confusão para cada comentário. A matriz de confusão tem a seguinte estrutura (Tabela 14):

Tabela 14: Matriz de Confusão

	Relevante	Não Relevante
Recuperado	V_p	F_P
Não Recuperado	F_n	V_n

A Tabela 14 é preenchida com a quantidade de itens (aspectos) que se encaixam com o cruzamento entre as linhas (Recuperado e Não Recuperado) e colunas (Relevante e Não Relevante). Os itens relevantes são aqueles que a ferramenta deveria ter tido como resposta. Os itens recuperados são aqueles que a ferramenta teve como resposta. Fazendo o cruzamento, tem-se os seguintes quantitativos:

- V_p (Verdadeiros-Positivos), a quantidade de itens relevantes recuperados
- F_P (Falso-Positivos), a quantidade de itens não relevantes recuperados

- F_n (Falsos-Negativos), a quantidade de itens relevantes não recuperados
- V_n (Verdadeiros-Negativos), a quantidade de itens não relevantes não recuperados

Foram usadas duas planilhas, uma para cada tipo de comentário (simples e complexo). Cada planilha contém a análise quantitativa e qualitativa das duas ferramentas. A análise quantitativa tem a finalidade de comparar o quão correto foi a identificação dos aspectos. Na análise qualitativa, além dos aspectos, compara também a opinião e a polaridade. Essas planilhas foram divididas em quatro seções: Análise Quantitativa da SentimentALL v1, Análise Qualitativa da SentimentALL v1, Análise Quantitativa da SentimentALL v2 e Análise Qualitativa da SentimentALL v2. A Figura 44 apresenta uma amostra de uma das seções (Análise Qualitativa da SentimentALL v2).

Figura 44: Amostra da análise qualitativa

Análise Qualitativa - SentimentALL v2

ID	Relevante	Não-Relevante		
Recuperado	Vp	Fp	Precision	Vp / (Vp + Fp)
Não-Recuperado	Fn	Vn	Recall	Vp / (Vp + Fn)
794289	Relevante	Não-Relevante		
Recuperado	4	1	Precision	80,00%
Não-Recuperado	2		Recall	66,67%
328129	Relevante	Não-Relevante		
Recuperado	2	0	Precision	100,00%
Não-Recuperado	0		Recall	100,00%
755543	Relevante	Não-Relevante		
Recuperado	3	0	Precision	100,00%
Não-Recuperado	0		Recall	100,00%
1056926	Relevante	Não-Relevante		
Recuperado	7	2	Precision	77,78%
Não-Recuperado	3		Recall	70,00%
500376	Relevante	Não-Relevante		
Recuperado	8	1	Precision	88,89%
Não-Recuperado	1		Recall	88,89%

Como pode ser observado na Figura 44, há uma matriz de confusão para cada comentário analisado, preenchidas conforme a Tabela 14. Porém, a célula resultante da intersecção de Não Relevante e Não Recuperada está vazia, pois essa informação não é possível obter, porque, para isso seria necessário informar

na análise manual o que não poderia ser classificado como aspecto. Ao lado de cada matriz de confusão são apresentados os valores Precision e Recall. Esses valores são calculados na próxima etapa.

Na quinta etapa é feito o cálculo dos valores de *Precision* e *Recall*. A equação de *Precision* e *Recall* são definidas da seguinte forma:

$$Precision = \frac{V_p}{V_p + F_p}$$

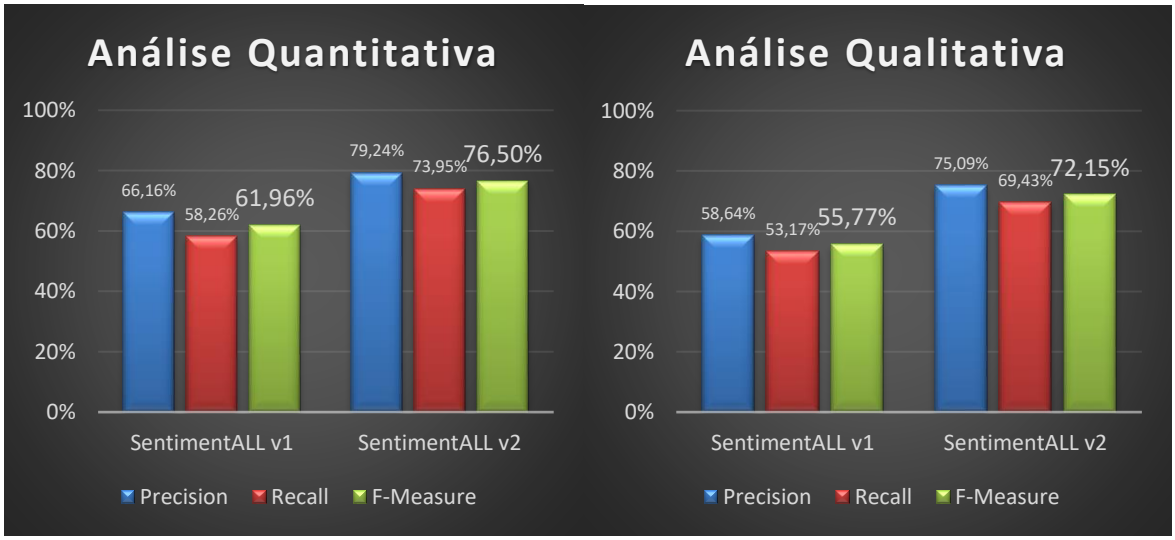
$$Recall = \frac{V_p}{V_p + F_n}$$

Precision é usado para saber o quão relevante são os itens recuperados e *Recall* para saber a porcentagem de itens relevantes foram recuperados. Usando esses valores é possível calcular o desempenho geral da ferramenta através do F-Measure, que possui a seguinte equação:

$$FMeasure = \frac{2 * Precision * Recall}{Precision + Recall}$$

Onde *Precision* e *Recall* representam, respectivamente, a média dos valores de *Precision* e *Recall* de cada comentário. A Figura 45 apresenta o resultado a análise quantitativa e qualitativa para os comentários simples.

Figura 45: Análise de comentários simples

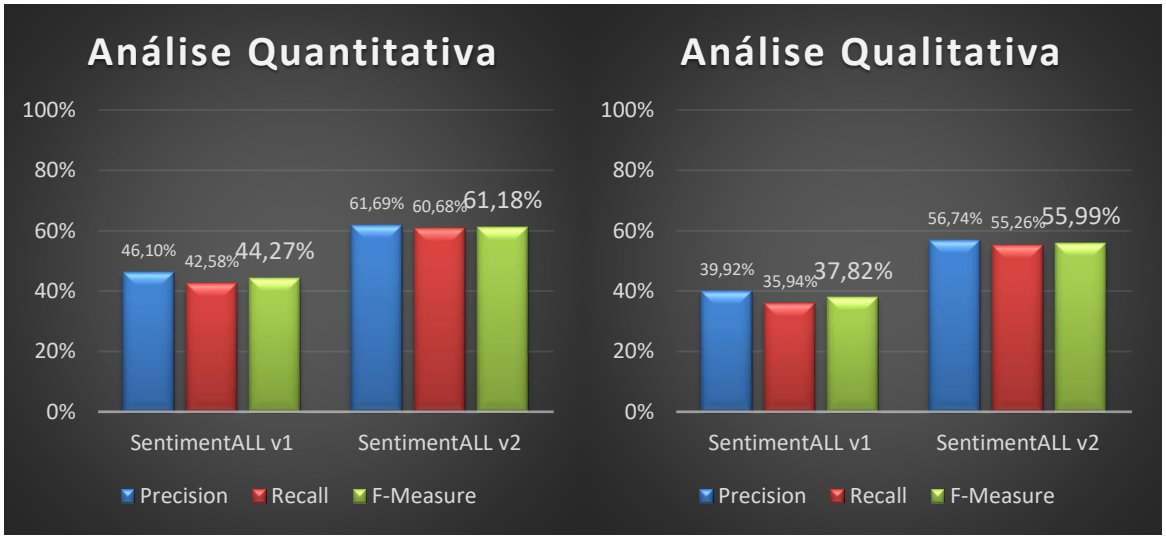


Como pode ser observado na Figura 45, houve uma melhora no desempenho da ferramenta, tanto quantitativamente quanto qualitativamente. Na análise quantitativa o F-Measure passou de 61,96% para 76,50%, tendo uma diferença de 14,54%. Na análise qualitativa teve um aumento um pouco maior, com

uma diferença de 16,38%, em que o F-Measure obtido para a SentimentALL v1 foi de 55,77% e para a SentimentALL v2 de 72,15%

A Figura 46 apresenta o resultado da análise quantitativa e qualitativa para os comentários complexos.

Figura 46: Análise de comentários complexos



Nota-se na Figura 46, que da mesma forma que nos comentários simples, houve um ganho no desempenho da SentimentALL v2 nos dois tipos de análise (quantitativa e qualitativa). Os resultados da análise quantitativa foram de 44,27% para a SentimentALL v1 e 61,18% para a SentimentALL v2, com diferença de 16,91%. Na análise qualitativa, houve a maior diferença dentre todas as análises, com 18,17%. A SentimentALL v1 obteve 37,82% e a SentimentALL v2, 55,99%.

Essa seção apresentou os resultados obtidos no desenvolvimento da segunda versão da SentimentALL. Foram apresentados a visão geral da ferramenta, a coleta dos dados, o pré-processamento, a análise de sentimentos e a avaliação da ferramenta. A coleta de dados incluiu a apresentação do processo de desenvolvimento das *spiders*, a interface gráfica do módulo de coleta e o processo de transformação e carga dos dados. No pré-processamento foram apresentadas as etapas de normalização, correção ortográfica, *PoS Tagging* e identificação de expressões multipalavras. Na análise de sentimentos foram apresentados o treinamento e teste do modelo para análise de dependência, a identificação de opiniões e a identificação de aspectos.

A seção a seguir aborda as considerações finais deste trabalho e os possíveis trabalhos futuros.

5 CONSIDERAÇÔES FINAIS

Esse trabalho teve como objetivo desenvolver a segunda versão da ferramenta SentimentALL, que realiza análise de sentimentos em comentários escritas na língua portuguesa. Para isso o trabalho foi dividido em coleta, pré-processamento, análise e por último a comparação dessa versão com a de Oliveira (2015).

Na coleta, o Scrapy se mostrou uma boa alternativa para criação de *spiders*. Contudo, o módulo desenvolvido funciona apenas no contexto do TripAdvisor. Logo, como trabalhos futuros poderia ser realizada algumas melhorias nesse sentido, através da possibilidade de se desenvolver *spiders* utilizando interface gráfica.

Na etapa de pré-processamento, a NLTK foi de grande importância. O módulo de tokenização de sentenças e palavras estão bastante precisos para a língua portuguesa e foram utilizados em quase todos os módulos. O módulo de *PoS tagging* do NLTK se mostrou capaz de realizar a etiquetagem morfológica nos comentários, porém ocorreu um problema na classificação das palavras com contração que foi corrigido atualizando o Mac-Morpho. Houve um problema também, em palavras que não estão no *corpus* Mac-Morpho, mas foi contornado utilizando uma lista de adjetivos, uma lista de verbos e uma expressão regular para a identificação de números. Na correção ortográfica, o resultado do teste foi satisfatório, pois o algoritmo corrigiu 88% das palavras incorretas de forma correta, porém, uma análise do contexto em que a palavra está inserida poderá melhorar os resultados do algoritmo.

Na análise de sentimentos houve um problema no *corpus* do UD que continha *PoS tags* diferentes das utilizadas no Mac-Morpho. Contudo esse problema foi contornado substituindo as *PoS tags* do *corpus* do UD pelas mesmas do Mac-Morpho. O MaltParser implementa nove algoritmos, que foram todos testados utilizando os parâmetros padrão. Porém, para cada algoritmo existem diversos parâmetros que podem ou não alterar o modelo treinado. Por questão de tempo, não foi possível ser testado todos os parâmetros de cada algoritmo para encontrar o que teria melhor resultado. Portanto sugere-se como trabalhos futuros que uma análise seja efetuada nesse sentido.

Na avaliação das ferramentas, foram realizadas análises quantitativas e qualitativas em comentários simples e complexos. Com essas análises, pode-se concluir que houve um aumento no desempenho da ferramenta SentimentALL, em que o resultado mais significativo usando o F-Measure, foi na análise qualitativa dos comentários complexos, com diferença de 18,17%. A SentimentALL v1 obteve 37,82% e a SentimentALL v2, 55,99%.

A seguir, um comparativo entre as duas ferramentas.

- **Coleta de Dados:** A versão 1 usou para a extração de dados a ferramenta Import.io. Na versão 2 foi desenvolvido um módulo para coleta de dados, usando a biblioteca Scrapy. Os trabalhos dos alunos do CEULP/ULBRA, Pedro Henrique Gomes Camargo e Kevin Martins Araújo contribuíram para o desenvolvimento.
- **Normalização:** Boa parte das expressões regulares foram desenvolvidas na primeira versão. Na segunda versão algumas expressões foram alteradas e outras adicionadas.
- **Corretor Ortográfico:** Presente apenas na segunda versão.
- ***PoS Tag*:** A etiquetagem morfológica foi feita pelo CoGrOO na SentimentALL v1. Na SentimentALL v2 essa etapa foi realizada com o auxílio do módulo de *PoS tag* da biblioteca NLTK. Essa etapa foi desenvolvida baseado no trabalho do aluno do CEULP/ULBRA, Matheus Rodrigues Leal.
- **Expressões Multipalavras:** Os algoritmos usados nas duas versões são semelhantes, porém, como a quantidade de dados coletados foi maior na segunda versão, logo foram encontradas o dobro de expressões candidatas.
- **Identificação de opiniões e polaridade:** Na primeira versão, essa etapa foi realizada usando os léxicos de sentimentos LIWC e SentiLex-PT. Além desses, foi adicionado OpLexicon na segunda versão.
- **Identificação de aspectos:** Na versão 1, os aspectos são identificados a partir da proximidade com opiniões. Na segunda versão, os aspectos são identificados através das relações de

dependência com opiniões. Para o desenvolvimento das relações de dependência foi usado a *corpora* do UD para treinar um modelo no MaltParser. Ao contrário da primeira versão, essa não considera aspectos com opiniões fora da sentença.

Os módulos de pré-processamento e análise de sentimentos foram implementados com a intenção de serem livres de contexto e que sua utilização seja relativamente simples. Para cada módulo há uma entrada e uma saída, sendo que para os módulos é irrelevante a origem dos dados ou o que será feito com a saída. Com isso, pode se criar programas que carregam os dados de arquivos de texto, banco de dados, e entre outros, contanto que os dados de entrada estejam no formato requerido pelo módulo, não há problema. O mesmo acontece com a saída, que pode ser salva em um arquivo de texto, banco de dados, e entre outros. Todos módulos são independentes, portanto algumas etapas podem ser ignoradas, por exemplo, a normalização ou a correção ortográfica. Contudo, nesse trabalho todas as etapas foram utilizadas e os dados tiveram como origem e destino o banco de dados.

Para que a ferramenta trabalhe melhor em diversos contextos, é necessário que sejam desenvolvidos, em trabalhos futuros, léxicos de sentimentos para cada contexto. Para isso, podem-se utilizar os métodos apresentados em Hatzivassiloglou e McKeown (1997) ou Qiu et al (2009, 2011). Porém, deve-se considerar uma etapa de identificação do contexto do texto, para que a ferramenta utilize o léxico respectivo.

REFERÊNCIAS

ALBA, F. Javier. **Basic Sentiment Analysis with Python.** 2012. Disponível em: <http://fjavieralba.com/basic-sentiment-analysis-with-python.html>. Acesso em: 17 nov. 2016.

ALLEN, James. **Natural Language Understanding.** 2. ed. S.l: Pearson, 1995. 654 p.

BALAGE FILHO, Pedro P.; PARDO, Thiago A. S.; ALUÍSIO, Sandra M. **An Evaluation of the Brazilian Portuguese LIWC Dictionary for Sentiment Analysis.** In: PROCEEDINGS OF THE 9TH BRAZILIAN SYMPOSIUM IN INFORMATION AND HUMAN LANGUAGE TECHNOLOGY - STIL, 2013, Fortaleza. p. 215 - 219.

BIG DATA BUSINESS. **Os grandes (e impressionantes) números de Big Data.** 2016. Disponível em: <http://www.bigdatabusiness.com.br/os-grandes-e-impressioantes-numeros-de-big-data/>. Acesso em: 08 nov. 2016.

BIRD, Steven; KLEIN, Ewan; LOPER, Edward. **Natural Language Processing with Python**. [s.l.]: O'reilly, 2009. 504 p.

BISHOP, Christopher M.. **Pattern Recognition and Machine Learning.** New York, Usa: Springer, 2007. 738 p.

BOOS, Rodrigo Augusto Scheller; PRESTES, Kassius Vargas; VILLAVICENCIO, Aline. **Identification of Multiword Expressions in the brWaC**. In: Language Resources and Evaluation Conference (LREC), 9., 2014, Reykjavik (iceland). Disponível em: <http://www.lrec-conf.org/proceedings/lrec2014/pdf/518_Paper.pdf>. Acesso em: 19 dez. 2016.

BRITO, P. F.; OLIVEIRA, W. C. C.; SOUZA, J. G.; SILVA, E.M. **SentimentALL**. Fábrica de Software: CEULP/ULBRA, 2015.

BROMBERG, Andy. **Second Try:** Sentiment Analysis in Python. 2013. Disponível em: <http://andybromberg.com/sentiment-analysis-python/>. Acesso em: 17 nov. 2016.

CHRISTHIE, William; BRITO, Parcilene Fernandes. **Utilização do Pointwise Mutual Information na Identificação de Expressões Multipalavras.** In: XVII ENCOINFO – CONGRESSO DE COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO, 17., 2015, Palmas. p. 25 - 34.

CHURCH, Kenneth Ward; HANKS, Patrick. **Word Association Norms, Mutual Information, and Lexicography.** Computational Linguistics. Cambridge, Ma, EUA, p. 22-29. mar. 1990.

COPPIN, Ben. **Inteligência Artificial.** Rio de Janeiro: Ltc, 2013. 636 p.

CRUYS, Tim van de. **Two multivariate generalizations of pointwise mutual information.** In: WORKSHOP ON DISTRIBUTIONAL SEMANTICS AND COMPOSITIONALITY DISCO, 11, 2011, Stroudsburg, Pa, EUA. Proceedings. Stroudsburg, Pa, EUA: Association for Computational Linguistics, 2011. p. 16 - 20.

DALE, Rober. **Classical Approaches to Natural Language Processing**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 1. p. 3-7.

DOMINGUES, Miriam Lúcia Campos Serra. **ABORDAGEM PARA O DESENVOLVIMENTO DE UM ETIQUETADOR DE ALTA ACURÁCIA PARA O PORTUGUÊS DO BRASIL.** 2011. 137 f. Tese (Doutorado) - Curso de Programa de Pós-graduação em Engenharia Elétrica, Instituto Tecnológico, Universidade Federal do Pará, Belém, 2011.

FACELI, Katti et al. **Inteligência Artificial:** Uma Abordagem de Aprendizado de Máquina. Rio de Janeiro: Ltc, 2011. 378 p.

FERNEDA, Edberto. **Recuperação de Informação:** Análise sobre a contribuição da Ciência da Computação para a Ciência da Informação. 2003. 147 f. Tese (Doutorado) - Curso de Ciências da Comunicação, Universidade de São Paulo, São Paulo, 2003.

GOEBEL, Michael; GRUENWALD, Le. **A SURVEY OF DATA MINING AND KNOWLEDGE DISCOVERY SOFTWARE TOOLS.** Sigkdd Explorations, [s.l], v. 1, n. 1, p.20-33, jun. 1999. Semestral. Disponível em: <http://www.kdd.org/exploration_files/survey.pdf>. Acesso em: 17 dez. 2016.

GÜNGÖR, Tunga. **Part-of-Speech Tagging**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 10. p. 205-235.

HAHN, Sangyun; OSTENDORF, Mari. **A Comparison of Discriminative EM-Based Semi-Supervised Learning algorithms on Agreement/Disagreement classification.** In: NEURAL INFORMATION PROCESSING SYSTEMS, 2., 2008, Canada. 2008. 7 p.

HATZIVASSILOGLOU, Vasileios; MCKEOWN, Kathleen R. **Predicting the semantic orientation of adjectives**. In: PROCEEDINGS OF THE EIGHTH CONFERENCE ON EUROPEAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 97., 1997, Madrid, Spain. p. 174 - 181.

HOBBS, Jerry R.; RILOFF, Ellen. **Information Extraction**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing**.** 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 10. p. 524-545.

HUSSEIN, Doaa Mohey El-din Mohamed. **A survey on sentiment analysis challenges**. Journal Of King Saud University: Engineering Sciences. Cairo, Egito, 9 p. abr. 2016.

JURAFSKY, Daniel; MARTIN, James H. **Speech and Language Processing:** An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Englewood Cliffs, New Jersey: Prentice Hall, 2000. 934 p.

KALAIVANI, P.; SHUNMUGANATHAN, K. L. **SENTIMENT CLASSIFICATION OF MOVIE REVIEWS BY SUPERVISED MACHINE LEARNING APPROACHES**. Indian Journal Of Computer Science And Engineering. Indian, p. 285-292. set. 2013.

KOERICH, Alessandro L. **Aprendizagem de Máquina:** Aprendizagem de Árvores de Decisão. 2005. Disponível em: <http://www.ppgia.pucpr.br/~alekoe/AM/2005/3-ArvoresDecisao-ApreMaq.pdf>. Acesso em: 25 nov. 2016.

LEAL, Matheus Rodrigues; OLIVEIRA, Willian Christie C.; BRITO, Parcilene Fernandes. **AVALIAÇÃO DE DESEMPENHO DE UMA FERRAMENTA DE ANÁLISE DE SENTIMENTOS BASEADA EM ASPECTOS.** In: CONGRESSO DE COMPUTAÇÃO E TECNOLOGIAS DA INFORMAÇÃO, 19., 2017, Palmas, TO.

LIU, Bing. **Sentiment Analysis.** New York, USA: Cambridge University Press, 2015. 769 p.

LIU, Bing. **Sentiment Analysis and Opinion Mining.** [s.l.]: Morgan & Claypool Publishers, 2012. 168 p.

LIU, Bing. **Sentiment Analysis and Subjectivity**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 26. p. 627-666.

LJUNGLÖF, Peter; WIRÉN, Mats. **Syntactic Parsing**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing. 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 4. p. 59-91.

LORENA, Ana Carolina; CARVALHO, André C. P. L. F. de. **Uma Introdução às Support Vector Machines**. Rita - Revista de Informática Teórica Aplicada**,** Porto Alegre, v. 14, p.43-67, 2007. Semestral.

NLTK. **NLTK 3.0 documentation.** Disponível em: <http://www.nltk.org>. Acesso em: 21 set. 2016.

NORVIG, Peter. **How to Write a Spelling Corrector.** 2016. Disponível em: <http://norvig.com/spell-correct.html>. Acesso em: 26 abr. 2017.

MENEZES, Paulo Blauth. **Linguagens Formais e Autômatos.** 6 ed. Porto Alegre: Bookman, 2011. 256 p.

MITCHELL, Tom M. **Machine Learning.** [s.l.]: Mcgraw-hill, 1997. 421 p.

OLIVEIRA, William Christhie Caproni de. **SENTIMENTALL:** Ferramenta para análise de sentimentos em português.2015. 86 f. TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas, Tocantins, 2015. Disponível em: <https://ulbra-to.br/bibliotecadigital/publico/home/documento/151>. Acesso em: 20 mar. 2016.

PALMER, David D. **Text Preprocessing**. In: INDURKHYA, Nitin; DAMERAU, Fred J. (Ed.). Handbook of Natural Language Processing**.** 2. ed. Boca Raton, Fl: Chapman and Hall/crc, 2010. Cap. 2. p. 9-30.

QIU, Guang et al. **Expanding Domain Sentiment Lexicon through Double Propagation.** In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, 21., 2009, [s.l.]. Proceedings**.** 2009. p. 1199 - 1024.

QIU, Guang et al. **Opinion Word Expansion and Target Extraction through Double Propagation**. Association For Computational Linguistics. [s.l], p. 9-27. 2011.

REIS, Julio C. S. et al. **Uma abordagem Multilíngue para Análise de Sentimentos.** In Proceedings of the Brazilian Workshop on Social Network Analysis and Mining (BraSNAM), Recife, Pe, 21 Julho 2015. 12. Disponível em: <http://homepages.dcc.ufmg.br/~fabricio/download/brasnam15-multi.pdf>. Acesso em: 21 Setembro 2016

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial.** 3. ed. Rio de Janeiro: Elsevier, 2013. 1016 p. Tradução de: Regina Célia Simille.

SAG, Ivan A. et al. **Multiword Expressions**: A Pain in the Neck for NLP. In: CICLING INTERNATIONAL CONFERENCE ON INTELLIGENT TEXT PROCESSING AND COMPUTATIONAL LINGUISTICS, 3., 2002, Mexico-city. Proceeding. London, Uk: Springer-verlag, 2002. p. 1 - 15.

SASAKI, Yutaka. **The Truth of the F-measure.** 2007. Disponível em: <http://www.cs.odu.edu/~mukka/cs795sum10dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf>. Acesso em: 15 jun. 2017.

SILVA, Mário J.; CARVALHO, Paula; SARMENTO, Luís. **Building a Sentiment Lexicon for Social Judgement Mining.** In Lecture Notes in Computer Science (LNCS) / Lecture Notes in Artificial Intelligence (LNAI), International Conference on Computational Processing of Portuguese (PROPOR), 17-20 April, 2012, Coimbra.

SILVA, Tércio de Morais Sampaio. **EXTRAÇÃO DE INFORMAÇÃO PARA BUSCA SEMÂNTICA NA WEB BASEADA EM ONTOLOGIAS.** 2003. 93 f. Dissertação (Mestrado) - Curso de Pós-graduação em Engenharia Elétrica, Universidade Federal de Santa Catarina, Florianópolis, 2003.

SOUZA, Marlo; VIEIRA, Renata; BUSETTI, Débora; CHISHMAN, Rove; Alves, Isa Mara. **Construction of a Portuguese Opinion Lexicon from multiple resources**. 8th Brazilian Symposium in Information and Human Language Technology, 2012

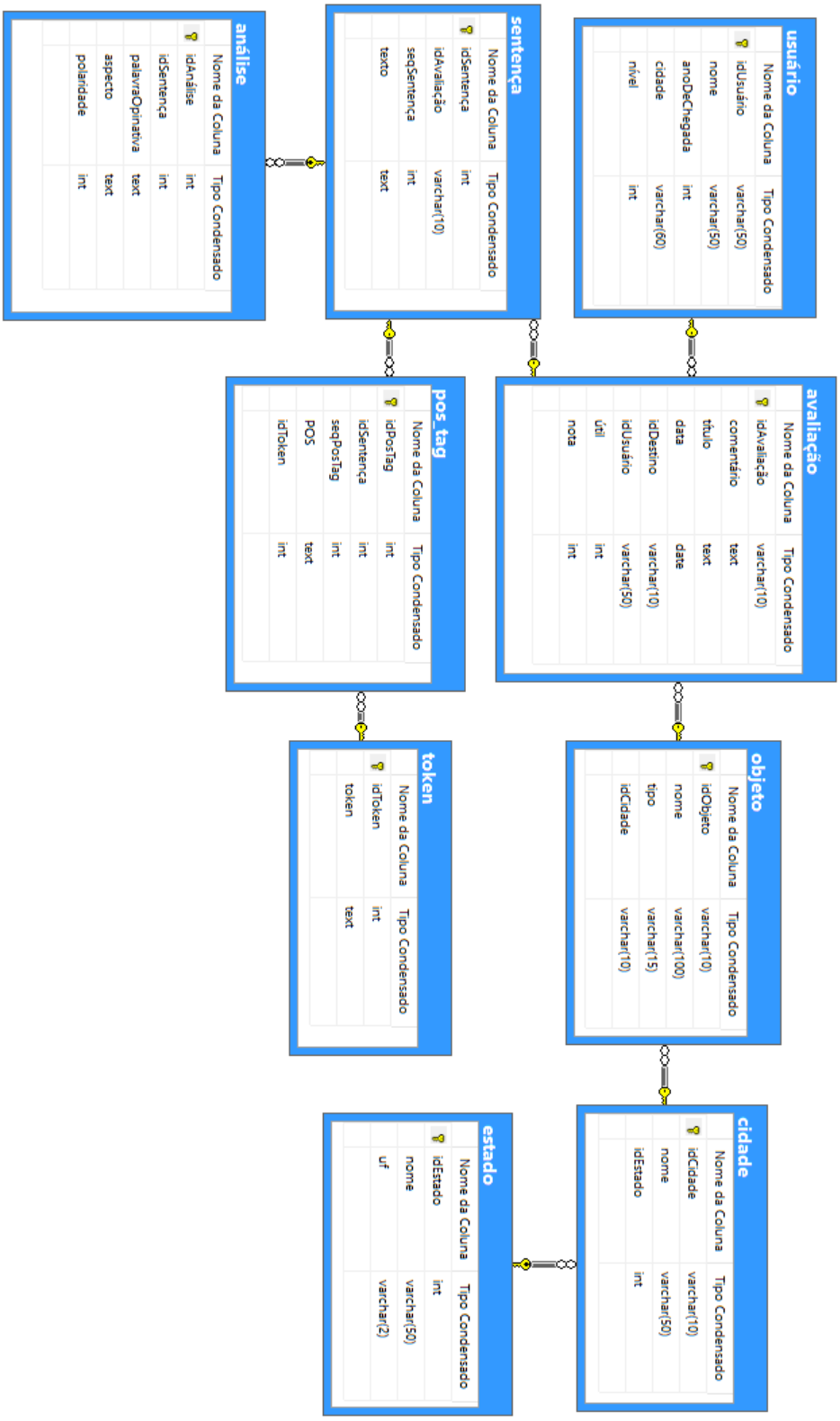
SU, Fangzhong; MARKERT, Katja. **From Words to Senses:** A Case Study of Subjectivity Recognition. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL LINGUISTICS, 22., 2008, Manchester. p. 825 - 832.

VAPNIK, Vladimir. **The Nature of Statistical Learning Theory.** New: Springer-verlag, 1995. 188 p.

APÊNDICES

APÊNDICE A – Modelagem completa do banco de dados

Figura 47: Modelo Relacional



APÊNDICE B – Os cem destinos brasileiros mais avaliados

Cidade	Estado	Quantidade
São Paulo	SP	877486
Rio de Janeiro	RJ	668520
Curitiba	PR	279913
Gramado	RS	265627
Brasília	DF	253343
Fortaleza	CE	227794
Belo Horizonte	MG	212288
Natal	RN	192082
Salvador	BA	182790
Foz do Iguaçu	PR	172774
Florianópolis	SC	164388
Porto Alegre	RS	162285
Maceió	AL	154571
Recife	PE	148112
Campos do Jordão	SP	119670
João Pessoa	PB	94483
Porto de Galinhas	PE	86134
Armação dos Búzios	RJ	84519
Campinas	SP	82208
Manaus	AM	80923
Canela	RS	74131
Aracaju	SE	72253
Porto Seguro	BA	71887
Belém	PA	70707
Goiânia	GO	68713
Balneário Camboriú	SC	63558
Jericoacoara	CE	61675
Vitória	ES	60750
Petrópolis	RJ	56083

Santos	SP	51597
Ribeirão Preto	SP	50856
Paraty	RJ	49464
Niterói	RJ	49346
Bonito	MS	48436
Fernando de Noronha	PE	48342
Bento Gonçalves	RS	46845
São Luís	MA	45672
Ilhabela	SP	41898
Arraial d'Ajuda	BA	38548
Ouro Preto	MG	38443
Ubatuba	SP	38011
Maragogi	AL	35761
Praia da Pipa	RN	35009
Caldas Novas	GO	33481
Monte Verde	MG	33083
Tiradentes	MG	33039
Londrina	PR	32779
Campo Grande	MS	32760
São José dos Campos	SP	32662
Poços de Caldas	MG	32494
Cuiabá	MT	31871
Joinville	SC	31462
Blumenau	SC	30446
Morro de São Paulo	BA	30161
Cabo Frio	RJ	29982
Praia do Forte	BA	29847
Guarujá	SP	28914
Guarulhos	SP	28423
Juiz de Fora	MG	26434
Vila Velha	ES	25786
Itacaré	BA	25031

Arraial do Cabo	RJ	24958
Santo André	SP	24265
Sorocaba	SP	23743
Olímpia	SP	23094
Teresópolis	RJ	22453
Uberlândia	MG	22278
Pirenópolis	GO	22140
São Sebastião	SP	21513
Jundiaí	SP	21347
Aquiraz	CE	21124
Bombinhas	SC	20678
Teresina	PI	20448
Penha	SC	20378
Maringá	PR	20353
São José do Rio Preto	SP	20132
Caxias Do Sul	RS	19743
Trancoso	BA	19549
Penedo	RJ	19326
São Bernardo do Campo	SP	18751
Piracicaba	SP	17432
Olinda	PE	16932
Ilhéus	BA	16850
Canos Quebrada	CE	16497
Ilha Grande	RJ	16484
Alto Paraíso de Goiás Restaurantes	GO	16363
Angra Dos Reis	RJ	16342
Tamandaré	PE	15927
Guarapari	ES	15374
Atibaia	SP	15278
Rio Quente	GO	14750
Barueri	SP	14448
Nova Petrópolis	RS	13864

São Lourenço	MG	13573
Sã Roque	SP	13571
Lençóis	BA	13011
Barreirinhas	MA	12988
Bauru	SP	12460
Taubaté	SP	12439
Palmas	TO	12427

APÊNDICE C – Expressões regulares para normalização

As expressões regulares que estiverem com o texto da cor verde, são as expressões que foram adicionadas nessa nova versão da SentimentALL. As expressões que estiverem laranjadas são as expressões alteradas e as pretas são as que permaneceram iguais.

Expressão	Substituição
Remove link e e-mail. EX: http://www.tripadvisor.com.br	
(^|[(\[{\"“])(https?://|ww)[0-9a-z.-]+\.[a-z.]{2,6}[/\w.-]*/?\??[^]*([.!?,;)\]\}]*|$)	.
(^|[(\[{\"“])[A-Za-z0-9._-]+[A-Za-z0-9]@[A-Za-z0-9._-]+[.][A-Za-z]{2,4}([.!?,;)\]\}]*|$)	.
Remove datas. EX: 01/abril/2017	
(^|[(\[{\"“])(\d{1,2}([\\/.-]))?\d{1,2}([\\/.-])\d{2,4}([- .!?,;)\]\}\"”'_:]|$)	$1data$4
(^|[(\[{\"“])(\d{1,2}([\\/.-]))?(jan(eiro)?|fev(ereiro)?|mar(ço)?|abr(il)?|mai(o)?|jun(ho)?|jul(ho)?|ago(sto)?|set(embro)?|out(ubro)?|nov(embro)?|dez(embro)?)([\\/.-])\d{2,4}([- .!?,;)\]\}\"”'_:]|$)	$1data$18
Remove horas. Ex: 10h15min	
(^|[(\[{\"“])[0-9]{1,2}:?h[0-9]{1,2}(m|(min))?([- .!?,;)\]\}\"”'_:]|$)	$1hora$4
Remove valores. Ex: US$345,99	
(^|[(\[{\"“])[a-zA-Z]{0,2}\$ *\d+([.,]\d*)*([- .!?,;)/\]\}\"”'_:]|$)	$1valor$3
Remove emoticons duplicados	
(^|[(\[{\"“])+((feliz)|(tristeza)|(neutro))([.!?,;)\]\}])?(*\1?\2\6?)+	$1$2$6
Remove espaço antes de pontuações	
\s+([,.;!?ºª:])	$1
Remove espaços repetidos	
\s{2,}	
Adiciona espaços antes e depois de aspas	
(^'| ')([^])	$1 $2
([^])(')([.,;?!\")\]\}]|$)	$1 $2$3
Remove aspas repetidas	
('')	
Remove pontuações repetidas	
([^.!?,;:])([.!?,;:])([.!?;,:]+)	$1$2
Remove aspas erradas	

[ˋˊ]	
[“”]	“
Adiciona espaço depois e antes de parêntesis, chaves e colchetes	
([^])([(\[{])	$1 $2
([(\[{])([^])	$1 $2
([)\]}])([^ .!?,;)\]\}”"])	$1 $2
([^])([)\]}])	$1 $2
Adiciona espaço depois de pontuação	
([!?;:\\/ºª]+)([^])	$1 $2
([.,])([^ 0-9])	$1 $2
([^ 0-9A-ZÀ-Ü]|[A-ZÀ-Ü]{2,})([/]+)([^])	$1 $2 $3
([A-ZÀ-Ü])([.,])(\d)	$1$2 $3
Adiciona espaço antes de “\” e “/”	
([\\/])	$1
Remove símbolos especiais (¨ˋˊ~*)	
[¨ˋˊ~*]	
Adiciona espaço antes e depois de símbolos especiais (+)	
([+])	$1
Remove símbolos especiais (_)	
[_]	
Acerta “+-“, “+/-“, “+ou-“ e “+ ou –“ em “mais ou menos”	
\+(\\|/|(?ou ?))?\-	mais ou menos
Acerta “+” em “mais”	
(\+)	mais
Acerta “NÃO”, “Não”, “ñ”, “NÃO”, “Não” e “nao” em “não”	
(^|[(\[{\"“])(((NÃ|NA|Nã|Na|na)([Oo]))|ñ)([- .!?,;)\]\}\"”"_:]|$)	não$6
Acerta “N” e “n” em “não”	
(^|[(\[{\"“])(n|N)([^0-9])	não
Remove letras repetidas menos cc, ff, pp, rr, ss e as vogais	
([!-/<=>bdghj-nqtv-à-õ])\1+	$1
Remove letras repetidas vogais (volta para primeira ocorrência)	
([aeiou])\1+	$1
Remove letras repetidas considerando rr, ss, cc e pp	

(([rscp])\2)\2+	$1
Substituir risos. Ex: rsrsrs	
(^|[(\[{\"“])(([rs]{3,}|[k]{2,}|[ha]{4,}|[he]{4,}|[hua]{3,})+ *)+([- .!?,;)\]\}\"”'_:]|$)	$1sorridente$4
Substituir notas ruins, nota zero, etc.	
(^|[(\[{\"“])nota([:])?(é|foi)? ?(([01234](,\d+)?)|zero|m[ií]nima|um|dois|tr[eê]s|quatro)([- .!?,;)\]\}\"”'_:]|$)	$1ruim$7
Substituir notas boas, nota dez, mil, 10, etc.	
(^|[(\[{\"“])nota[:]?(é|foi)? ?((([56789]|1\.?0+)(\.\d{3}|,\d)*)|dez|mil|m[aá]xima|cem|cinco|seis|sete|oito|nove)([- .!?,;)\]\}\"”'_:]|$)	$1ótimo$7
Substituir recomend(ad)o	
(^|[(\[{\"“])r+-?e+-?c+-?o+-?m+-?e+-?n+-?d+-?(a+-?d+-?)?o([- .!?,;)\]\}\"”'_:]|$)	$1gostei$3
Corrige variações de ótimo e ótimoo	
(^|[(\[{\"“])[aeiouà-ü]*-?t+-?i+-?m+-?([oa])+-?(s)*([- .!?,;)\]\}\"”'_:]|$)	$1ótim$2$3$4
Corrige variações do excelente	
(^|[(\[{\"“])e+-?x+-?c+-?e+-?l+-?e+-?n+-?t+-?e+-?(s)*([- .!?,;)\]\}\"”'_:]|$)	$1excelente$2$3
Corrige variações do fantástico	
(^|[(\[{\"“])f+-?a+-?n+-?t+-?[aeiouà-ü]*-?s+-?t+-?i+-?c+-?([oa])+-?(s)*([- .!?,;)\]\}\"”'_:]|$)	$1fantástic$2$3$4
Corrigir ma-ra-vi-lho-so	
ma-ra-vi-lho-s([oa])(s)*	maravilhos$1$2
(^|[(\[{\"“])m+-?a+-?r+-?a+-?v+-?i+-?l+-?h+-?o+-?s+-?([oa])+-?(s)*([- .!?,;)\]\}\"”'_:]|$)	$1maravilhos$2$3$4
Corrige variações do bom	
(^|[(\[{\"“])b+-?o+-?[mn]+-?([- .!?,;)\]\}\"”'_:]|$)	$1bom$2
Corrige variações do agradável	
(^|[(\[{\"“])a+-?g+-?r+-?a+-?d+-?[aeiouà-ü]*v+-?e+-?(l|is)+-?([- .!?,;)\]\}\"”'_:]|$)	$1agradáve$2$3
Corrige variações do indispensável	
(^|[(\[{\"“])i+-?n+-?d+-?i+-?s+-?p+-?e+-?n+-?s+-?[aeiouà-ü]*v+-?e+-?(l|is)+-?([- .!?,;)\]\}\"”'_:]|$)	$1indispensáve$2$3
Corrige variações do incrível	

(^|[(\[{\"“])[aeiouà-ü]*-?n+-?c+-?r+-?[aeiouà-ü]*v+-?e+-?(l|is)+-?([-.!?,;)\]\}\"”"_:]|$)	$1incríve$2$3
Corrige variações do impecável(mente)	
(^|[(\[{\"“])i+-?m+-?p+-?e+-?c+-?[aeiouà-ü]*-?v+-?e+-?(l|is)+-?(mente)?([- .!?,;)\]\}\"”"_:]|$)	$impecáve$2$4
Variações do superlativo (tentativa de redução que funciona para a maioria das vezes	
[ií]+-?(s-?)+[ií]+-?m+-?([oa])+-?(s)*([- .!?,;)\]\}”"_:]|$)	$2$3$4
Corrige variações de péssimo	
(^|[(\[{\"“])p+-?[aeiouà-ü]*-?(s-?)+i+-?m+-?([oa])+-?(s)*([- .!?,;)\]\}\"”"_:]|$)	$1péssim$3$4$5
Corrige variações do ridículo	
(^|[(\[{\"“])r+-?i+-?d+-?[aeiouà-ü]*-?c+-?u+-?l+-?([oa])+-?(s)*([-.!?,;)\]\}\"”"_:]|$)	$1ridícul$2$3$4
Corrige variações do horrível	
(^|[(\[{\"“])h?-?o+-?(r-?)+[aeiouà-ü]*-?v+-?e+-?(l|is)+-?([- .!?,;)\]\}\"”"_:]|$)	$1horríve$3$4
Corrige variações de terrível(mente)	
(^|[(\[{\"“])t+-?e+-?(r-?)+[aeiouà-ü]*-?v+-?e+-?(l|is)+-?(mente)?([-.!?,;)\]\}\"”"_:]|$)	$1terríve$3$5
Corrige variações de decepção	
(^|[(\[{\"“])d+-?e+-?c?s?-?e+-?p+-?[scç]+-?[aeiouà-ü]*-?o+-?([-.!?,;)\]\}\"”"_:]|$)	$1decepção$2
Corrige variações do (des)ilusão	
(^|[(\[{\"“])(d+-?e+-?s+-?)?i+-?l+-?u+-?s+-?[aeiouà-ü]*-?o+-?([-.!?,;)\]\}\"”"_:]|$)	$1ilusão$3
Corrige variações de chateação	
(^|[(\[{\"“])c+-?h+-?a+-?t+-?e+-?a+-?[cç]+-?[aeiouà-ü]*-?o+-?([-.!?,;)\]\}\"”"_:]|$)	$1chateação$2
Corrige são	
(^|[(\[{\"“])s+-?[aã]+-?o+-?([- .!?,;)\]\}\"”"_:]|$)	$1são$2
Adiciona espaço entre hashtags	
([0-9a-zà-õ])(#[^])	$1 $2
Remove cerquilhas e asteriscos das hashtags	
(^|[(\[{\"“])[#*]([0-9a-zà-õ])	$1$2
Corrigir eh	

eh([- .!?,;)\]\}\"”'_:]|$)	é$1
Adiciona espaço em palavras começadas com não* **	
(^|[(\[{\"“'])não([^ .!?,;)\]\}\"”'-_:])	$1não $2
Trata a expressão vale a pena	
(^|[(\[{\"“'])vale([uw]|r[{aá])?(+(muito)|(realmente))? +[àa] +pena([-.!?,;)\]\}\"”'_:]|$)	$1gostei$6
Trata abreviações	
(^|[(\[{\"“'])vo?ce?([- .!?,;)\]\}\"”'_:]|$)	$1você$2
(^|[(\[{\"“'])q([- .!?,;)\]\}\"”'_:]|$)	$1que$2
(^|[(\[{\"“'])pq([- .!?,;)\]\}\"”'_:]|$)	$1que$2
(^|[(\[{\"“'])c/?([- .!?,;)\]\}\"”'_:]|$)	$1com$2
(^|[(\[{\"“'])p(\\\|/|(ra))?([- .!?,;)\]\}\"”'_:]|$)	$1para$4
(^|[(\[{\"“'])tbm([- .!?,;)\]\}\"”'_:]|$)	$1também$2
(^|[(\[{\"“'])d[ ]?(\+)+([- .!?,;)\]\}\"”'_:]|$)	$1demais$3
Trata variações de substantivos	
(^|[(\[{\"“'])c+-?o+-?m+-?i+-?d+-?a+-?([- .!?,;)\]\}\"”'_:]|$)	$1comida$2
(^|[(\[{\"“'])a+-?t+-?e+-?n+-?d+-?i+-?m+-?e+-?n+-?t+-?o+-?([-.!?,;)\]\}\"”'_:]|$)	$1atendimento $2
(^|[(\[{\"“'])p+-?i+-?(z-?)+a+-?([- .!?,;)\]\}\"”'_:]|$)	$1pizza$3
(^|[(\[{\"“'])gar[sç]+o[nm]([- .!?,;)\]\}\"”'_:]|$)	$1garçom$2
(^|[(\[{\"“'])c+-?h+-?o+-?p+-?e*([- .!?,;)\]\}\"”'_:]|$)	$1chope$2

Tabela 15: Emoticons positivos

Expressão	Substituição	Emoticon
(^|[(\[{])(:\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:)
(^|[(\[{])(:\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:]
(^|[(\[{])(:})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:}
(^|[(\[{])(:o\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:o)
(^|[(\[{])(:o\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:o]
(^|[(\[{])(:o})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:o}
(^|[(\[{])(:-\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-]
(^|[(\[{])(:-\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-)
(^|[(\[{])(:-})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-}
(^|[(\[{])(=\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=)

(^|[(\[{])(=\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=]
(^|[(\[{])(=})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=}
(^|[(\[{])(=\^\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^]
(^|[(\[{])(=\^\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^)
(^|[(\[{])(=\^})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^}
(^|[(\[{])(:B)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:B
(^|[(\[{])(:-D)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-D
(^|[(\[{])(:-B)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-B
(^|[(\[{])(:\^D)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^D
(^|[(\[{])(:\^B)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^B
(^|[(\[{])(:\^D)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^D
(^|[(\[{])(:\^B)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^B
(^|[(\[{])(=B)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=B
(^|[(\[{])(=\^B)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^B
(^|[(\[{])(=\^D)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^D
(^|[(\[{])(:'\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:')
(^|[(\[{])(:'\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:']
(^|[(\[{])(:'})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:'}
(^|[(\[{])(=' \))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=')
(^|[(\[{])(=' \])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=']
(^|[(\[{]) (='})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	='}
(^|[(\[{])(<3)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	<3
(^|[(\[{])(\^\.\^)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	^.^
(^|[(\[{])(\^-\^)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	^-^
(^|[(\[{])(\^_\^)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	^_^
(^|[(\[{])(\^\^)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	^^
(^|[(\[{])(:*)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:*
(^|[(\[{])(=*)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=*
(^|[(\[{])(:-*)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-*
(^|[(\[{])(;\))(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	;)
(^|[(\[{])(;\])(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	;]
(^|[(\[{])(;})(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	;}
(^|[(\[{])(:-p)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-p
(^|[(\[{])(:-P)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-P

(^|[(\[{])(:-b)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:-b
(^|[(\[{])(:\^p)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^p
(^|[(\[{])(:\^P)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^P
(^|[(\[{])(:\^b)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:^b
(^|[(\[{])(=P)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=P
(^|[(\[{])(=p)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=p
(^|[(\[{])(\\o\\)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	\o\
(^|[(\[{])(/o/)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	/o/
(^|[(\[{])(:P)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:P
(^|[(\[{])(:p)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:p
(^|[(\[{])(:D)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:D
(^|[(\[{])(:d)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:d
(^|[(\[{])(:b)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	:b
(^|[(\[{])(=b)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=b
(^|[(\[{])(=\^p)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^p
(^|[(\[{])(=\^P)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^P
(^|[(\[{])(=\^b)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=^b
(^|[(\[{])(\\o/)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	\o/
(^|[(\[{])(*-*)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	*-*
(^|[(\[{])(=D)(\2+|[.!?,;\)\]\}]+|$)	$1feliz$3	=D

Tabela 16: Emoticons Negativos

Expressão	Substituição	Emoticon
(^|[(\[{])(D:)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	D:
(^|[(\[{])(D=)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	D=
(^|[(\[{])(D-:)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	D-:
(^|[(\[{])(D\^:)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	D^:
(^|[(\[{])(D\^=)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	D^=
(^|[(\[{])(:\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:(
(^|[(\[{])(:\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:[
(^|[(\[{])(:\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:{
(^|[(\[{])(:o\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:o(

(^|[(\[{])(:o\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:o[
(^|[(\[{])(:\^\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:^(
(^|[(\[{])(:\^\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:^[
(^|[(\[{])(:\^\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:^{
(^|[(\[{])(=\^\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=^(
(^|[(\[{])(=\^\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=^{
(^|[(\[{])(>=\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>=[
(^|[(\[{])(>=\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>={
(^|[(\[{])(>=\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>=(
(^|[(\[{])(>:-\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>:-{
(^|[(\[{])(>:-\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>:-[
(^|[(\[{])(>:-\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>:-(
(^|[(\[{])(>=\^\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>=^[
(^|[(\[{])(:-\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:-[
(^|[(\[{])(:-\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:-(
(^|[(\[{])(=\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=(
(^|[(\[{])(=\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=[
(^|[(\[{])(=\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	={
(^|[(\[{])(=\^\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=^[
(^|[(\[{])(>:-=\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>:-=(
(^|[(\[{])(>=\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>=[
(^|[(\[{])(:‟\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:’(
(^|[(\[{])(:’\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:’[
(^|[(\[{])(:’\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:’{
(^|[(\[{])(=’\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=’{
(^|[(\[{])(=’\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=’(
(^|[(\[{])(=’\[)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=’[
(^|[(\[{])(=\\)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=\
(^|[(\[{])(:\\)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:\
(^|[(\[{])(=/)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=/
(^|[(\[{])(:/)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:/
(^|[(\[{])(=\$)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	=$

(^|[(\[{])(o\.O)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	o.O
(^|[(\[{])(O_o)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	O_o
(^|[(\[{])(Oo)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	Oo
(^|[(\[{])(:\$:-\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:$:-{
(^|[(\[{])(>=\^\()(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>=^(
(^|[(\[{])(>=\^\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	>=^{
(^|[(\[{])(:o\{)(\2+|[.!?,;\)\]\}]+|$)	$1tristeza$3	:o{

Tabela 17: Emoticons Neutros

Expressão	Substituição	Emoticon
(^|[(\[{])(:|)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:|
(^|[(\[{])(=|)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	=|
(^|[(\[{])(:-|)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:-|
(^|[(\[{])(>\.<)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	>.<
(^|[(\[{])(><)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	><
(^|[(\[{])(>_<)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	>_<
(^|[(\[{])(:o)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:o
(^|[(\[{])(:O)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:O
(^|[(\[{])(:0)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:0
(^|[(\[{])(=O)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	=O
(^|[(\[{])(:@)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:@
(^|[(\[{])(=@)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	=@
(^|[(\[{])(:\^o)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:^o
(^|[(\[{])(:\^@)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:^@
(^|[(\[{])(-\.-)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	-.-
(^|[(\[{])(-\.-')(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	-.-'
(^|[(\[{])(-_-)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	-_-
(^|[(\[{])(-_-')(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	-_-'
(^|[(\[{])(:x)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:x
(^|[(\[{])(=X)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	=X
(^|[(\[{])(=#)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	=#
(^|[(\[{])(:-x)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:-x

(^|[(\[{])(:-@)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:-@
(^|[(\[{])(:-#)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:-#
(^|[(\[{])(:\^x)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:^x
(^|[(\[{])(:\^#)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:^#
(^|[(\[{])(:#)(\2+|[.!?,;\)\]\}]+|$)	$1neutro$3	:#

ANEXOS

ANEXO A – Tabela de Etiquetas

Tabela 18: Etiquetas Morfológicas

Classe Gramatical	Etiqueta
Adjetivo	ADJ
Advérbio	ADV
Advérbio Conectivo Subordinativo	ADV-KS
Advérbio Relativo Subordinativo	ADV-KS-REL
ARTIGO (def. ou indef.)	ART
Conjunção Coordenativo	KC
Conjunção Subordinativa	KS
Interjeição	IN
Nome	N
Nome Próprio	NPROP
Numeral	NUM
Particípio	PCP
Palavra Denotativa	PDEN
Preposição	PREP
Pronome Adjetivo	PROADJ
Pronome Conectivo Subordinativo	PRO-KS
Pronome Pessoal	PROPESS
Pronome Relativo Conectivo Subordinativo	PRO-KS-REL
Pronome Substantivo	PROSUB
Verbo	V
Verbo Auxiliar	VAUX
Símbolo de Moeda Corrente	CUR
Etiquetas Complementares (Estrangeirismo; Apostos; Dados; Números de Telefone; Datas; Horas e Disjunção)	|EST |AP |DAD |TEL |DAT |HOR |[|]
Contrações e Ênclises	|+
Mesóclises	|!

ANEXO B – Lista Completa das Dependências do UD PT-BR

Dependência	Descrição
acl	clausal modifier of noun (adjectival clause)
acl: inf	acl: inf
acl:part	acl:part
acl:relcl	acl:relcl
advcl	adverbial clause modifier
advmod	adverbial modifier
amod	adjectival modifier
appos	appositional modifier
aux	auxiliary
auxpass	passive auxiliary
case	case marking
cc	coordinating conjunction
ccomp	clausal complement
compound	compound
conj	conjunct
cop	copula
csubj	clausal subject
csubjpass	clausal passive subject
dep	unspecified dependency
det	determiner
det:poss	det:poss
discourse	discourse element
dislocated	dislocated elements

dobj	direct object
expl	expletive
foreign	foreign words
goeswith	goes with
iobj	indirect object
list	list
mark	marker
mwe	multi-word expression
name	name
neg	negation modifier
nmod	nominal modifier
nsubj	nominal subject
nsubjpass	passive nominal subject
nummod	numeric modifier
parataxis	parataxis
punct	punctuation
remnant	remnant in ellipsis
reparandum	overridden disfluency
root	root
vocative	vocative
xcomp	open clausal complement
xcomp:adj	xcomp:adj

ANEXO C – Adversativos, Negativos e delimitadores

Palavras negativas:

não; tampouco; nem; nunca; jamais; mal

Palavras adversativas:

mas; porém; contudo; todavia; entretanto; embora; apesar