Publicação: Fevereiro de 2019

Digital Object Identifier https://doi.org/10.33911/singular-etg.v1i1.22

Desenvolvimento do Módulo de Pre-processamento da ferramenta SentimentALL

PARCILENE FERNANDES BRITO e LUAN GOMES DE ALMEIDA ARAÚJO.

Centro Universitário Luterano de Palmas, Teotônio Segurado 1501 Sul Palmas - TO CEP 77.019-900 Caixa Postal nº 85
(e-mail: {parcilene, luangomes2121}@gmail.com)
Autor Correspondente: Parcilene Fernandes Brito (e-mail: parcilene@gmail.com).

RESUMO - Os sistemas que utilizam em seu processo de análise a linguagem natural (por exemplo, sistemas que analisam comentários de sites, reviews de filmes, posts em redes sociais etc.) geralmente usam técnicas relacionadas ao Processamento de Linguagem Natural (PLN). Neste artigo, serão apresentadas as etapas de PLN utilizadas no desenvolvimento do Módulo de Pré-Processamento da Ferramenta SentimentALL, que tem como propósito realizar análises de textos que emitem opiniões sobre produtos. O módulo foi desenvolvido para ser usado em qualquer contexto, mas neste trabalho foi utilizado na análise dos comentários extraídos do site de turismo TripAdvisor. O módulo é composto das etapas de normalização, correção ortográfica, pos-tagging e identificação de expressões compostas. Para a sua implementação, foi utilizada a biblioteca NLTK, da linguagem de Programação Python.

PALAVRAS-CHAVE - Processamento de Linguagem Natural, SentimentALL, Pos-Tagging, NLTK.

I. INTRODUÇÃO

Processamento de Linguagem Natural (PLN), ou em inglês Natural Language Processing (NLP), é uma subárea da linguística computacional. Segundo [8] 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 ([8]; [17]).

A maioria das informações presentes na internet está na forma não estruturada e em linguagem natural. Portanto, para a resolução de determinados problemas, é necessária a utilização de técnicas de PLN para extrair tais informações, transformando-as para a forma estruturada. 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.

O módulo apresentado neste artigo é parte do projeto SentimentALL de [20] e está vinculado diretamente à implementação da segunda versão da ferramenta SentimentALL, desenvolvida em [2] e [19]. Neste projeto, foram desenvol-

vidos três módulos: coleta de dados, pré-processamento e análise de sentimentos. Na coleta de dados foram extraídos mais de seis milhões de comentários das páginas do site de turismo TripAdvisor. Esses dados foram pré-processados utilizando as técnicas de PLN, foco deste trabalho, que serão apresentadas nas próximas seções. O pré-processamento consistiu em realizar a normalização dos dados, correção ortográfica, identificação da classe morfológica das palavras e identificação de expressões multipalavras.

II. ETAPAS DE ANÁLISE EM PLN

As análises tradicionais (sintaxe, semântica e pragmática) servem, no máximo, como um ponto de partida para o processamento de textos reais em linguagem natural ([7]). Então, quando se lida com dados reais, os estágios são decompostos como na Figura 1.

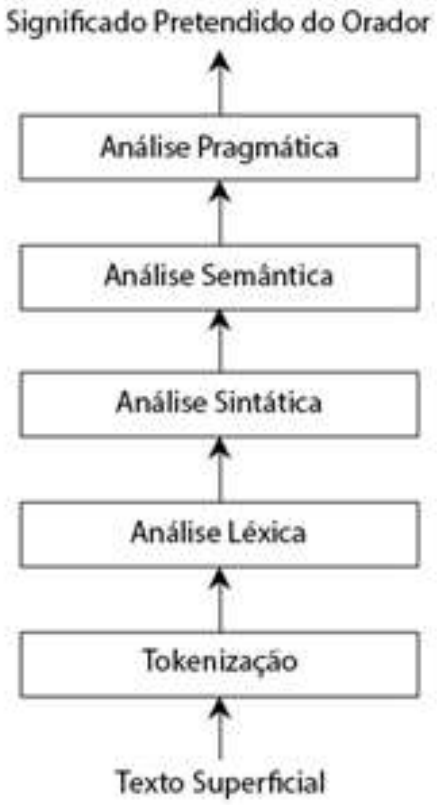

Figura 1. Os estágios de análise em PLN. [7] (traduzido)

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.

1) Tokenização

Segundo [16], 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.

[16] 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) 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 [5], 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.

3) Análise Sintática

A análise sintática verifica se a estrutura gramatical de uma frase está correta. Segundo [11], 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:

$$
\begin{aligned}
FRASE &\rightarrow SINTAGMANOMINAL\ SINTAGMAVERBAL \\
SINTAGMANOMINAL &\rightarrow ARTIGO\ SUBSTANTIVO \\
&\mid SUBSTANTIVO \\
SINTAGMAVERBAL &\rightarrow VERBO\ SINTAGMANOMINAL \\
&\mid VERBO \\
ARTIGO &\rightarrow a \mid o \mid um \mid uma \\
SUBSTANTIVO &\rightarrow João \mid Maria \mid cachorro \\
VERBO &\rightarrow ganhou \mid deu \mid recebeu
\end{aligned}
$$

Figura 2. Regras de Produção

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 [12], "é a substituição de uma sub-palavra 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):

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

Figura 3. Exemplo de Derivação

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:

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.

4) Análise Semântica

A análise semântica trata do significado da sentença, sobre aquilo que é possível entender através de um determinado

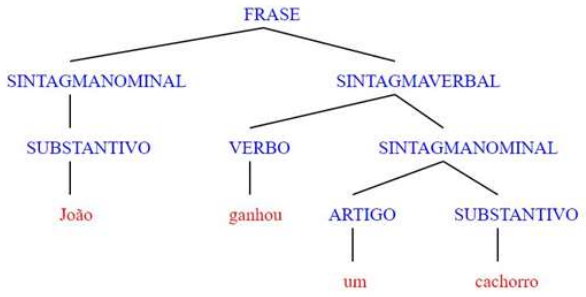

Figura 4. Árvore de Sintaxe

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" ([5], 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:

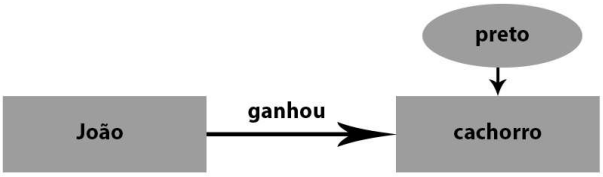

Figura 5. Rede Semântica

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".

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:

"Pitoco é um cachorro. "

"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. [7] 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.

A. POS TAG

PoS tagging é uma tarefa que ocorre na etapa de análise léxica em PLN. Segundo [10], 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 tags de um tagset, para cada palavra da sentença ([1]; [9]).

[9] 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, [9] 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 1representa 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	Substativo

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.

B. EXPRESSÕES MULTIPALAVRAS (MWE)

Expressões Multipalavras, ou em inglês Multiwords Expression (MWE), são, segundo [18], 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 [4], é o seguinte.

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)} \quad (1)$$

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:

$$\begin{aligned} I(cachorro, quente) &= \log_2 \frac{P(cachorro, quente)}{P(cachorro)P(quente)} \\ &= \log_2 \frac{41\%}{23\% * 11\%} \\ &= 4,02 \end{aligned}$$

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 [6].

$$SI(x_1, x_2, ..., x_n) = \log_2 \frac{P(x_1, x_2, ..., x_n)}{\Pi_{i=1}^{n} P(x_i)} \qquad (2)$$

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 ($P(x_1, x_2, ..., x_n)$) e é realizado o produto da probabilidade individual de todas as palavras ($\Pi_{i=1}^{n} P(x_i)$).

O cálculo de PMI é realizado sobre um corpus. Nesse corpus é realizado o PoS tagging e depois, são identificados padrões morfológicos. [3], 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. [3]

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

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.

III. METODOLOGIA

O módulo desenvolvido neste trabalho utilizou os dados obtidos na ferramenta SentimentALL, extraídos do site TripAdvisor. Especificamente, foram usados comentários feitos pelos usuários do site, em um total de 6.438.497 comentários dos 100 destinos turísticos brasileiros mais avaliados no site entre início de fevereiro e final de março de 2017.

Para o desenvolvimento do módulo foi utilizada a linguagem Python, codificada por meio da Integrated Development Environment (IDE) PyCharm da Jet Brains. 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 etc. 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 e implementado o algoritmo de PMI para a definição de expressões compostas. Esse módulo teve como entrada os dados coletados (comentários do TripAdvisor) e como saída os dados normalizados, corrigidos ortograficamente, etiquetados e as possíveis expressões multipalavras identificadas.

IV. RESULTADOS

A. 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.

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 em [2]. A maioria das expressões regulares foi criada por [14] 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 ")". [2] apresenta todas as expressões alteradas, adicionadas e que permaneceram iguais.

B. CORREÇÃO ORTOGRÁFICA

Essa etapa tem a função de corrigir a grafia das palavras. O algoritmo utilizado foi baseado em [13], 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 explicado 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 [13]. 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. As letras utilizadas na adição e substituição estão no seguinte 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.

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. [13] 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 [13] está em inglês, foi necessário criar um novo. Para isso, foram coletadas notícias nos sites da 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. 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".

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.

C. 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*.

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 7 apresenta um exemplo do contexto de um *3-gram tagger*, também chamado de *trigram tagger*.

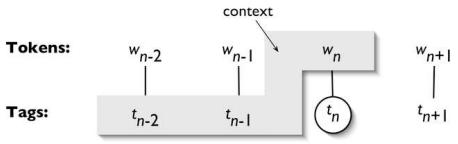

Figura 7. Contexto do Tagger. Bird, Klein e Loper (2009)

A área sombreada em cinza na Figura 7 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 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 os artigos, alguns pronomes e alguns advérbios.

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 palavra que pode ser contraída com a anterior, então é feita a contração. Quando uma contração é realizada, é 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 o problema das contrações e combinações foi resolvido.

Um outro problema identificado foi o fato de 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, era verificado se ela estava na lista de adjetivos, se estivesse, era definida a *tag* como "ADJ". Se não fosse um adjetivo, verificava se ela estava na lista de verbos, se estivesse, definia a *tag* como "V". Se não fosse um verbo, verificava se ela era um número usando a expressão regular "\d+([,\.]\d+)?" e etiquetava como "NUM". Se depois de todas as verificações nenhuma condição fosse satisfeita, as palavras eram etiquetadas 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.

D. 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 4 apresenta esses padrões.

O algoritmo utilizado para realizar o primeiro passo é baseado no pseudocódigo usado em [15]. A mudança está apenas no fato de que em [15] 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. 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 formem um padrão, 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 candi-

data é encontrada, calcula-se o seu PMI, usando a Equação 2, proposta por [6] (2011):

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.

$$Precision = \frac{V_p}{V_p + F_p} \tag{3}$$

$$Recall = \frac{V_p}{V_p + F_n} \tag{4}$$

$$FMeasure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5}$$

Onde V_p estão erradas; e F_n é a quantidade de expressões que deveriam ter sido reconhecidas, mas não foram.

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 8 apresenta o gráfico com os valores de *Precision*, *Recall* e *F-Measure*, para cada valor de x.

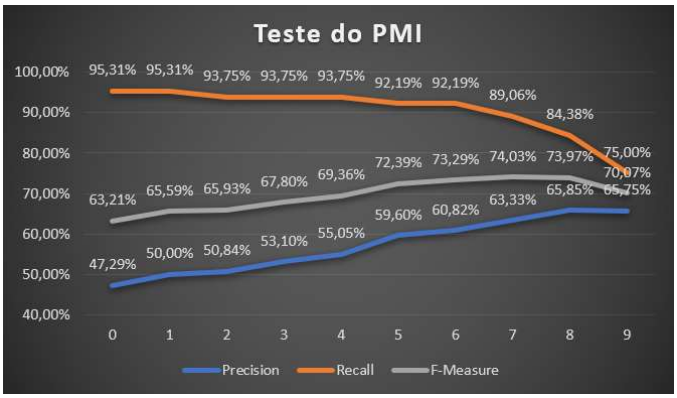

Figura 8. Resultado do teste do PMI

Como pode se observar na Figura 8, 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%.

V. CONCLUSÕES

O módulo de pré-processamento foi implementado com o propósito de ser livre de contexto e de que sua utilização fosse relativamente simples, considerando que será usado por outras equipes do grupo de pesquisa. Para cada etapa do módulo há uma entrada e uma saída, sendo que nestas etapas 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, dentre outros, contanto que os dados de entrada estejam no formato requerido pelo módulo. O mesmo acontece com a saída, que pode ser salva em um arquivo de texto, banco de dados etc. Todas as etapas que compõem o módulo são independentes, portanto algumas delas 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.

O módulo de tokenização de sentenças e palavras estão bastante precisos para a língua portuguesa na biblioteca NLTK do Python, isso forneceu uma dinamicidade maior na implementação do módulo. O 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, também, um problema em relação a palavras que não estão no corpus Mac-Morpho, mas foi contornado utilizando uma lista de adjetivos e verbos, disponíveis no site Linguateca 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.

Referências

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

[2] ARAUJO, L. G. A. SENTIMENTALL VERSÃO 2: Desenvolvimento de Análise de Sentimentos em Python, 103 f., TCC (Graduação) - Curso de Bacharelado em Ciências da Computação, Centro Universitário Luterano de Palmas, Palmas, Tocantins. 2017.

[3] 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).

[4] Brito, Parcilene Fernandes de. RELATOS VERBAIS DE CONSUMIDORES EM AVALIAÇÕES ON-LINE: PROSPECÇÃO COMPUTACIONAL E INTERPRETAÇÕES COM BASE NO BEHAVIORAL PERSPECTIVE MODEL (BPM). 2018.182 fl. Tese(Programa de Pós-Graduação STRICTO SENSU em Psicologia) - Pontifícia Universidade Católica de Goiás, Goiânia-GO .

[5] Brito, P. F., Oliveira, W. C. C., Souza, J. G. (2015). SentimentALL. Fábrica de Software: CEULP/ULBRA.

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

[7] COPPIN, Ben. Inteligência Artificial. Rio de Janeiro: Ltc, 2013. 636 p.

[8] 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.

[9] 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.
[10] 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.
[11] 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.
[12] 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.
[13] 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.
[14] MENEZES, Paulo Blauth. Linguagens Formais e Autômatos. 6 ed. Porto Alegre: Bookman, 2011. 256 p.
[15] NORVIG, P. "How to Write a Spelling Corrector". 2016. Disponível em: <http://norvig.com/spell-correct.html>, Acesso em: 12 jan. 2019.
[16] OLIVEIRA, W. C. C. "SENTIMENTALL: Ferramenta para análise de sentimentos em português", 86 f., TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Centro Universitário Luterano de Palmas, Palmas, Tocantins. 2018.
[17] OLIVEIRA, W. C. C. e BRITO, P. F. "Utilização do Pointwise Mutual Information na Identificação de Expressões Multipalavras", In: XVII ENCOINFO – CONGRESSO DE COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO, Palmas. p. 25 - 34. 2015.
[18] 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.
[19] RUSSELL, Stuart; NORVIG, Peter. Inteligência Artificial. 3. ed. Rio de Janeiro: Elsevier, 2013. 1016 p. Tradução de: Regina Célia Simille.
[20] 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.

PARCILENE FERNANDES BRITO Doutora em Psicologia (PUC/GO), Mestre em Ciências da Computação (UFSC), Especialista em Ciências da Computação (UFSC), Especialista em Informática para Aplicações Empresariais (ULBRA), Graduada em Psicologia (CEULP/ULBRA), Graduada em Processamento de Dados (UNITINS). Pesquisadora do Grupo de Pesquisa Engenharia Inteligente de Dados (CEULP/ULBRA). Atualmente é Diretora Acadêmica do Ceulp/Ulbra, Coordenadora e Professora dos cursos de Sistemas de Informação, Ciência da Computação e Engenharia de Software na mesma IES. Tem experiência na área de Computação e Psicologia, com ênfase em Lógica Formal, Análise de Sentimentos, Ontologias, Psicologia do Consumidor e Informática na Educação.

LUAN GOMES DE ALMEIDA ARAÚJO Possui graduação em Ciências da Computação (CEULP/ULBRA), Técnico em Informática (IFTO) e Técnico em Mecatrônica (IFTO) e especialização em andamento em Prática de Metodologias Ágeis (UniCesumar). Atualmente trabalha no Tribunal Regional Federal da 1ª Região, desenvolvendo sistemas judiciais.