Análise de sentimentos – Utilizando Python para construir e treinar modelos de machine learning

Ao final deste texto, você deve apresentar os seguintes aprendizados:

* Descrever processos de implementação de algoritmos de análise de sentimentos
* Programar algoritmos de análise de sentimentos.
* Analisar os resultados gerados por algoritmos de análise de sentimentos.

**Capítulo 1. Descrever processos de implementação de algoritmos de análise de sentimentos**

O aprendizado de máquina (do inglês, *machine learning*) é uma área da Ciência da Computação ligada à Inteligência Artificial, responsável por treinar sistemas e dispositivos a executarem tarefas que geralmente seriam realizadas por humanos. Neste sentido, muitas técnicas são utilizadas para que estes treinamentos sejam efetivos. O propósito do *machine learning* é humanizar os processos, aproximando cada vez mais os dispositivos das capacidades cognitivas humanas.

Uma destas técnicas está pautada na construção de algoritmos de análise de sentimentos que, dentre suas muitas aplicabilidades, auxiliam empresas a entenderem o quão satisfeitos estão os clientes em relação aos seus produtos e/ou serviços, identificando e categorizando computacionalmente as opiniões expressas em um conjunto de dados textual, a fim de determinar se a atitude do escritor em relação a um tópico, produto, etc. é positivo, negativo ou neutro.

Portanto, neste capítulo, você vai aprender os processos de implementação de um algoritmo de análise de sentimentos, além de programá-lo utilizando a linguagem Python e, por fim, poderá analisar os resultados gerados a partir desta implementação.

Neste capítulo falaremos sobre Naive Bayes, citado por Martin (2019), abordaremos também conceitos de filtros em Bag of Words, que significa mineração de palavras, segundo o mesmo autor. No final falaremos da matriz de confusão, que avalia a qualidade do código.

As Sete Etapas no Processo de Análise de Sentimentos

A figura a seguir exemplifica as sete etapas no processo de análise de sentimentos. Após a explicação das etapas 1 e 2, vamos implementar um algoritmo de análise de sentimentos utilizando a linguagem Python e você poderá observar em detalhes a descrição do que cada linha de código faz. No exemplo, utilizaremos o Google Colab.

Você pode usar a IDE PyCharm ao invés de Spyder, e em ambas as IDE podem mudar a cor do fundo. O PyCharm, pode ser baixado através do link

**Emoções e sentimento**

Em primeiro lugar se faz necessário entender alguns conceitos relacionados a sentimentos. De acordo com Martin (2019), o primeiro conceito é a emoção, ela é um reflexo de eventos internos relevantes. Embora o autor supracitado fale sobre oito emoções, é mais comum utilizarem nas classificações seis, sendo os sentimentos: Bravo, Triste, Alegre, Medroso, Envergonhado, Orgulhoso, Entusiasmado, Desesperado e os mais abordados são: Tristeza, Raiva, Surpresa, Medo, Desgosto e Alegria.

O autor ainda cita oscilações de humor como eventos difusos sem causa aparente, como exemplo, temos os seguintes humores: Alegre, Sombrio, Irritável, Apático, Deprimido, Dinâmico. Além disso, ele cita também algumas atitudes, traços e posturas, que trazem mais variações, mas nos ateremos às seis emoções citadas. Diante de tudo isso, se faz necessário identificar palavras e emoções atribuídas a estes sentimentos, para tanto observe os tópicos a seguir:

1. Palavras relacionadas a Alegria: Animação, belo, anseio, admirar e etc.
2. Palavras relacionadas a Desgosto: Abominável, enjoo, repugnante, chatear e etc.
3. Palavras relacionadas a Medo: Brutal, calafrio, espantoso, estremecer, escuridão e etc.
4. Palavras relacionadas a Raiva: Aversão, aborrecer, doido, diabólico e etc.
5. Palavras relacionadas a Surpresa: Expectativa, fantástico, encantamento, prodígio e etc.
6. Palavras relacionadas a Tristeza: Abatido, contrito, culpa, abandonar e etc.

Portanto, a primeira etapa no processamento de emoções é listar as seis emoções citadas e definir quais palavras entrarão em cada emoção*.*

**Seleção de textos**

Em uma construção civil, os revestimentos e materiais reais apresentam diversos aspectos e características que os tornam únicos, por exemplo, em uma parede de tijolos aparentes é fácil perceber que o tijolo é um material mais rugoso, irregular e que entre cada tijolo há um espaçamento de massa, que no modo de instalação correta, essa massa fica mais recuada. Quando pensamos em uma textura de tijolo para aplicar em uma superfície, que será a parede do nosso ambiente, temos que buscar uma imagem que apresente as características mais próximas possíveis do objeto real.Neste sentido, é importante buscar uma imagem de qualidade em bancos de imagens onde podemos controlar a qualidade da imagem buscada.

Para fazer mineração de textos e análise de emoções, é comum usar a linguagem NTLK, de python, a Natural Language Toolkit é uma biblioteca de python destinada a fazer processamento de linguagem natural.

Da mesma forma, antes de aplicar o NLTK, se faz necessário conhecer alguns conceitos da mineração de emoções em textos. No cenário atual, existe um prato cheio para a mineração de textos, pois podemos encontrar opiniões de pessoas postadas nas mais diversas redes sociais em forma de palavras, bem como no google e nos sites e mecanismos de pesquisa. Sabe-se por exemplo, que o Google Trends, analisa a estatística das palavras mais pesquisadas no google, e que os criadores de site utilizam meta *tags* para otimizar pesquisas orgânicas.

Vamos abordar três conceitos, a saber: Token, Documentos e Coleção. No universo da mineração de textos, palavras são chamadas por *token*. Elas refletem sentimentos e emoções e trazem verdades sobre opiniões, apreços ou rejeição por determinado post, produto ou notícia. Você sabe o que são documentos? Em mineração de texto, um documento é um arquivo único, podendo variar tanto entre e-mails, artigos web, opiniões, posts e comentários, quanto entre avaliações de cinco, quatro, três estrelas e etc. Um texto, seja post, seja artigo ou e-mail é um documento. Vários textos formam uma chamada coleção. Agora, imagine que neste universo de palavras, documentos e coleções, se faz necessário identificar emoções. Para isso, se faz necessário alguns passos de tratamento de textos, feitos quase sempre em python, através da importação do conjunto de bibliotecas do NLTK, que veremos mais adiante.

Em um documento, ao usar NLTK, deve-se excluir palavras “stop word”, ou seja, palavras sem significado, como “e”, “ou” e etc. Além disso, é importante que as palavras estejam na mesma formatação, ou seja, em letras minúsculas. Antes de tudo, ao analisar um texto é necessário calcular a frequência, denominada TF - taxa de frequência (FOSTER, FAWCETT, 2016). Cuja fórmula está abaixo:

Para exemplificar, vamos usar um exemplo de mineração de emoções, onde em 1 documento analisado: A avaliação “Bom atendimento” aparece 3 vezes, “Agilidade” aparece 2 vezes, e “Qualidade” aparece 1 vez. Para calcular a taxa de frequência das palavras fazemos um cálculo de normalização:

Agora, em outro exemplo, na coleção inteira de documentos analisados, existem 8 documentos analisados. Ao analisar a frequência dessas palavras na coleção inteira, percebe-se que nem sempre uma emoção muito presente em um texto está na maioria dos textos, tem-se: Na coleção em questão: “Bom atendimento” (ocorre em 5 documentos), “Agilidade” (ocorre em 1 documentos), “Qualidade” (ocorre em 8 documentos). Assim, é necessário calcular a quantidade de repetição do corpo completo.

* Índice de avaliação de Bom atendimento = 3/3 x 5 = 5
* Índice de Importância de Agilidade = 2/3 x 1 = 0,66
* Índice de Importância de Qualidade 1/3 X 8 = 2,66

~~Em uma outra análise, que pode ser considerada melhor ainda, quando se aplica o conceito de “clusterização” de público alvo, antes de aplicar cálculos, por exemplo, quando se divide o conjunto de documentos em três coleções de opiniões, uma coleção de jovens entre 20 e 30 anos, outra coleção entre adultos de 30 a 40 anos e outra coleção entre adultos de 40 a 50 anos.~~

Usar cálculos de palavras mais citadas ajuda a saber o que de fato é importante, além disso, existe um conceito chamado clusterização, que auxilia na venda de produtos combinados ou oferta por público-alvo. Em clusterização por produtos combinados, por exemplo, sabendo que dentre todos que falam de pipoca em um texto, 80% falam também de refrigerante, pode-se identificar que é possível fazer um combo para promoções. Em clusterização por público-alvo, por exemplo, pode-se dividir as análises de textos por clusters por idade, avaliando uma coleção de textos de jovens entre 20 e 30 anos, outra coleção entre adultos de 30 a 40 anos e outra coleção entre adultos de 40 a 50 anos, pode-se descobrir semelhanças úteis entre as opiniões de cada grupo, que auxiliam na formação de futuras propagandas.

**Capítulo 2. Programar algoritmos de análise de sentimentos.**

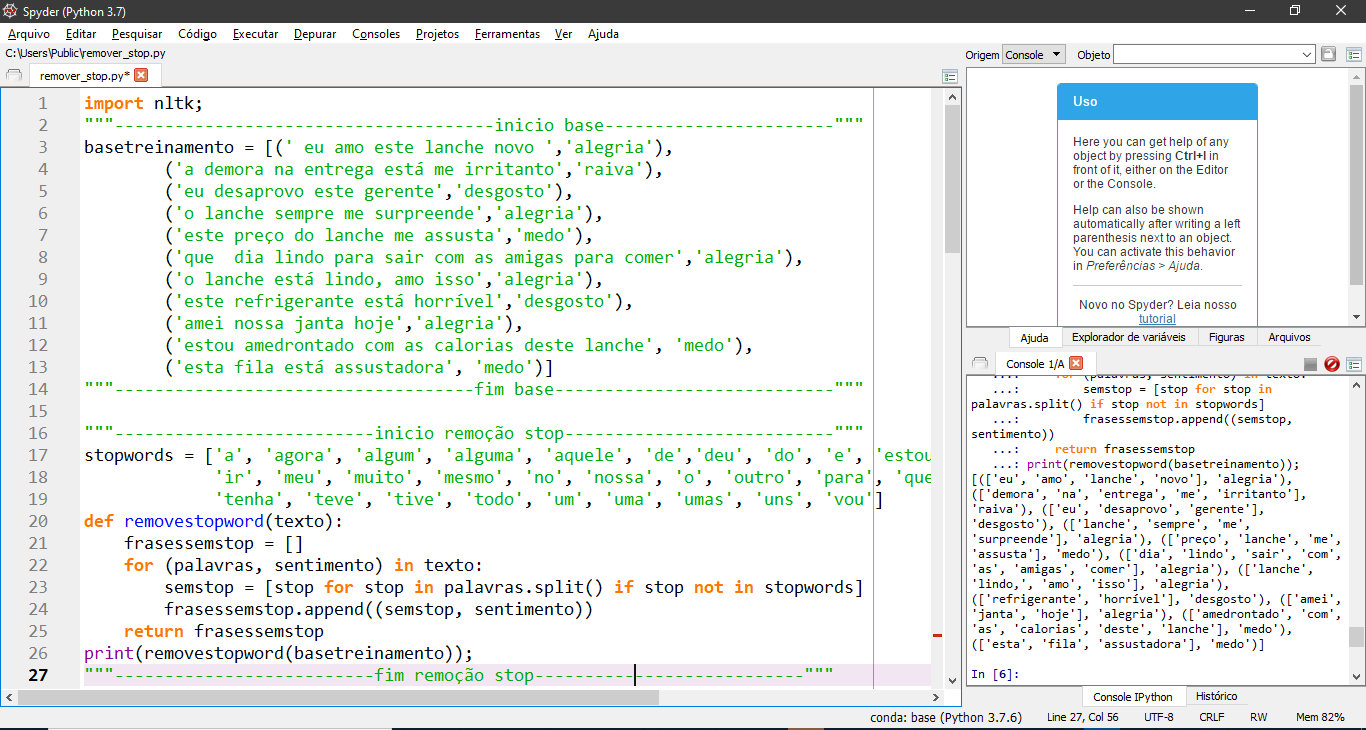
**Remoção das *stopwords***

Agora entraremos na parte prática e, abra o Google Colabe e siga as etapas explicadas a seguir.

O material base desta disciplina estará no item Saiba+. **O material referente a construção completa do Naive Bayes, começando na remoção de Stop Words e chegando no Naive Bayes arquivo chama mineracao.py.**  A etapa referente ao código completo com matriz de confusão está no item Saiba+ e chama **codigo\_geral.py**

No exemplo a seguir, foi feita uma pesquisa em 10 posts de comentários sobre uma rede de *fast food*, e identificou-se que as frases contidas na variável basetreinamento, se repetiram de uma forma ou de outra, com mais frequência em toda a coleção. Nosso objetivo aqui, é mostrar como remover *stopwords* da base de treinamento de emoção, por isso, leia os códigos, mas antes de colocar em prática, observe a Figura 1, pois ela mostra no canto esquerdo o código inteiro e no canto direito inferior o resultado. Para executar cada comando do código, selecione a frase completa, cole no Colab e aperte **Alt Enter**. No nosso exemplo usamos o Spyder, mas use o Colab.

Observe bem que a Figura 1 mostra o resultado no canto direito sem *stopwords*. Faça um teste, ao copiar a função que inicia com “def” e mantenha o mesmo espaçamento (indentação).



Função que remove stopwords

Resultado da linha 26, que exibe a base sem stopwords

Definição de stopwords

Criação de frases que apareceram frequentemente na coleção de post;

Figura 1. Como remover *stopwords*

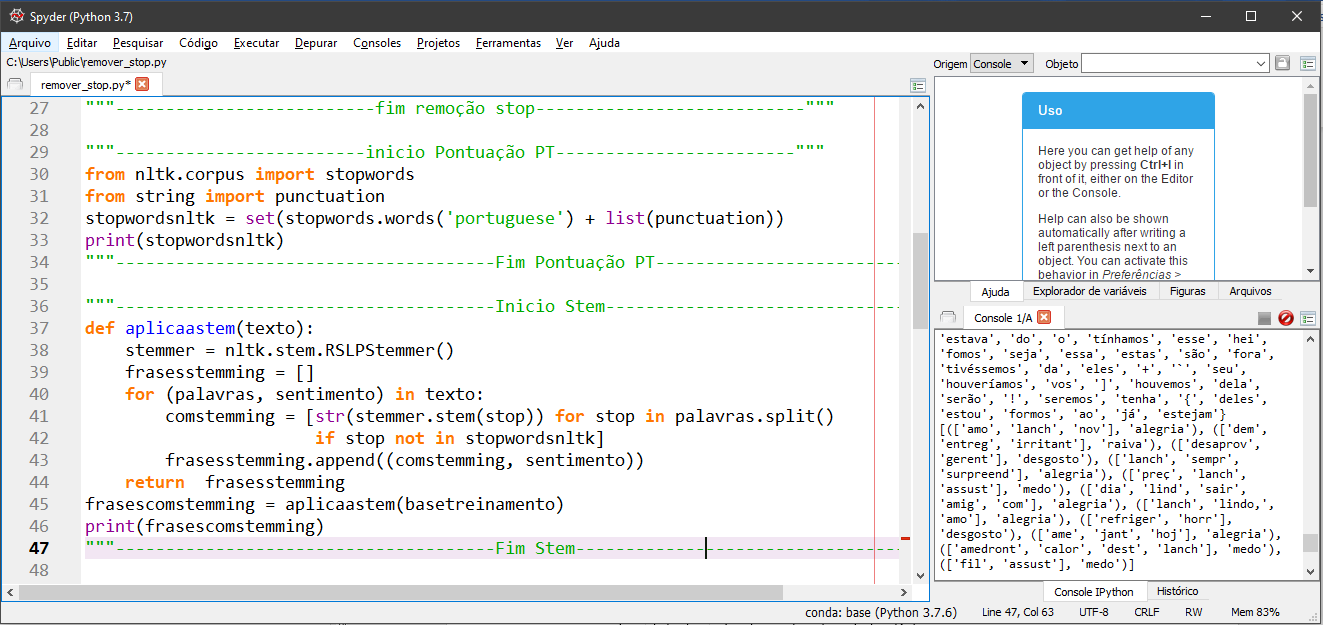
Fonte: Do Autor - 2020

***Stemização***

Agora falaremos de *Stemização*, isto é, a remoção de sufixos, manteremos apenas os prefixos das palavras, assim, o prefixo “am” por exemplo, das palavras amar, amou e etc, serão igualmente calculados.

Novamente, vamos explicar passo a passo na figura 2:

Aqui importamos duas bibliotecas que auxiliam na tradução da sintaxe para português:



A função descrita abaixo removerá o sufixo (fará o stemm) das palavras através da divisão (split) e da remoção do sufixo

Colocamos na variável stopwordsnltk as stopwords em português

Figura 2. Stemização

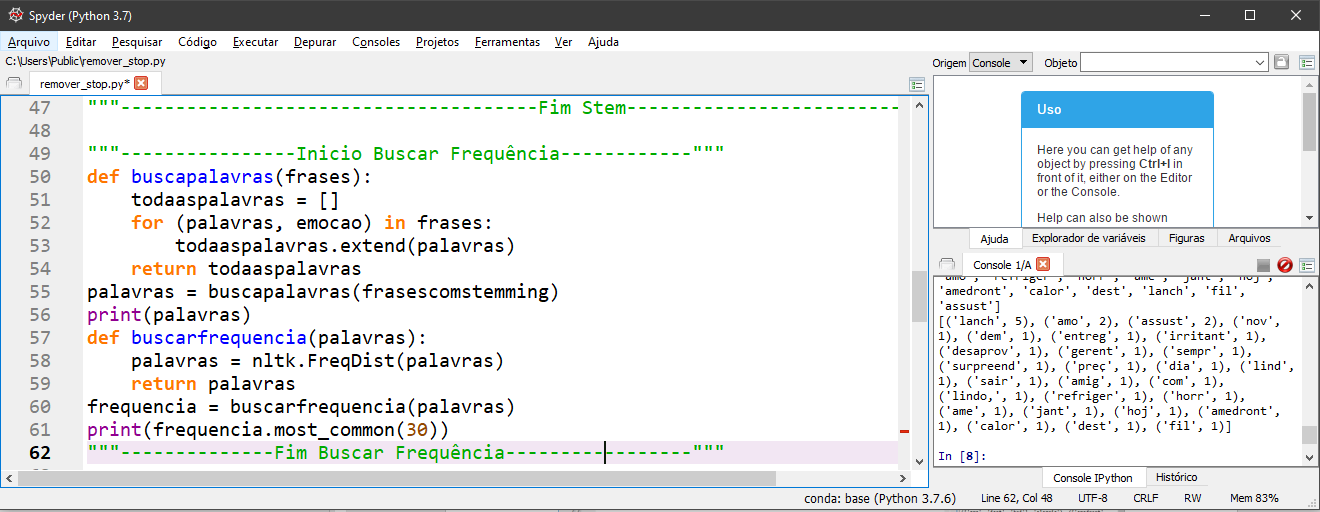
Palavras sem sufixo, resultado da linha 46

Fonte: Do autor - 2020

Objetivo 1 - Descrever processos de implementação de algoritmos de análise de sentimentos

**Treinar a base para o aprendizado de máquina**

Nesta etapa, executaremos alguns passos. O primeiro deles é contar quantas vezes aparecem as 30 palavras principais. Isso é feito através da palavra nativa chamada FreqDist.



Primeiro note que vamos dissociar as palavras das emoções e pagar só as palavras

Perceba que usaremos o FreqDist para calcular a frequência, depois imprimiremos na tela stopwords em português

Resultado das 30 mais comuns

Figura 3. Cálculo de frequência de palavras

Fonte: Do autor - 2020

Até aqui, com estes dados, já é possível criar uma tabela, relacionando palavras com sentimentos, a fim de proceder com tomadas de decisões em relação a filtros de bancos de dados. Pode-se navegar em meio a um banco de dados textual, e procurar as palavras contidas na primeira linha da tabela, para agilizar pesquisas de sentimentos. É recomendável fazer a remoção de *stopwords*, a *stemização* e o levantamento de frequência da maior quantidade de linhas, para então criar uma tabela mais complexa do que está embaixo.

Do autor - 2020

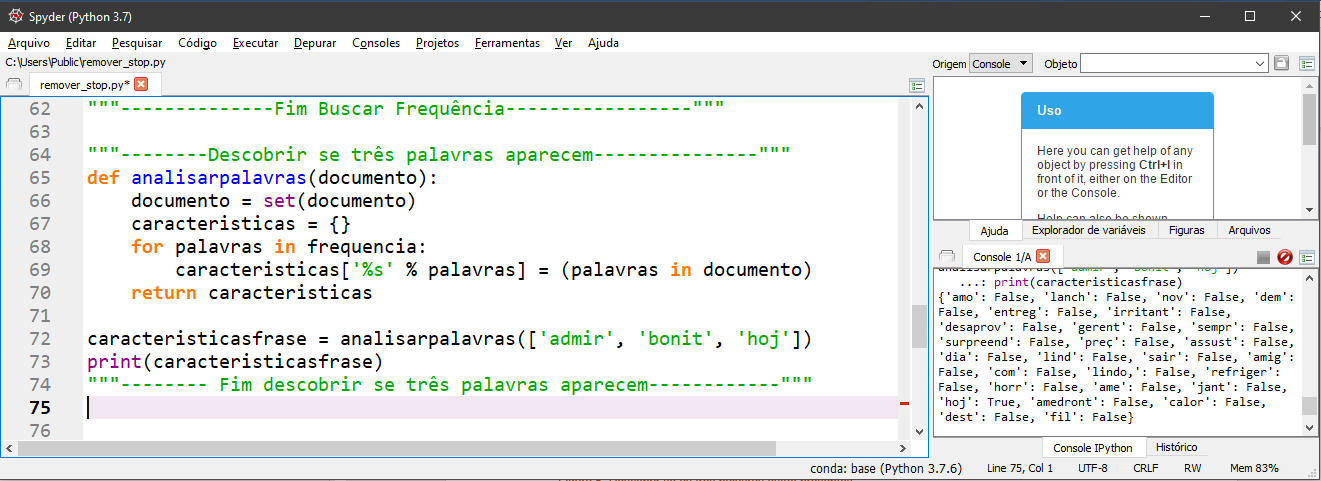
Analisando palavras para pesquisas em bancos de dados

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Lanche | Amo | Assus | Entreg | Irritant | Dem | Surpreend | Desaprov |
| Alegria | 3 | 2 |  |  |  |  | 1 |  |
| Surpresa | 2 |  |  |  |  |  |  |  |
| Medo |  |  | 2 |  |  |  |  |  |
| Tristeza |  |  |  |  |  |  |  |  |
| Desgosto |  |  |  |  |  |  |  | 1 |
| Raiva |  |  |  | 1 | 1 | 1 |  |  |

**Aplicar o teste de palavras**

Vamos supor que você precisa saber se uma determinada palavra, ou algumas palavras, estão no texto, e ainda, quais sentimentos ela reflete. Você quer saber se as palavras, em seus radicais, “admir”, “hoje’ e “bonit” estão no texto, e quais sentimentos estão associados.

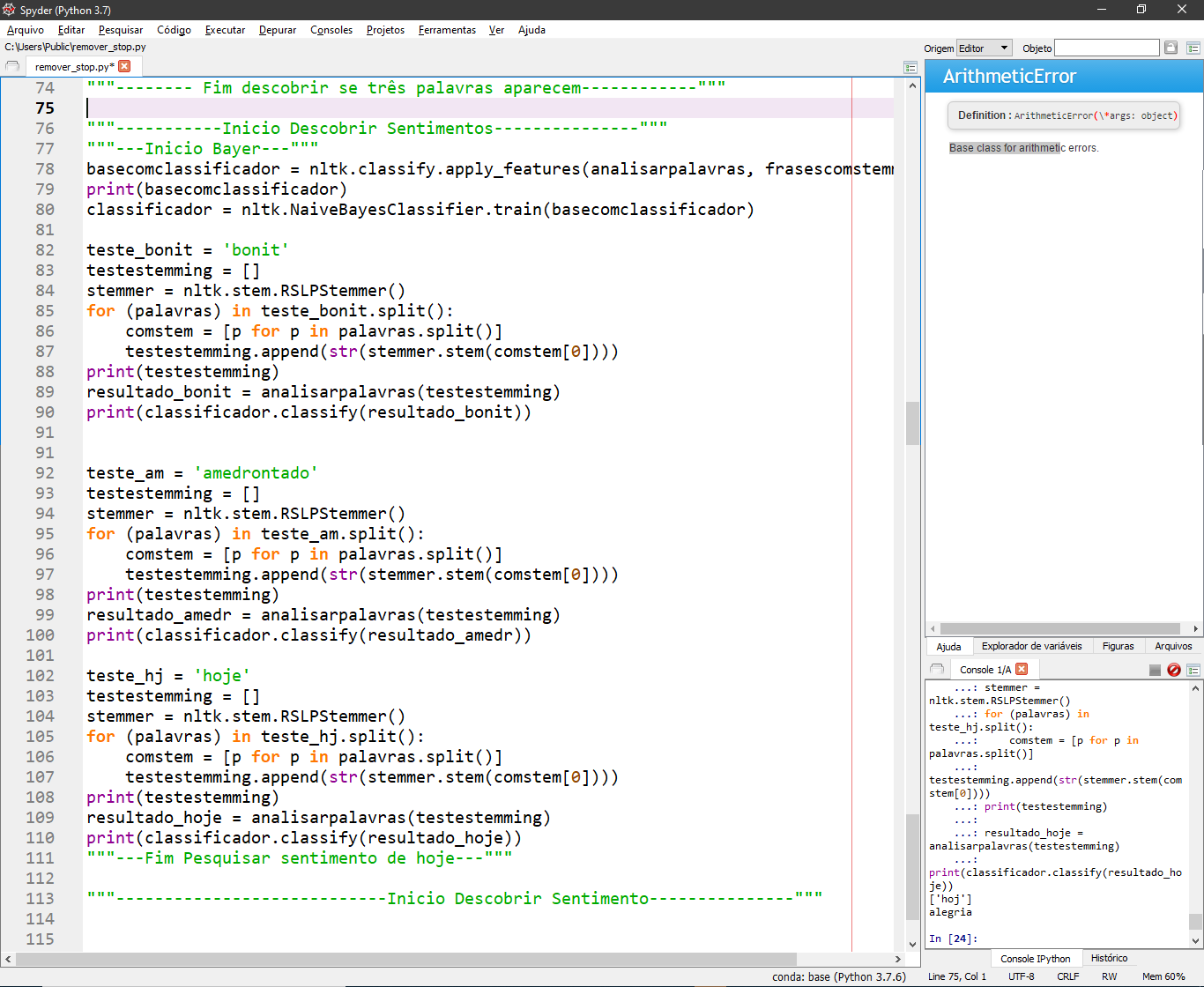
Em banco de dados, para descobrir se as palavras estão presentes, costuma-se usar o símbolo de % no meio da palavra, para descobrir se aquele trecho está presente, assim, criaremos a função analisarpalavras:

Figura 4. Descobrir se as três palavras estão presentes

O resultado mostrará que tem “admir”, tem “hoj”, e tem “bonit”.

Fonte: Do autor - 2020

Agora vamos descobrir os sentimentos associados a cada uma destas palavras, através da utilização do classificador Naïve Bayes. Vamos explicar este classificador após o script, mas se refere ao modelo matemático de probabilidade que faz treinamento de probabilidade *a priori*, e depois aplica ele a *posteriori*. De acordo com Martin (2019), a todo momento nossos rostos, vozes e palavras dizem algo, e criar modelos que treinem bases para classificar sentimentos é uma das muitas ações da NLTK aplicada em Bayes.



Descobre qual sentimento a palavra “'hoje ” na variável “teste\_hj” denota, resultado = medo

Descobre qual sentimento a palavra “'amedrontado'” na variável “teste\_am” denota, resultado = medo

Descobre qual sentimento a palavra “boni” na variável “teste\_bonit” denota, resultado = alegria

Aplica o método bayesiano na classificação

Figura 5. Avaliando palavras

Fonte: Do autor - 2020

Um questionamento que pode surgir é: Como o algoritmo de Bayes faz uma decisão?

A equação de Bayes é feita da seguinte forma:

Para entender esta equação, vamos supor que analisamos 8 pessoas diferentes:

Do autor - 2020

Pesquisas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PESSOA | FAIXA SOCIAL | ESTADO CIVIL | TEM PAIS VIVOS | COMPRAR CASA? |
| 1º Pessoa | MÉDIA | SOLTEIRO | SIM | SIM |
| 2º Pessoa | BAIXA | CASADO | NÃO | NÃO |
| 3º Pessoa | ALTA | DIVORCIADO | NÃO | SIM |
| 4º Pessoa | MÉDIA | SOLTEIRO | NÃO | SIM |
| 5º Pessoa | MÉDIA | CASADO | SIM | NÃO |
| 6º Pessoa | ALTA | SOLTEIRO | NÃO | NÃO |
| 7º Pessoa | BAIXA | CASADO | SIM | SIM |
| 8º Pessoa | BAIXA | CASADO | SIM | NÃO |

Agora vamos aplicar a fórmula para identificar por faixa social, quem tem maior probabilidade de comprar uma casa:

Do autor - 2020

Contagem Bayesiana

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Não** | **Sim** |  |  |
| BAIXA | 2 | 1 | =3/8 | 0,375 |
| MÉDIA | 1 | 2 | =3/8 | 0,375 |
| ALTA | 1 | 1 | =2/8 | 0,25 |
| Total | 4 | 4 |  |  |
|  | 4/8 | 4/8 |  |  |
|  | 0,5 | 0,5 |  |  |

Quem é de classe alta compra casa (sim) com qual probabilidade?

Quem é classe média compra casa (sim) com qual probabilidade?

Quem é classe baixa compra casa (sim) com qual probabilidade?

Então surgem algumas dúvidas: a classe média compra mais casas que a classe alta? Na verdade, pode ser que sim, ou pode ser que não, por isso é necessário calcular o Bayer dos índices para traçar a probabilidade de cada cluster (grupo específico).

Este é o cálculo feito no classificador de emoções contidos nas variáveis citadas:

* **basecomclassificador = nltk.classify.apply\_features(analisarpalavras, frasescomstemming)**
* **print(basecomclassificador)**

**Capítulo 3. Analisar os resultados gerados por algoritmos de análise de sentimentos.**

Cada negócio, cada dataset e banco, cada conjunto de letras e cada caso possui uma lógica interna, formada por fatores econômicos, geográficos, demográficos, psicológicos, mercadológicos, processuais, legais, sociográficos, governamentais, climáticos... Ou seja, cada caso é um caso, cada planilha de dados coletados possui uma probabilidade e lógica interna, e descobrir esta lógica é a base do Data Science.

A matriz de confusão avalia na primeira coluna se o que era para dar positivo, deu positivo, ou se o que era para dar negativo, deu negativo, então teríamos verdadeiro positivo e verdadeiro negativo. A última coluna avalia se o previsto foi falso, se o que era para dar positivo, deu falso negativo, ou se eo que era para dar negativo, deu falso positivo.

Suponha tenhamos 4 palavras: Feliz, Alegre, Triste, Chateada. O código de Naive Bayes colocou erroneamente “chateada” como “alegria”, ele colou “Feliz e Alegre” certamente como alegria, e colocou “triste” como não alegria. Logo ele acertou em três e errou em dois, ele errou colocando chateada como “não alegria”. A linha representa a realidade.

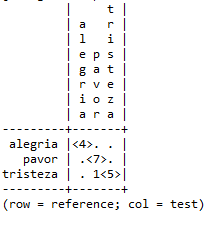
Do autor - 2020

Matriz de confusão

|  |  |  |
| --- | --- | --- |
| É ALEGRIA? | | |
| Previsto | | | | |
| Realidade |  | | Positivo (coluna 1) | Negativo (coluna2) |
| Positivo | | Feliz e Alegre  (VERDADEIRO POSITIVO) | Chateada (FALSO NEGATIVO) |
| Negativo | | (VERDADEIRO NEGATIVO) | Triste  (VERDADEIRO NEGATIVO) |

**Avaliar acurácia e matriz de confusão**

Agora que você já sabe analisar textos, precisa verificar se a análise está certa. Neste sentido, a Matriz de confusão analisa se o esperado se cumpriu. Sempre que se faz um algoritmo, é necessário testar. No exemplo a seguir, você perceberá que o teste acertou em tudo, exceto em uma palavra: “tristeza”. Isso ocorre pois o treinamento de “tristeza” foi pequeno. O ideal é colocar no mínimo 10 frases e tentar descobrir por que o teste errou, verificando qual palavra foi usada.



Vamos fazer a matriz de confusão no python. A Figura 6 a seguir mostra a criação de uma base para treino.

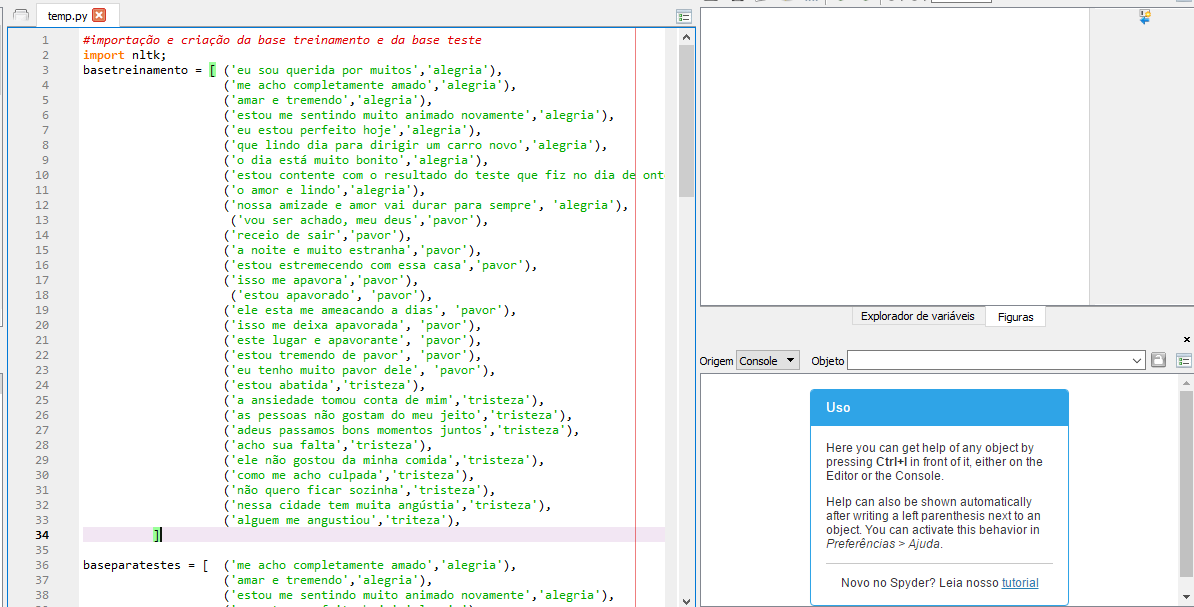


Figura 6. Treinamento base

Fonte: Do autor - 2020

A Figura 7 a seguir mostra a criação de uma base para teste, importação de ferramentas de idioma e tradução para português, bem como remoção de stopwords.

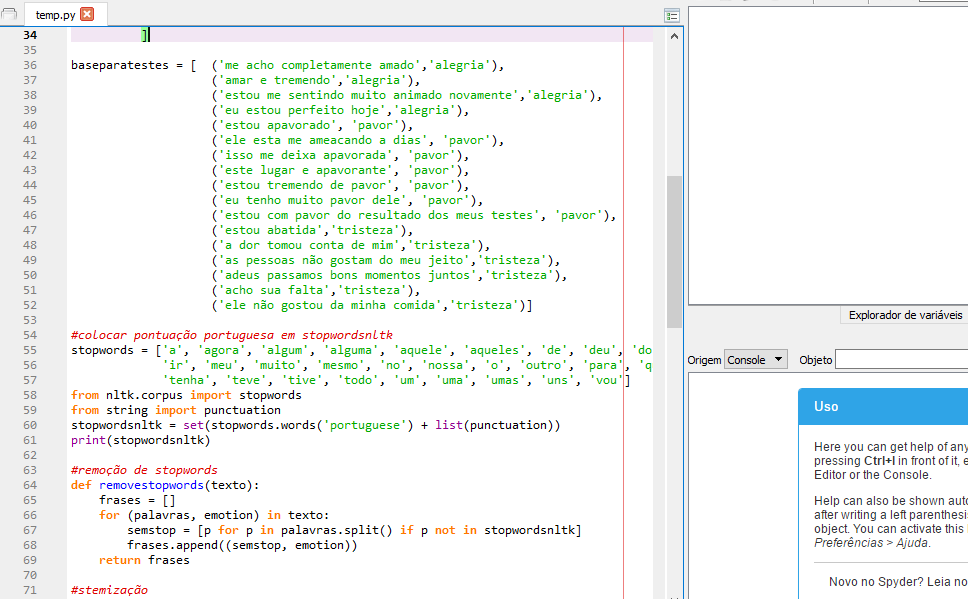


Figura 7. Remoção stops, e criação da base teste

Fonte: Do autor - 2020

A figura 8 mostra a stemmização tanto do treino quanto do teste, nas linhas 72 a 80 é criada a função de stemmizar, e nas linhas 80 e 81 são criadas as variáveis de stemm para ambas as bases. Nas linhas 84 a 90, a figura mostra como coloca as todas palavras de cada base em duas variáveis, palavrastreina, palavrasteste, aqui ela remove as emoções após a virgula, mas as deixa armazenadas na variável “emotion” para usos futuros.

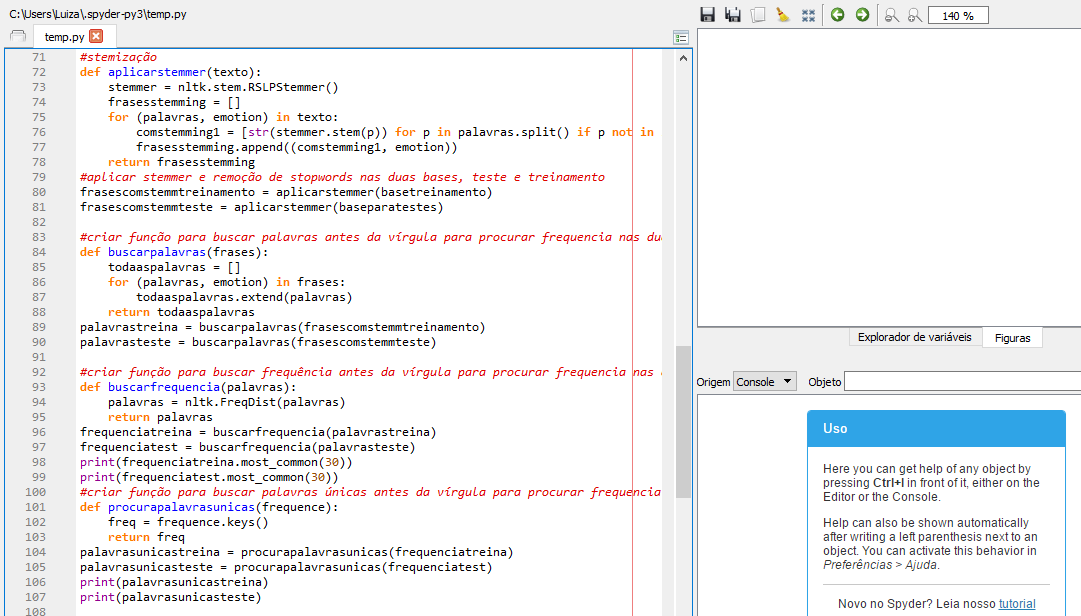


Figura 8. Stemmizar e separar palavras de emoções

Fonte: Do autor - 2020

Na figura 9, na linhas 93 até 95, como o Freq.Dist, a função buscarpalavras calcula a frequencia de cada palavra de cada base, e esta frequencia geral de cada base será colocada, nas linhas 89 e 90, nas variáveis frequenciatest e frequenciatreina.Nas linhas 101 até 107, a função procurapalavrasunicas descobre quais palavras são únicas em cada base.

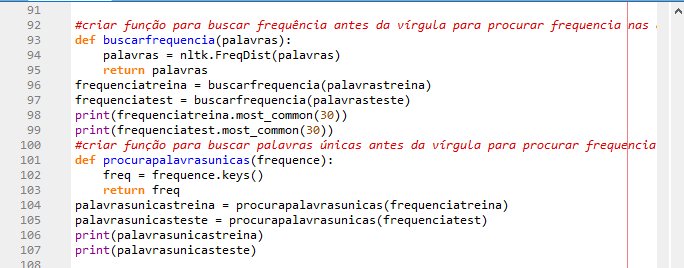


Figura 8. Buscar frequência e palavras únicas

Fonte: Do autor - 2020

Na figura 10 o algoritmo descobre se as palavras entre aspas estão no código. Pode-se pesquisar frases também.

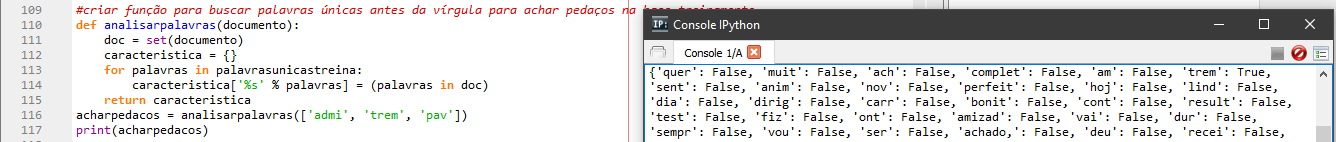


Figura 10. Pesquisar palavras ou frases

Fonte: Do autor - 2020

Na figura 11, é criado o classificador, através do nltk.classify, nas duas bases, teste e treinamento, nas linhas 120 e 121. A linha 124 passa e treina a base treinamento no Naive Bayes, para compreender as emoções e armazenar, e para posteriormente na análise de confusão de matriz da função da linha 128 , comparar com a basecheia na linha 131. O resultado está a direita.

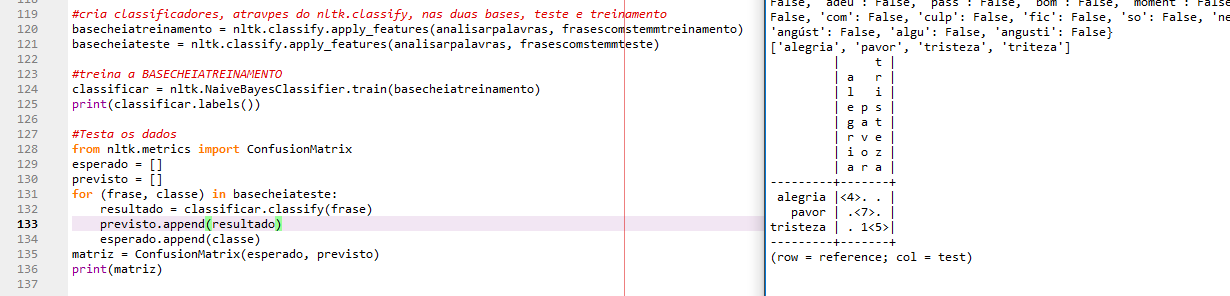
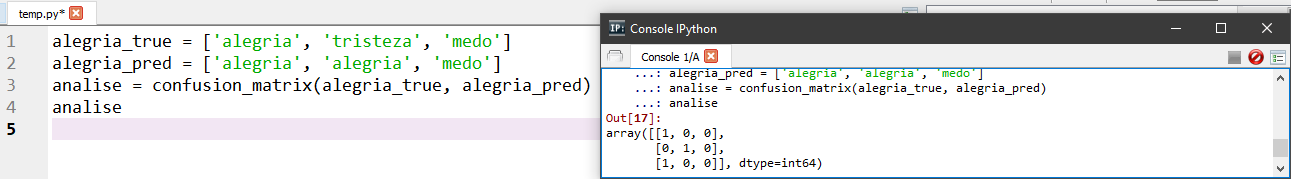


Figura 10. Realização da matriz de confusão

Fonte: Do autor - 2020

Também é possível analisar matriz de confusão com SkLearn, que por sua vez,a scikit-learn (originalmente scikits.learn) é se trata de uma biblioteca de [aprendizado de máquina](https://pt.wikipedia.org/wiki/Aprendizado_de_m%C3%A1quina) de código aberto, a vantagem da sklearn é que ela também faz cálculos rápidos de regressão linear, que analisa quando um indicador influencia em outro, ela faz cálculos de classificação e entre outros cálculos estatísticos. Além disso ela é usada com matplotlib que é uma biblioteca que desenha gráficos, assim, as conclusões matemáticas de sklearn, se dão por desenhos mais visuais e agradáveis.

Isso pode ser analisado rapidamente, onde temos o verdadeiro na linha e o previsto da 3, e o resultado é dado da seguinte maneira:



|  |  |  |  |
| --- | --- | --- | --- |
|  | alegria | medo | alegria |
| alegria | 1 |  |  |
| medo |  | 1 |  |
| tristeza | 1 |  |  |

Figura 12. Matriz de confusão no sklearn

Fonte: Do autor - 2020

Para concluir, segue mais um exemplo, agora de forma ordenada.

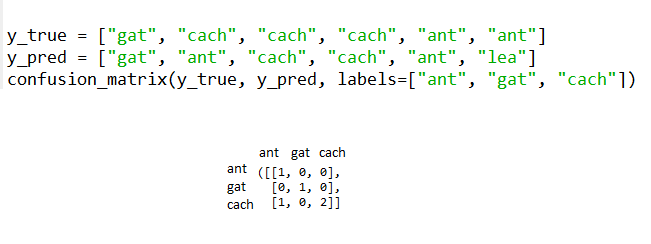


Figura 13. Matriz de confusão no sklearn ordenada

Fonte: Do autor - 2020

Isso pode ser feito com grandes bases de textos importadas.Basta somente substituir as linhas 1 e 2 por importações de arquivos csv ou arrays da biblioteca numpy.

Conclui-se que a linguagem python, através da remoção de stopwords, da stemmização, da análise de frequência, do treinamento com bayes, pode avaliar emoções de uma frase por inteligência artificial. Sempre se faz necessário analisar a eficácia do algoritmo para evitar análises erradas, e para isso é importante fazer a matriz de confusão.

FOSTER, P.; FAWCETT, P. Data Science para negócios: O que você precisa saber sobre mineração de dados e pensamento analítico de dados? Rio de Janeiro: Alta books, 2016.

MARTIN, J.; JURAFSKY, M. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition EUA: 2019.

Davenport, Thomas. H. Big data no trabalho. Rio de Janeiro: Alta books. 2017.

TEOREY, Toby. et al. Projeto e Modelagem de Banco de Dados.

Rio de Janeiro: Elsevier. 2014, p.6. Disponível em: //www.fatecsp.br/dti/tcc/tcc0025.pdf. Acesso em: 08/07/2019. P.35

SILBERSCHATZ, Abraham. Et al. Sistema de Banco de Dados. São Paulo: 2004, Pearson-Makro Books, p.3.

RUIZ, Evandro. Modelo de Dados Relacional. São Paulo: 2017, USP, p.3.