ANALIZO - ANÁLISE TEXTUAL PARA VALIDAÇÃO DE COMUNICAÇÃO CORPORATIVA

Johny William de Oliveira Alves, Walter Gomes Pedroso Junior
Especialização em Gestão de Projetos em Tecnologia da Informação, Faculdade de Tecnologia de Rio Preto, São José do Rio Preto-SP

e-mail: contato@johnywalves.com.br, walter.pedroso@fatecriopreto.edu.br

Resumo: Neste trabalho apresentamos conceitos de lexema linguístico, aprendizado de máquina e alguns algoritmos utilizados na linguagem Python, explorando uma aplicação de indicação de palavras não recomendadas e classificação binária de sentenças com uso de manipulação, transformação e carregamento de dados com processamento de linguagem natural, apontando métodos e resultados para desenvolver um sistema para indicação e treinamento de comunicação corporativa.

Palavras-chave: Aprendizado de máquina, Processamento de Linguagem Natural, Tratamento de dados

Abstract: *In this work we present concepts of linguistic lema, machine learning and some algorithms used in the Python language, exploring an application of indication of non-recommended words and binary classification of sentences using manipulation, transformation and loading of data with natural language processing, pointing methods and results to develop a system for corporate communication referral and training.*

Keywords: *Machine Learning, Natural Language Processing, Data Transform*

1. Introdução

Na busca de vencer a concorrência ou manter a dominância de um mercado, a imagem da marca possui uma importância fundamental, como dito por Ribeiro (2012) “O seu relacionamento com seus clientes não tem como base a realidade. Os relacionamentos mudam, evoluem e se completam em torno de percepção” para trabalhar nesse relacionamento é necessário o engajamento de toda equipe, principalmente os que possuem contato direto com o cliente, como a equipe de publicidade, venda, pós-venda, entre outros.

Com base neste contexto, o trabalho presente tem por objetivo a estruturação do desenvolvimento de uma ferramenta para validar essa comunicação, através do uso de dicionário de

palavras e processamento de linguagem natural, com uso de inteligência artificial, registrando esse processo para garantir auditoria e uma base de melhoria do processo.

O presente trabalho está organizado da seguinte maneira. A seção 2 descreve a justificativa para a formulação da ferramenta. A seção 3 descreve fundamentação teórica base para a aplicação das teorias. A seção 4 descreve a fundamentação teórica com a base teórica das aplicações. A seção 5 descreve o desenvolvimento da ferramenta com a transformação e carga dos dados para treino dos modelos de aprendizado de máquina. A seção 6 descreve a conclusão do estudo. A seção 7 apresenta considerações sobre o trabalho e as possibilidades de expandi-lo futuramente.

2. Justificativa

Na busca de um lugar no mercado devemos nos munir de uma boa publicidade, devemos nos preocupar com a visão do cliente e funcionários, como apresentado por Sampaio (2013) “Apesar de buscarmos a racionalidade do ceticismo, de incentivarmos a defesa da indiferença, sempre há uma mensagem publicitária que nos atrai, interessa e convence.”. Para alcançar essa atração e convencimento devemos possuir uma comunicação que se alinhe com os valores defendidos pela corporação.

Quanto mais amplia uma empresa, cresce a imagem de distância dos seus clientes e funcionários, eles se tornam muitos para garantir uma escala adequada a fim de suportar o aumento constante, isso ocorre pela falta de capacidade de manter a comunicação corporativa e institucional que mantenha a imagem positiva da marca para os interessados.

> Não se pode perder de vista que a marca é uma experiência total (no famoso conceito alemão de gestalt), e isso inclui tudo o que está ligado ao produto, do design gráfico da marca ao serviço pós-venda, passando pela propaganda, pelo canal de venda, pela entrega e pela performance do produto em si, dentre diversos fatores (SAMPAIO, 2013)

Procurando manter essa linha de diálogo com os consumidores “Para isso você precisa montar um projeto para a empresa que envolva todas as pessoas que trabalhem nela” (RIBEIRO, 2012) envolvendo conscientização e treinamento, mesmo no contexto de contato por mensageiros de textos, geração de material publicitário, documento para imprensa, entre outros.

Uma ferramenta de indicação e educação no momento em que o texto é gerado com capacidade de centralizar essas decisões pode ser de grande ajuda a equipe responsável por decidir a linha da conversa da marca, também para acompanhar a evolução na geração destas mensagens melhorando o processo de comunicação com os consumidores.

3. Metodologia

O presente trabalho possui caráter exploratório, pois busca identificar os principais critérios para uma análise de um conjunto de expressões em linguagem natural com uso de aprendizado de máquina disponível na linguagem Python e suas bibliotecas.

Etapa 1: Levantamento bibliográfico

O levantamento bibliográfico foi realizado a partir de leituras de textos voltados para estudo de mercado e construção de uma marca na percepção dos funcionários e clientes. Com isso direcionamos a formulação dos materiais para o estudo.

Etapa 2: Estudo de caso

Para conduzir o caso de estudo, fizemos uso de uma empresa e produtos fictícios, e uma lista de frases geradas pelos autores e coletadas nos meios publicitários da Docile Alimentos Ltda (Docile), Fini Comercializadora Ltda (Fini) e Haribo Brasil Indústria e Comércio de Produtos Alimentícios Ltda (Haribo), que podem ser usadas pela equipe de atendimento ou publicidade, geradas para o estudo simulando uma comunicação.

Etapa 3: Investigação de algoritmos para classificação

A partir do conteúdo da empresa como nome, catálogo de produtos e exemplos de comunicação foi realizado uma investigação do algoritmo que melhor se encaixa para realizar a classificação das expressões.

4. Fundamentação Teórica

Na fundamentação teórica apresentamos os conceitos utilizados do desenvolvimento, deste trabalho com uma introdução: aprendizado de máquina (*Machine Learning*), lexema linguístico, linguagem de programação Python e alguns modelos de predição como: Floresta Aleatória (*Random Forest*), Árvore de Decisão (*Decision Tree*), Multi-Layer Perceptron Classifier e Multinomial Naive Bayes.

4.1. Aprendizado de máquina (*Machine Learning*)

O aprendizado de máquina ou *machine learning* pode ser explicado como "Coleta métodos para a extração (previsão) de modelos a partir de dados, agora conhecidos como métodos de aprendizado de máquina” (PROVOST e FAWCETT, 2016). Em outras palavras, o aprendizado de máquina é baseado em aplicar um conjunto de técnicas matemáticas, lógicas ou ambas para

encontrar padrões em um grupo de dados, ao replicar essas técnicas em outros dados podemos realizar previsões de valores; esse conjunto de técnicas é chamado de modelo preditivo.

Os modelos preditivos, podem ser divididos em supervisionados ou não supervisionados, essa supervisão é caracterizada por informar ao modelo ao valor desejado, para esse estudo focamos em modelos supervisionados para classificação.

Modelos de classificação são desenvolvidos para segmentar os dados em subgrupos pelos seus atributos, por exemplo, se uma pessoa pode receber um crédito imobiliário ou qual o grupo nutricional de determinado alimento. No exemplo da Tabela 4.1 podemos ver a aptidão dos atletas para participarem de uma competição de triathlon olímpico. Como atributos, temos nome e distância percorridos nas modalidades de natação, bicicleta e corrida e classificação se ele está apto para participar da competição.

Tabela 4.1 - Exemplo de classificação de atletas aptos para disputar um triathlon olímpico

Nome	**Natação (Km)**	**Bicicleta (Km)**	**Corrida (Km)**	**Apto**
João	0.8	30.0	10.2	Não
Maria	1.5	40.0	5.0	Sim
José	1.6	45.0	10.0	Sim
André	1.4	40.0	5.0	Não
Luciana	0.4	35.0	3.5	Não

Fonte: elaborado pelos autores

Com sua acurácia medida pelo percentual de acertos entres as classificações preditas e esperadas com valores variando de 0 para ruim e 1 para ótimo. O cálculo de precisão é a aplicação dos $positivos\ verdadeiro\ /\ (positivos\ verdadeiros\ +\ falsos\ positivos)$ com valores variando de 0 para ruim e 1 para ótimo.

Os modelos utilizados neste trabalho são:

4.1.1. Multinomial Naive Bayes

O algoritmo Naive Bayes trabalha com cálculos de probabilidades, embora muito simples leva em contas as evidências características dada pela equação descrita na Figura 4.1.

Figura 4.1 - Equação Naive Bayes

$$p(c \mid E) = \frac{p(e_1 \mid c) \,.\, p(e_2 \mid c) \ldots p(e_k \mid c) \,.\, p(c)}{p(E)}$$

Fonte: Adaptado de PROVOST e FAWCETT (2016)

Sendo:

c: Ocorrência que estamos analisando, exemplo: Chuva no dia anterior;

E: Evidência de ocorrência, exemplo: O asfalto está molhado;

$p(c \mid E)$: a probabilidade de **c** condicionada a **E**, exemplo: A possibilidade de ter chovido caso o asfalto estava molhado;

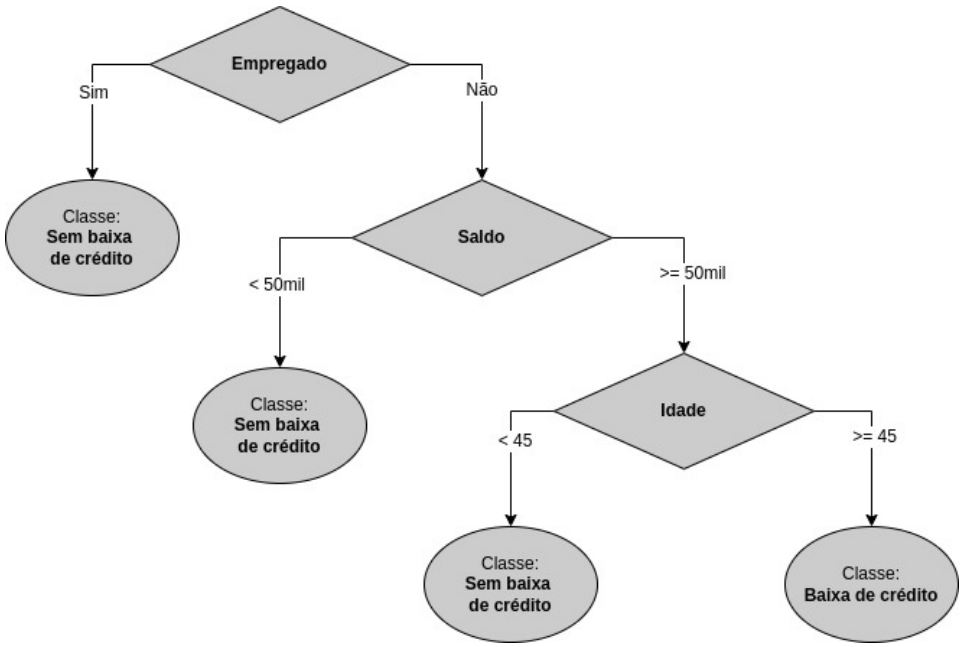
$p(E)$: probabilidade de **E** ocorrer sem inferência, nesse caso uma lista de evidências, exemplo: Possibilidade do chão estar molhado, independe de chovido ou não;

$p(e_n|c)$: a possibilidade de **c** condicionada por data evidência.

4.1.2. Árvore de Decisão (*Decision Tree*)

Uma árvore de decisão, ou árvore de classificação, é uma algoritmo de segmentação estruturado em nós de decisão, como no exemplo da Figura 4.2, no processo de aprendizado gera um caminho baseado na maior possibilidade de sucesso.

Figura 4.2 - Um simples árvore de classificação



Fonte: Adaptado de PROVOST e FAWCETT (2016)

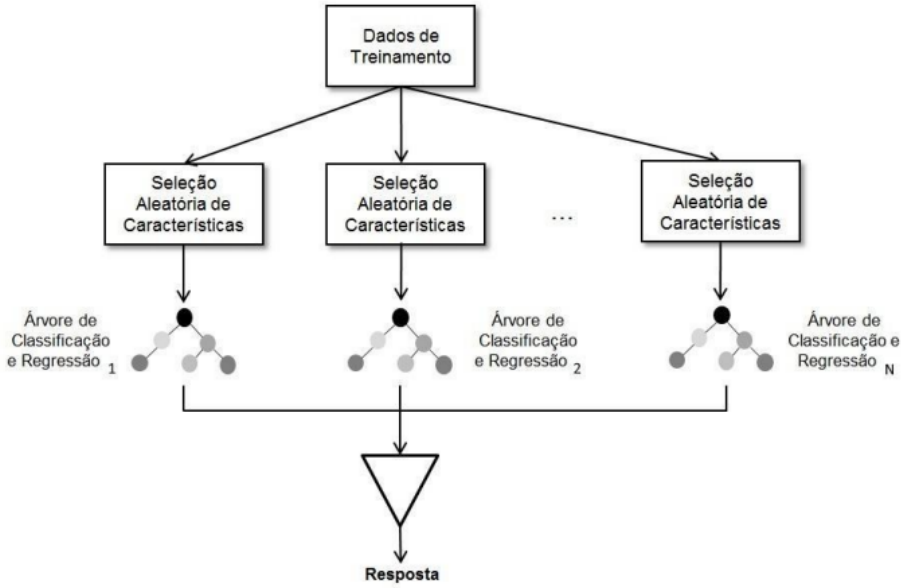
4.1.3. Floresta Aleatória (*Random Forest*)

O modelo da floresta aleatória é uma aplicação de várias árvores de decisão de classificação geradas aleatoriamente, o resultado é definido para o maior número de ocorrência em resultados.

> O método random forest (floresta aleatória) foi desenvolvido por Breiman (2001) para combinar vários modelos, especificamente árvore de classificação e/ou regressão, com base em vetores de características (covariáveis), os quais são gerados de maneira aleatória e independente a partir do conjunto de dados. (BORGES, 2016)

O aprendizado de máquina gera as árvores de forma aleatória e seleciona o conjunto das com os melhores resultados como apresentado na Figura 4.3.

Figura 4.3 - Exemplo de desenvolvimento de uma Random Forest

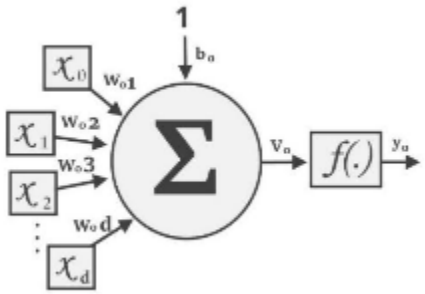


Fonte: Borges Jr. et al. (2016)

4.1.4. Multicamadas de Perceptrons (*Multi-Layer Perceptron*)

Um perceptron é uma aplicação de um neurônio artificial, um neurônio artificial segundo Silva et al. (2016) "o neurônio artificial é, na realidade, uma função que mapeia entradas e saídas.". Como apresentado na Figura 4.4 temos o recebimento de valores de entrada $\{x_1, x_2 \dots x_d\}$ que são multiplicados pelos seus respectivos pesos $\{W_{o1}, W_{o2} \dots W_{od}\}$ somados, acrescidos de um valor externo (b_o) esse resulta por sua vez é aplicado em uma função para classificar o valor como 1 ou 0.

Figura 4.4 - A estrutura de um neurônio artificial do tipo Perceptron.



Fonte: adaptado de Silva et al. (2016)

No processo de descoberta do aprendizado o modelos estipula e valida os valores e pesos de cada perceptron que forma o sistema de multicamadas.

4.2. Lexema linguístico

O lema, ou lexema, é a forma canônica de uma palavra, reduzida ao seu significado mais básico, normalmente usado como verbete em dicionários, também podemos descrever como:

> Entende-se por lexema a unidade linguística dotada de *significado léxico*, isto é, aquele significado que aponta para o que se apreende do mundo extralinguístico mediante a linguagem. Assim, em *amor*, *amante*, *amar*, amavelmente o significado léxico é comum a todas as palavras da série. (BECHARA, 2009)

Cada palavra de uma sentença pode ter um atributo dotado de flexibilização de gênero, número, grau, entre outros. Para reduzir o número de atributos analisados pelo aprendizado de máquina, fizemos o uso do lema das palavras.

4.3. Linguagem Python e suas bibliotecas

Em uma tradução livre do VanderPlas (2017) “O Python emergiu nas últimas duas décadas como uma ferramenta de primeira classe para pesquisas científicas, incluindo a análise e visualização de grandes conjuntos de dados”, Python é uma linguagem de fácil aprendizado que é largamente usada pela área de ciência de dados por possuir uma vasta coleção de pacotes, conjunto de funções prontas para agilizar o processo de desenvolvimento incluindo aprendizado de máquina.

4.3.1. Pandas

A biblioteca Pandas cria um objeto de conjuntos de dados chamado de *DataFrame*, uma tabela com linhas e colunas com rótulos que aceita vários formatos de tipos de informações, para

guardar e manipular dados de forma eficiente, também permite ser carregado e salvo em arquivos de formatos diferentes.

4.3.2. Scikit-Learn

Scikit-Learn possui uma vasta, eficiente e claras implementações para aprendizagem de máquina com os mais conhecidos e usados algoritmos e ferramentas de transformações para adaptar os dados para os formatos necessários para aplicações nos algoritmos.

4.3.3. spaCy

Pacote de funções, rotinas e modelos de processamento de linguagem natural, como identificação da classe gramatical e conversão do texto para seu lexema.

4.3.4. FastAPI

Ferramenta em Python focada em aplicações web pela disponibilização de serviço rodando em servidores, provendo uma *Application Programming Interface* (API) para processamento de dados e disponibilização de informações, conhecida pela alta performance, facilidade de uso e geração de documentação automática.

5. Desenvolvimento

O presente trabalho foi desenvolvido seguindo os passos de criar uma dicionário de palavras válidas, lematização e vetorização das frases, treinamento do aprendizado de máquina e disponibilização do serviço, detalhados a seguir.

5.1. Dicionário de palavras

Como caso de estudo foi utilizado uma empresa fictícia de doces com o nome de Lutscher, palavra em alemão para pirulito, escolhida arbitrariamente, e o catálogo de produtos descrito na Tabela 5.1, o nome da empresa e a listagem dos produtos devem constar como palavras aceitas para nossa validação:

Tabela 5.1 - Catálogo de produtos da Lustcher

Nome	**Descrição**
Chupscher	Pirulito com vários sabores de frutas
DulCacau	Barra de chocolate com recheio com sabores de doces tradicionais
Marsher	Marshmallow com sabores diversos

Fonte: elaborado pelos autores

Algumas palavras não são recomendadas por passar uma mensagem que deve se evitar ser ligada a marca, alguns exemplos descritos na Tabela 5.2.

Tabela 5.2 - Radical para palavras removidas

Radical	**Motivo**	**Exemplo de palavras**
diab-	Referência a uma doença alimentícia e religiosa	diabete, diabetes, diabos, diabrura
trist-	Sentimento contra vender alegria	triste, tristeza, tristona, trississimo
amarg-	Sabores oposto a doce	amarga, amargado, amargáveis

Fonte: elaborado pelos autores

Fazendo uso da linguagem Python e da biblioteca Pandas, lemos a listagem de palavras do Instituto de Matemática e Estatística Universidade de São Paulo (IME-USP, 2022) adicionamos o nome da empresa e o catálogo de produtos gerando um arquivo com essa listagem (Anexo 1) com o trecho de código, também removemos algumas palavras que não devem fazer parte do tipo de mensagem escolhido para comunicação do estudo de caso

No final do processo foi gerado o arquivo **palavras.csv** contendo 261.710 termos, que foi usado nas próximas etapas.

5.2. Lematização e vetorização das frases

Fizemos uso da base de frases com classificação, gerada pelos autores com coleta e adaptação de itens encontrados nos sites e redes sociais da Docile, Fini e Haribo, em 100 frases divididas igualmente em duas categorias positiva e negativa, ambas com 50 frases apresentadas no Anexo 2, arquivo salvo no formato CSV com o nome **frases.csv** e carregada com a biblioteca Pandas como apresentada na Tabela 3.

Tabela 5.3 - Carregar base de treinamento com a biblioteca Pandas

```
# Importar Pandas - biblioteca para manipulação de datasets
import pandas as pd

# Carregar listagem de frases
df_frases = pd.read_csv('./data/frases.csv', encoding='utf-8')
```

Fonte: elaborado pelos autores

As sentenças possuem classes gramaticais tanto com flexibilização de gênero, número, grau, tempo, modo e pessoa. Para possibilitar as análises, fizemos o uso do lexema das palavras, extraído por um conjunto de algoritmos de processamento de linguagem natural presente na biblioteca spaCy.

O pacote spaCy instalado pelo gestor de pacotes do Python (Pip), com o comando "`pip install spacy`" e o modelo já treinado em português, pela execução do "`python -m spacy download pt_core_news_lg`", criamos uma função para fazer o uso da biblioteca e do modelo (Anexo 3), extraindo somente as classes gramaticais de substantivos, verbos, advérbios e adjetivos sem suas flexibilizações, como exemplo a frase "Uma colorida e irresistível variedade" no formato dos lexemas e separação das classes apresenta como "colorido irresistível variedade".

Com a lematização em mãos, precisamos adequar o conjunto de palavras para uma padrão numérico usado pelas bibliotecas de aprendizado de máquina, com uma lista de valores posicionada na coluna por palavras como nos exemplos listados na Tabela 5.4.

Tabela 5.4 - Exemplo de vetorização dos lemas

Léxicos	**afeto**	**ajudar**	**alegria**	**amar**	**diversidade**	**família**
diversidade alegria família	0	0	1	0	1	1
afeto amar	1	0	0	1	0	0
ajudar família	0	1	0	0	0	1

Fonte: elaborado pelos autores

Para realizar a transformação dos lexemas em um vetor de de existência, fizemos o uso da biblioteca *CountVectorizer* presente no pacote do Scikit-Learn, demonstrada no Anexo 4, com os vetores gerados armazenados no *DataFrame* para facilitar consultas posteriores.

Na primeira versão a listagem apresentou um formato de 100 linhas de 233 colunas vetorizadas, o que atrapalha o processo de aprendizado pois atributos não agregam decisões se aparecem um ou poucas vezes, a listagem foi atualizada para aumentar a repetição de palavras e ampliar a acurácia, trocando alguns termos por seus sinônimos que aparecem em outras frases, para melhorar as referências entre frases, listada no Anexo 2, gerando uma nova vetorização com dimensões 100 linhas e 216 colunas.

5.3. Treinamento do aprendizado de máquina

Cada frase possui um vetor léxico, para realizar o treinamento e teste dos algoritmos separamos as fontes em 50 positivas e 50 negativas e posteriormente uma relação de 70% e 30%, para treinamento e teste respectivamente, implementadas no Anexo 5.

A fonte de dados de treinamento foi usada para realizar o aprendizado de máquina, ensinando pelos algoritmos como chegar no resultado de classificação. Na Tabela 5.5 podemos ver a importação e parâmetros usados para cada modelo treinado, também a execução da função `analizar_modelo` (declarada no Anexo 6) colecionando os resultados de acurácia e precisão.

Tabela 5.5 - Disponibilização do serviço de análise pela API

```
# Random Florest Classifier
from sklearn.ensemble import RandomForestClassifier
modelo_rf = RandomForestClassifier(max_depth=10, n_estimators=10, random_state=42)
resultado_metricas = analizar_modelo('Random Florest Classifier', modelo_rf)

# Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
modelo_dtc = DecisionTreeClassifier(random_state=42)
resultado_metricas = analizar_modelo('Decision Tree Classifier', modelo_dtc)

# Multi-Layer Perceptron Classifier
from sklearn.neural_network import MLPClassifier
modelo_clf = MLPClassifier(random_state=42, max_iter=1000)
resultado_metricas = analizar_modelo('Multi-Layer Perceptron Classifier', modelo_clf)

# Multinomial Naive Bayes
from sklearn.naive_bayes import MultinomialNB
modelo_mnb = MultinomialNB()
resultado_metricas = analizar_modelo('Multinomial Naive Bayes', modelo_mnb)
```

Fonte: elaborado pelos autores

No objeto do `resultado_metrica` colecionamos as métricas de resultado e o próprio modelo treinado, podemos observar esses resultados na Tabela 5.6. Com a informação desta maneira temos como visualizar e selecionar o melhor modelo para ser usado posteriormente.

Tabela 5.6 - Resultados de acurácia e precisão das modelagem testadas

Name	Acurácia	Precisão
Random Forest Classifier	0.7333	0.7511
Decision Tree Classifier	0.8000	0.8000
Multi-Layer Perceptron Classifier	0.7666	0.7678
Multinomial Naive Bayes	0.8000	0.8000

Fonte: elaborado pelos autores

Observando os resultados vimos que os modelos possuem acurácias semelhantes para os dados e parâmetros usados, ao invés de escolher uma das opções criamos uma função para realizar essa escolha baseado na melhor acurácia. Como o resultado pode variar dependendo dos dados de

treinos usado para o treinamento com o código na Tabela 5.7 podemos garantir que sempre vamos ter o modelo com o melhor resultado para ser utilizado posteriormente.

Tabela 5.7 - Seleção do modelo com melhor acurácia entre os resultados

```
top_accuracy = 0
name_selected = ''
model_selected = None

for index, metrica in resultado_metricas.iterrows():
  if top_accuracy < metrica.Accuracy:
    top_accuracy = metrica.Accuracy
    name_selected = metrica.Name
    model_selected = metrica.Classifier

print('Modelo selecionado é {} com acurácia {}'.format(name_selected, top_accuracy))
```

Fonte: elaborado pelos autores

Para nos dar capacidade de replicação do processo de vetorização e classificação das frases, mesmo após todo programa descarregado da memória, armazenamos o modelo de classificação selecionado e o vetorizador treinado, pela biblioteca pickle, responsável por serialização e deserialização de objetos em Python, gerando arquivo e permitindo o resgate.

Tabela 5.8 - Gerar os arquivos com configurações para uso posterior

```
# Importação de biblioteca de serialização
import pickle

# Geração do arquivo 'model_selected.sav' com o modelo e vetorizador como 'vectorizer.sav'
pickle.dump(model_selected, open('./data/model_selected.sav', 'wb'))
pickle.dump(vectorizer, open('./data/vectorizer.sav', 'wb'))
```

Fonte: elaborado pelos autores

5.4. Disponibilizando o serviço

Nessa etapa do trabalho tínhamos quatros artefatos, listados na tabela Tabela 5.9, com modelos treinados para gerar e vetorizar os lexemas, realizar a classificação e a listagem das palavras recomendadas.

Tabela 5.9 - Artefatos para disponibilização de serviços

Nome	Descrição
palavras.csv	Coleção das palavras permitidas com o nome da empresa e produtos
pt_core_news_lg	Coleção de modelos treinados da biblioteca spaCy para filtro das classes gramaticais e conversão em lexemas
vectorizer.sav	Gerador dos vetores com os lexemas
model_selected.sav	Modelo treinando e selecionado com a melhor acurácia

Fonte: elaborado pelos autores

Com o uso da biblioteca FastAPI em Python, código apresentado no Anexo 7, criarmos uma interface de comunicação (API) com nosso serviço, para responder a requisição com uma frase dentro de um objeto como no exemplo: `{ “frase”: “Gostoso demais” }`.

A API aplica, detalhado no Anexo 7, os modelos filtro da classe gramática e lematização da frase, vetorização, aplicação do algoritmo de classificação e indicação das palavras não recomendadas. A resposta é enviada por um objeto, modelo descrito na Tabela 5.10, contendo pontuações de palavras para serem evitadas, bem como a classificação realizada pelo modelo.

Tabela 5.10 - Dicionário de dados para a resposta da API

Nome	**Descrição**	**Tipo**
Frase	Texto original com marcação nas palavras não indicadas	Texto
Classificação	Classificação de Positiva ou Negativa	Texto
Data_Resposta	Data e Hora do envio da resposta	Texto

Fonte: elaborado pelos autores

Essa chamada pode ser realizado por qualquer ferramenta que possibilita uma chamada a uma REST API com um o método POST, como o programa curl pela execução apresentada na Tabela 5.11.

Tabela 5.11 - Seleção do modelo com melhor acurácia entre os resultados

```
curl --location --request POST 'http://127.0.0.1:8000' \
 --header 'Content-Type: application/json' \
 --data-raw '{ "frase": "Gostoso demais" }'
```

Fonte: elaborado pelos autores

6. Resultados e discussões

O presente trabalho permitiu o desenvolvimento da abordagem Analizo, a qual possui a aplicação de um conjunto de algoritmos de processamento de linguagem natural e os algoritmos de Floresta Aleatória (*Random Forest*), Árvore de Decisão (*Decision Tree*), Multi-Layer Perceptron Classifier e Multinomial Naive Bayes para classificação que dentro do conjunto de 100 sentenças divididas igualmente em positivas e negativas, geradas para o estudo, fazendo uso do lexema e seleção de algumas classes gramaticais, o pontuou como uma acurácia de 0.7333 a 0.8000.

Dentro do proposto e da baixa quantidade das sentenças presentes na fonte de dados a abordagem conseguiu uma boa margem de acertos, mostrando que seu uso é aconselhável com o treinamento adequado.

7. Considerações finais

Com a atualização da fonte de dados com as sentenças e classificações corretas, bem como a possibilidade de adicionar modelos algoritmos, sendo necessário um estudo para encontrar os mais indicados, mantendo em dia constantemente os classificadores podemos sempre garantir que a mensagem cada vez mais próxima da correta está sendo compartilhada, para ampliar a funcionalidade da ferramenta a incorporação de correções ortográficas e sugestões de continuidade de texto tende a ampliar sua usabilidade, bem como sua incorporação nas ferramentas de uso diário dos usuários.

8. Referências

ALVES, Igor. **Lemmatization vs. stemming: quando usar cada uma?**. 2021. Disponível em: <https://www.alura.com.br/artigos/lemmatization-vs-stemming-quando-usar-cada-uma>. Acesso em 17 de junho de 2022;

BECHARA, Evanildo. **Moderna Gramática Portuguesa**, 37ª edição atualizada pelo novo Acordo Ortográfico. Rio de Janeiro, RJ: Editora Nova Fronteira, 2009;

BORGES JUNIOR, Sergio Ricardo et al. **SEnsembles - uma abordagem para melhorar a qualidade das correspondências de instâncias disjuntas em estudos observacionais explorando características idênticas e ensembles de regressores**. 2016;

HUDSON, Richard. **spaCy: Linguistic Features**. 2022. Disponível em: <https://spacy.io/usage/linguistic-features>. Acesso em 01 de setembro de 2022;

IME-USP. Instituto de Matemática e Estatística da Universidade de São Paulo. **Lista de todas as palavras do português brasileiro em codificação UTF-8**. Disponível em: <https://www.ime.usp.br/~pf/dicios/br-utf8.txt>. Acesso em 16 de março de 2022;

PROVOST, Foster, FAWCETT, Tom. **Data Science para Negócios**. Rio de Janeiro, RJ: Alta Books, 2016;

RAMÍREZ, Sebastián. **FastAPI framework, high performance, easy to learn, fast to code, ready for production**. 2022. Disponível em: <https://fastapi.tiangolo.com/>. Acesso em 01 de setembro de 2022;

RIBEIRO, Julio. **Marketing de Atitude: como fazer suas equipes e seus clientes gostarem de você**. São Paulo: Dash Editora, 2012;

SAMPAIO, Rafael. **Propaganda de A a Z: como usar a propaganda para construir marcas e empresas de sucesso**. Rio de Janeiro: Elsevier, 2013;

SILVA, Leandro Augusto da; PERES, Sarajane Marques; BOSCARIOLI, Clodis. **Introdução à Mineração de Dados: Com aplicações em R**. Rio de Janeiro: Editora Elsevier, 2016;

VANDERPLAS, Jake. **Python Data Science Handbook**. Sebastopol: O'Reilly Media, 2017.

Anexos

Anexo 1 - Geração da listagem de palavras aceitas

```
import pandas as pd

# Carregar listagem com a base de palavras do
# Instituto de Matemática e Estatística Universidade de São Paulo (IME-USP)
palavras = pd.read_csv('https://www.ime.usp.br/~pf/dicios/br-utf8.txt', header=None)[0]

# Adicionar o nome da empresa
nomes = pd.DataFrame(['Lutscher'])

# Adicionar os produtos do catálogo da empresa
produtos = pd.DataFrame(['Chupscher', 'DulCacau', 'Marsher'])

# Limpeza de palavras que devem ser evitadas
palavras_limpas = palavras[
    # Remoção de referências com diabo ou diabetes
    ~palavras.str.startswith('diab') &
    # Remoção de referências com tristeza
    ~palavras.str.startswith('trist') &
    # Remoção de referências com diabo ou diabetes
    ~palavras.str.startswith('amarg')
]

# Adicionar na empresa para a listagem de palavras
nova_listagem = pd.concat([palavras_limpas, nomes, produtos])

# Salvar listagem de palavras tratadas
nova_listagem.to_csv('./data/palavras.csv', header=False, index=False)
```

Fonte: elaborado pelos autores

Anexo 2 - Texto e Classificação das frases usadas para o treinamento e teste do aprendizado de máquina

Texto	**Classificacao**
Com diversidade e alegria para toda família	Positiva
Uma colorida e irresistível variedade	Positiva
O que posso fazer por você?	Positiva
Aprecie as pequenas coisas, como um brigadeiro bem doce	Positiva
Eu quero a vida com muita cobertura e muito recheio	Positiva
O ingrediente mais importante é o amor, e aqui é o que mais usamos!	Positiva
A felicidade mora em um doce	Positiva
Felicidade é só uma questão de chocolate	Positiva
Confeitar é um jeito de amar	Positiva
Com açúcar e com gentileza, fiz o seu doce amado	Positiva
Escolha a sobremesa do seu dia	Positiva
Que a nossa doçura seja contagiante!	Positiva
De todos os ingredientes da vida o amor é o mais doce	Positiva
Delícia mesmo é adoçar a vida ao lado de quem a gente ama	Positiva
Viva de bem; passe sempre pela confeitaria	Positiva
Ande sempre leve, tenha pirulitos no bolso	Positiva
A felicidade é um marshmallow	Positiva
Melhor que marshmallow, só beijo com sabor de caramelo	Positiva
Aqui tem frutas embaladas em sonhos	Positiva
Temos o seu sabor preferido	Positiva
Permita-se a alegria de nossos sabores	Positiva
Leve um pacote de felicidade	Positiva
Os mais macios e gostosos	Positiva
Direto da embalagem ou no espeto, são ideais para assar	Positiva
Em volta da fogueira ou chocolate, da maneira que você quiser	Positiva
Tudo que é bom é doce	Positiva
Sua festa ainda mais alegre com a nossa ajuda	Positiva
Deixe sua comemoração mais colorida e doce com nossos kit festas	Positiva
Festança para sua casa	Positiva
Produtos coloridos e deliciosos	Positiva
Trazer alegria para crianças e adultos	Positiva
Sinta a refrescância	Positiva
Um recheio super macio e cremoso envolvido em uma cobertura fina e consistente	Positiva
Descobrir é delicioso	Positiva
Sonhando com um doce	Positiva

Para adoçar ainda mais a sua compra	Positiva
Doce ou travessuras	Positiva
Novidade que dá gosto	Positiva
Vontade de compartilhar	Positiva
Espalhar doces gentilezas	Positiva
Trouxe um presentinho para você	Positiva
Bora adoçar?	Positiva
Deixe a gentileza gerar o mundo mais doce	Positiva
Assista uma série com alguém especial e compartilhe um marshmallow	Positiva
Expressar o quanto uma pessoa é importante para você	Positiva
O gostoso da vida está nos pequenos gestos	Positiva
Eu queria pela primeira vez de novo	Positiva
O mundo mágico da alegria	Positiva
Um parceria de milhões de sabores	Positiva
Seu melhor momento	Positiva
Doce mania que vicia	Negativa
Não podemos te ajudar no momento	Negativa
Brigadeiro não engorda, preenche o corpo de felicidade	Negativa
Briga? Só se for brigadeiro!	Negativa
Diga não ao controle	Negativa
Sem dieta, sem balança, sem preocupação	Negativa
Diga não à tristeza, preocupe-se menos com a balança	Negativa
Prejudicial aqui é só o seu recalque	Negativa
Experiência ruim nunca mais	Negativa
De madrugada é mais gostoso	Negativa
Maluquice e descontrole todo dia	Negativa
Fruta é bom, mas já experimentou nossos doces	Negativa
Não fique na vontade, abra um doce	Negativa
Vontade gostosa que não passa	Negativa
Alegria toda hora	Negativa
Sorria, tem alegria todo dia	Negativa
Não tenho como compartilhar com você	Negativa
Itens essenciais para seu dia	Negativa
Afogar minhas inseguranças em doce	Negativa
Vontade gigante de doce não sai de mim	Negativa
Deliciar um doce é moleza demais	Negativa
Gaveta cheia de doces é um ótimo refúgio para o antiestresse	Negativa
Um doce toda hora vale ouro	Negativa
A verdade é que um pacote de doce é tão delicioso que acaba em um piscar de olhos	Negativa
Ansioso? Um doce resolve	Negativa
A paixão por doce cresce a cada dia	Negativa
Competidor por doce é gostoso demais	Negativa
No hora da fome pode sair umas combinações meio diferentes	Negativa
Nessa competição, passe para o lado do doce da força	Negativa
Todos os momentos deveria ter um doce	Negativa
A companhia doce para a vida	Negativa
Você não vai querer compartilhar o pacote	Negativa
O azar passa longe de quem tem doce	Negativa
Seu estômago quer doce	Negativa
Lista de coisas que melhoram o meu humor	Negativa
Vida sem doce não vale a pena	Negativa
Não deixe seu pacote perdido por aí	Negativa
Como você devora seus marshmallow	Negativa
Durante horas um pacote como companhia	Negativa
Esconda seus pacotes, doce não se compartilha	Negativa
Sempre existe um pacote maior	Negativa
Você pode comer o pacote todo, sim	Negativa
Não deixe para amanhã o que você pode comer hoje	Negativa
Como está seu humor hoje? Deixe um doce melhorar	Negativa
Tinha um problema, agora tem um pacote de doce	Negativa
Café da manhã, almoço ou jantar? Não importa toda hora é hora de doces	Negativa
Se o que não vivo sem? Um café e um bom doce	Negativa
Doce no final de semana? Claro e na semana toda também	Negativa
Pacote de doce é companhia para a vida toda	Negativa
Melhor que fruta? Somente um doce de sobremesa	Negativa

Fonte: elaborado pelos autores, juntamente com coleta e adaptação nas mídias da Docile Alimentos Ltda, Fini Comercializadora Ltda e Haribo Brasil Indústria e Comércio de Produtos Alimentícios Ltda

Anexo 3 - Declaração da função de filtro de classes gramaticais e conversão para lexemas

```
# Importar biblioteca de processamento linguagem natural
import spacy
# Carregar tokenização e treinamento em português
nlp = spacy.load('pt_core_news_lg')

def lematizar(list):
  # Definição de classes gramaticais Substantivos, Verbos, Advérbios e Adjetivos
  classeGramatical = ['NOUN', 'VERB', 'ADV', 'ADJ']

  # Criar listagem de lematização
  lema = []

  # Loop para criação das frases lematizadas
  for frase in list:
      doc = nlp(frase)
      spacy_lema = [token.lemma_ for token in doc if token.pos_ in classeGramatical]
      lema.append(' '.join(spacy_lema))

  return lema
```

Fonte: elaborado pelos autores

Anexo 4 - Gerar vetorização das frases convertidas seus lexema

```
# Alimentar a lista e dataset com a lematização dos corpos
df_frases['Lematizacao'] = lematizar(df_frases['Texto'].tolist())

# Importação da biblioteca para vetorização
from sklearn.feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer()
vector_lematizacao = vectorizer.fit_transform(climatizacao)

# Agregação da vetorização para a base
df_frases['Vetorizado'] = vector_lematizacao.toarray().tolist()
```

Fonte: elaborado pelos autores

Anexo 5 - Separação e geração das frases com classificação para treinamento e geração

```
# Biblioteca de Separação de Treino/Teste
from sklearn.model_selection import train_test_split

# Separação das frases para treino do modelo e geração de métricas
## Negativas
negativas = df_frases.query("Classificacao == 'Negativa'")
vector_neg_lematizacao = negativas['Vetorizado'].tolist()
classificacoes_neg = negativas['Classificacao'].tolist()

vector_neg_treino, vector_neg_teste, classificacoes_neg_treino, classificacoes_neg_teste =
train_test_split(vector_neg_lematizacao, classificacoes_neg, test_size=0.3, random_state=42)

## Positivas
positivas = df_frases.query("Classificacao == 'Positiva'")
vector_pos_lematizacao = positivas['Vetorizado'].tolist()
classificacoes_pos = positivas['Classificacao'].tolist()

vector_pos_treino, vector_pos_teste, classificacoes_pos_treino, classificacoes_pos_teste =
train_test_split(vector_pos_lematizacao, classificacoes_pos, test_size=0.3, random_state=42)

# Concatenar listagem para treino e teste
vector_treino = vector_neg_treino + vector_pos_treino
vector_teste =  vector_neg_teste + vector_pos_teste
classificacoes_treino = classificacoes_neg_treino + classificacoes_pos_treino
classificacoes_teste =  classificacoes_neg_teste + classificacoes_pos_teste
```

Fonte: elaborado pelos autores

Anexo 6 - Declaração das funções de análise de frase e modelo

```
# Biblioteca de Métricas
from sklearn import metrics

# Frases de amostra  que estão no dicionário
frases_amostra = [
    'Sua confeitaria a todo momento',
    'Sobremesa para alegrar o dia',
    'Brigar por doce, pode'
  ]

# Coleção de métricas do modelo
resultado_metricas = pd.DataFrame(data={
    'name': [], 'accuracy': [], 'precision': [],
    'recall': [], 'F1 score': [], 'classifier': []})

# Geração da análise por frases
def analizar_frase(texto, modelo):
  # Lematizar o texto recebido
  lematizada = lematizar([texto])

  # Vetorizar o texto lemaizado
  vetorizada = vectorizer.transform(lematizada).toarray()

  # Realizar predição modelo oferecido
  predicao = modelo.predict(vetorizada)

  return predicao[0]

# Apresentar análise do modelo
def analizar_modelo(name, modelo):
  # Gerar fit do modelo para predições
  modelo.fit(vector_treino, classificacoes_treino)

  # Loop de apresentação dos conceitos
  for frase in frases_amostra:
    analise = analizar_frase(frase, modelo)
    print(frase + ' | ' +  analise)

  # Cálculo das métricas
  predicoes_teste = modelo.predict(vector_teste)

  accuracy = metrics.accuracy_score(classificacoes_teste, predicoes_teste)
  precision = metrics.precision_score(
      classificacoes_teste, predicoes_teste, average='weighted')
  recall = metrics.recall_score(
      classificacoes_teste, predicoes_teste, average='weighted')
  f1_score = metrics.f1_score(
      classificacoes_teste, predicoes_teste, average='weighted')

  print("")
  print("Acurária:", "{:.4f}".format(accuracy))
  print("Precisão:", "{:.4f}".format(precision))
  print("Recall Score:", "{:.4f}".format(recall))
  print("F1 Score:", "{:.4f}".format(f1_score))

  df_metric = pd.DataFrame(data={
      'Name': [name], 'Accuracy': [accuracy], 'Precision': [precision],
      'Recall': [recall], 'F1 Score': [f1_score], 'Classifier': [modelo]})

  return resultado_metricas.append(df_metric)
```

Fonte: elaborado pelos autores

Anexo 7 - Disponibilização do serviço de análise pela API

```
# Importação de biblioteca de serialização
import pickle
# Importação de biblioteca para lidar com APIs
from fastapi import FastAPI
# Biblioteca de validação de tipos
from pydantic import BaseModel
# Biblioteca para possibilitar uso nulo para tipos
from typing import Union
# Importar biblioteca de processamento linguagem natural
import spacy
# Importar Pandas - biblioteca para manipulação de datasets
import pandas as pd
# Bilbioteca de
from datetime import datetime

# Iniciar instância para lidar com API
app = FastAPI()

# Carregar tokenização e treinamento em português
nlp = spacy.load('pt_core_news_lg')

# Carregar o arquivo com o modelo gerado, vetorizador e dicionário de palavras
loaded_model = pickle.load(open('./data/model_selected.sav', 'rb'))
vectorizer = pickle.load(open('./data/vectorizer.sav', 'rb'))
dicionario = pd.read_csv('./data/palavras.csv', header=None)

def lematizar(list):
    # Definição de classes gramaticais Substantivos, Verbos, Advérbios e Adjetivos
    classe_gramatical = ['NOUN', 'VERB', 'ADV', 'ADJ']
    # Criar listagem de lematização
    lema = []
    # Loop para criação das frases lematizadas
    for frase in list:
        doc = nlp(frase)
        spacy_lema = [
            token.lemma_ for token in doc if token.pos_ in classe_gramatical]
        lema.append(' '.join(spacy_lema))
    # Retornar frase lematizzada
    return lema

def analizar_frase(texto, modelo):
    # Lematizar o texto recebido
    lematizada = lematizar([texto])
    # Vetorizar o texto lemaizado
    vetorizada = vectorizer.transform(lematizada).toarray()
    # Realizar predição modelo oferecido
    predicao = modelo.predict(vetorizada)
    # Retornar a análise da frase
    return predicao[0]

def marcar_palavras_desaconselhadas(texto):
    result = []
    for word in texto.split():
        if word.lower() in dicionario.values:
            result.append(word)
        else:
            result.append('<notfound>' + word + '</notfound>')
    return " ".join(result)

class Item(BaseModel):
    frase: str
    classificacao: Union[str, None] = None
    data_resposta: Union[datetime, None] = None

@app.get("/")
def check_healthy():
    return "Funcionando"

@app.post("/")
def check_phase(item: Item):
    item.classificacao = analizar_frase(item.frase, loaded_model)
    item.data_resposta = datetime.today()
    item.frase = marcar_palavras_desaconselhadas(item.frase)

    return item
```

Fonte: elaborado pelos autores