UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

ITeligence: Sistema de Apoio à Análise de Intenções.

Gabriel Morais Teixeira
Bruno Ferreira Brum

Orientador
Fernanda Baião
Kate Revoredo

RIO DE JANEIRO, RJ – BRASIL
JULHO DE 2018

Catalogação informatizada pelo(a) autor(a)

T266

Teixeira, Gabriel Morais
ITeligence: Sistema de Apoio à Análise de Intenções. / Gabriel Morais Teixeira. -- Rio de Janeiro, 2018.
96

Orientadora: Fernanda Baião.
Coorientadora: Kate Revoredo.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2018.

1. Análise de Intenções. 2. Mineração de Textos. I. Baião, Fernanda, orient. II. Revoredo, Kate, coorient. III. Título.

ITeligence: Sistema de Apoio à Análise de Intenções.

Gabriel Morais Teixeira
Bruno Ferreira Brum

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____
Fernanda Araujo Baião, D. Sc. (UNIRIO)

_____
Kate Revoredo, D. Sc. (UNIRIO)

_____
Flávia Maria Santoro, D. Sc. (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.
JULHO DE 2018

Agradecimentos

Gostaríamos de agradecer a todos nossos familiares pelo apoio durante essa nossa jornada, ajudando e nos incentivando em todos os momentos difíceis e tortuosos dessa jornada.

Agradecemos também a nossas orientadoras Fernanda e Kate, pelas ideias, sugestões e apoio durante todo o percurso de nosso projeto, sempre indicando por qual caminho deveríamos seguir.

RESUMO

Com o avanço tecnológico e o grande volume de informações disponíveis, as organizações vêm adotando grandes mudanças de forma a se adequar e usufruir de todas essas informações. Diante desse novo cenário comercial, grandes e pequenas organizações em todo o mundo começam a virar seus olhos para novos paradigmas tecnológicos que visam obter com cada vez mais precisão toda essa informação que tramita nos meios de comunicação, o advento do Big Data.

A partir disso a Mineração de Textos, uma subárea da como Descoberta de Conhecimento que visa extrair padrões a partir de dados não estruturados/semiestruturados, vem recebendo grande destaque. Nos dias de hoje as redes sociais se tornaram o principal meio de comunicação entre as pessoas, com constantes atualizações e disseminação de informações e opiniões, com isso tais mídias se tornam um alvo perfeito para a mineração de texto e análise de sentimentos, para descobrir tendências e padrões. Tendo em vista que é inviável executar uma Análise de Intenções sem o suporte computacional, devido à grande quantidade de dados a serem manipulados e analisados, a proposta e o uso de técnicas automáticas para o processamento de dados e mineração de textos tem se disseminado.

Uma das redes sociais mais utilizadas atualmente para Análise de Intenções é o Twitter, o qual possui uma REST API de fácil acesso para a coleta de seus dados. O Twitter atualmente possui cerca de 342.000.000(trezentos e quarenta e dois milhões) ativos, os quais publicam média 58.000.000 (cinquenta e oito milhões) de *tweets* novos por dia e cerca de 9.100 (nove mil e cem) por segundo.

O presente trabalho propõe o desenvolvimento da ferramenta ITeligence capaz de extrair dados do Twitter e realizar uma análise de intenções de viagens em um determinado período de tempo, onde foram avaliados o processo de análise dos resultados e a qualidade das intenções descobertas. A análise foi feita com *tweets* coletados no idioma inglês.

Palavras-chave: Mineração de Intenções, Análise de Intenções, Mineração de Textos, Twitter

ABSTRACT

With the technological advance and the large volume of information available, organizations have been forced to apply great changes in order to adapt and benefit from all the information. In this context, large and small companies from all over the world are adopting new technological paradigms in order to deal with Big Data scenarios with increasing precision.

Text Mining, as part of the Knowledge Discovery in Databases (KDD) area, has been receiving great attention. The popularity of social networks as a prominent communication channel among people has greatly increased, disseminating and constantly updating information and opinions, thus becoming a perfect forum for the discovery of trends and patterns through the application of automatic text mining and intention analysis techniques. Several new intention analysis techniques have been recently proposed, and some of them incorporate Cognitive Computing concepts so as to extract more sophisticated patterns from social networks, such as user intentions.

Twitter is one of the most popular social networks used for Sentiment Analysis due to its great popularity and the availability of a REST API. Twitter has currently 342,000,000 (three million, forty-two million) users that post an average of 58,000,000 (fifty-eight million) new tweets per day and about 9,100 per second.

The present work describes the development of ITeligence, a tool for extracting data from Twitter and performing Intention Mining on the domain of Travelling. The developed tool was used in a real scenario over tweets in English, where we evaluated the analysis results and the quality of the discovered intentions.

Keywords: Intention Mining, Sentiment Analysis, Text Mining, Twitter

Índice

Índice de Tabela

Índice de Figura

Índice de Equações

1. Introdução

1.1. Motivação

Técnicas de Inteligência e Análise de Negócio (*Business Intelligence and Analytics – BI & A*), associadas a cenários de *Big Data*, tornaram-se cada vez mais importantes nas comunidades acadêmicas e empresarial nas últimas duas décadas; Estudos da indústria destacaram este importante desenvolvimento (Chen, Chiang, & Storey, 2012).

Atualmente o volume de dados disponíveis em redes sociais é gigantesco, tendo um volume e velocidade de disseminação muito alta, diariamente são publicados milhões de novas entradas em diversas redes sociais por pessoas do mundo inteiro. Esse grande número de dados disponíveis nas redes sociais são um convite para realização de mineração de dados para aplicação de *BI & A*, para tomada de decisões, ou simplesmente para identificar um comportamento da sociedade dentro de um cenário. Essas técnicas vêm recebendo grande atenção da comunidade científica nos últimos anos.

Este projeto foi motivado principalmente pelo tópico de mineração de textos, mas também por análise de intenções, e processamento de linguagem natural, mais particularmente no que se refere a intenção de viajar para o local em que um evento aconteceu (por exemplo, um ataque terrorista, um grande evento esportivo, um festival de música, etc.). As etapas seguidas pelo trabalho envolveram coleta de dados da rede social Twitter, aplicação de técnicas de mineração de textos, como pré-processamento, execução de análise de sentimentos nesses dados e tratamento linguístico nos dados.

Além disso, a possibilidade de entender como a população cibernética reage aos acontecimentos globais nos traz muita ansiedade para iniciar nossas pesquisas nesse campo.

1.2. Objetivos

O objetivo deste trabalho é desenvolver uma API com interface gráfica capaz de realizar todos os passos necessários para a realização de mineração de textos, desde a coleta até a análise dos dados, necessitando apenas da execução de uma classificação manual de uma massa de dados para treinamento, analisando e classificando dados que denotam a intenção dos usuários que utilizam a rede social Twitter, além de demonstrar a utilização do programa

no cenário de intenções de viagens na época da Copa do Mundo FIFA de 2018. Tais dados podem ser extraídos em tempo real de postagem ou em algum espaço de tempo fechado no passado, possibilitando ao usuário uma grande variedade de dados disponíveis a serem coletados.

Como objetivos específicos, pode-se citar:

- Criar uma interface gráfica amigável para o usuário.
- Criar um sistema que intermedia a coleta de dados.
- Coletar dados textuais (Tweets) do Twitter de acordo com as preferências dos usuários.
- Fazer o pré-processamento desses dados para classificação e análise.
- Classificar uma nova base de textos, manualmente, que servirá como base de treinamento para o algoritmo Naive Bayes.
- Analisar e classificar automaticamente os dados previamente coletados de acordo com a base de treinamento.
- Gerar gráficos com os adjetivos, substantivos e países mais mencionados nos dados classificados, entre **com intenção** e **sem intenção**.
- Analisar os objetos de maior desejo e relacioná-los com o cenário de vendas de produtos e serviços.

1.3. Organização do Texto

O presente trabalho está estruturado em capítulos e, além desta introdução, inclui:

- Capítulo II: Aspectos teóricos e conceituais envolvidos no desenvolvimento do trabalho. Nesta seção serão apresentadas todas as ferramentas, conceitos, tecnologias e informações importantes que possuem alguma relação com o trabalho.
- Capítulo III: Apresentação da proposta do trabalho e detalhamento da criação do ITeligence descrevendo todas as ferramentas e conceitos utilizados durante o processo.
- Capítulo IV: Seção de utilização do software e onde os resultados são estudados, analisados e comparados.
- Capítulo V: Aqui concluímos o trabalho e propomos um estudo futuro..

2. Aspectos Teóricos e Conceituais

Com a finalidade de sustentar teoricamente este projeto, neste capítulo será abordado: "Mineração de Dados", "Mineração de Texto" com foco em linguagem natural, "Análise de Intenções", abordaremos também as técnicas "Naive Bayes" e "Cross Validation" que serão utilizadas em nosso sistema, e a definição da rede social "Twitter".

2.1. Mineração de Dados

Aparecendo como uma das alternativas mais eficazes para extrair conhecimento a partir de grandes volumes de dados, a mineração de dados nos ajuda a descobrir relações ocultas, padrões e a gerar regras para predizer e correlacionar dados que ajudam as instituições em suas tomadas de decisões rápidas ou a atingir um maior grau de confiança (Machado, et al., 2010).

Sendo uma prática relativamente recente no mundo computacional, a mineração de dados, ou *data mining*, utiliza técnicas de reconhecimento de padrões, inteligência artificial, técnicas de recuperação de informação e reconhecimento de padrões e de estatística para encontrar correlações entre diferentes dados que permitam adquirir um conhecimento benéfico para uma empresa ou indivíduo. Para uma empresa, a *data mining* pode ser uma importante ferramenta que potencia a inovação e lucratividade (Amo, 2004).

Dependendo do objetivo a ser alcançado através da mineração de dados, pode ser necessário aplicar técnicas de detecção de possíveis ruídos e pontos fora da curva, aprendizagem de regras de associação, agrupamento dos dados (*clustering*), classificação dos dados, regressão de dados ou sumarização. O processo de mineração de dados geralmente ocorre se utilizando de dados contidos em um depósito digital para dados (Turban, Sharda, & Delen, 2010).

Na figura 1 podemos observar todo o processo de mineração de dados, e o que se deseja obter com ela, o conhecimento.

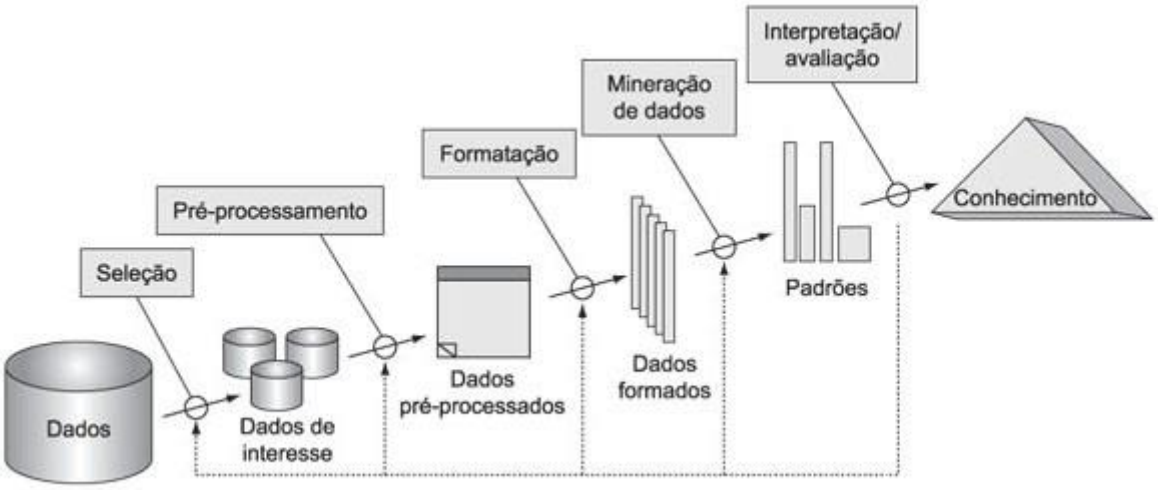


Figura 1: Processo de mineração de dados. (Piatetsky-Shapiro, Fayyad, & Smyth, 1996)

Mineração de dados (data mining) é definida como "[...] o processo não-trivial de identificar, em dados, padrões válidos, novos, potencialmente úteis e ultimamente compreensíveis". (Piatetsky-Shapiro, Fayyad, & Smyth, 1996). É tida como uma área multidisciplinar de pesquisa, onde diversas disciplinas estão inclusas. Por mais que haja uma vasta gama de informações sobre o assunto, ainda não existe uma classificação universalmente aceita sobre o assunto, o que dificulta a condução de projetos para interessados da área (Cardoso & Machado, 2008).

É importante destacar também que o processo tem como finalidade levar resultados relevantes para o usuário. Caso a informação obtida não seja a desejada, é importante que haja uma redefinição sobre os conhecimentos aos quais se desejam obter informações, repetindo o processo até que o resultado obtido seja satisfatório.

2.2. Mineração de Textos

Enquanto a mineração de dados está sendo aplicada a dados estruturados, a mineração de textos aplica-se em analisar textos não estruturados. Com o advento da Web e mídia sociais, este se tornou um dos conceitos mais importantes no cenário da mineração.

A mineração de textos é uma extensão da mineração de dados, e pode ser definida como um processo de extração de informações desconhecidas e úteis de documentos textuais escritos em linguagem natural. Como a maioria das informações são armazenadas em forma de texto, a mineração de textos possui alto valor comercial, e pode ser aplicada em áreas como medicina e atendimento ao cliente (Pezzini, 2016).

Graças ao desenvolvimento da Internet e das redes de computadores os documentos virtuais se transformaram no principal método de armazenamento de informações, principalmente as informações comerciais, que segundo estimativas armazena cerca de 85% de suas informações em documentos de texto. Porém, os paradigmas mais tradicionais de desenvolvimento de software não são capazes de entender o relacionamento confuso e geralmente ambíguo que existe nos documentos de texto virtuais (Machado, et al., 2010).

A mineração de textos permite compreender a imprevisibilidade presente nos textos não estruturados. Assim como a mineração de dados, a mineração de textos consiste em um processo utilizado para descoberta de conhecimento, essa técnica utiliza-se de assimilação e extração de dados a partir de textos e documentos.

Assim no processo de KDD, o processo de Descoberta de Conhecimento em Textos consiste de várias etapas, as quais envolvem preparação dos dados não estruturados, busca por padrões, avaliação do conhecimento e refinamento, todos repetidos em múltiplas iterações. Esse processo é composto por quatro passos bem definidos: seleção, pré-processamento, mineração, análise / assimilação (Morais & Ambrósio, 2007). Essas etapas podem ser vistas no processo ilustrado na figura 2.

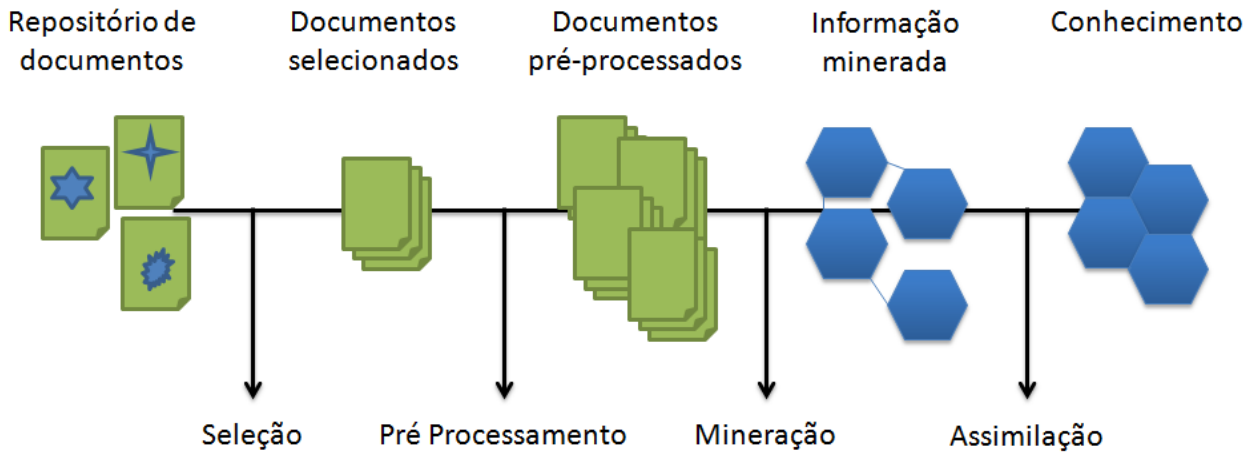


Figura 2: Processo de mineração de textos. (Devmedia, 2014)

Podemos observar ainda na figura 2, que o processo de mineração de textos é iterativo, pois suas transições não possuem um fluxo definido, podendo se repetir quaisquer dos passos quantas vezes forem necessárias, o que pode vir a ocorrer, pois ao chegarmos na fase de assimilação pode ser observado que têm dados expressivos no conhecimento gerado que não são necessários para determinada análise, então a qualquer momento poderá voltar

ao passo de pré-processamento e refazer o passo-a-passo para que se obtenha um melhor resultado. A seguir será apresentada uma breve definição sobre as etapas desse processo.

2.2.1. Seleção (Coleta de dados)

Podemos definir a seleção ou coleta de dados como sendo um processo de seleção e coleta de dados com fins de pesquisa, se utilizando de diversas técnicas e restrições. Está é a etapa inicial da mineração de textos, a partir da coleta que obteremos os dados necessários para poder se efetuar as demais etapas, sendo assim uma etapa crucial para a realização do projeto.

Diversos sites na web hoje em dia permitem efetuar essa coleta, no nosso trabalho utilizaremos o site Twitter como fonte de extração. A seleção é feita a partir de alguma ferramenta específica efetuando uma requisição com certas informações a um determinado site, que será retornado os valores desejados.

2.2.2. Pré-Processamento

Está etapa é de grande importância para o presente trabalho e se dá início logo após o término da coleta dos dados, visto que para podermos efetuar uma análise eficiente dos dados os mesmos devem ter uma boa qualidade.

A etapa de pré-processamento, no processo de mineração de textos, compreende a aplicação de várias técnicas para captação, organização, tratamento e a preparação dos dados. É uma etapa que possui fundamental relevância no processo de mineração de textos. Compreende desde a correção de dados errados até o ajuste da formatação dos dados para os algoritmos de mineração de textos que serão utilizados (Neves, 2003).

Os dados coletados do Twitter podem conter ruídos, tuplas repetidas, emoticons, caracteres especiais indesejados, links, abreviações e gírias, tudo isso citado anteriormente não é interessante para o prosseguimento do trabalho, sendo assim, este passo é necessário para preparar os dados para a fase de mineração.

Entre as técnicas utilizadas no pré-processamento estão presentes limpeza de dados, tokenização e remoção de stopwords, que são técnicas do Processamento de Linguagem Natural (PLN).

Caso não houvesse um pré-processamento, a análise executada neste trabalho poderia ser afetada negativamente, apresentando resultados que não refletem o cenário atual.

2.2.2.1. Tokenização

A tokenização consiste em nada mais do que decompor frases em um conjunto de palavras, essas palavras podem ser chamadas de *tokens* (Stanford NLP, 2009), realizando a exclusão de espaços eventualmente presentes no texto. A frase que passa pelo processo de tokenização passa a ser tratada como uma "lista" (Sette & Martins, 2017).Um exemplo que podemos dar de tokenização é representado na tabela 1 abaixo.

Tabela 1: Tokenização de uma frase.

Entrada	Saída
Muitos ataques terroristas vem ocorrendo no mundo ultimamente.	[Muitos, ataques, terroristas, vem, ocorrendo, no, mundo, ultimamente, .]

Neste exemplo as palavras foram identificadas como sendo aquelas separadas por um espaço em branco, ao se realizar a tokenização da frase a mesma foi delimitada por colchetes, e cada segmento ou palavra que foi identificada foi separada por uma vírgula, sendo montada assim uma estrutura de lista.

2.2.2.2. Remoção de Stopwords

Stopwords consistem basicamente em palavras que são consideradas irrelevantes para o processo de mineração de dados, sendo elas, palavras que contribuem apenas para entendimento geral da fala, todas essas *stopwords* devem ser relacionadas em uma lista que é chamada de *stoplist*, a *stoplist* geralmente é composta por palavras que geralmente são vistas em qualquer pedaço de texto como, por exemplo, artigos, pontuações, pronomes e conjunções. Essas palavras na maioria dos casos não são necessárias para realizarmos a mineração então todas elas são retiradas do texto após a tokenização (Morais & Ambrósio, 2007).

A remoção das *stopwords* também alteram o tamanho do documento e consequentemente a melhoria da análise dos dados. Elas podem afetar também a eficiência da análise devido ao fato natural de não carregarem valores que agregam o que pode acarretar em um grande processamento não produtivo. Estudos confirmam que sua retirada pode aumentar a eficácia das análises entre 30 e 50% (El-Khair, 2006).

2.2.2.3. Stemming e Lematização

Ambos têm como função base a redução de uma palavra e há uma diferença bem sutil entre ambos conceitos (Jivani, 2011). No *stemming* a preocupação está em reduzir para o seu radical, já para a lematização a preocupação é o lema, onde o gênero e o número não interessam.

A lematização lida com a obtenção do "lema" de uma palavra que envolve a redução das formas das palavras à sua origem básica, após o entendimento do contexto da palavra. Em contraste disso, o radical do *Stemming* é encontrado após a aplicação de um conjunto de regras, mas sem preocupação com o discurso.

2.2.3. Mineração

Na mineração, o minerador irá detectar os padrões com base no algoritmo escolhido. E por fim, na assimilação, os usuários irão utilizar o conhecimento gerado para apoiar as suas decisões (Devmedia, 2014).

2.2.4. Assimilação

A assimilação, também conhecida como análise ou pós-processamento, é a última etapa presente na mineração de texto, consiste em apresentar os dados de maneira clara para se verificar se o conhecimento adquirido até aqui é útil ou não para a análise de sentimentos, ou para uma tomada de decisões.

Assimilação visa utilizar ainda técnicas visuais, para a apresentação dos resultados do pós-processamento ao usuário, de modo a apresentá-los todo o conhecimento adquirido nas etapas até aqui (Côrtes, Porcaro, & Lifschitz, 2002).

2.3. Análise de Intenções

A análise do sentimento, também chamada de mineração de opiniões, é o campo de estudo que analisa opiniões, sentimentos, avaliações, atitudes, e emoções acerca de entidades como produtos, serviços, organizações, indivíduos, problemas, eventos, tópicos e seus atributos (Liu, 2012).

Já por outro lado, intenções podem ser definidas como “Aquilo que uma pessoa espera que aconteça; vontade; o que se pretende fazer; o que se almeja; o que se busca; desejo: sua intenção era brigar”. (Dicio, 20-)

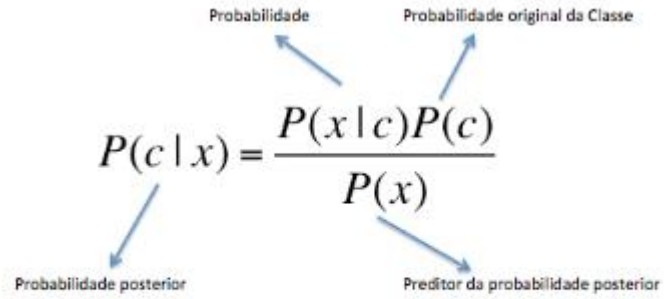
Com isso podemos assimilar que a mineração de intenções está voltada para a análise de atos, atividades, e desejos, com o objetivo de aplicar o aprendizado dessas análises em cenários futuros. Para isso é muito importante que haja uma base de dados com uma quantidade relevante de dados para o aprendizado.

2.4. Naive Bayes

O Naive Bayes é tido por muitos como o melhor algoritmo de aprendizado indutivo. Sendo muito eficiente e eficaz, seu desempenho competitivo é classificado como surpreendente pois a independência condicional na qual se baseia raramente é verdadeira em aplicações no mundo real. (ZHANG, 2004)

O método é projetado para uso em tarefas de indução supervisionadas, onde prever com precisão a classe de instâncias dos testes, e em quais instâncias de treinamento estão incluídas informações de classe, é o principal objetivo (John & Langley, 1995).

Segundo o teorema de Bayes, é possível encontrar a probabilidade de certo evento ocorrer, dada a probabilidade de um outro evento que já ocorreu: Probabilidade (B dado A) = Probabilidade (A e B)/Probabilidade(A) (Camilo & Silva, 2009).



$$P(c \mid X) = P(x_1 \mid c) x P(x_2 \mid c) x \ldots x P(x_n \mid c) x P(c)$$

Equação 1: Teorema de Bayes (Vooo, 2016)

Onde podemos observar que “P (c | x) é a probabilidade posterior da classe (c, alvo) dada preditor (x, atributos). P (c) é a probabilidade original da classe. P (x | c) é a probabilidade que representa a probabilidade de preditor dada a classe. P (x) é a probabilidade original do preditor.” (Vooo, 2016).

Alguns exemplos onde a aplicação do algoritmo Naive Bayes se destaca são: Previsões em tempo real, Previsões multi-classes, Sistema de Recomendação e Classificação de textos/Filtragem de spam/Análise de sentimento. (Jurafsky & Martin, 2017)

Neste trabalho o Naive Bayes foi o algoritmo escolhido por ter um melhor resultado em problemas de classes múltiplas e regra de independência.

2.5. Cross Validation

A validação cruzada é um método de avaliação estatístico, onde os dados são divididos em dois segmentos: Um usado para treinar o modelo e outro usado para testar o modelo. Na hora de testar o modelo os conjuntos de treinamento e testes precisam se cruzar em rodadas sucessivas, até que cada ponto tenha sido pelo menos uma vez testado. A forma básica de validação cruzada é a *k-fold* (Refaeilzadeh, Tang, & Liu, 2008).

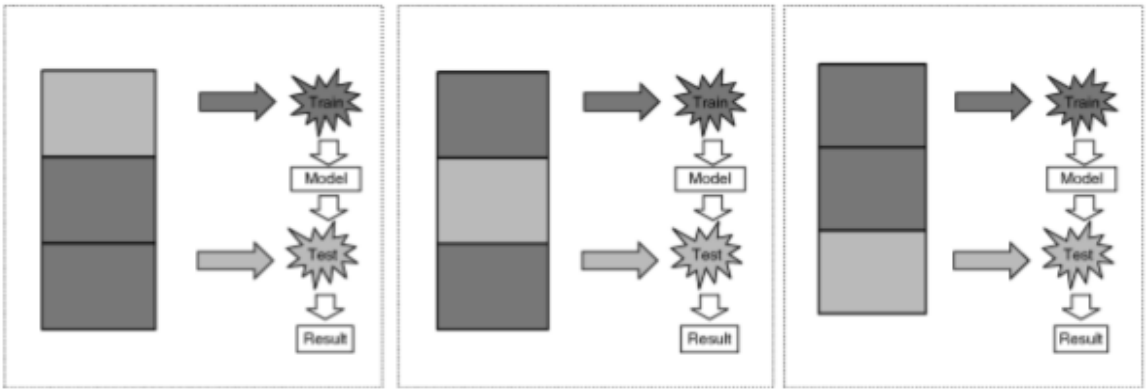


Figura 3: 3-*fold cross validation*. (Refaeilzadeh, Tang, & Liu, 2008)

A validação cruzada permite aumentar a robustez de um modelo gerado a partir da análise de dados de treinamento (Training data), tornando-o menos sensível à variabilidade dos dados de entrada. Essa validação oferece uma estimativa de como o modelo se comporta ao analisar um conjunto de novos dados, Na figura 3 podemos observar o funcionamento de um *cross validation* com três *folds*.

É normalmente aplicada a algum aprendizado de máquina com o objetivo de comparar e selecionar um modelo para um determinado problema de modelagem preditiva, porque é de fácil entendimento e fácil implementação, e geralmente traz resultados com uma margem de erro menor do que outros métodos. O procedimento possui um parâmetro único chamado K que se refere ao número de grupos que um simples grupo de dados precisa ser

dividido. Dessa forma algumas vezes o procedimento é chamado de “validação cruzada em K dobras”.

2.6. Twitter

Twitter é uma rede social e um servidor para *microblogging* que possui atualmente mais de 600.000.000 (seiscentos milhões) de usuários, sendo destes cerca de 342.000.000(trezentos e quarenta e dois milhões) ativos atualmente (Static Brain, 2016).

O Twitter pode ser definido como uma rede social que permite que usuários postem curtas atualizações de status, essas postagens são chamadas de *tweets. Tweets* atualmente podem conter até 280 caracteres em sua mensagem, além de poderem conter menções a outros usuários, marcações de lugares do mundo real, *hashtags*, *URL’s* e anexos, esse conjunto de características são chamadas de *entities* e *places*. O Twitter também possui atualmente uma *RESTful* API que nos permite coletar seus dados (Russel, 2014). Também é possível na rede social realizar *retweets*, que consiste em compartilhar o *tweet* de outro usuário em seu próprio perfil.

Atualmente são feitos em média 58.000.000 (cinquenta e oito milhões) de tweets novos por dia e cerca de 9.100 (nove mil e cem) por segundo (Static Brain, 2016), levando em conta o grande número de usuários ativos atualmente nesta rede social, e o grande número de novo conteúdo que vem a ser publicado diariamente, foi a rede social escolhida para este projeto.

Este trabalho utilizará o Twitter para fazer a extração dos *tweets* dos usuários que possuam um ou mais dos termos que serão definidos para a coleta dos mesmos, utilizando-se também de sua RESTful API.

3. Proposta: Desenvolvimento da ferramenta ITeligence

Neste capítulo iremos falar sobre o propósito do trabalho, o processo de desenvolvimento do sistema ITeligence e suas especificações, além de abordarmos as tecnologias que foram utilizadas durante o desenvolvimento.

O propósito deste trabalho é oferecer para os usuários uma interface capaz de extrair, processar, e analisar dados do Twitter em tempo real ou em algum espaço de tempo fechado no passado, oferecendo também gráficos e opções que auxiliem a tomada de decisão. A ferramenta gerará gráficos analíticos para o usuário (coluna, barra, linhas). Para que isso aconteça o usuário deverá ter instalado em seu computador uma versão compatível do Python juntamente de um banco de dados MySQL, com os devidos módulos instalados para o armazenamento e processamento dos dados.

Além disso, o usuário poderá também exportar as informações extraídas e processadas do programa para um documento CSV, para manter um histórico e consultá-lo quando necessário.

O grupo se interessou pelo tema devido a grande demanda mundial de se obter cada vez mais informações dos usuários, clientes e outros *stakeholders* envolvidos no dia a dia do negócio de uma companhia.

3.1. Visão Geral

A proposta principal de nosso sistema é oferecer para os usuários uma interface gráfica simples e intuitiva, para qualquer usuário, o sistema deverá ser capaz de executar todos os passos presentes na mineração de textos, podendo coletar dados, processar, analisar e classificar.

O cenário do sistema será para utilização exclusiva na rede social Twitter, e com suporte apenas para o idioma inglês. Será oferecida a possibilidade ainda de o usuário coletar dados em tempo real com delimitação de datas passada, oferecendo também a possibilidade de criar gráficos dos dados já classificados para auxiliar em tomadas de decisões. É importante frisar que o sistema é genérico, porém para que seu funcionamento seja o melhor possível para qualquer domínio, deverá ser criada uma massa de testes manualmente.

A ferramenta gerará gráficos analíticos para o usuário (coluna, barra, linhas). Esses gráficos permaneceram salvos em pastas internas do programa para posterior acesso, além disso o usuário poderá fazer a exportação dos dados classificados para um arquivo no formato CSV, de modo que o mesmo poderá realizar outras análises de forma mais personalizada.

Para o início da implementação do programa foi criada uma documentação de requisitos contemplando casos de usos, diagramas de atividade e modelos conceituais, esse documento poderá ser acessado através do APÊNDICE A. Na figura 4 podemos observar o modelo de dados conceitual do sistema, esse modelo tem como principal objetivo apresentar de uma maneira simples e de alto nível as informações são manipuladas pelo sistema.

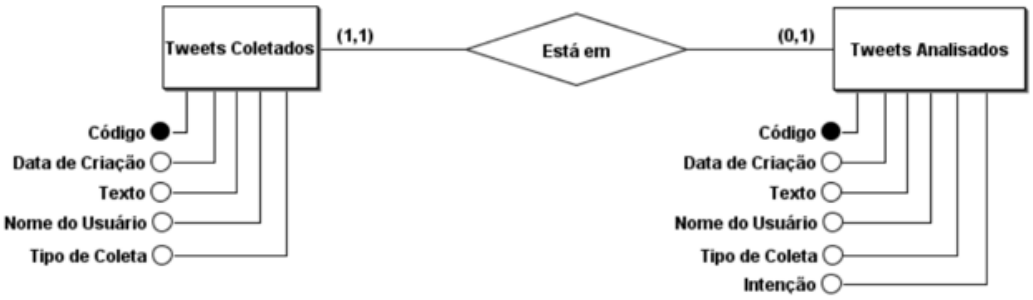


Figura 4: Modelo de dados conceitual ITeligence

Podemos a partir desse modelo observar que existem 2 (dois) tipos de dados, um dos tipos são os dados que ainda não foram classificados pelo sistema, e o outro tipo, os dados já classificados, não poderá existir dados classificados sem que eles tenham sido coletados, e um dado só poderá ser classificado apenas uma única vez. O atributo “Tipo de Coleta”, se refere a como os dados foram coletados, sendo o valor 1 (um) para dados coletados em tempo real e 0 (zero) para dados que foram coletados em um espaço de tempo passado. Já nos dados que já foram classificados há o atributo intenção, que irá indicar a classificação das tuplas, sendo ela 1 (um) para com intenção e 0 (zero) para sem intenção

3.2. Tecnologias Utilizadas

O sistema foi codificado utilizando-se a linguagem de programação Python, além da utilização de um bando de dados SQL para armazenamento dos dados utilizando-se do SGBD MySQL, e da plataforma GitHub para versionamento e compartilhamento do código do sistema.

3.2.1. Python

Para a implementação do sistema foi utilizada a linguagem de programação Python em sua versão 3.6.3.

A documentação do Python define a linguagem como: “Uma linguagem de programação interativa, interpretativa e interativa. Ela incorpora módulos, exceções, *dynamic typing*, um nível alto de tipos de dados dinâmicos e classes. O Python combina um poder notável com uma sintaxe clara. Possui interfaces para muitas chamadas de sistema e bibliotecas, bem como para vários sistemas de janelas, e é extensível em C ou C ++. Também é utilizável como uma linguagem de extensão para aplicativos que precisam de uma interface programável. Por fim, o Python é portátil: ele é executado em muitas variantes do Unix, no Mac e no Windows 2000 e posterior.” (Python, 201-?).

Python foi escolhido como linguagem de programação por possuir diversos módulos e plataformas que são voltados para a mineração de textos e dados, como o *Natural Language Toolkit* (NLTK), por exemplo.

Para explicar um pouco melhor do que se trata a plataforma NLTK, buscamos a definição diretamente no site oficial da mesma, sendo definida como: “Uma plataforma líder para criar programas em Python para trabalhar com dados em linguagem natural. Ele fornece interfaces fáceis de usar para mais de 50 recursos de corpora e lexicais como o *WordNet*, juntamente com um conjunto de bibliotecas de processamento de texto para classificação, tokenização, stemming, *tagging*, análise e raciocínio semântico, *wrappers* para bibliotecas NLP de força industrial, e um fórum de discussão ativo.” (NLTK, 201-?).

3.2.2. Anaconda

Para fazer o gerenciamento dos pacotes e módulos que foram usados durante toda a implementação foi utilizada a ferramenta Anaconda em sua versão 4.5.4.

Segundo podemos ver na documentação fornecida pela empresa Anaconda (desenvolvedora responsável pela ferramenta), o software Anaconda trata-se de um programa onde podemos gerenciar e ter acesso a pacotes *open source*, tendo um número atualmente maior que 250 pacotes disponíveis para download e atualização (Anaconda, 201-?).

3.2.3. Spyder

O programa Spyder foi utilizado durante o projeto como a IDE para o desenvolvimento do sistema, sendo usada sua versão 3.2.4.

Segundo podemos encontrar na documentação da ferramenta, o Spyder trata-se de um ambiente de desenvolvimento científico para Python, sendo definido também como: “um poderoso ambiente de desenvolvimento interativo para a linguagem Python com recursos avançados de edição, testes interativos, depuração e introspecção. O Spyder também pode ser usado como uma biblioteca que fornece poderosos *widgets* relacionados ao console para seus aplicativos baseados em PyQt - por exemplo, ele pode ser usado para integrar um console de depuração diretamente no layout de sua interface gráfica com o usuário.” (PythonHosted, 201-?).

3.2.4. NLTK

A *Natural Language Toolkit* (NLTK) é um módulo utilizado no auxílio de construção de programas em Python para manipulação de dados de linguagem natural, ou seja, humana. Este módulo fornece inúmeras interfaces para uso com mais de 50 corpora e recursos léxicos, além de conter um grande conjunto de bibliotecas voltadas para uso em mineração de textos para tokenização, *stemming* lematização, classificação, análise e raciocínio semântico (NLTK, 2017).

Esse módulo será usado em várias etapas de nosso programa, por conter diversas bibliotecas que serão utilizadas durante todo o processo de mineração de texto.

3.2.5. SQL

Para a manipulação dos dados, foi utilizada a linguagem SQL.

SQL se trata de uma linguagem usada para realizar o armazenamento, recuperação e manipulação de dados (W3Schools, 200-?).

SQL trata-se de uma linguagem utilizada para base de dados relacionais, que são definidas como dados organizados baseados em um modelo de dados relacional (Codd, 1970). No modelo de dados relacional, os dados são representados em tabelas. Uma tabela contém um conjunto de atributos, esses atributos podem ser considerados meta dados das tabelas, e cada um deles e associado a um tipo, como por exemplo, *integer* ou *string*. As linhas dessa tabela são chamadas de tuplas, quando uma tabela possui N atributos cada tupla

contém N componentes, e cada um desses componentes deve pertencer ao tipo especificado (Näsholm, 2012).

Para o controle dessas tabelas e dados é utilizado um sistema que conhecemos como SGBD.

3.2.6. MySQL

Para armazenamento e controle dos dados foi escolhido o SGBD que utiliza linguagem SQL, MySQL, na versão 5.7.12. Foi utilizado o pacote MySQLdb para Python para realizar a conexão entre o sistema e o banco de dados.

Segundo apontado pela pesquisa realizada pelo portal DB-engines, o MySQL é o segundo SGBD mais popular, perdendo apenas para o Oracle, tendo suporte total a linguagem de banco de dados SQL, e suporte a diversas linguagens de programação como C, C++, Python, Java, entre muitas outras (DB Engines, 2018).

Um sistema gerenciador de banco de dados (SGBD) se define por um conjunto de dados inter-relacionados e um conjunto de programas voltados para o acesso a estes dados a qualquer momento. Desse modo, um SGBD deve prover um ambiente ideal para que possa haver a recuperação e armazenamento das informações do banco de dados gerenciado (Silberschatz, Korth, & Sudarshan, 1999).

3.2.7. GitHub

Como ferramenta para versionamento, controle e compartilhamento do código foi utilizada a ferramenta GitHub, utilizando-se de uma conta universitária para que o repositório fosse privado durante todo o desenvolvimento do sistema.

GitHub é uma ferramenta da Microsoft, que permite o versionamento e controle de códigos de diversas linguagens diferentes. Com o uso da ferramenta foi possível exercer a portabilidade do código, onde o mesmo poderia ser acessado de qualquer computador, além de permitir o controle e versionamento do mesmo, além da colaboração no desenvolvimento.

3.3. Desenvolvimento

Nessa seção iremos abordar o processo de desenvolvimento, indicando o funcionamento de cada parte do sistema e os problemas que foram encontrados no decorrer do mesmo, bem como as alterações que foram realizadas.

3.3.1. Interface Gráfica

Foi desenvolvida uma interface gráfica (GUI – *Graphical User Interface*) para o sistema utilizando o framework TKinter, que permite a criação de interfaces gráficas utilizando a linguagem de programação Python, que foi escolhido por ser nativo da linguagem possuindo todos os recursos necessários para a implementação da GUI.

Tkinter é definido como uma biblioteca exclusiva para a linguagem de programação Python, que vem junto com a instalação padrão, e permite que sejam desenvolvidas interfaces gráficas com a utilização desta biblioteca, ou seja, qualquer computador que possua um interpretador Python instalado e capaz de efetuar a criação de GUI (Otávio, 2016).

Com intuito de fornecer um programa com fácil uso para qualquer usuário que não possua conhecimentos aprofundados em computação, foi utilizado o conceito de usabilidade e intuitividade para a construção da interface do sistema, utilizando como modelo uma interface similar a usada atualmente para dispositivos móveis.

A usabilidade mede a facilidade do usuário em completar objetivos específicos com eficácia, utilizando um produto projetado para proporcionar eficiência e satisfação ao usuário, em um contexto específico (ISO 9241-11).

Para conseguirmos atender a usabilidade que seria um premissa do sistema, utilizamos uma linguagem simples para o usuário, com imagens intuitivas. Foi focado durante a implementação em tratar o maior número de erros que possam vir a acontecer, sempre fornecendo um feedback claro para o usuário durante cada processamento ou cada erro que possa vir a acontecer no programa, além do sistema manter um log em tempo real que o usuário pode acessar a qualquer momento caso queira ter uma descrição melhor do funcionamento do sistema e os erros que possam a vir a acontecer durante esse funcionamento.

Na figura 5 podemos observar a interface principal do programa que é basicamente composta por botões e uma barra de tarefas, a partir dessa interface o usuário poderá acessar e executar qualquer funcionalidade do sistema em no máximo 3 (três) cliques.

Figura 5: Página inicial do sistema ITeligence

Outra tela presente no programa é a tela destinada a alteração das preferências do usuário como podemos observar na figura 6, nesta tela o usuário poderá alterar suas preferências para coleta e análise, sendo tratado qualquer texto inserido pelo usuário.

Figura 6: Janela de preferências do usuário no sistema ITeligence.

Por fim a última tela presente no programa é a galeria de imagens, onde o usuário poderá visualizar todos os gráficos que foram criados pelo programa na etapa de análise dos dados, na figura 7 podemos observar essa tela.

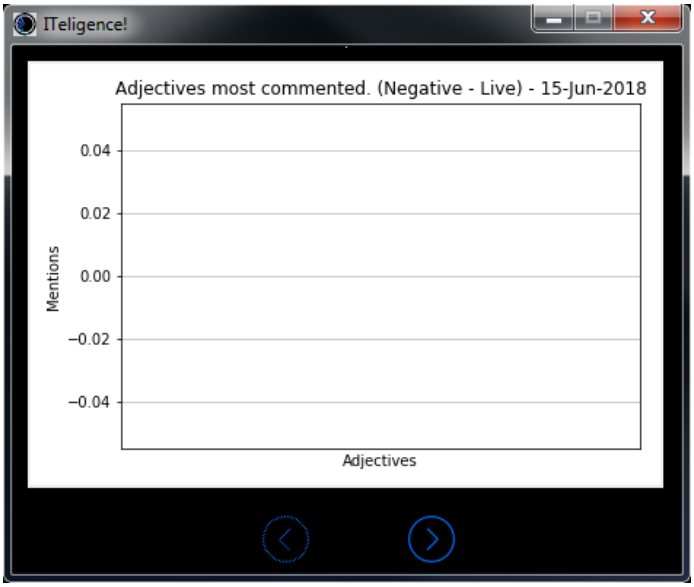


Figura 7: Galeria de imagens.

3.3.2. Extração de Tweets

Para efetuar a etapa de coleta da mineração de textos, o usuário terá duas opções no sistema, ele poderá realizar a coleta dos dados do Twitter em tempo real, ou seja, enquanto os usuários forem *tweetando* dados que se encaixam na preferência indicada pelo usuário esses dados serão coletados e salvos, ou poderá efetuar uma coleta selecionando um espaço no tempo, onde serão coletados dados que já foram postados em algum momento e ainda continuam disponíveis no Twitter.

3.3.2.1. Extração em Tempo Real

Para que o usuário tivesse a possibilidade de efetuar coleta de dados do Twitter em tempo real, foi utilizada uma biblioteca disponível para Python chamada tweepy, essa biblioteca fornece as funções necessárias para efetuar uma conexão diretamente com a API disponibilizada pelo Twitter e então realizar a coleta dos dados requeridos. Para a coleta e

busca foram feitas customizações nas funções disponibilizadas, de modo a atender as necessidades do sistema.

Para que o consumo dos dados da API do Twitter seja realizado, é necessário conseguir junto ao *Twitter Application Manager* as seguintes chaves: *Consumer Key* (*API Key*), *Consumer Secret* (*API Secret*), *Access Token* e *Access Token Secret*, que podemos ver na figura 8, podendo assim realizar a coleta de dados para a mineração de texto ser realizada. Para possuir acesso a essas chaves deve ser criada uma conta de usuário na *Twitter Application Manager*, com a conta já criada, deverá ser feita a criação de uma *application* para então ter acesso às chaves mencionadas.

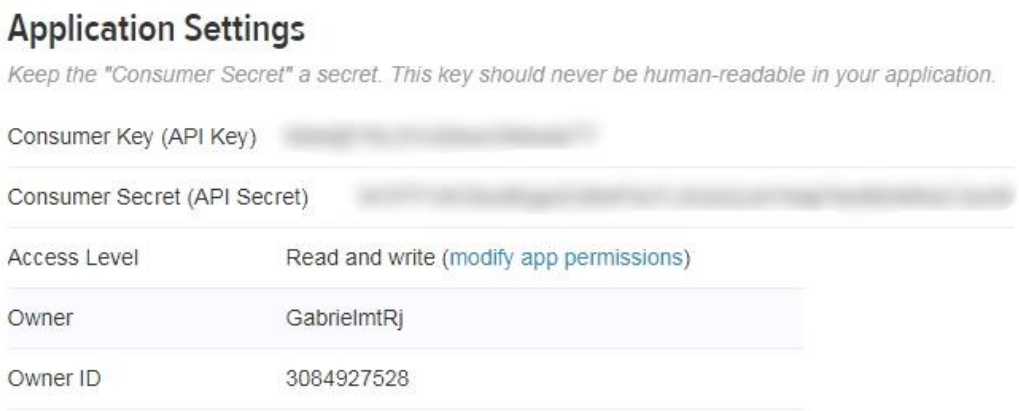


Figura 8: *Consumer Key* da API do Twitter.

Nossas chaves, por questões de segurança, estão armazenadas em um arquivo criptografado com a AES (*Advanced Encryption Standard*), para que a qualquer momento o sistema possa recuperar estas chaves de acesso e realizar o consumo dos dados da API.

Com a conexão com a API realizada, o usuário poderá executar a coleta dos dados, para isso ele pode definir suas preferências no sistema, uma dessas preferências é chamada de *Keywords*, que nada mais é que um conjunto de palavras ou frases que serão utilizadas para efetuar um filtro de dados a serem coletados da API, o outro parâmetro é o *runtime* que será o tempo, definido pelo usuário, que irá durar a coleta, esse tempo é definido em segundos, e o último parâmetro que já é definido pelo sistema é o parâmetro “Lang” que indica a linguagem dos dados a serem coletados, no nosso caso a linguagem definida foi inglês, sendo assim a coleta só ocorrerá para dados que foram definidos pelo Twitter como sendo do idioma Inglês.

É importante ser definido também como o algoritmo irá interpretar as Keywords, para realizar buscas podem ser utilizados dois operadores lógicos: E e OU. Para utilizarmos

do operador lógico OU será utilizado o caractere ponto e vírgula (;), ou seja, caso as *Keywords* sejam definidas como "*travel;trip*", equivale a buscar "*travel* OU *trip*", então serão coletados dados que possuam as palavras *travel* ou *trip* ou as duas. Para podermos utilizar o operador lógico E será utilizado o caractere "espaço", então caso os *Keywords* sejam definidos como "*travel trip*", equivaleria a buscar "*travel* E *trip*", ou seja, será feita a coleta de dados que possuam as palavras *travel* e *trip* simultaneamente, a chamada da função de coleta pode ser vista no trecho de código na figura 9.

```
 98    ##Execute the streamer, just english tweets, and the execution is done in background
 99    streamer.filter(languages=["en"], track = keyWords, stall_warnings=True, async=True)
100    ##
101    ##After runtime is over the stremer is finalized
102    time.sleep(config.runtime)
103    streamer.disconnect()
```

Figura 9: Filtro para coleta de dados do Twitter.

Após recebermos da API os dados, de acordo com as *Keywords* definidas, é verificado se as *Keywords* estão presentes somente no campo de texto dos dados (ou seja no *tweet*), pois elas podem estar presentes em outros campos sem que estejam presentes no campo de texto, caso as *Keywords* estejam presentes no campo de texto dos dados, os campos definidos (texto, nome de tela do usuário e data de criação) para a coleta são salvos em variáveis locais, então é realizada a chamada de uma função que realiza o *INSERT* dessas variáveis na tabela "*tweets*" de nosso banco de dados por meio de um query criada para essa inserção, como podemos ver na figura 10. A tabela ainda possui o atributo "*live*" este atributo identifica com os valores 0 (zero) ou 1 (um) se o *tweet* foi coletado em tempo real, ou se foi coletado por intervalo de tempo, sendo 1 (um) para tempo real é 0 (zero) para espaço de tempo. Assim esses dados poderão ser acessados pelo sistema a qualquer momento que o usuário necessitar, seja para realizar análise ou para excluí-los.

```
17 def store_data(created_at, text, screen_name):
18     db=MySQLdb.connect(host=config.sqlHost, user=config.sqlUser,
19                        passwd=config.sqlPasswd, db=config.sqlDataBase,
20                        charset="utf8")
21     cursor = db.cursor()
22     insert_query = "INSERT INTO tweets (screen_name, created_at, text, live) VALUES (%s, %s, %s, %s)"
23     cursor.execute(insert_query, (screen_name, created_at, text, '1'))
24     db.commit()
25     cursor.close()
26     db.close()
27     return
```

Figura 10: Inserindo dados na tabela *tweets*

Durante a implementação desta funcionalidade alguns problemas foram encontrados para serem tratados. Um destes problemas encontrados durante a utilização do programa foi que o método de busca do tweepy realiza a busca das *Keywords* em todo o JSON que é retornado da API do Twitter, ou seja, quando procuramos por "*travel;trip*" (*travel* OU *trip*), caso o nome do usuário, *login* ou qualquer outro atributo presente no JSON contenha uma dessas palavras, o *tweet* será coletado e salvo pela aplicação. A mitigação para isto foi realizar uma verificação assim que esse JSON é retornado, assim que é encontrado algum dado que atende aos requisitos, é feita uma conversão do campo texto desse dado para letras minúsculas e então é feita a comparação com os termos que foram utilizados para filtrar os dados (*Keywords*), caso não exista nenhum dos termos definidos no corpo do tweet então ele não será salvo, caso contrário será armazenado pela aplicação, assim garantimos que só serão coletados dados que possua as *Keywords* em seu texto.

Outro problema encontrado foi um erro recorrente que acontece durante a coleta, o *imcompleteRead*, esse erro ocorre quando existe uma quantidade muito elevada de dados sendo retornada por segundo, com isso, a aplicação não dava conta dessa quantidade, quando este erro ocorria a extração era finalizada sem tratamento, o que levava ao usuário crer que a coleta ainda estava ocorrendo, porém a mesma já havia parado. Para realizar uma mitigação para este problema, foi feito um *overwrite* da função que é chamada quando ocorre uma exceção (*on_exception*) da classe *StreamListener* da biblioteca tweepy, para caso ocorra um erro deste tipo ele irá efetuar novamente a chamada da coleta com os mesmos parâmetros da anterior, sem que o runtime seja zerado, então mesmo que o problema ocorra a aplicação não sofrerá consequências, o erro será registrado no log e a coleta de *tweets* continuará a ocorrer normalmente durante o tempo estipulado.

3.3.2.2. Extração em Intervalo Temporal

O método para a coleta de dados de um intervalo temporal pré-definido teve de ser implementado sem a utilização da biblioteca tweepy, visto que a API disponibilizada pelo Twitter só fornece dados que foram publicados em no máximo 2 (duas) semanas atrás.

Dado esse cenário, para realizar essa coleta de dados mais antigos, é feita uma pesquisa por *tweets* diretamente na URL do Twitter destinada a busca avançada. A pesquisa é realizada nesta URL, e é retornado para aplicação um HTML contendo os dados a serem coletados, que será convertido para um JSON somente com os dados desejados (texto, nome de tela do usuário e data de criação), como mostrado na figura 11.

```
[
    {
        "tweet_id": 980593709443047400,
        "text": "Remains of Ur. The alleged location of Abraham's house.",
        "user_screen_name": "MaxergonDotCom",
        "created_at": 1522626733000,
        "lang": "en",
        "user_id": 977006803622944800
    }
]
```


Figura 11: JSON convertido para coleta de dados.

A busca é realizada a exemplo da funcionalidade anterior, utilizando-se das Keywords como parâmetro, sendo definidas da mesma maneira, e utilizando-se também do campo “Lang” para que a coleta seja feita apenas de dados do idioma inglês. A diferença ocorre na utilização de 2 (dois) novos parâmetros, o parâmetro *until* que define a data de início da busca pelos dados, e o parâmetro *since* que define a data final, assim, são buscados dados que estejam nessa range de datas que será definida pelo usuário, os parâmetros são inseridos no formato aaaa-mm-dd (ano-mês-dia), a chamada da função com os parâmetros pode ser vista na figura 12.

```
    ##Using twitter slicer to collect old tweets
    since = datetime.datetime.strptime(since, '%Y-%m-%d')
    until = datetime.datetime.strptime(until, '%Y-%m-%d')
    searchQuery = keyWords
    ##
    ##Call function to search for tweets
    twitSlice = TwitterSlicer(rateDelaySeconds, errorDelaySeconds, since, until)
    twitSlice.search(searchQuery)
```

Figura 12: Chamada de função para coleta de tweets em intervalo de tempo.

Após os dados serem retornados, é verificado se a data dos mesmos existe e se eles são do idioma inglês, caso contrário eles não serão salvos, a exemplo do que ocorre na extração em tempo real, após essa verificação dos dados, é verificado se o corpo do *tweet* (campo de texto) possui as palavras que foram definidas no *Keywords*, caso contrário os *tweets* não serão salvo, sendo descartados. Caso positivo o processo de coleta prosseguirá, os campos (texto, nome de tela do usuário e data de criação) são separados do restante dos dados e então são inseridos em nossa tabela “*tweets*” no banco de dados. Porém desta vez o valor do atributo *live* será salvo como sendo 0 (zero) por não se tratar de uma coleta em tempo real.

Uma das dificuldades encontradas para a criação dessa funcionalidade foi para escolhermos os campos que gostaríamos de extrair dos dados retornados, para isso teve de ser feita uma análise do HTML recebido como resposta da URL, podemos ver uma parte do HTML na figura 13, retornado pela URL do Twitter, a partir daí podemos definir os campos a serem salvos à vontade.

```
      <li class="js-stream-item stream-item stream-item
" data-item-id="980591879166177280"
id="stream-item-tweet-980591879166177280"
data-item-type="tweet"
><div class="tweet js-stream-tweet js-actionable-tweet js-profile-popup-actionable dismissible-content
       original-tweet js-original-tweet
"
data-tweet-id="980591879166177280"
data-item-id="980591879166177280"
data-permalink-path="/Secretarial_AP/status/980591879166177280"
data-conversation-id="980591879166177280"

data-tweet-nonce="980591879166177280-486b3098-c5f3-4c02-b1b9-0966df35ef9e"
data-tweet-stat-initialized="true"

  data-screen-name="Secretarial_AP" data-name="ð•Šð•–ð•”ð•£ð•–ð•¥ð•’ð•£ð•šð•’ð•_ð•”,â„™" data-user-id="794678393501581312"
  data-you-follow="false"
  data-follows-you="false"
  data-you-block="false"
```

Figura 13: Pequeno trecho de HTML retornado da coleta de dados.

Outro problema intermitente e que quando esse tipo de coleta e executado diversas vezes seguidas, ela é bloqueada pelo próprio Twitter e não é possível efetua-la novamente por um tempo, nesse período de tempo é retornado um HTML vazio pela URL.

3.3.3. Mineração de Intenções

3.3.3.1. Pré-Processamento

Antes que o treinamento do classificador e a classificação dos dados já coletados pelo usuário sejam feitos, os dados devem passar pela etapa de pré-processamento, por isso foi criado um script especificamente para essa função, para chegar ao processo de pré-processamento atual, foram realizados diversas validações utilizando-se do método de *cross validation k-fold* (que será abordado mais à frente).

A função de pré-processamento é chamada sempre que o treinamento ou classificação dos dados é executado, iremos indicar todas as etapas realizadas em nosso pré-processamento demonstrando as entradas e saídas dos dados.

Em um primeiro momento pode ser observado na tabela 2 as entradas que serão recebidas na função, que correspondem ao atributo texto dos dados coletados.

Tabela 2: Entrada crua dos dados para pré-processamento.

Oddly enough same look when we tell #flatearth believers to get off their ass and travel and put their money where… https://t.co/03bbOMYPGK
The latest The Architecture News Daily! https://t.co/ZzoYHHZBWp Thanks to @dulger_s @josetomasfr @excitingleon #travel

Assim que esses dados são recebidos, eles são convertidos para letra minúscula, pois assim ficará mais fácil de realizar a limpeza necessária nos dados, visto que os procedimentos são *case sensitive*, ou seja, diferenciam letras maiúsculas de minúsculas. Após a conversão ter sido feita é realizada a remoção de todas as *hashtags* (#) dos *tweets*, porém o texto do mesmo é mantido, pois queremos avaliar as *hashtags* (#) como sendo palavras comuns e não *tags*, a saída dessa etapa pode ser vista na tabela 3.

Tabela 3: Saída de dados da primeira etapa de pré-processamento

oddly enough same look when we tell flatearth believers to get off their ass and travel and put their money where… https://t.co/03bbomypgk
the latest the architecture news daily! https://t.co/zzoyhhzbwp thanks to @dulger_s @josetomasfr @excitingleon travel

A próxima etapa a ser realizada e a retirada de todas as menções e links presentes nos *tweets*, pois os mesmos podem causar ruídos e não são necessários para o treinamento e análise. Então por se tratar de um pré-processamento de linguagem natural, é removido os espaços que podem ser excessivos em alguns *tweets*, de modo que não haja mais de um espaço entre cada palavra, a saída da segunda etapa pode ser observada na tabela 4.

Tabela 4: Saída da segunda etapa do pré-processamento.

oddly enough same look when we tell flatearth believers to get off their ass and travel and put their money where…
the latest the architecture news daily! thanks to travel

Agora que os dados já foram devidamente "limpos" pode ser feita a tokenização dos mesmos, para realizar a tokenização de nossos dados utilizamos o módulo casual do NLTK com um sub módulo específico para tokenizar *tweets* chamado *TweetTokenizer*, podemos ver na figura 14 um exemplo de utilização deste sub módulo, onde podemos observar que ele mantém juntos *emoticons*, e na tabela 5 podemos observar a saída após a realização da tokenização.

```
In [1]: from nltk.tokenize import TweetTokenizer

In [2]: tknzr = TweetTokenizer()

In [3]: text = "This is a coool #dummysmiley: :-) :-(<3 and some other stuffs ->
:-P"

In [4]: tknzr.tokenize(text)
Out[4]:
['This',
 'is',
 'a',
 'coool',
 '#dummysmiley',
 ':',
 ':-)',
 ':-(',
 '<3',
 'and',
 'some',
 'other',
 'stuffs',
 '->',
 ':-P']
```

Figura 14: Exemplo do uso do sub módulo *TweetTokenizer*.

Tabela 5: Saída após tokenização.

[*'oddly', 'enough', 'same', 'look', 'when', 'we', 'tell', 'flatearth', 'believers', 'to', 'get', 'off', 'their', 'ass', 'and', 'travel', 'and', 'put', 'their', 'money', 'where', '…'*]
[*'the', 'latest', 'the', 'architecture', 'news', 'daily', '!', 'thanks', 'to', 'travel'*]

Depois de realizar a tokenização, a próxima etapa será realizar a remoção das *stopwords* dos dados, que consiste na remoção de palavras que são consideradas irrelevantes para o treinamento e análise dos dados, para efetuar a remoção das *stopwords*, utilizamos a *stoplist* (lista com palavras a serem removidas), no idioma inglês, fornecida pelo NLTK, porém, foi realizada a exclusão desta lista os pronomes pessoais (*i, you, he, she, it, we, they*) e os pronomes possessivos (*my, your ,his , her, its, our , their*), durante a *cross validation* foi verificado que não é interessante realizar a exclusão de pontuações e emojis, pois isso nos

permitirá melhor classificar a intenção dos tweets, nos indicando uma possível dúvida ou sarcasmo, a saída desta etapa pode ser verificada na tabela 6.

Tabela 6: Saída após remoção de stopwords.

['*oddly*', '*enough*', '*look*', '*we*', '*tell*', '*flatearth*', '*believers*', '*get*', '*their*', '*ass*', '*travel*', '*put*', '*their*', '*money*', '*…*']
['*latest*', '*architecture*', '*news*', '*daily*', '*!*', '*thanks*', '*travel*']

Com a remoção das *stopwords* feitas podemos realizar a stemização das palavras dos *tweets* com intuito de normalizar as mesmas, a stemitização nada mais é que simplificar palavras até a sua raiz. Para stemização foi utilizada a função *Snowball Stemmer*, que possui suporte para o idioma inglês, mas também realizamos testes utilizando o Porter *Stemmer*, podemos ver uma singela diferença entre os dois na figura 15. Podemos analisar também na tabela 7 o retorno final do pré-processamento.

```
>>> print(SnowballStemmer("english").stem("generously"))
generous
>>> print(SnowballStemmer("porter").stem("generously"))
gener
```

Figura 15: Comparação entre *english* e Porter *Stemmer*. (NLTK, 2017)

Tabela 7: Retorno do pré-processamento.

['*odd*', '*enough*', '*look*', '*we*', '*tell*', '*flatearth*', '*believ*', '*get*', '*their*', '*ass*', '*travel*', '*put*', '*their*', '*money*', '*…*']
['*latest*', '*architectur*', '*news*', '*daili*', '*!*', '*thank*', '*travel*']

Com isso nosso pré-processamento é finalizado, podendo dar-se início ao treinamento do algoritmo de classificação ou análise dos *tweets* coletados. Na figura 16 podemos ver o processo de pré-processamento.

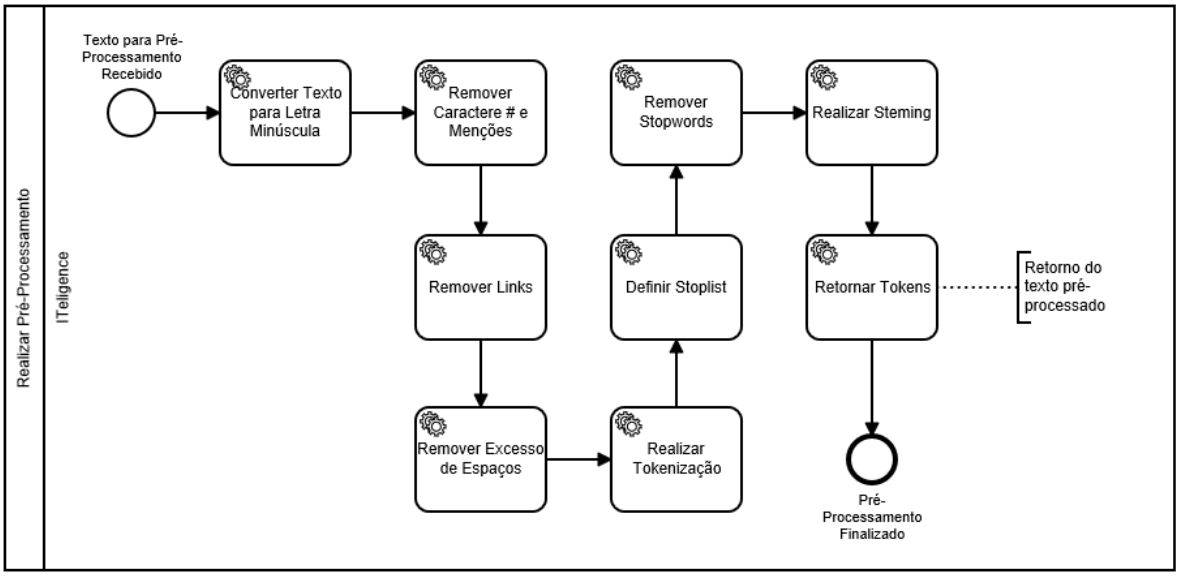


Figura 16: Processo de pré-processamento.

3.3.3.2. Treinamento

Para o treinamento do algoritmo de classificação ser realizado, primeiramente devemos possuir um conjunto de dados classificados manualmente para ser utilizado para criar o classificador, partindo da premissa que esse arquivo já existe, o sistema realiza a leitura do mesmo realizando a separação dos dados em duas listas distintas, uma delas para valores positivos (que expressará a intenção de realizar ou tornar algo real) denotados pelo número 1 (um) e a outra lista para dados negativos (demonstrará que a pessoa não está expressando uma intenção) representados pelo número 0 (zero). Os dados deveram ser classificados como dados que expressam uma intenção ou não, assim não haverá a classificação neutra dos dados, pois iremos levar em consideração que uma frase irá expressar ou não uma intenção, não havendo um meio termo nesta classificação.

Após ter sido realizada a leitura dos dados previamente classificados, e terem sido devidamente separados nas listas de dados positivos e negativos de acordo com suas classificações, será criada uma lista concatenando essas duas listas respectivamente, ficando assim ordenada com dados positivos e logo após dados negativos. A lista então passará linha por linha pelo método de pré-processamento, sendo retornado ao final os *tokens* que serão utilizados para o treinamento.

Com os *tokens* já acessíveis, é chamada de frequência de palavras ,que pode ser vista na figura 17, recebendo esses *tokens* como parâmetro, que retornará como resultado uma

lista de palavras ordenada de forma decrescente pela frequência de menção de cada palavra, essa lista então é salva em uma pasta interna do sistema no formato de arquivo .pickle (arquivo que permite que objetos sejam serializados e desserializados no Python em tempo de execução) de modo que ela possa ser acessada a qualquer momento pelo sistema, para a realização da análise e classificação.

```
27 ##Calculate word frequencies then eliminate duplicates
28 def getwordfreqs(training):
29     uniquelist = []
30     wordfreq = nltk.FreqDist(training)
31     ##For each word
32     for x in training:
33         uniquelist.append((x,wordfreq[x]))
34     ##Eliminate duplicates
35     uniquelist = list(set(uniquelist))
36     ##Order the uniquelist by word frequency
37     uniquelist = sorted(uniquelist, key= lambda tup: tup[1])
38     ##Reverse the list to show higher frequencies first
39     uniquelist.reverse()
40     ##
41     return uniquelist
```

Figura 17: Função de frequência de palavra.

Com a lista de palavras mais frequentes devidamente salva ainda devemos realizar a criação do classificador para ser utilizado pelo algoritmo Naive Bayes. Para a criação desse classificador foi realizado primeiramente uma extração das *features* já classificadas, junto com seus respectivos *labels* (classificação), essa função irá converter cada valor de entrada em *features*, cada uma com seu respectivo *label*, que serão aplicados para a criação do classificador, esse classificador é gerado utilizando-se das *features*, e então é salvo a exemplo da lista de frequência de palavras como um arquivo pickle, podemos ver uma representação da criação do classificador na figura 18 item a.

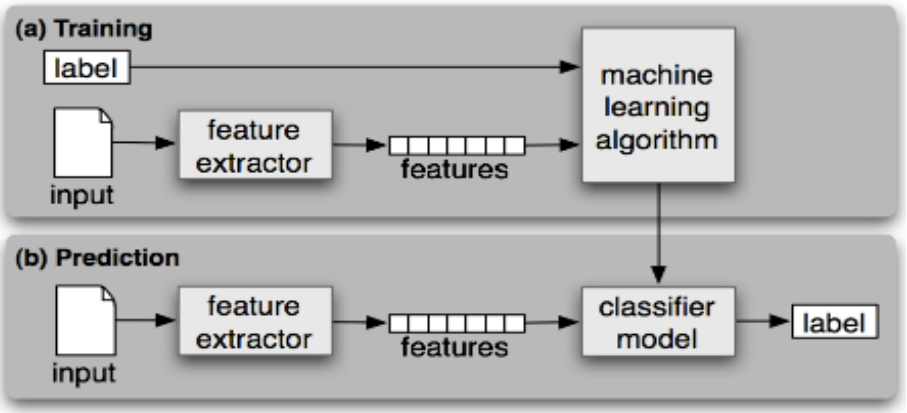


Figura 18: Representação da criação do classificador. (NLTK, 2017)

3.3.3.3. Classificação

Para que o sistema realize a análise e classificação dos dados coletados foi utilizado o algoritmo de classificação disponibilizado pela NLTK, que utiliza o classificador Naive Bayes. Para que a análise seja realizada é necessário que o usuário já tenha efetuado os passos de coleta de dados e de treinamento, caso contrário ele não conseguirá prosseguir com a classificação.

Para que os dados com classificação positiva sejam separados dos dados que possuem classificação negativa, são criados previamente dois *data frames*, esses *data frames* serão o local de armazenamento temporário dos dados conforme eles forem sendo classificados.

Partindo da premissa que o processo de treinamento já tenha sido executado e os arquivos necessários para o nosso classificador tenham sido salvos, poderemos realizar a análise dos dados. Então podemos realizar a seleção dos dados que foram coletados na etapa de coleta de dados, podendo eles serem coletados em tempo real ou coletados dentro de um espaço de tempo. Esses dados são armazenados temporariamente em um *data frame*, e então cada este data frame é passado pela função de pré-processamento, verificando a existência de dados duplicados, *retweets* e quebras de linha, fazendo assim a base de dados ficar mais homogênea é sem ruídos por conter dados duplicados e *retweets*, que poderiam vir a ser spams, e os *retweets* nada mais são do que usuários propagando um *tweet* de outro usuário do Twitter, e o foco da análise de intenções realizada será na intenção do usuário, e não no que ele compartilha, pois seria a intenção de um terceiro.

Após efetuar essa primeira limpeza no *data frame*, cada registro do mesmo, ou seja, cada texto do tweet, irá passar pela função de pré-processamento que foi abordada neste capítulo, está função irá retornar o texto tokenizado, sem as *stopwords* e stemitizado, esse retorno será salvo no *data frame* substituindo o texto anterior de cada registro respectivo ao mesmo. Após os dados de entrada da análise já estarem pré-processados será feita a abertura do arquivo classificador que foi criado na etapa de treinamento, então, a exemplo do que ocorreu na etapa de treinamento, é realizada a extração de *features* de cada registro e o resultado dessa função será passado como parâmetro na função de classificação de dados do classificador. Para a criação de *features* é utilizada a listagem de palavras com maior frequência que foi gerada na etapa de treinamento, para cada palavra contida na lista de palavras mais frequentes, caso essa palavra exista no *tweet*, uma *feature* é criada, o processo principal da classificação pode ser visto na figura 19.

```
107     ##
108     ##For each line at the dataframe will be done several processes
109     for index, row in tweets.iterrows():
110         ##Call pre-process function to text into each line of dataframe
111         tokens = ppc.main(row["text"])
112         ##
113         logging.info('Classifying tweets')
114         ##Call function to classify the line accordin to the pickle files
115         result = classifier.classify(feature_extractor(tokens))
116         ##
117         ##If classifier returns wish results
118         if(result == 'wish'):
119             ##Append the line into poslist dataframe
120             poslist = poslist.append({'id':row['id'], 'screen_name':row['screen_name'],
121                                       'created_at':row['created_at'],
122                                       'text':row['text'], 'intention':'1'}, ignore_index=True)
123         ##Else classifier returns non wish
124         elif (result == 'non-wish'):
125             ##Append the line into poslist dataframe
126             neglist = neglist.append({'id':row['id'], 'screen_name':row['screen_name'],
127                                       'created_at':row['created_at'],
128                                       'text':row['text'], 'intention':'0'}, ignore_index=True)
```

Figura 19: Processamento dos dados coletados.

O retorno da função de classificação do classificador será *wish* para o caso do tweet possuir intenção e *non-wish* para o caso do *tweet* não possuir intenção. Se o resultado for *wish*, o *tweet* é adicionado ao *data frame* de dados positivos, e a coluna do *data frame intention* receberá o valor 1 (um), caso contrário o *tweet* será adicionado ao data frame negativo e o valor de *intention* será definido como 0 (zero). Com as listas já completas, é realizado um *UPDATE* na base de dados para cada tupla que foi classificada, como podemos ver na figura 20, inserindo assim o valor da intenção do tweet na base para futuras análises.

```
135     ##
136     ##Update each analyzed tweet
137     ##Insertion of positive intentions
138     for index, row in poslist.iterrows():
139         update_query = ('UPDATE tweets SET intention = ' + str(1) + ' WHERE id = ' + row['id'])
140         cursor.execute(update_query)
141         ##Commit changes
142         db.commit()
143     ##
144     ##Insertion of negative intensions
145     for index, row in neglist.iterrows():
146         update_query = ('UPDATE tweets SET intention = ' + str(0) + ' WHERE id = ' + row['id'])
147         cursor.execute(update_query)
148         ##Commit changes
149         db.commit()
```

Figura 20: *Update* de *tweets* classificados.

O processo de classificação pode ser visto na figura 21. Após a realização da classificação dos dados o usuário ainda poderá optar por realizar a exportação dos dados classificados, que se encontram disponíveis na tabela de *tweets*, para o formato CSV, de modo que possa ver os resultados e manipular o mesmo caso desejado.

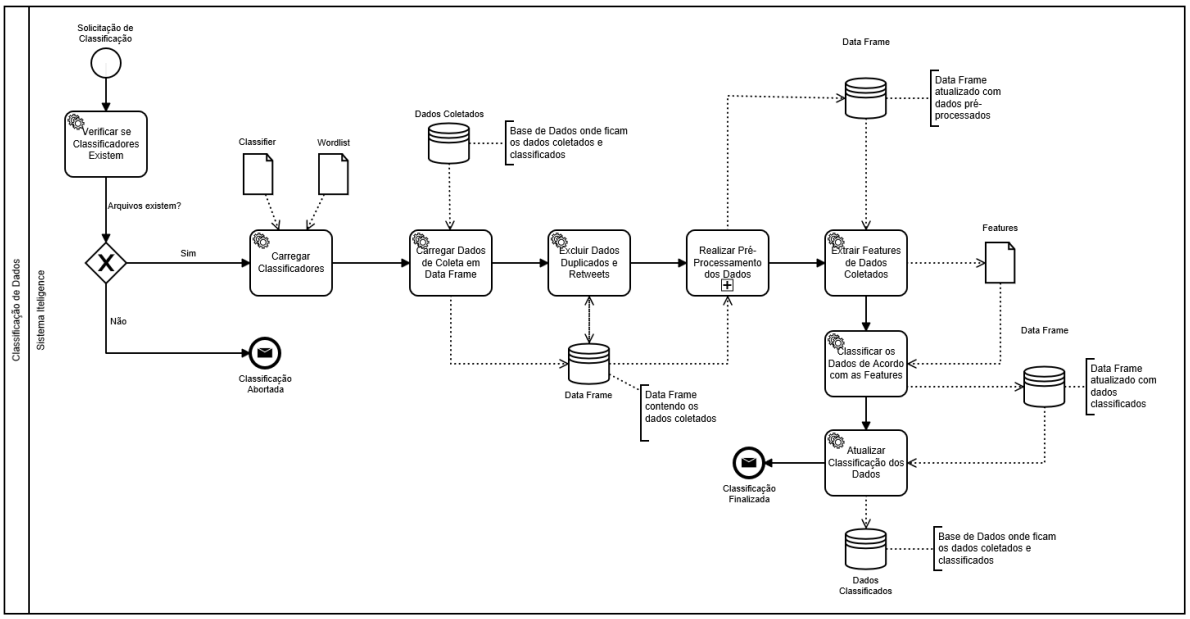


Figura 21: Processo de classificação dos dados.

Um problema encontrado durante a implementação da análise foi a dificuldade de recuperação do *index* do *tweet* da base de dados, pois quando era executado o armazenamento no *data frame* ele estava perdendo sua formatação, para contornar este problema, após o armazenamento dos dados deve ser feita a conversão do index, para refletir o index presente na base de dados.

3.3.3.4. Cross Validation

Após terminarmos a classificação manual dos dados da nossa massa de dados que foi coletada para treinamento, realizamos o *cross validation* utilizando essa massa de dados para avaliarmos se nosso método de treinamento e classificação estava com uma margem de erros aceitável. Após a realização repetidamente do *cross validation* algumas mudanças foram feitas no pré-processamento dos dados, fazendo o mesmo ser ao que é hoje, abaixo iremos descrever o processo que foi realizado para essas mudanças.

Para a realização do *cross validation*, escolhemos o método *k-fold*, definindo k como 10 (dez), ou seja, a cada uma das dez iterações, nossa massa de dados de treinamento foi separada em 10 (dez) conjuntos com o mesmo tamanho, cada um contendo 10% do total da massa de dados, de modo que não houvessem registros repetidos em nenhum dos conjuntos, e os mesmo não fossem sequenciais. Cada um desses conjuntos será classificado com o classificador gerado na etapa de treinamento utilizando os 9 (nove) conjuntos restantes, pode ser observado na figura 22 a separação de conjuntos, e então sua taxa de erro será retornada,

ao final das 10 execuções as taxas serão somadas e divididas pelo número de execuções ou seja 10.

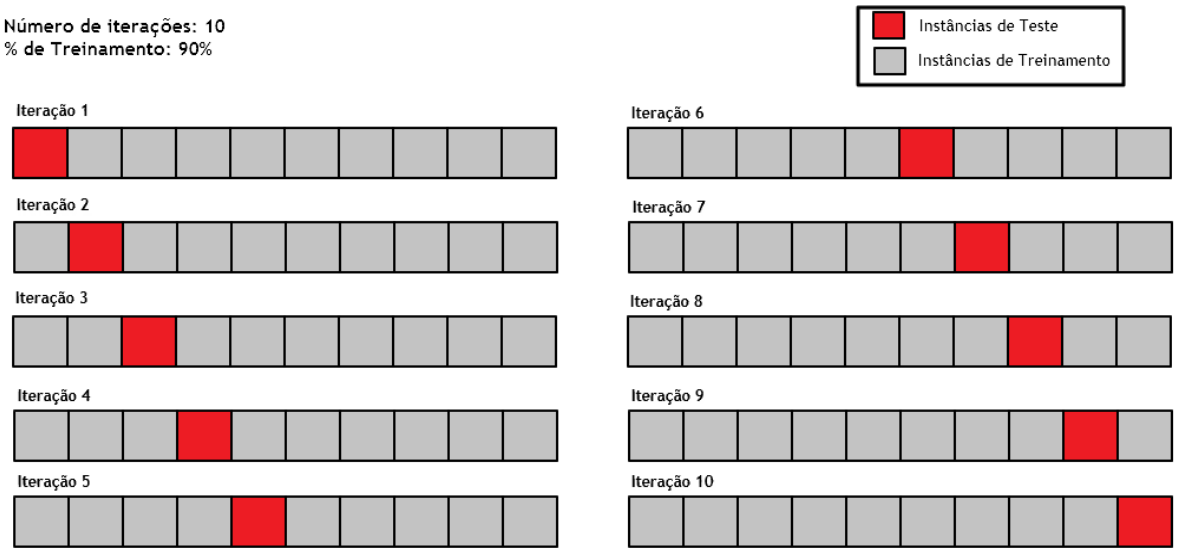


Figura 22: Conjuntos do *cross validation*.

A fórmula utilizada para o retorno do percentual de erro após o *cross validation* é mostrada na equação 2.

$$Ac_f = \frac{1}{v}\sum_{i=1}^{v}(y_i - \hat{y}_i)$$

Equação 2: Fórmula utilizada no *cross validation*.

Interpretando a fórmula acima, a variável v é o número de *folds* utilizados para efetuar a *cross validation*, 10 para o nosso caso, y_i é a porcentagem inicial, ou seja 1 (100%), e $\hat{y}_i$ a porcentagem de classificações realizadas corretamente, sendo assim, a expressão (y_i - $\hat{y}_i$), retorna a porcentagem de erro de uma classificação, podemos assim dizer que a fórmula é o somatório do percentual de erro de cada iteração dividido pelo número de iterações, ou seja a média da taxa de erro.

Na primeira execução do método de *cross validation*, a média de taxa de erro retornada foi de 0.1794 (17,94%), porém percebemos que poderíamos abaixar está taxa manipulando apenas o pré-processamento, sem realizar alterações no treinamento ou classificação.

A primeira mudança a ser realizada foi a troca da função de *tokenizer*, em um primeiro momento a função *wordpunct_tokenize* era a função utilizada para realizar a tokenização dos dados, que é representada na figura 23, porém após leitura da documentação do pacote *tokenizer* do módulo NLTK, foi verificado que havia uma função que se adequaria melhor para o nosso cenário de análise de intenções do Twitter, trata-se da função o *TweetTokenizer*, com essa simples alteração conseguimos reduzir a taxa de erro da classificação para 0.1758 (17,58%).

```
>>> from nltk.tokenize import regexp_tokenize, wordpunct_tokenize, blankline_tokenize
>>> regexp_tokenize(s, pattern='\w+|\$[\d\.]+|\S+')
['Good', 'muffins', 'cost', '$3.88', 'in', 'New', 'York', '.',
'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
>>> wordpunct_tokenize(s)
['Good', 'muffins', 'cost', '$', '3', '.', '88', 'in', 'New', 'York',
 '.', 'Please', 'buy', 'me', 'two', 'of', 'them', '.', 'Thanks', '.']
>>> blankline_tokenize(s)
['Good muffins cost $3.88\nin New York.  Please buy me\ntwo of them.', 'Thanks.']
```

Figura 23: Exemplos de tokenização. (NLTK, 2017)

Após a alteração realizada na tokenização dos dados, foi realizada a verificação se a função de *stemmer* utilizada no pré-processamento poderia ser alterada para a diminuição da taxa de erro. Inicialmente era utilizada a função *SnowBall Stemmer* com o parâmetro de linguagem para o idioma inglês, porém testamos a utilização do *SnowBall Stemmer* com o parâmetro de linguagem como *Porter*, que se caracteriza como um tipo específico de stemização o *Porter Stemmer*, e conseguimos obter um retorno de taxa de erro de 0,1635 (16,35%), a diferença entre os dois processos de *stemmer* pode ser observada na figura 24.

```
>>> print(SnowballStemmer("english").stem("generously"))
generous
>>> print(SnowballStemmer("porter").stem("generously"))
gener
```

Figura 24: Diferenciação dos processos de *stemmer*. (NLTK, 2017)

Durante a classificação realizada manualmente da massa de dados para criação do classificador, verificamos que uma classe gramatical de palavras que pode ser a grande identificadora de expressões que contêm a intenção de uma pessoa ou não, foram essas: pronomes pessoais (*i, you, he, she, it, we, they*), os pronomes possessivos (*my, your ,his ,*

her, its, our, their) e as pontuações, como podemos observar na tabela 8, o impacto dessas palavras.

Tabela 8: Frases com e sem intenção.

Sem intenção	Com Intenção
"you want to make a trip to japan? #japanculture"	*"i want to make a trip to japan... #japanculture"*
"are you planning a trip to the beautiful emerald coast?"	*"our next trip to tapachula will take place from october 13th through the 21st."*

Fizemos o teste deste cenário para verificar qual o impacto que teria na classificação, não realizando a remoção desses pronomes e nem de palavras que possuíssem menos de três caracteres, pois como os dados são coletados de redes sociais, os dados poderiam possuir também indicadores de emoções que podem mudar totalmente o sentido de uma frase (como por exemplo ";)" que pode indicar sarcasmo ou ":´(" que pode indicar algo que não irá ocorrer). Essas alterações foram as que causaram mais impacto na taxa de erro, após a realização do *cross validation*, com elas já aplicadas, tivemos uma taxa de erro de 0,1358 (13,58%), na tabela 9 abaixo podemos ver o impacto de cada mudança.

Tabela 9: Comparação das taxas de erro.

Alteração	Taxa de Erro	Diferença
-	17,94%	-
Tokenização	17,58%	0,36%
Stemização	16,35%	1,23%
Stopwords	13,58%	2,77%

3.3.3.5. Análise de Resultados

Com os dados já devidamente classificados e salvos em nossa base de dados, o usuário poderá realizar a análise desses dados, ele terá a opção de poder realizar a exportação desses dados para CSV, e também selecionar para que o programa gere gráficos para sua análise.

3.3.3.6. Países Mais Mencionados

Depois que a análise e classificação dos dados é realizada, o usuário poderá optar por fazer uma série de atividades manipulando esses dados já classificados, uma dessas atividades é a criação de um gráfico em barras que indica os 35 (trinta e cinco) países que foram mais mencionados nos dados classificados previamente.

Para realizar a criação desses gráficos primeiramente realizamos o armazenamento temporário do atributo texto dos dados que já foram classificados, já efetuando a divisão desses dados em 2 (dois) data frames, um de dados que possuem intenção e outro de dados que foram classificados como sem intenção. Com os dois *data frames* já populados podemos realizar a chamada do módulo *geotext*, mas somente após o pré-processamento.

O pré-processamento nesta funcionalidade é mais simplificado, os únicos passos feitos são: remoção de caracteres que não sejam alfanuméricos, remoção de menções a outros usuários, e remoção de possíveis links. Com isso deixamos o texto pronto para o processamento.

Então feita a chamada do módulo *geotext*, passando os *data frames* como parâmetro, o retorno desta função será um objeto do tipo *geotext*, este objeto permitirá chamar funções do módulo de modo que os gráficos possam ser criados

A função do módulo *geotext* que utilizamos para receber como retorno os países que possuem mais menções é a função *countrie_mentions*. Essa função identifica no texto países, cidades, lugares e estados utilizando-se de um dicionário para sua identificação, a função consegue identificar ainda utilizando-se do seu próprio dicionário a qual país os dados remetem, ou seja, caso ele reconheça a cidade “Rio de Janeiro”, será retornado uma menção para o Brasil. O retorno da função será um *ordered dictionary* (dicionário ordenado) com os países e seus números de menções em ordem decrescente, ou seja, com os países mais mencionados na frente, a entrada e saída podem ser observadas na tabela 10.

Com esse retorno separamos os 35 (trinta e cinco) maiores valores, e então, é chamado um dicionário que foi criado para que as abreviações dos países sejam substituídas por seu nome completo.

Tabela 10: Dados processados pelo geotext.

Entrada	Saída	Substituição
Rio de Janeiro is a beautiful place to visit, but is dangerous!	*OrderedDict([('BR', 3), ('US', 3), ('IT', 2), ('FR', 2), ('NL', 1), ('GB', 1)])*	*OrderedDict([('Brazil', 3), ('United States', 3), ('Italy', 2), ('France', 2), ('Netherlands', 1), ('United Kingdom', 1)])*
Salvador is the city of carnival		
all day is party day in Miami		
Paris the city of light and love		
Nice is a beach city		
Texas what a beautiful city		
New York, the big apple		
Brazil, the country of soccer		
Amsterdam is a great place to visit		
let's eat pasta in Italy		
i love to go to London, call me #travel		
Rome is a great place to live		

Com a substituição realizada poderemos utilizar este *ordered dictionary* para a criação dos gráficos, para o desenho dos gráficos foi utilizado o módulo maptolib. Os gráficos criados serão do tipo gráfico em barras, um exemplo de gráfico utilizando o *ordered dictionary* da coluna “Substituição” da tabela 10 pode ser observado na figura 25.

Antes de executar a função da criação dos gráficos é feita uma verificação se existem dados disponíveis para o gráfico em questão, seja ele negativo, positivo, de dados coletados em tempo real ou de dados coletados em intervalo de tempo. Após o gráfico já ter sido criado o mesmo é salvo como um arquivo de imagem em uma pasta interna do sistema.

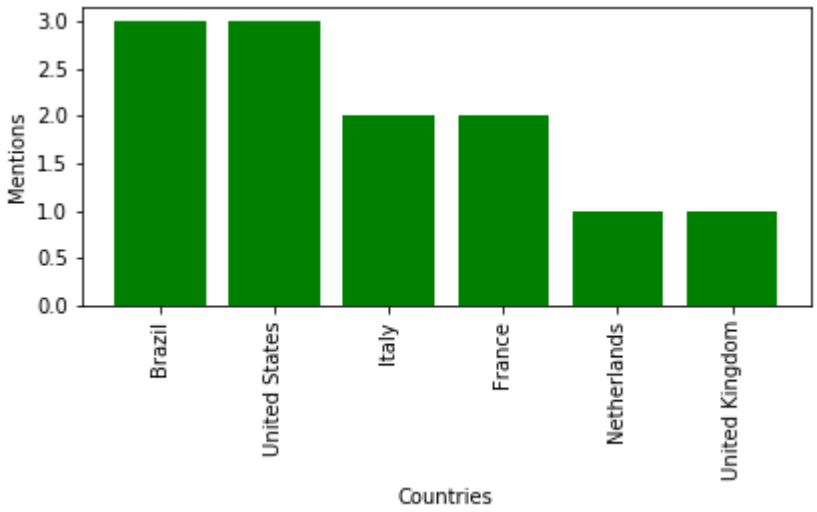


Figura 25: Gráfico em barras gerado pelo maptoblib.

Essas imagens estarão disponíveis para que o sistema possa acessá-las a qualquer momento, quando o usuário entrar na galeria de imagens.

3.3.3.7. Substantivos e Adjetivos mais mencionados

Além de poder gerar gráficos de países com mais menções nos dados que foram classificados, o usuário também poderá gerar gráficos dos substantivos e adjetivos mais mencionados, esses gráficos serão gráficos em linha, divididos da seguinte maneira: Tipo de coleta, sendo gerados 4 gráficos para cada um dos dois tipos, 2 gráficos de substantivos mais mencionados e 2 gráficos de adjetivos mais mencionados, sendo um desses para dados que expressam uma intenção e um gráfico para dados que não expressam intenção.

Para realizar a criação desses gráficos primeiramente realizamos o armazenamento temporário do atributo texto dos dados que já foram classificados, já efetuando a divisão desses dados em dois *data frames*, um de dados que possuem intenção e outro de dados que foram classificados como sem intenção. Com os dois *data frames* já populados podemos chamar a função do pré-processamento dos dados, porém de uma maneira um pouco diferente da função de pré-processamento que é executada para a criação do classificador e a classificação dos dados.

No pré-processamento realizado para desenho destes gráficos é realizada a conversão para letras minúsculas, para que não haja diferenciação das palavras por conta de letras maiúsculas, depois é feita a remoção do caractere *hashtag* (#), a remoção de todas as menções, e então a remoção de qualquer tipo de link presente no texto, após serem feitos esses passo, são removidas também dos dados todas as palavras que foram definidas pelo usuário que não devem ser contabilizadas nos gráficos, essas palavras são definidas nas configurações de preferências dos usuários. Com o texto já limpo de dados não necessários e formatado, é realizada a tokenização dos dados, utilizando-se da função *TweetTokenizer*, e então é feita a remoção das *stopwords*, desta vez todas as *stopwords* da lista disponibilizada no módulo NLTK são removidas, pois não desejamos efetuar contabilização de pronomes nos gráficos, assim como também são removidas todas as palavras ou pontuações que possuam menos de 3 (três) caracteres pois não desejamos que sejam apresentados nos gráficos nenhum tipo de pontuação ou sinal, com isso o pré-processamento está concluído, pois para criação dos gráficos não desejamos realizar a stemitização das palavras, visto que queremos refletir exatamente como elas foram escritas, os textos pré-processados são retornados e então salvos como uma lista única de palavras.

Com o pré-processamento realizado e os dados pré-processados já armazenados temporariamente nos *data frames*, é realizada a etapa de *POS-tagger*, que é um serviço disponível no módulo NLTK para adicionar tags as palavras, o *POS-tagger* processa uma sequência de palavras e as classifica de acordo com a sua semântica, adicionando uma TAG a cada palavra, como pode ser observado na figura 26, onde o texto passa por uma tokenização e logo após passa pelo processo de *POS-tagger*.

```
>>> text = word_tokenize("They refuse to permit us to obtain the refuse permit")
>>> nltk.pos_tag(text)
[('They', 'PRP'), ('refuse', 'VBP'), ('to', 'TO'), ('permit', 'VB'), ('us', 'PRP'),
('to', 'TO'), ('obtain', 'VB'), ('the', 'DT'), ('refuse', 'NN'), ('permit', 'NN')]
```

Figura 26: *POS-tagger* de palavras. (NLTK, 2017)

Após as palavras já estarem classificadas de acordo com sua semântica são selecionados os substantivos e adjetivos presentes nos dados, separadamente, e então é chamada a função para realizar o cálculo de frequência de cada uma dessas palavras que podemos observar na figura 27, utilizando-se do método FreqDist disponibilizado pelo módulo NLTK, retornando cada palavra e sua distribuição de frequências para cada um dos *data frames*.

```
48 ##
49 ##Function to calculate the words frequencies
50 def getwordfreqs(analyzed):
51     ##Calls nltk FreqDist function to return the words frequency
52     wordfreq = nltk.FreqDist(analyzed)
53     ##Returns the words frequencies
54     return wordfreq
```

Figura 27: Função para retorno de frequência de palavras.

Porém para a criação dos gráficos é necessário efetuar a separação das palavras e suas frequências em duas listas, de modo que sua ordem não se perca para que a frequência possa ser ligada a palavra. A entrada inicial que é passada como parâmetro para a função de frequência e a saída final obtida após a separação das frequências podem ser observadas na tabela 11.

Tabela 11: Entrada e saída para criação dos gráficos.

Entrada	Saída
['beautiful', 'dangerous', 'big', 'great', 'call', 'great']	*['great', 'beautiful', 'dangerous', 'big', 'call']* *[2, 1, 1, 1, 1]*

Com isso feito, é verificado se existem dados para a criação dos gráficos, e então os mesmos são desenhados utilizando a biblioteca *matplotlib*, utilizando as frequências para montar o eixo y e as palavras para montar o eixo x, assim o gráfico é salvo como imagem em uma pasta interna do sistema, para que o possa ser acessado pela galeria de imagens a qualquer momento, o gráfico para a saída apresentada na tabela 11 pode ser visto na figura 28.

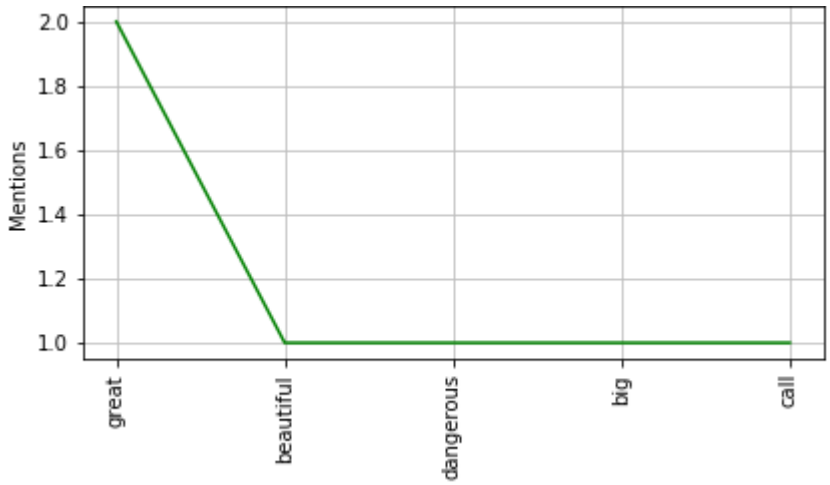


Figura 28: Gráfico de linha de adjetivos mais mencionados.

Como podemos observar no gráfico acima, o processo de POS-tagger, está sujeito a erros, podemos observar que a palavra *call* foi classificada como um adjetivo, porém a mesma se trata de um verbo ou substantivo, isso ocorre porque este método possui uma taxa de sucesso de 97% por *token*, ou seja, palavra (Manning, 2011). Com isso podemos afirmar que há a possibilidade de ocorrerem classificações erradas.

Com as funcionalidades do programa fechadas de acordo com a documentação podemos efetuar a utilização do mesmo.

4. Utilização do Programa

A cada quatro anos, é realizado um evento que é acompanhado pelo mundo todo, e esse evento é a copa do mundo, a Copa do Mundo é um evento esportivo que a cada edição possui sua sede em um país diferente, esse evento é um chamariz e tanto para pessoas de toda parte do mundo praticarem o turismo no país sede. No ano presente (2018) a Copa do Mundo será realizada na Rússia.

No ano de 2014 na Copa do Brasil foram registrados um milhão de turistas estrangeiros no país (R7; Agência Brasil & Estadão, 2014). Com isso esse evento é o cenário ideal para a realização de uma mineração de intenções no domínio de viagens, esperamos mostrar na prática uma das muitas possíveis utilizações de nosso sistema de análise de intenções.

4.1. Coleta dos dados

Para a coleta de dados, definimos que seriam coletados *tweets* que ocorreram antes e durante a realização da Copa do Mundo da Rússia, pois queremos avaliar o impacto que pode ter acontecido nas intenções de viagens em um curto período de tempo antes da Copa se iniciar, e nos dias durante a realização da mesma.

Para a coleta definimos as *Keywords* como "*travel;trip*", ou seja, serão coletados quaisquer dados que possuam em seu campo texto a palavra *travel*, *trip* ou até mesmo as duas. Na tabela 12 verificar todos os dados da coleta realizada para dados de antes do início da Copa do Mundo.

Tabela 12: Coleta dos dados antes da Copa do Mundo.

Começo da Coleta	Duração	Quantidade de Dados	Tipo de Coleta
01/05/2018 - 09:42:50	5 horas	51.639	Tempo Real
02/05/2018 - 07:05:44	5 horas	46.890	Tempo Real
03/05/2018 - 08:36:14	5 horas	51.609	Tempo Real

A quantidade de dados coletados se refere a dados contendo *retweets* e *tweets* repetidos, que não serão utilizados na análise, dessa forma a quantidade total irá diminuir

ainda consideravelmente após o pré-processamento. A coleta retornou um total de 150.138 registros.

Com a coleta de tweets feitos antes do início da Copa realizada, é a vez de coletarmos os tweets feitos durante a Copa do Mundo, dessa vez utilizamos a coleta de dados definindo um espaço de tempo no passado, o registro dos dados coletados pode ser observado na tabela 13.

Tabela 13: Coleta dos dados durante a Copa do Mundo.

Intervalo da Coleta	Quantidade de Dados	Tipo de Coleta
14/06/2018 - 23/06/2018	35.031	Intervalo Temporal

A quantidade de Dados coletados neste tipo de coleta se refere aos *tweets* já contendo a exclusão dos *retweets*, porém ainda com possíveis duplicatas.

Com a etapa de coleta já realizada, o próximo passo que devemos executar é a realização do treinamento, para a criação de um classificador a ser utilizado para a classificação de nossos dados.

4.2. Treinamento

Para o processo de treinamento do classificador, primeiro deverá ser criada uma base com dados para executar este treinamento, estes dados deverão ser classificados manualmente de forma que servirão como parâmetro para realizar um treinamento confiável.

Para a criação da massa de dados para treinamento foi realizada a coleta, utilizando nosso sistema, de 1701 registros sem dados duplicados, que foram coletados utilizando-se dos termos de busca *travel* ou *trip,* podendo assim serem retornados dados que possuam em seu texto as palavras *travel*, *trip* ou ainda as duas.

Essa massa de dados foi então convertida para uma lista separada por vírgulas, sendo também excluídos menções e links que poderiam estar presentes nos textos, essas menções e links iriam tornar a classificação manual confusa, e como os mesmos são excluídos durante o pré-processamento essa exclusão não irá afetar em nada o treinamento e classificação dos dados.

Todos esses dados foram classificados manualmente e cuidadosamente, de modo que a classificação fosse a mais precisa e criteriosa possível, desses 1701 *tweets* de treinamento, apenas 390 *tweets* foram classificados como *tweets* que possuíam intenção de viagem, isso

corresponde a 22.9% do total, ou seja, 77.1% correspondem à *tweets* que não expressaram uma intenção do usuário viajar. Durante a classificação pôde ser observado que o Twitter vem sendo muito utilizado para a realização de propagandas, por esse fator podemos ver uma diferença tão grande entre dados que possuem intenção de viagem e não possuem, na tabela 14 podemos observar a classificação de alguns *tweets* da massa de dados de treinamento, que será utilizada para a classificação se um texto possui ou não a intenção de viagem.

Tabela 14: Dados classificados manualmente.

Tweet	Classificação	Comentários
spring break in japan anyone? read how this group of current students planned an unforgettable trip	Sem intenção	*Tweet* dedicado para promover a leitura de um artigo sobre viagem.
i need to start packing for this weekend's beach trip	Com intenção	*Tweet* de uma pessoa que precisa começar a fazer as malas para viajar.
meghan used to travel to the uk with her friends family take pics outside the palace now she's marrying a prince	Sem intenção	*Tweet* referente a atriz que se casou com o príncipe inglês.
i love my trip to london, next year i will return there	Com intenção	*Tweet* de uma pessoa que fez uma viagem, e tem a intenção de voltar ao lugar.
we offer special rates for corporate travel, military/government travel, group travel, special events and club/comp	Sem intenção	*Tweet* de propaganda de uma agência de viagem.
travel safely!	Sem intenção	*Tweet* de uma pessoa desejando uma boa viagem a alguém.

Também é importante deixar claro que os dados foram classificados, levando em consideração apenas a intenção do usuário de viajar ou não, ou seja, qualquer outra intenção que possa vir a ocorrer que não seja relacionado à vontade do autor em viajar, será considerado como um registro que não possui intenção. Pois queremos apenas intenções de viajar.

Com a massa de treinamento pronta, podemos realizar a etapa de treinamento, e então executar a classificação dos dados coletados previamente.

4.3. Análise dos dados

Para começar toda a análise de dados foi realizada uma extração de mais de 150.000 *tweets* para identificarmos quais países possuem o maior desejo de viagens às vésperas da copa do mundo. Essa massa de dados se mostrou bastante diversificada, mas trouxeram resultados satisfatórios para a pesquisa. Essa sessão será dedicada a análise dos dados recolhidos do primeiro momento da extração, até o seu fim com a criação dos gráficos.

4.3.1. Antes da Copa do Mundo FIFA

Após o pré-processamento dos dados, com a exclusão de dados duplicados e os *retweets*, sobraram 67.417 *tweets* a serem classificados, observamos assim que a grande maioria dos dados coletados no Twitter se refere a dados que foram *retweets* ou spams.

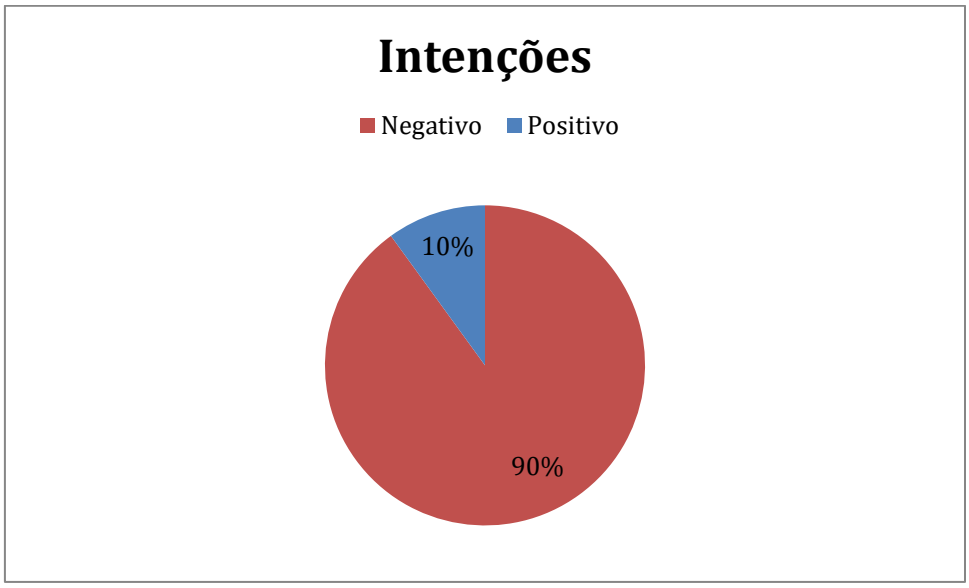


Figura 29: Gráfico de dados positivos e negativos.

Na figura 29 nota-se que apenas 6.741 *tweets* demonstraram intenção positiva de viagem. Em contrapartida o número de *tweets* classificados sem intenção alcançou o 60.671, disparando na frente dos *tweets* com intenção.

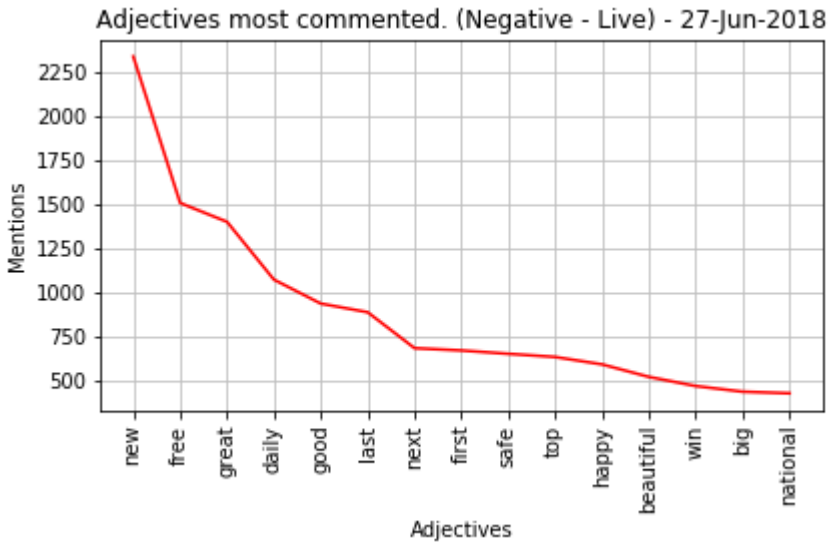


Figura 30: Adjetivos mais frequentes sem intenção.

Neste caso de adjetivos mais comentados sem intenção de viagens, apresentado na figura 30, podemos visualizar que de fato não há uma correlação forte entre as palavras e as frases que demonstram intenção de viagem

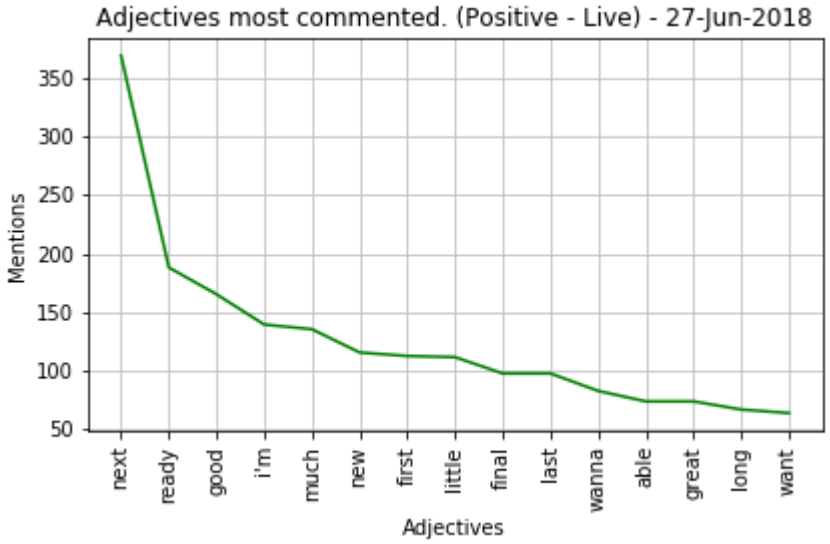


Figura 31: Adjetivos mais frequentes com intenção.

Na figura 31 conseguimos entender bastante da nossa análise, pois adjetivos mais citados como "*next*", "*ready*", "*good*", costumam estar ligadas a sentenças do tipo "*I'm going to travel next week*", "*I'm ready to travel*", etc.

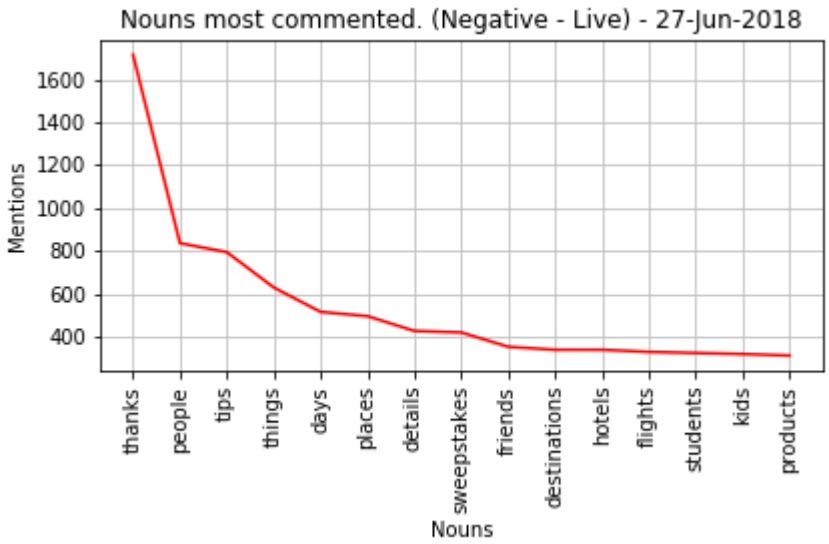


Figura 32: Substantivos mais frequentes sem intenção.

Novamente reforçamos com a figura 32, que não há uma forte correlação entre os substantivos e os desejos de viagens.

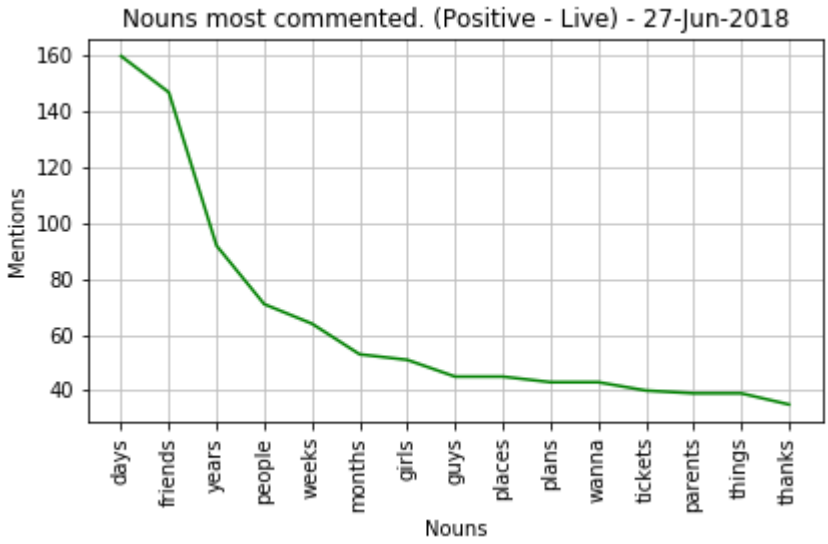


Figura 33: Substantivos mais frequentes com intenção.

Na figura 33 acima, entendemos que os substantivos mais mencionados possuem alto nível de correlação com frases do tipo "*Counting the day*", "*Travel with my friends*", "*I'm going to travel next year*".

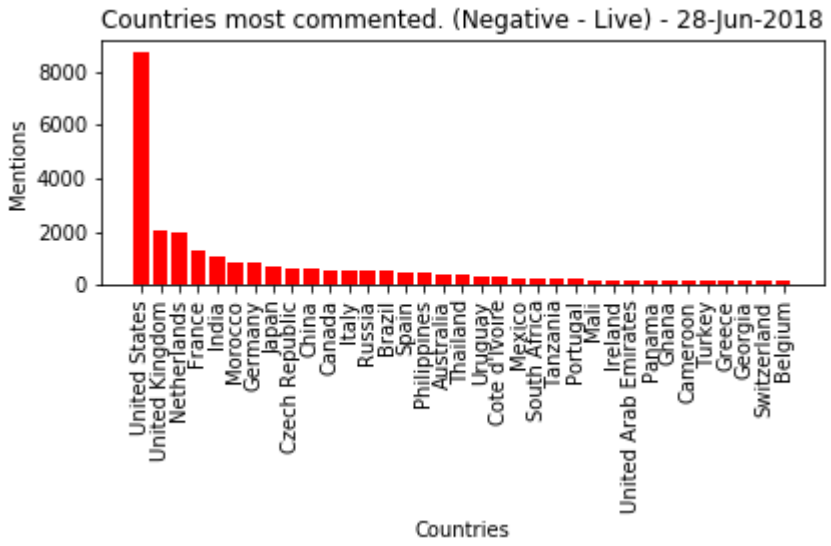


Figura 34: Países com mais menções sem intenção.

Partindo para a análise dos países com mais menções em *tweets* sem intenção de viagem, apresentado na figura 34, é importante frisar que independentemente de o *tweet* ser classificado como positivo, ou negativo os Estados Unidos despontam na frente. Há uma grande menção devido a forte influência que o país tem na economia global, fazendo o mesmo ser o centro das atenções quando tentamos colocamos um filtro por países. Na tabela 15 abaixo, está listado o top 5 (cinco) dos países mais comentados a partir da análise dos *tweets* sem desejo de viagem.

Tabela 15: Países mais mencionados em dados sem intenção.

Posição	País
1	Estados Unidos
2	Reino Unido
3	Holanda
4	França
5	Índia

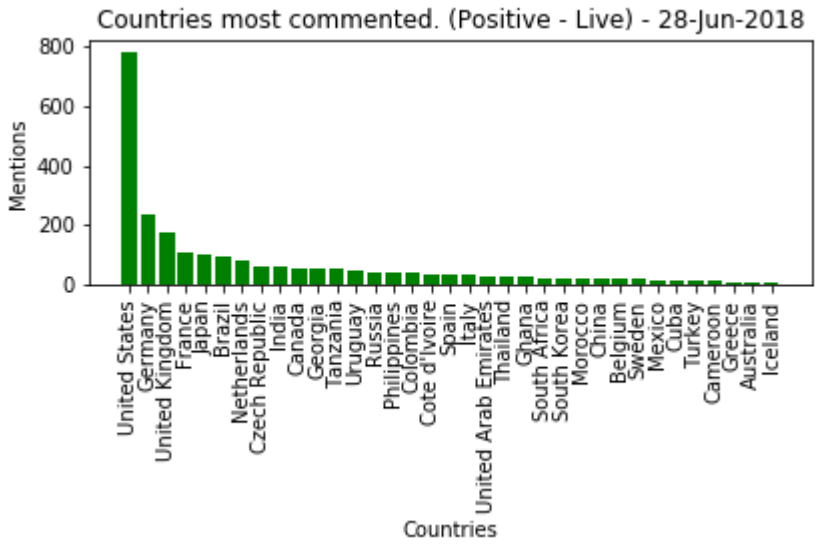


Figura 35: Países mais mencionados em dados com intenção

Neste momento veio a surpresa para a nossa análise, ao analisarmos os dados de países com mais menções em *tweets* classificados com intenção de viagem, apresentados na figura 35, deixaram a Rússia praticamente na mesma posição do que os classificados sem intenção de viagem. Por sediar a Copa do Mundo imaginávamos que receberia o maior número de intenções, mas após análises esse pensamento não foi ratificado. Na tabela 16 abaixo, está listado o top 5 (cinco) dos países mencionados com maior número de intenções de viagens:

Tabela 16: Países mais mencionados em dados com intenção.

Posição	País
1	Estados Unidos
2	Alemanha
3	Reino Unido
4	França
5	Japão

4.3.2. Depois da Copa do Mundo FIFA

Após o pré-processamento dos dados, com a exclusão de dados duplicados, sobraram 14.287 *tweets* a serem classificados, observamos assim que a grande maioria dos dados coletados no Twitter durante a Copa se referem a spams.

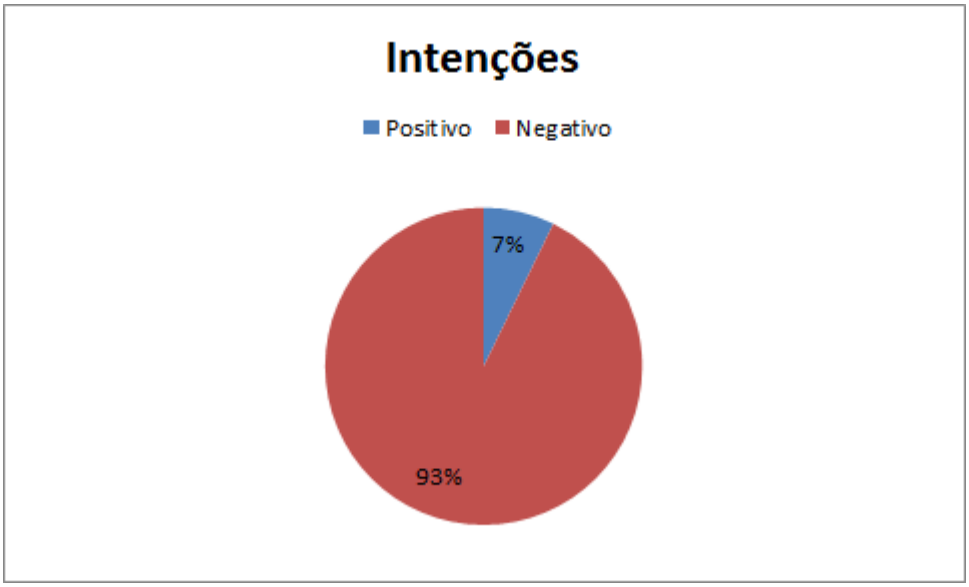


Figura 36: Gráfico de dados positivos e negativos.

Percentuais similares aos encontrados antes da copa do mundo.

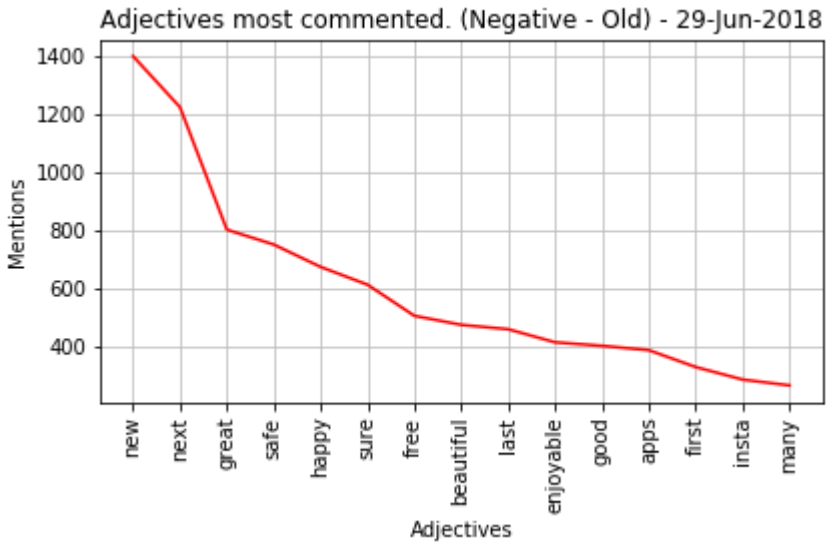


Figura 37: Adjetivos mais frequentes sem intenção.

Mesma análise da figura de adjetivos frequentes sem intenção antes da Copa do Mundo.

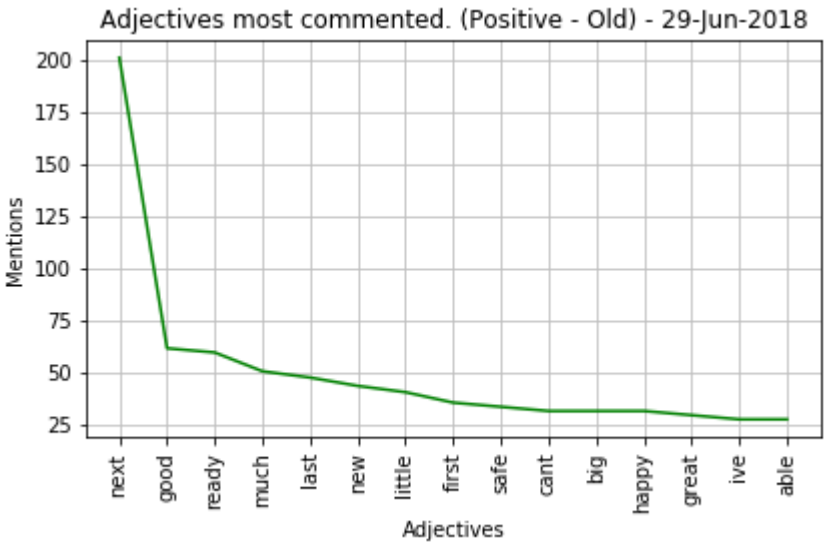


Figura 38: Adjetivos mais frequentes com intenção.

Mesma análise dos adjetivos mais frequentes antes da copa

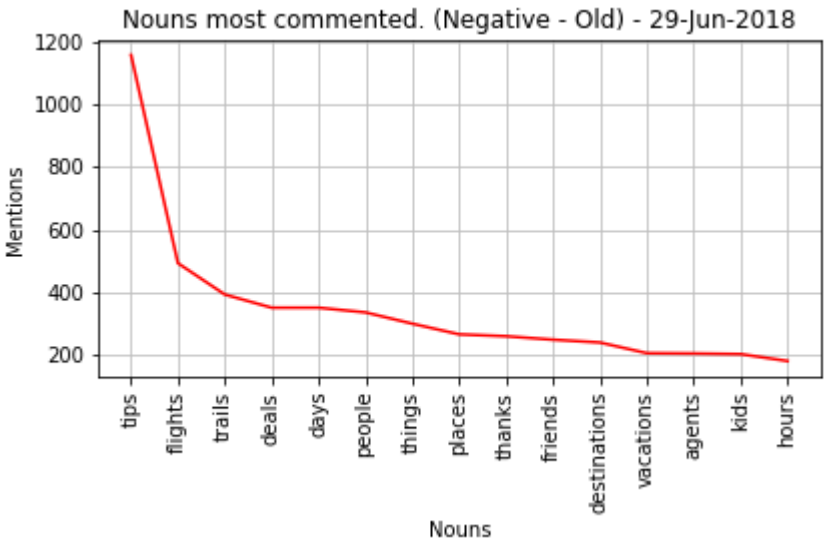


Figura 39: Substantivos mais frequentes sem intenção.

Comparando com a análise feita antes da copa do mundo, há apenas uma alteração quanto a ordem das palavras mais frequentes.

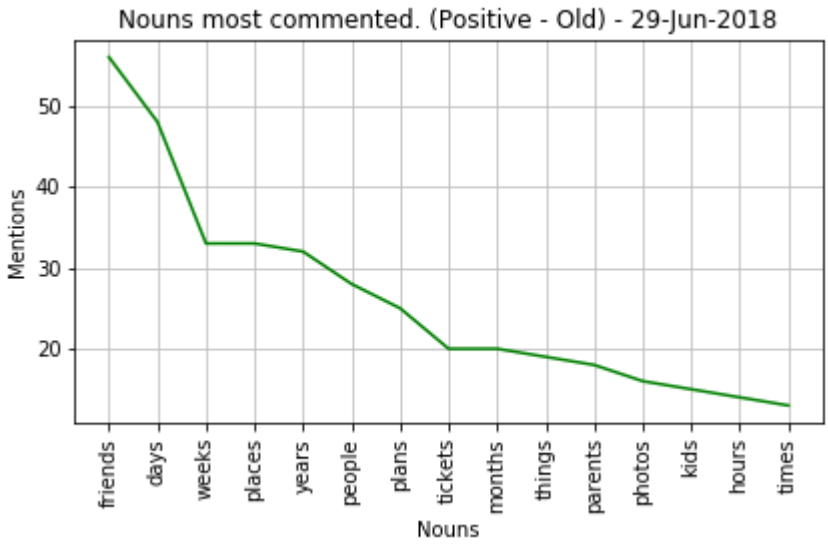


Figura 40: Substantivos mais frequentes com intenção.

Como podemos observar na figura 40, as palavras *Days, Friends, Weeks, Places* aparecem novamente como campeãs de frequência, o que mantém a forte correlação com os *tweets* com intenção.

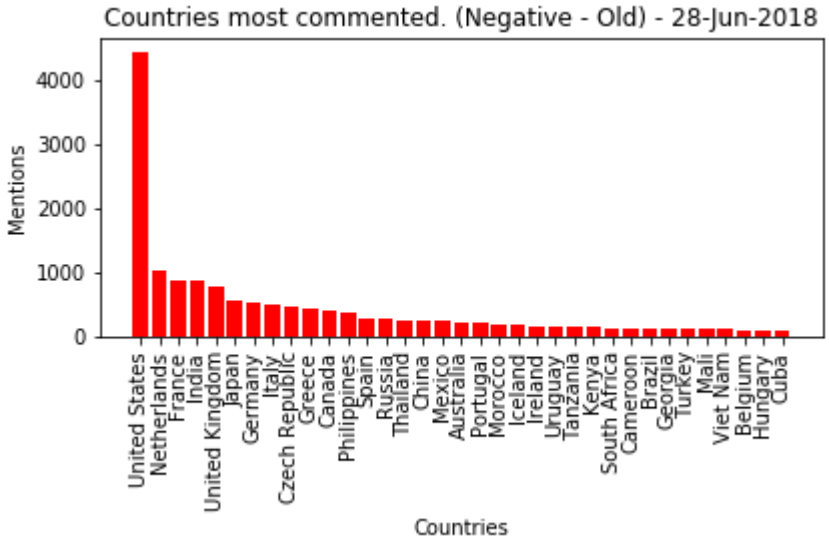


Figura 41: Países com mais menções sem intenção.

Partindo para um cenário durante a realização da Copa do Mundo, os países presentes no top 5 (cinco) dos países sem intenção de viagem continuam sendo os mesmos. No detalhe, que pode ser observado na figura 41, percebemos que Alemanha e Japão deram

lugar para a Holanda e Índia. Na tabela 17 abaixo, está listado o top 5 (cinco) dos países mencionados com maior número de intenções de viagens.

Tabela 17: Países mais mencionados em dados sem intenção.

Posição	Países
1	Estados Unidos
2	Holanda
3	França
4	Índia
5	Reino Unido

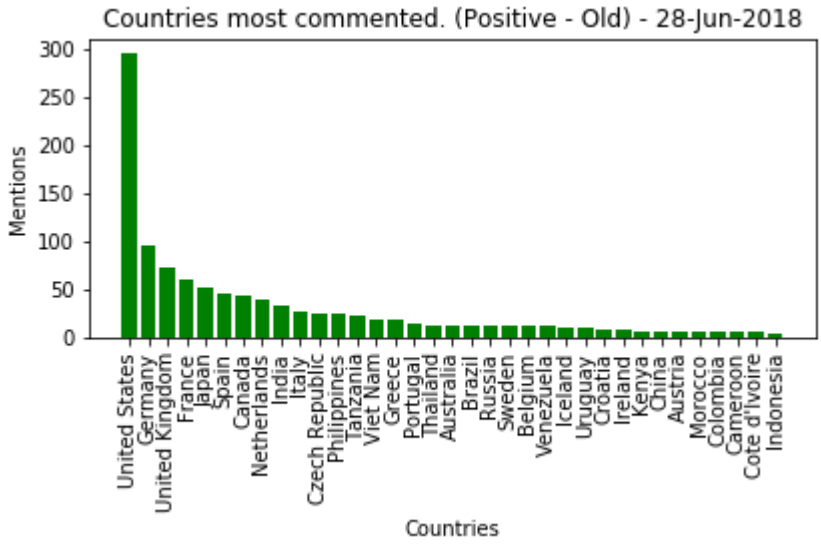


Figura 42: Países com mais menções com intenção.

Se antes da Copa do Mundo o 14º (décimo quarto) lugar da Rússia impressionava, para o ranking de países que possuíam mais menções em *tweets* com intenção de viagem, após o início da copa aparece apenas na 19ª (décima nona) posição, como podemos observar na figura 42. Na tabela 18 abaixo, está listado o top 5 (cinco) dos países com mais menções após o início da Copa do Mundo.

Tabela 18: Países mais mencionados em dados com intenção.

Posição	Países
1	Estados Unidos
2	Alemanha
3	Reino Unido
4	França
5	Japão

5. Conclusão

Este trabalho apresentou como foi feito o desenvolvimento de uma interface capaz de extrair e manipular dados extraídos do Twitter (*tweets*) e como aplicar o processo de Mineração de Textos nesse cenário. Um cenário de classificação de textos foi criado para que fosse permitido a aplicação da Análise de Intenções, além disso, a Copa do Mundo da FIFA Rússia 2018 ofereceu um bom *background* para análise, que nos permitiu entender como que os jogos da Copa do Mundo estão influenciando os desejos de viagens das pessoas. Os *tweets* foram classificados de dois modos, 1 (um) para com intenção, 0 (zero) para sem intenção.

Desta forma, o modelo de classificação desse trabalho nos permitiu entender que por mais que a Rússia esteja sediando os jogos, outros países como Estados Unidos e Alemanha ainda possuem as maiores intenções de viagem. É importante deixar claro que essa análise não comprova que não houve um aumento significativo no número de intenções de viagens para a Rússia. Comprova apenas que nesse período outros países são os mais desejados.

Além disso, é também apresentada nesse trabalho uma sequência de gráficos que auxiliaram todas as análises feitas pelo o grupo, dando credibilidade para a ferramenta desenvolvida. O usuário é capaz de criar gráficos de Pizza e em Linhas que simplificam sua vida na hora que precisarem entender quais foram os substantivos, adjetivos e até os países que apareceram com mais frequência em seu grupo de dados.

Sendo assim, a ferramenta desenvolvida neste trabalho poderia ser utilizada por uma companhia prestadora de serviços de viagens que queira saber quais são os países que aparecem com as maiores intenções de viagens, de forma a oferecer serviços e produtos, ou até ajustar valores prevendo uma maior ou menor demanda.

Visto que um grande evento como a Copa do Mundo não foi capaz de colocar o país anfitrião no primeiro colocado com as maiores intenções, o grupo propõe como trabalhos futuros entender como a análise de intenções de viagens sofre interferência de grandes eventos.

6. Referências Bibliográficas

Stanford NLP. (Abril de 2009). Acesso em 30 de Julho de 2018, disponível em Stanford.

Amo, S. d. (Julho de 2004). Técnicas de Mineração de Dados.

Anaconda. (201-?). *Anaconda Cloud Glossary*. Acesso em 23 de Junho de 2018, disponível em Anaconda: https://docs.anaconda.com/anaconda-cloud/glossary#cloud-glossary-cloud

Camilo, C. O., & Silva, J. C. (Agosto de 2009). Mineração de Dados: Conceitos, Tarefas, Métodos.

Cardoso, O. N., & Machado, T. M. (Junho de 2008). Gestão do conhecimento usando data mining: estudo de caso na Universidade Federal de Lavras. *42*(3).

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business Intelligence and Analytics from Big Data to Big Impact. *36*(4).

Codd, E. F. (1970, Junho). A relational model of data for large shared data banks. Communications of the ACM. *13*(6).

Côrtes, S. d., Porcaro, R. M., & Lifschitz, S. (Maio de 2002). Mineração de Dados - Funcionalidades, Técnicas e Abordagens.

DB Engines. (2018). *MySQL System Properties*. Acesso em 23 de Junho de 2018, disponível em db-engines: https://db-engines.com/en/system/MySQL

Devmedia. (14 de Julho de 2014). *Mineração de texto análise comparativa de algoritimos revista SQL magazine*. Acesso em 30 de Novembro de 2017, disponível em Devmedia: https://www.devmedia.com.br/mineracao-de-texto-analise-comparativa-de-algoritmos-revista-sql-magazine-138/34013

Dicio. (20-). *Intenção*. Acesso em 29 de Junho de 2018, disponível em Dicio: https://www.dicio.com.br/intencao/

El-Khair, I. A. (2006, Dezembro). Effects of stop Words Elimination for Arabic Information Retrieva: A Comparative Study.

Jivani, A. G. (Novembro de 2011). A Comparative Study of Stemming Algorithms.
John, G. H., & Langley, P. (1995, Agosto). Estimating Continuous Distributions in Bayesian Classifiers.

Jurafsky, D., & Martin, J. H. (2017). *Speech and Language Processing.*

Khodabandelo, G., Hug, C., Deneckere, R., & Salinese, C. (2013). Supervised intentional process models discovery using hidden markov models.

Liu, B. (2012). *Sentiment Analysis and Opinion Mining.* Morgan & Claypool Publishers.

Machado, A. P., Ferreira, R., Bottencourt, I., Elias, E., Brito, P., & Costa, E. (Julho de 2010). Mineração de Texto em Redes Sociais Aplicada à Educação a Distância. *6*(23).

Manning, C. D. (2011). Part-of-Speech Tagging from 97% to 100%:.

Morais, E. A., & Ambrósio, A. P. (2007). *Mineração de Textos.*
Näsholm, P. (2012, Janeiro). Extracting Data from NoSQL Databases A Step towards Interactive Visual Analysis of NoSQL Data.

Neves, R. d. (Abril de 2003). Pré-Processamento no Processo de Descoberta da Conhecimento em Banco de Dados. Fonte: Devmedia.

NLTK. (201-?). *NLTK - Documentation*. Acesso em 23 de Junho de 2018, disponível em nltk: https://www.nltk.org/

NLTK. (2017). *NLTK Natural Language Toolkit*. Acesso em 29 de Junho de 2018, disponível em NLTK: www.nltk.org

Otávio, J. (2016). *Tkinter interfaces gráficas em python*. Acesso em 23 de Junho de 2018, disponível em Devmedia: https://www.devmedia.com.br/tkinter-interfaces-graficas-em-python/33956

Pezzini, A. (Desembro de 2016). MINERAÇÃO DE TEXTOS: CONCEITO, PROCESSO E APLICAÇÕES.

Piatetsky-Shapiro, G., Fayyad, U., & Smyth, P. (1996). *Advances in knowledge discovery and data mining. American Association for Artificial Intelligence.* Menlo Park: American Association for Artificial Intelligence.

Python. (201-?). *What is Python*. Retrieved Junho 23, 2018, from Python: https://docs.python.org/3/faq/general.html#what-is-python

PythonHosted. (201-?). *Spyder - Documentation*. Retrieved Junho 23, 2018, from pythonhostedP: https://pythonhosted.org/spyder/

R7; Agência Brasil & Estadão. (14 de Julho de 2014). *Brasil recebeu 1 milhão de turistas estrangeiros durante a copa*. Acesso em 28 de Junho de 2018, disponível em Esportes R7: https://esportes.r7.com/futebol/copa-do-mundo-2014/brasil-recebeu-1-milhao-de-turistas-estrangeiros-durante-a-copa-14072014

Refaeilzadeh, P., Tang, L., & Liu, H. (Novembro de 2008). Cross-Validation.

Russel, M. A. (2014). *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub and more* (2nd ed.). Sebastopol: O'Relly Media.

Sette, B. S., & Martins, C. A. (2017). Pré-processamento textual para a extração de informação em bases de patentes.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (1999). *Sistema de Banco de Dados.* São Paulo: Makron Books.

Static Brain. (2016, Outubro 01). *Twitter Statics*. Retrieved Novembro 29, 2017, from Static Brain: https://www.statisticbrain.com/twitter-statistics/

Turban, E., Sharda, R. E., & Delen, D. (2010). *Decision support and business intelligence systems* (9th ed.). New Jersey: Prentice Hall.

Vooo. (30 de Abril de 2016). *6 passos fáceis para aprender o algoritmo Naive Bayes com o código em Python*. Acesso em 30 de Janeiro de 2018, disponível em Vooo: https://www.vooo.pro/insights/6-passos-faceis-para-aprender-o-algoritmo-naive-bayes-com-o-codigo-em-python/

W3Schools. (200-?). *SQL Tutorial*. Acesso em 23 de Junho de 2018, disponível em w3schools: https://www.w3schools.com/sql/

APÊNDICE A – Especificação de Requisitos de Software

ITeligence

Especificação de Requisitos de Software

Versão 0.0.4
Data: 10/06/2018

Histórico de Revisões

Data	Versão	Descrição	Autores
<08/06/2018>	<0.0.1>	Início do preenchimento	Gabriel Teixeira
<09/06/2018>	<0.0.2>	Inclusão dos Requisitos funcionais, não funcionais, diagramas e casos de uso.	Gabriel Teixeira
<10/06/2018>	<0.0.3>	Inclusão dos diagramas de atividades, modelos de dados e propósito.	Gabriel Teixeira, Bruno Brum
<11/06/2018>	<0.0.4>	Alterações no propósito e organização do documento.	Gabriel Teixeira

Especificação de Requisitos de Software

1. Introdução

Este documento visa fornecer a especificações de requisitos da ferramenta de mineração de intenções do twitter – Iteligence.

As funcionalidades serão descritas a partir do momento em que o usuário final seleciona a ferramenta de RV no sistema de aprendizagem eletrônica. Funções de suporte a diferentes tipos de usuário são descritas (aluno/professor).

1.1 Propósito

O propósito é oferecer para os usuários uma interface capaz de extrair, processar, e analisar dados do twitter em tempo real ou em algum espaço de tempo fechado no passado, oferecendo também gráficos e opções que auxiliem a tomada de decisão. A ferramenta gerará gráficos analíticos para o usuário (coluna, barra, linhas).

Para que isso aconteça o usuário deverá ter instalado em seu computador uma versão compatível do Python juntamente de um banco de dados MySQL, com os devidos módulos instalados para o armazenamento e processamento dos dados.

Além disso, o usuário poderá também exportar as informações extraídas e processadas do programa para um documento Excel, para manter um histórico e consultá-lo quando necessário.

1.2 Escopo

O sistema especificado deve servir como forma de ferramenta para análise de intenções na rede social Twitter, o sistema irá permitir que o usuário faça a coleta de dados diretamente do Twitter podendo a mesma ser feita em tempo real ou então de dados mais antigos, para coletas em tempo real ele permite que o usuário defina quais as palavras que servirão como parâmetro para a busca dos dados e por quanto tempo está coleta será realizada, para coletas de dados mais antigos também será possível definir as palavras chaves, e escolher qual a data inicial e final dos dados a serem coletados, após essa coleta o usuário poderá realizar o treinamento do algoritmo de classificação dos dados por meio de uso de uma massa de dados para treinamento, e então poderá ser feita a análise e classificação dos dados previamente coletados, que poderão ser exportados para CSV, após sua classificação, o usuário também poderá optar pela criação de gráficos representando os países mais mencionados na massa de dados já classificada ou os adjetivos e substantivos mais recorrentes nos dados que já foram classificados, esses gráficos serão salvos em imagens para que o usuário possa acessar a qualquer momento.

O sistema só irá fornecer suporte para o idioma inglês, e somente para a rede social Twitter.Para realização do treinamento do algoritmo de classificação, o usuário deverá fornecer a massa de dados já classificada previamente.

1.3 Stakeholders

Profissionais da área de gerenciamento de projeto e desenvolvimento de software.

1.4 Organização do Documento

Este documento esta organizado da seguinte forma: A seção 2 descreve uma visão geral do sistema. Os requisitos funcionais e não funcionais do sistema estão contemplados nas seções 3 a seção 4 apresentará os modelos de casos de uso. Os diagramas de atividade e modelos de dados são representados respectivamente nas seções 5 e 6.

2. Visão Geral do Sistema

O sistema é parte de um trabalho para conclusão de curso, do curso de bacharelado em sistemas de informação na instituição de ensino UNIRIO.

2.1 Tecnologia

O sistema será desenvolvido utilizando a linguagem de programação Python, com auxílio da ferramenta Spyder, um ambiente de desenvolvimento integrado para python, e uso do SGBD MySQL.

2.2 Dependências

A coleta do sistema dependerá de conexão a internet para seu funcionamento, além de depender da conexão com a plataforma do Twitter.

2.3 Restrições

O sistema não irá fornecer massa de dados de treinamento, somente para o caso de estudos de intenção de viagens, esses dados deveram ser disponibilizados pelos usuários para a realização do treinamento do algoritmo de classificação.

O sistema para seu funcionamento deverá possuir o Python em sua versão 3 instalado, com os devidos módulos instalados e configurados, além do SGBD MySQL, devidamente instalado e configurado.

3. Requisitos do Sistema

Nessa seção irá ser definido todos os requisitos referentes ao sistema ITeligence.

3.1 Requisitos Funcionais

[RF1] Coleta de dados do Twitter em tempo real

a. O sistema deverá permitir que o usuário defina quais as expressões a serem utilizadas para o filtro aplicado na coleta de dados do Twitter.
b. O sistema deverá permitir o usuário qual será o tempo que a coleta de dados do twitter irá durar.
c. O sistema deverá permitir que o usuário efetue a coleta de dados do Twitter em tempo real.

[RF2] Coleta de dados do Twitter de datas passadas

a. O sistema deverá permitir que o usuário defina quais as expressões a serem utilizadas para o filtro aplicado na coleta de dados do Twitter.
b. O sistema deverá permitir que o usuário defina a data de ínicio da coleta de dados do Twitter.
c. O sistema deverá permitir que o usuário defina a data final da coleta de dados do Twitter.
d. O sistema deverá permitir que o usuário efetue a coleta de dados do Twitter de datas passadas.

[RF3] Treinamento do algoritmo de classificação

a. O sistema deverá permitir que o usuário realize o treinamento para uso do algoritmo de classificação desde que exista uma massa de dados de treinamento disponível.
b. O sistema deverá salvar esse classificador para que o algoritmo de classificação faça uso do mesmo posteriormente.
c. O sistema deverá realizar o pré-processamento dos dados antes de realizar o treinamento.

[RF4] Análise de dados coletados

a. O sistema deverá permitir que o usuário efetue a análise de dados coletados em tempo real ou de datas passadas.
b. O sistema deverá realizar pré-processamento dos dados antes de efetuar sua análise.
c. O sistema deverá realizar a análise dos dados coletados e classifica-los.

[RF5] Gerar gráficos para análise

a. O sistema deverá fazer o pré-processamento dos dados antes de gerar os gráficos.
b. O sistema deverá permitir que o usuário gere gráficos dos substantivos mais comuns nos dados classificados.
c. O sistema deverá permitir que o usuário gere gráficos dos adjetivos mais comuns nos dados classificados.
d. O sistema deverá permitir que o usuário gere gráficos dos países com maior número de menções nos dados classificados.
e. O sistema deverá permitir que o usuário escolha quais palavras não devem aparecer nos gráficos de substantivos e adjetivos mais comuns.
f. O sistema deverá salvar os gráficos gerados como imagem para estarem disponíveis para o usuário.

[RF6] Galeria de gráficos

a. O sistema deve permitir que o usuário acesse uma galeria para a visualização das imagens geradas.

[RF7] Exportar dados classificados

a. O sistema deve permitir que o usuário exporte os dados classificados para um CSV.
b. O sistema deve salvar o CSV em uma pasta interna.

[RF8] Exclusão de dados

a. O sistema deve permitir que o usuário exclua todos os dados sejam eles coletados ou classificados.

3.2 Requisitos Não-Funcionais

[RN1] Usabilidade

a. O sistema deverá ser totalmente intuitivo para o usuário, remetendo o mesmo a uma interface mobile.

b. O usuário deverá poder acessar e utilizar qualquer funcionalidade do sistema utilizando-se no máximo de 3 (três) cliques.

[RN2] Confiabilidade

a. O sistema dependerá da conexão com o Twitter para poder realizar a coleta de dados.

[RN3] Restrições

a. O sistema dependerá de conexão com a internet para a realização de coleta de dados.

b. O sistema terá suporte de coleta e classificação apenas para o idioma inglês.

c. O computador deverá possuir o Python versão 3.x instalado, assim como os módulos necessários devidamente instalados e configurados.

d. O computador deverá possuir o SGBD MySQL devidamente instalado e configurado para o funcionamento do sistema.

[RN4] Documentação

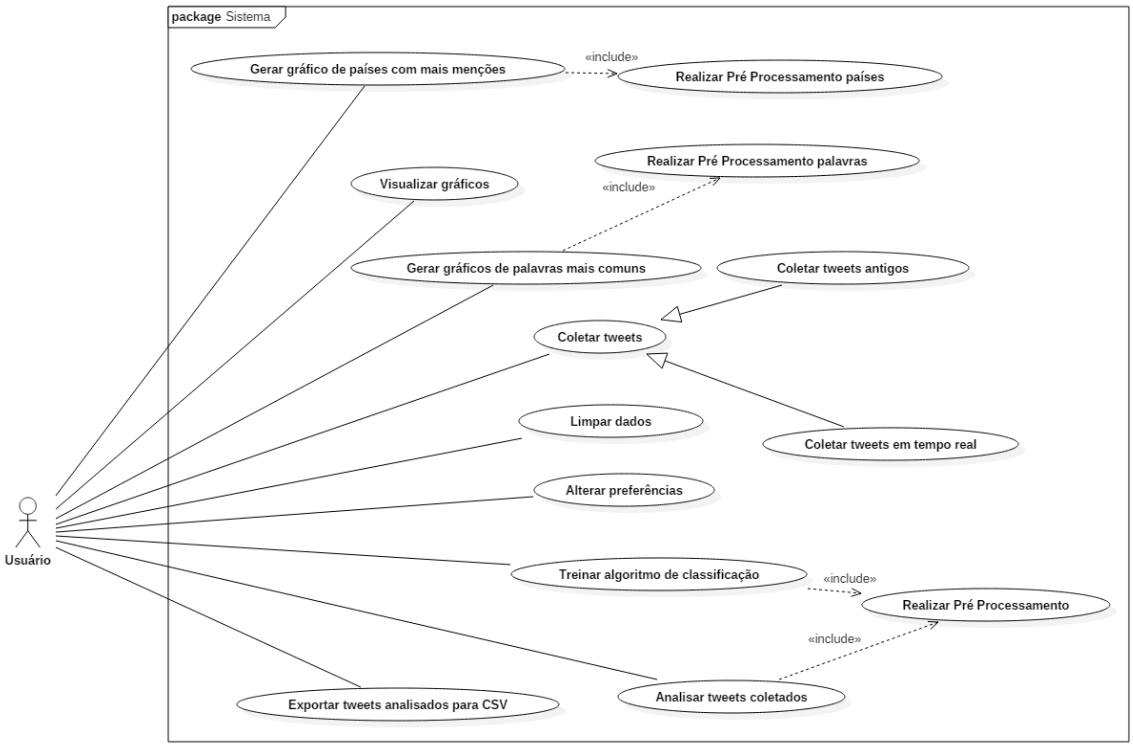
a. O sistema não possuirá ajuda on-line para os usuários.

b. O sistema não possuirá um manual do usuário.

4. Mapeamento de Requisitos com Casos de Uso

Nessa seção iremos descrever o sistema por meio de casos de uso, mostrando o funcionamento do mesmo pelo ponto de vista do usuário.

4.1 Casos de Uso



4.2 Casos de uso estendidos

[CU1] Coletar tweets antigos

a. Atores
   i. Usuário
   ii. Twitter

b. Fluxo de Eventos
   i. Fluxo Principal
      1. O evento se inicia quando o usuário seleciona para coletar Tweets de datas antigas;
      2. O sistema indica para o usuário que os Tweets estão sendo coletados;
      3. O sistema solicita os dados ao Twitter;
      4. O Twitter retorna os dados para o sistema;
      5. O sistema salva os dados indicando que não foram coletados em tempo real;

6. O sistema retorna para o usuário indicando que a coleta foi realizada.

ii. Fluxo Alternativo

1. Se ocorrer algum erro de conexão ou coleta o sistema indica o erro ocorrido para o usuário.

c. Pré-condições

i. O usuário ter definido pelo menos uma palavra para a busca de dados;

ii. O usuário ter definido as datas da coleta ao menos uma vez;

iii. O usuário possuir conexão com a internet.

d. Pós-condições

i. Os dados terem sido salvos no banco de dados.

[CU2] Coletar tweets em tempo real

a. Atores

i. Usuário

ii. Twitter

b. Fluxo de Eventos

i. Fluxo Principal

1. O evento se inicia quando o usuário seleciona para coletar Tweets em tempo real;
2. O sistema indica para o usuário que os Tweets estão sendo coletados;
3. O sistema estabelece conexão com o Twitter;
4. O Twitter retorna os dados para o sistema;
5. O sistema salva os dados indicando que foram coletados em tempo real;
6. O sistema retorna para o usuário indicando que a coleta foi realizada.

ii. Fluxo Alternativo

1. Se ocorrer algum erro de conexão ou coleta o sistema indica o erro ocorrido para o usuário.

c. Pré-condições

i. O usuário ter definido pelo menos uma palavra para a busca de dados;

ii. O usuário ter definido o tempo de coleta ao menos uma vez;

iii. O usuário possuir conexão com a internet.

d. Pós-condições

i. Os dados terem sido salvos no banco de dados.

[CU3] Alterar preferências

a. Atores
    i. Usuário
b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário seleciona na barra de tarefas para alterar as preferências;
        2. O sistema abre a tela de preferências para o usuário com as preferências atuais;
        3. O usuário altera suas preferências (palavras a serem utilizadas na coleta, runtime da coleta em tempo real, data de coleta de tweets antigos, palavras a serem excluídas dos gráficos);
        4. O usuário escolhe salvar suas preferências;
        5. O sistema salva as preferências em arquivos do sistema;
        6. O sistema indica ao usuário que as preferências foram alteradas;
    ii. Fluxo Alternativo
        1. O usuário preenche dados errados
        2. O sistema indica que os dados foram preenchidos de forma errada
    iii. Fluxo Alternativo
        1. O usuário a qualquer momento pode cancelar a alteração de preferências;
        2. O sistema fecha a janela de preferencias.
c. Pré-condições
    i. N/A
d. Pós-condições
    i. As preferências alteradas pelo usuário serem salvas de modo que possam ser acessadas posteriormente.

[CU4] Treinar algoritmo de classificação

a. Atores
    i. Usuário
b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário seleciona para realizar o treinamento do algoritmo de classificação;
        2. O sistema indica ao usuário que o treinamento está sendo executado;
        3. O sistema abre o arquivo com a massa de dados de treinamento;
        4. O sistema realiza o pré-processamento dos dados;

5. O sistema realiza o treinamento e salva uma lista com palavras mais comuns em arquivo pickle;
6. O sistema realiza o treinamento e salva um classificador em arquivo pickle;
7. O sistema indica ao usuário que o treinamento foi realizado;

ii. Fluxo Alternativo

1. O usuário não forneceu um arquivo para o treinamento;
2. O sistema indica que não existe arquivo para realizar o treinamento.

c. Pré-condições

i. Existir um arquivo com os dados para o treinamento.

d. Pós-condições

i. O sistema deve salvar os arquivos necessários para a realização da classificação de tweets.

[CU5] Analisar tweets coletados

a. Atores

i. Usuário

b. Fluxo de Eventos

i. Fluxo Principal

1. O usuário seleciona para realizar a análise/classificação de tweets antigos ou coletados em tempo real;
2. O sistema indica ao usuário que a análise está sendo executado;
3. O sistema abre os arquivos criados durante o treinamento;
4. O sistema realiza o pré-processamento dos dados a serem classificados;
5. O sistema realiza a análise e classificação dos tweets;
6. O sistema adiciona a classificação dos tweets no banco de dados;
7. O sistema indica ao usuário que a análise foi realizada;

ii. Fluxo Alternativo

1. O usuário não fez a coleta de dados;
2. O sistema não irá realizar a análise e classificação dos dados.

c. Pré-condições

i. A coleta de dados ter sido realizada.

d. Pós-condições

i. O sistema deve salvar a classificação referente a cada tweets analisado.

[CU6] Realizar Pré Processamento

a. Atores

i. Usuário

b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário solicita alguma função que necessite de pré-processamento;
        2. O sistema inicia treinamento ou análise;
        3. O sistema realiza o pré-processamento dos dados.
    ii. Fluxo Alternativo
        1. N/A
c. Pré-condições
    i. O usuário ter solicitado a execução do treinamento ou analise.
d. Pós-condições
    i. Os dados pré-processados estarem disponíveis para o uso do sistema.

[CU7] Realizar Pré Processamento países

a. Atores
    i. Usuário
b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário solicita a criação de gráfico de países com mais menções;
        2. O sistema inicia geração de gráficos de países com mais menções;
        3. O sistema realiza o pré-processamento dos dados.
    ii. Fluxo Alternativo
        1. N/A
c. Pré-condições
    i. O usuário ter solicitado a execução da geração de gráficos de países com mais menções.
d. Pós-condições
    i. Os dados pré-processados estarem disponíveis para o uso do sistema.

[CU8] Realizar Pré-Processamento palavras

a. Atores
    i. Usuário
b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário solicita a criação de gráfico de substantivos e adjetivos com mais menções;
        2. O sistema inicia geração de gráficos de palavras mais comuns;
        3. O sistema realiza o pré-processamento dos dados.
    ii. Fluxo Alternativo

1. N/A

c. Pré-condições
    i. O usuário ter solicitado a execução da geração de gráficos de palavras mais comuns.
d. Pós-condições
    i. Os dados pré-processados estarem disponíveis para o uso do sistema.

[CU9] Gerar gráfico palavras mais comuns

a. Atores
    i. Usuário
b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário seleciona para gerar gráficos de palavras mais comuns;
        2. O sistema indica que os gráficos estão sendo criados;
        3. O sistema seleciona os dados já classificados;
        4. O sistema realiza pré-processamento dos dados selecionados;
        5. O sistema cria os gráficos de substantivos mais comuns sem intenção;
        6. O sistema cria os gráficos de substantivos mais comuns com intenção;
        7. O sistema cria os gráficos de adjetivos mais comuns sem intenção;
        8. O sistema cria os gráficos de adjetivos mais comuns com intenção;
        9. O sistema salva os gráficos como imagem;
        10. O sistema indica ao usuário que os gráficos foram feitos.
    ii. Fluxo Alternativo
        1. Não há dados para algum dos gráficos;
        2. O sistema não cria o gráfico.
c. Pré-condições
    i. Existir dados classificados
d. Pós-condições
    i. O sistema deve salvar as imagens dos gráficos de modo que fiquem disponíveis para visualização.

[CU10] Gerar gráfico países com mais menções

a. Atores
    i. Usuário
b. Fluxo de Eventos
    i. Fluxo Principal
        1. O usuário seleciona para gerar gráficos de países com maior quantidade de menções;

2. O sistema indica que os gráficos estão sendo criados;
3. O sistema seleciona os dados já classificados;
4. O sistema realiza pré-processamento dos dados selecionados;
5. O sistema cria os gráficos de países com mais menções sem intenção;
6. O sistema cria os gráficos de países com mais menções com intenção;
7. O sistema salva os gráficos como imagem;
8. O sistema indica ao usuário que os gráficos foram feitos.

ii. Fluxo Alternativo

1. Não há dados para algum dos gráficos;
2. O sistema não cria o gráfico.

c. Pré-condições

i. Existir dados classificados.

d. Pós-condições

i. O sistema deve salvar as imagens dos gráficos de modo que fiquem disponíveis para visualização.

[CU11] Exportar tweets analisados para CSV

a. Atores

i. Usuário

b. Fluxo de Eventos

i. Fluxo Principal

1. O usuário seleciona na barra de tarefas para exportar para CSV;
2. O sistema indica que os tweets analisados estão sendo exportados;
3. O sistema exporta os tweets classificados para CSV;

ii. Fluxo Alternativo

1. Não há dados para serem exportados;
2. O sistema indica que não há dados para serem exportados.

c. Pré-condições

i. Existir dados classificados.

d. Pós-condições

i. O sistema deve salvar o CSV de modo que fique disponível para o usuário.

[CU12] Limpar dados

a. Atores

i. Usuário

b. Fluxo de Eventos

i. Fluxo Principal

1. O usuário seleciona na barra de tarefas para acessar opções do banco;
2. O sistema abre a tela de opções do banco;
3. O usuário seleciona para limpar os dados;
4. O sistema indica que os dados estão sendo limpos;
5. O sistema exclui os dados no banco;
6. O sistema indica que os dados foram limpos.

ii. Fluxo Alternativo

1. N/A

c. Pré-condições

i. N/A

d. Pós-condições

i. O dados coletados e classificados antes da limpeza não devem mais estar disponíveis.

[CU13] Visualizar gráficos

a. Atores

i. Usuário

b. Fluxo de Eventos

i. Fluxo Principal

1. O usuário seleciona para acessar a galeria;
2. O sistema carrega os gráficos disponíveis;
3. O sistema abre a tela exibindo a imagem do gráfico disponível;
4. O usuário pode visualizar todos os gráficos disponíveis;

ii. Fluxo Alternativo

1. Não existem gráficos disponíveis;
2. O sistema informa que não há gráficos disponíveis.

c. Pré-condições

i. Possuir gráficos disponíveis para a exibição.

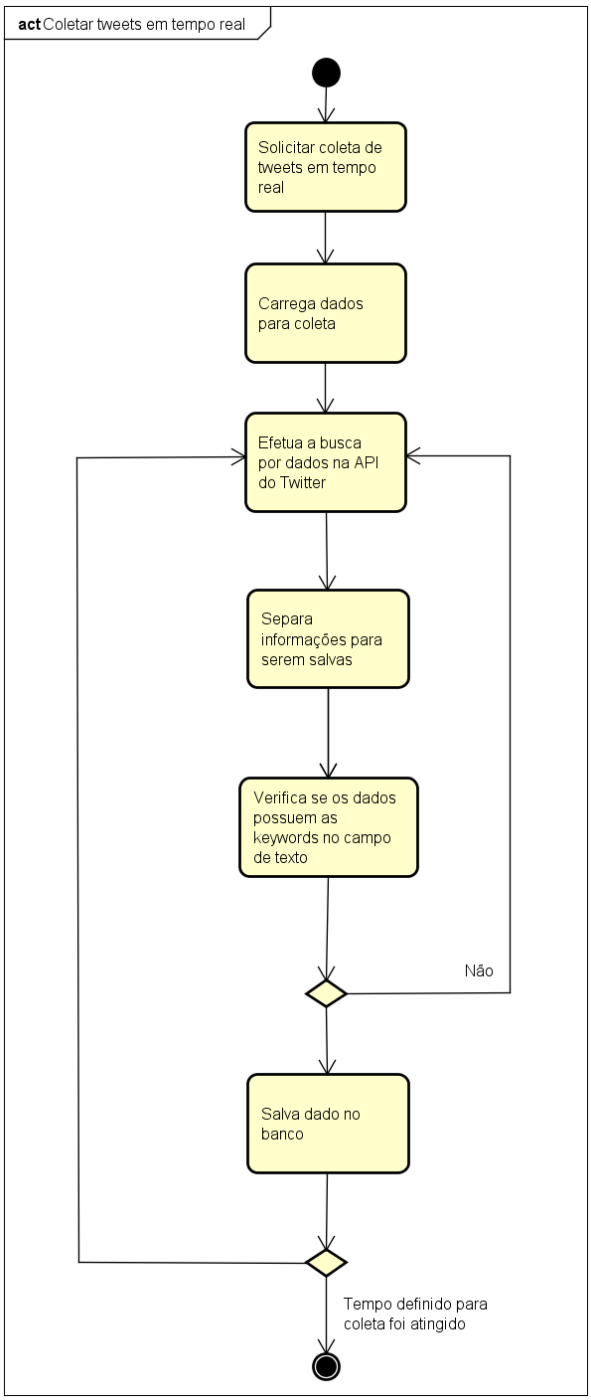
d. Pós-condições

i. N/A

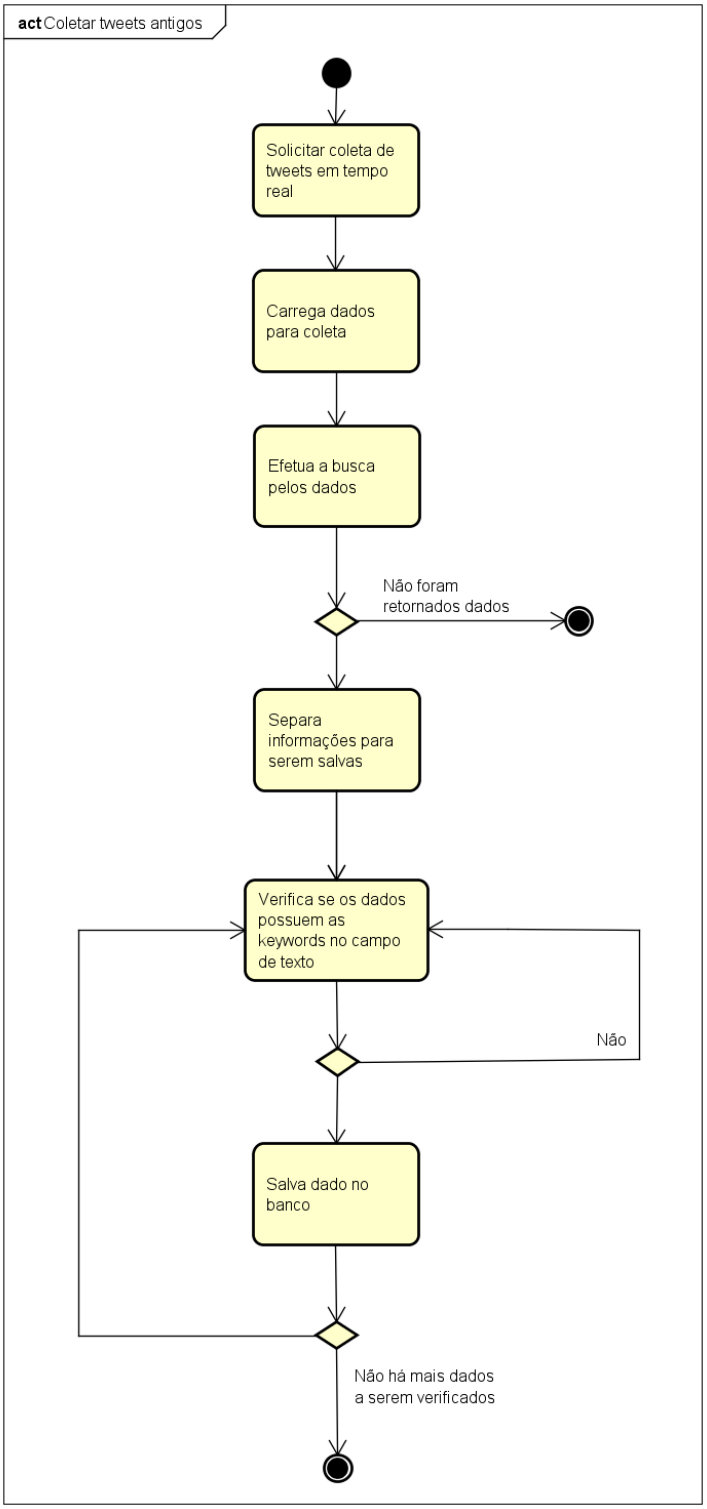
5. Diagramas de atividades

Nessa seção iremos apresentar os diagramas de atividades referentes a cada um dos casos de uso apresentados na seção anterior, sendo basicamente os fluxos de cada um dos processos.

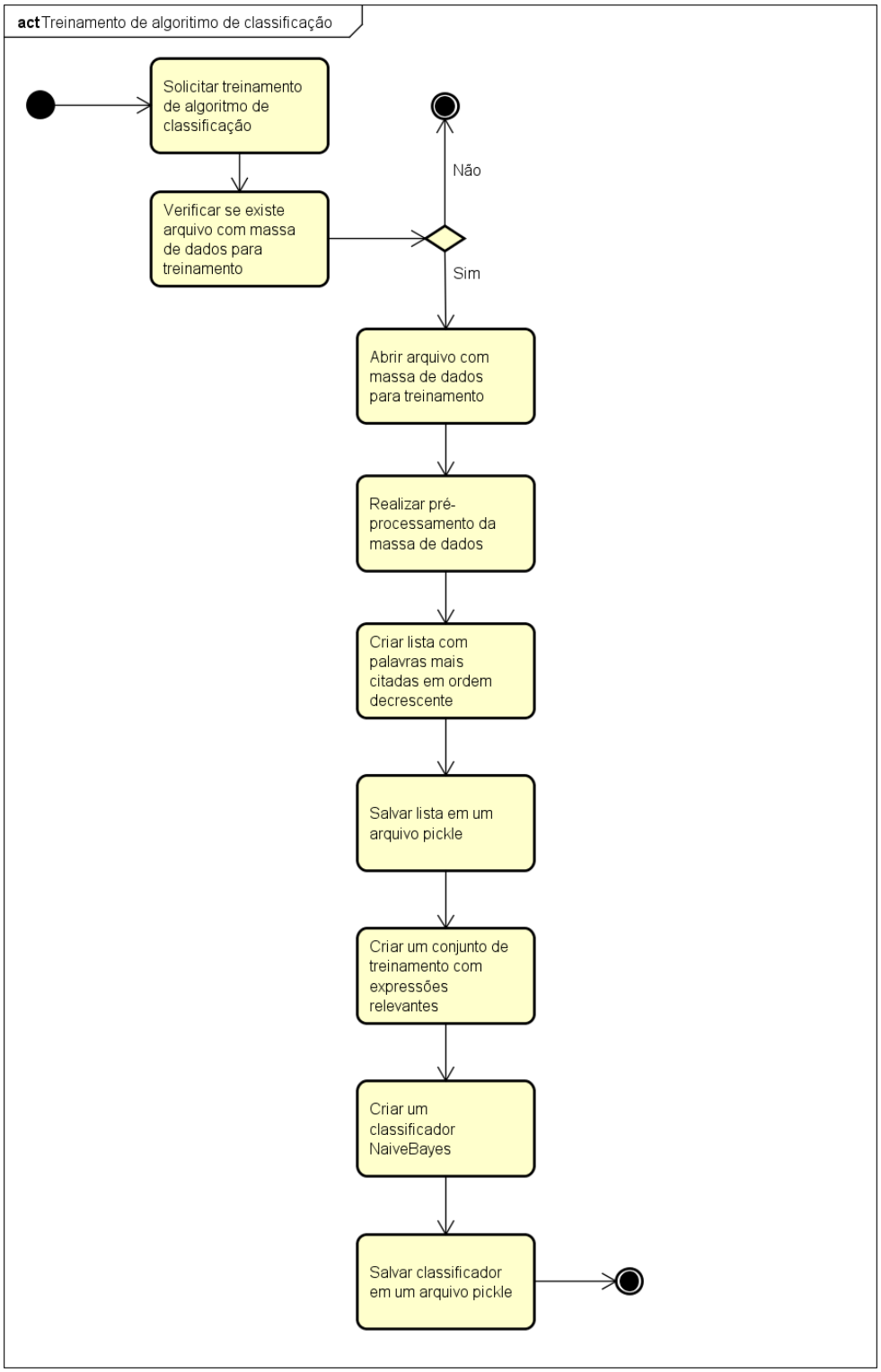
[DA1] Coletar tweets em tempo real



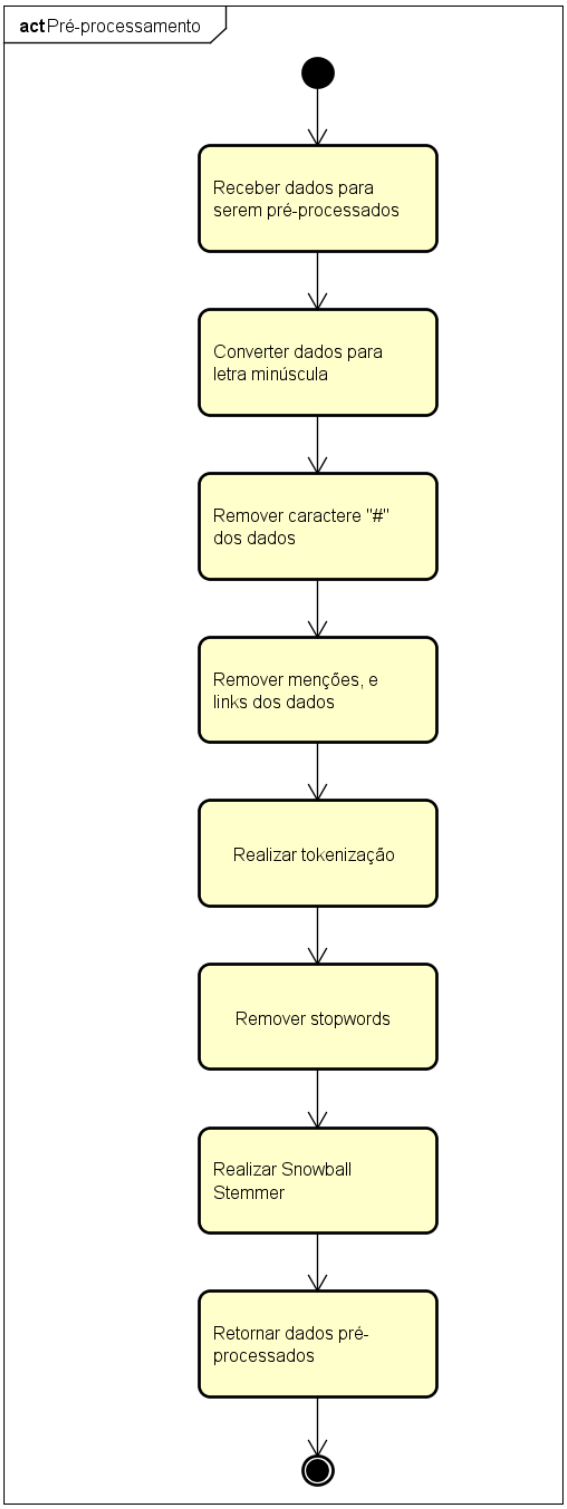
[DA2] Coletar tweets antigos



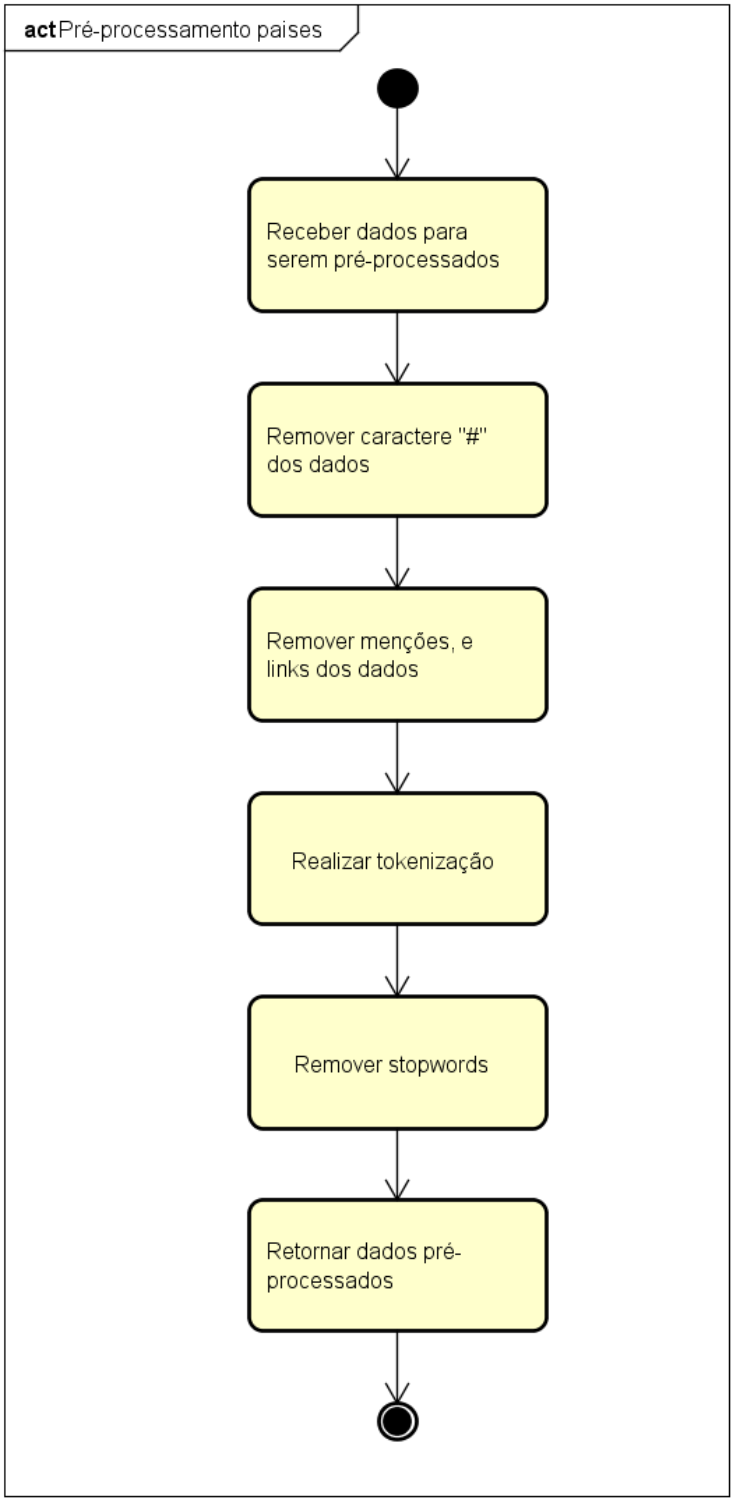
[DA3] Treinar algoritmo de classificação



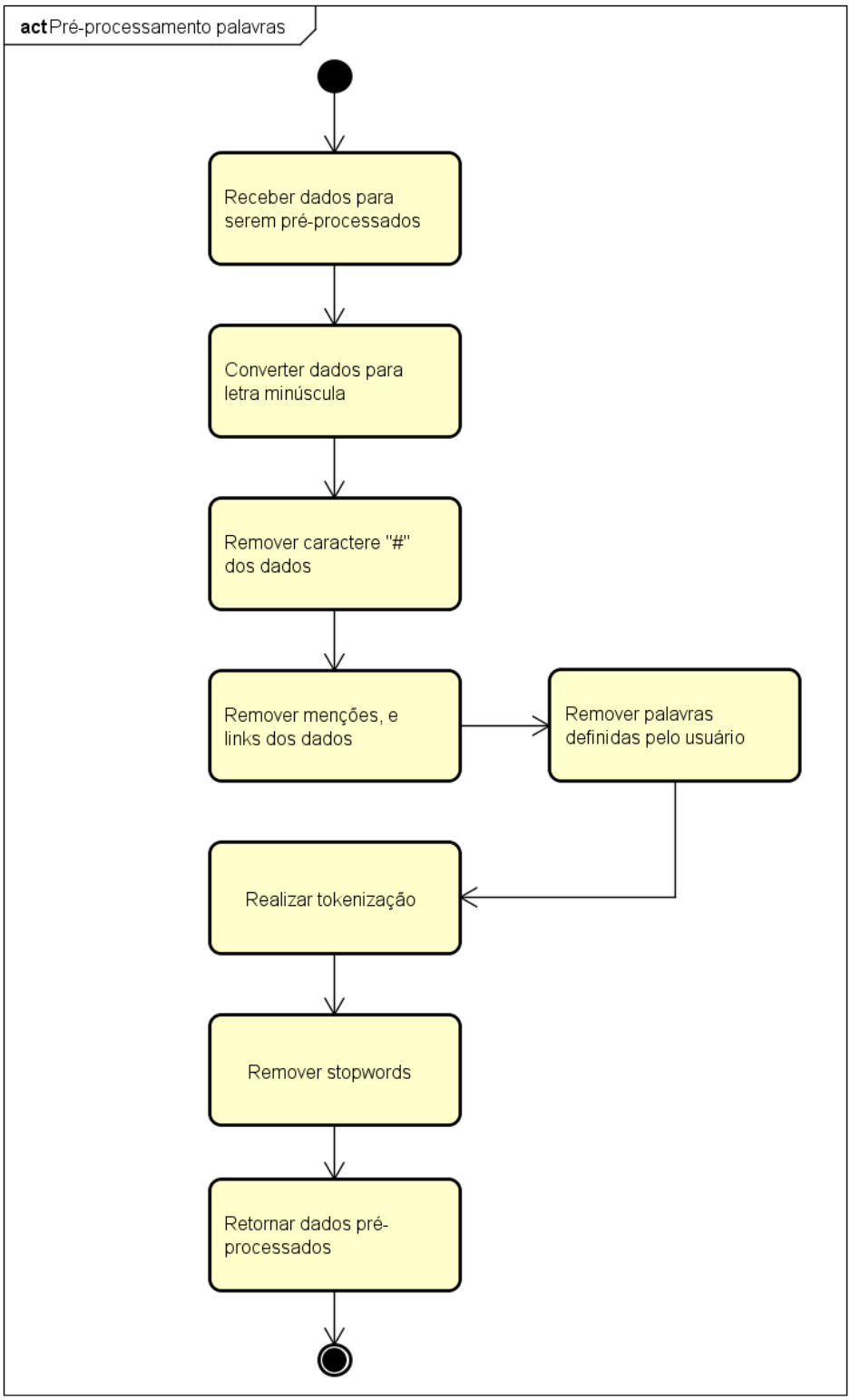
[DA4] Pré processamento



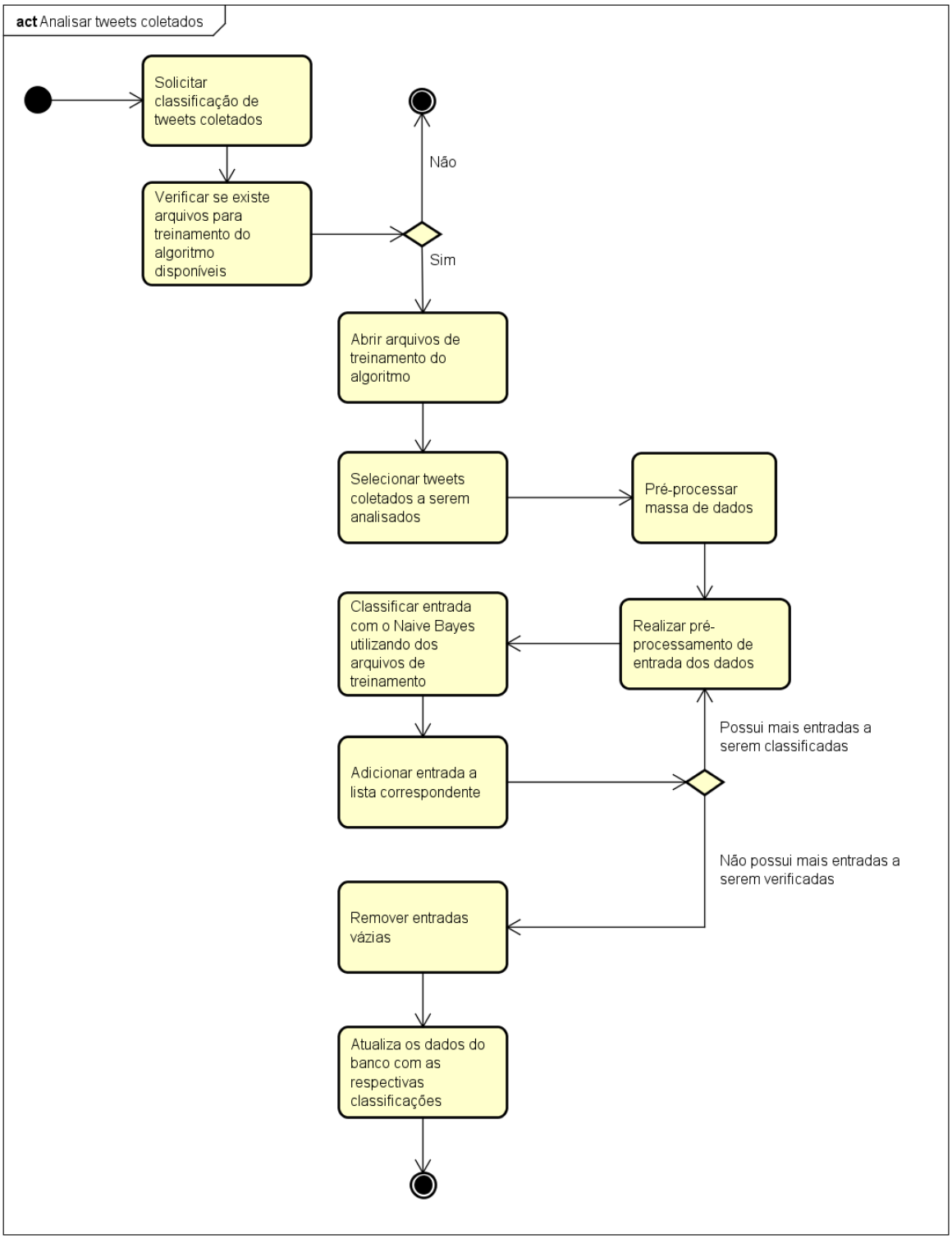
[DA5] Pré processamento países



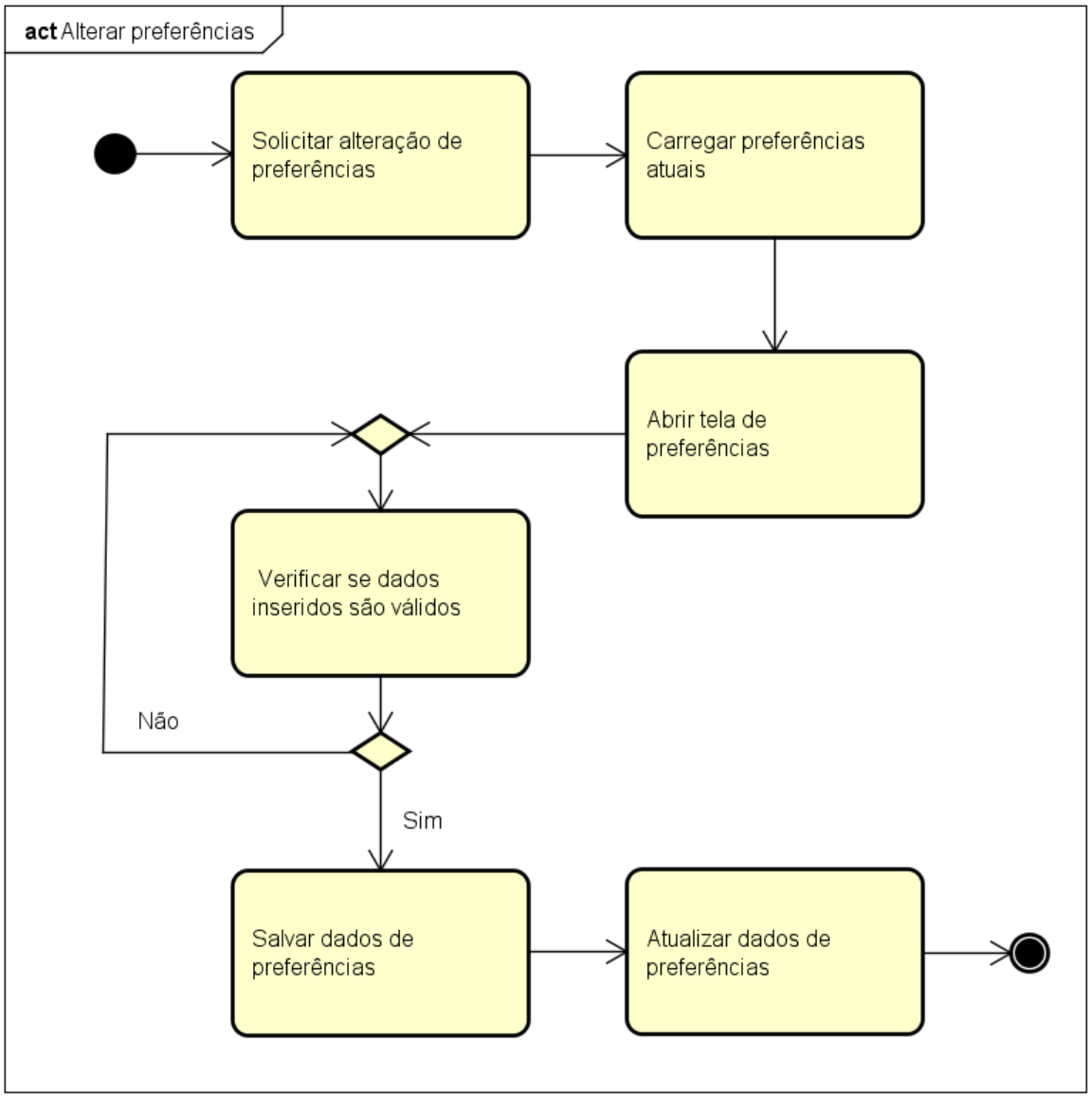
[DA6] Pré processamento palavras



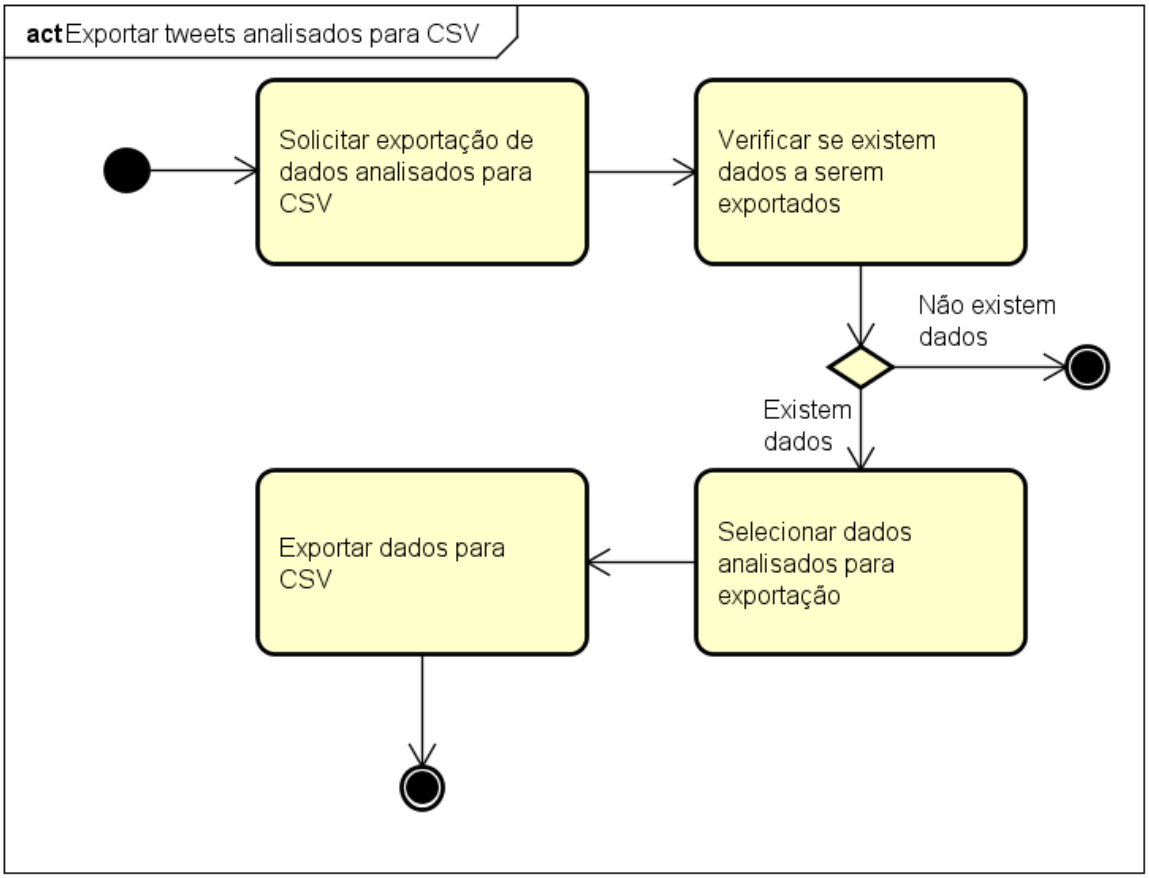
[DA7] Analisar tweets coletados



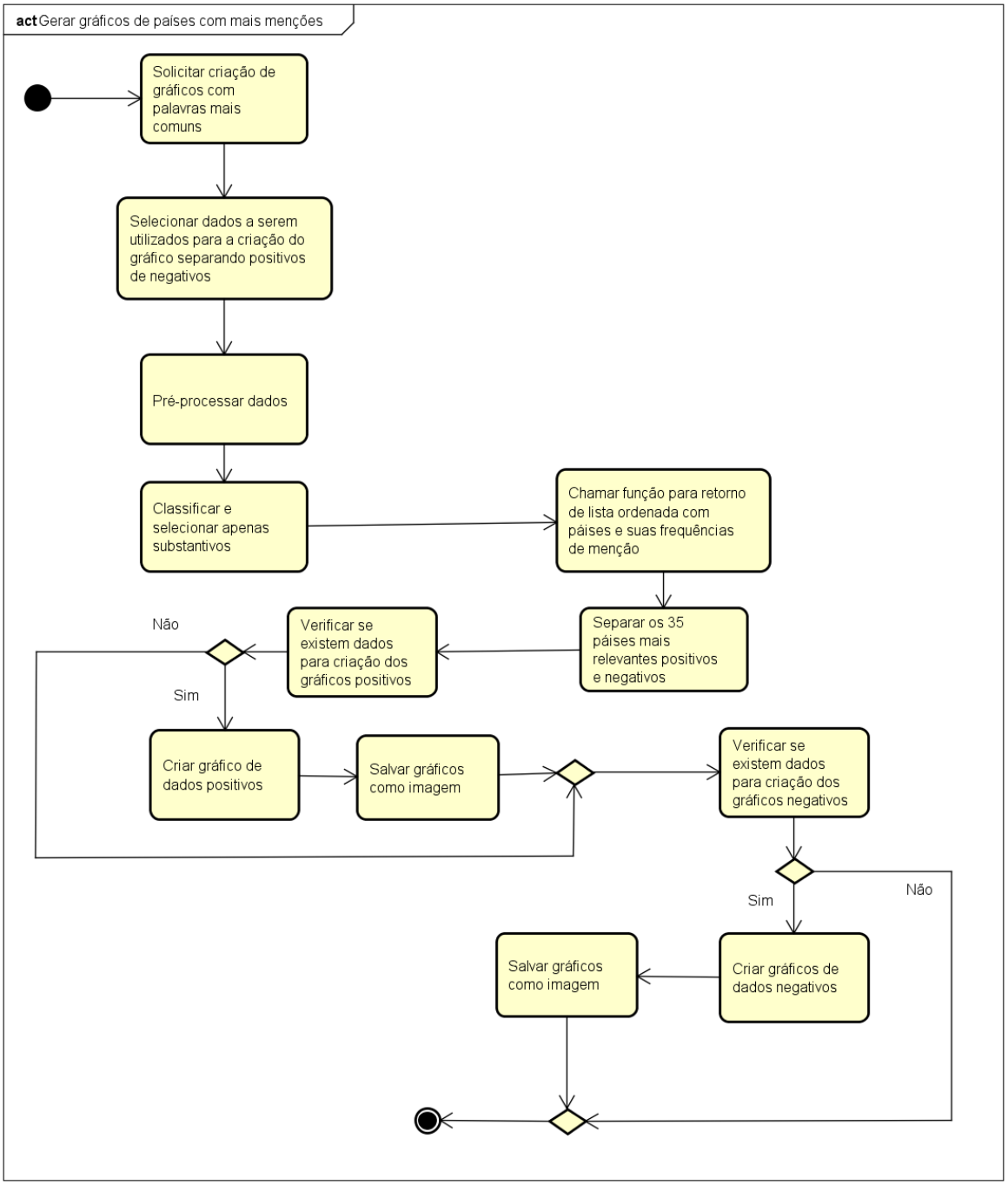
[DA8] Alterar preferências



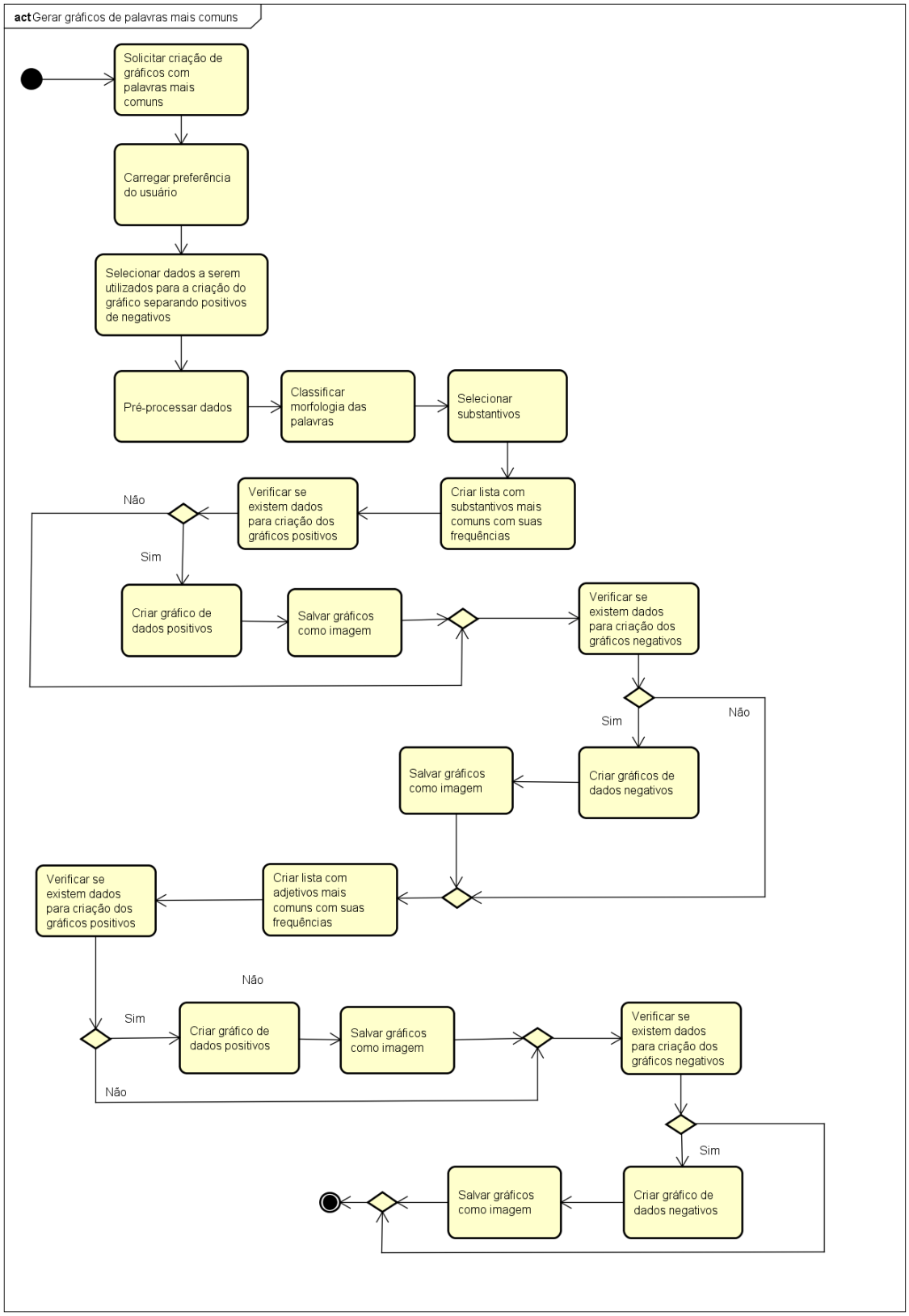
[DA9] Exportar tweets analisados para CSV



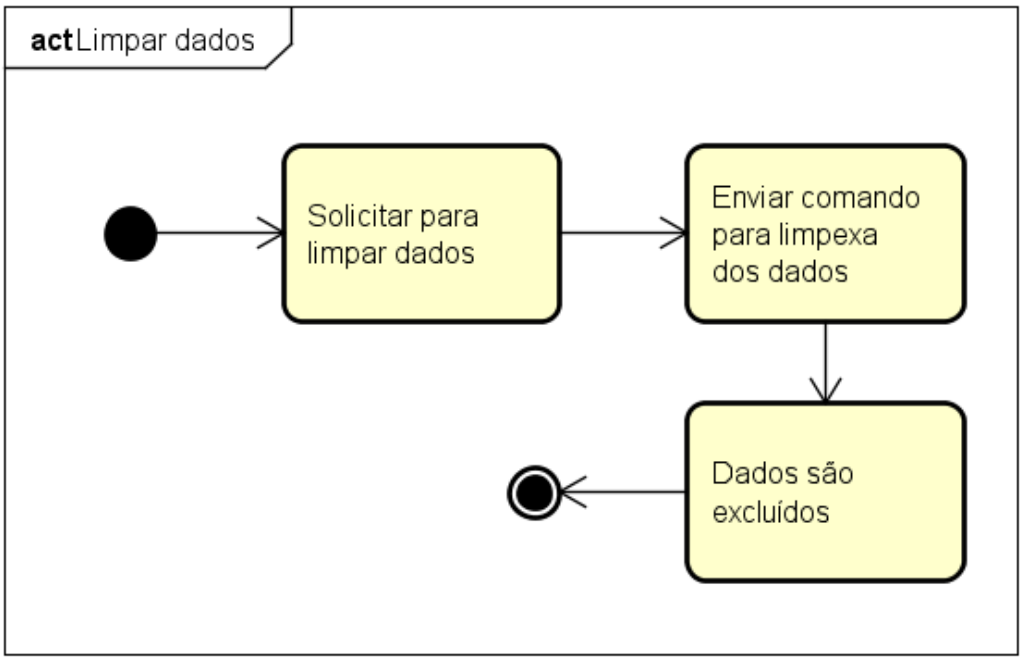
[DA10] Gerar gráficos de países com mais menções



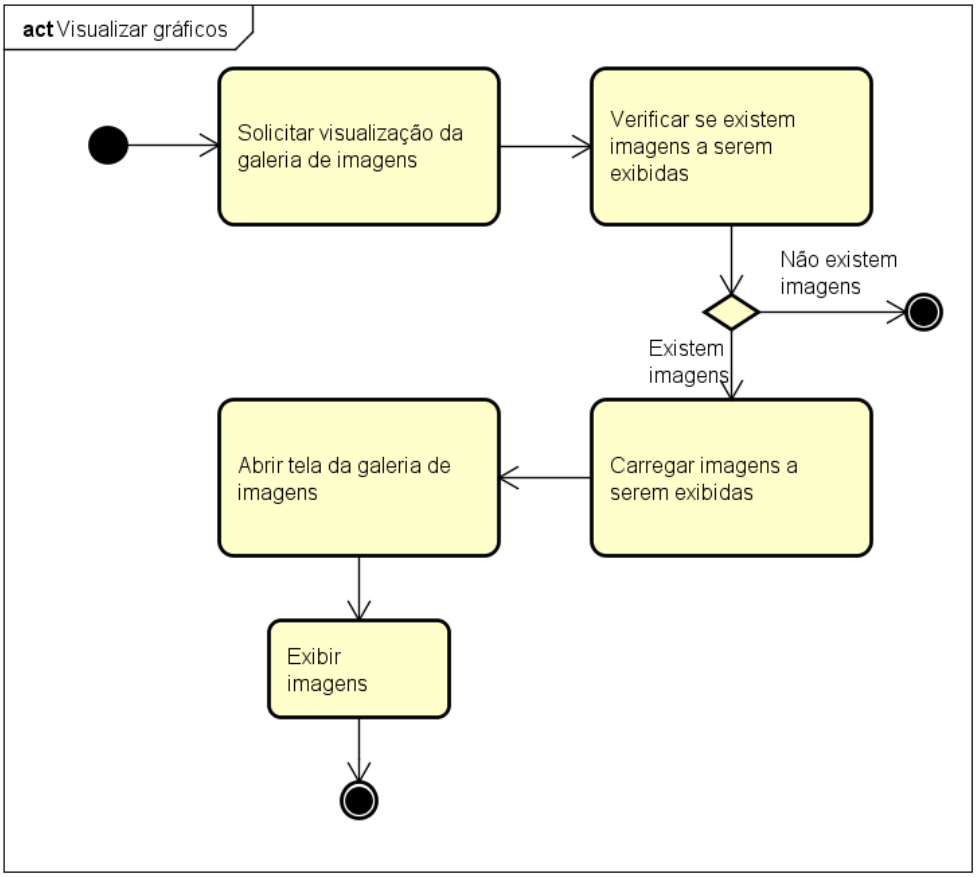
[DA11] Gerar gráficos de palavras mais comuns



[DA12] Limpar dados



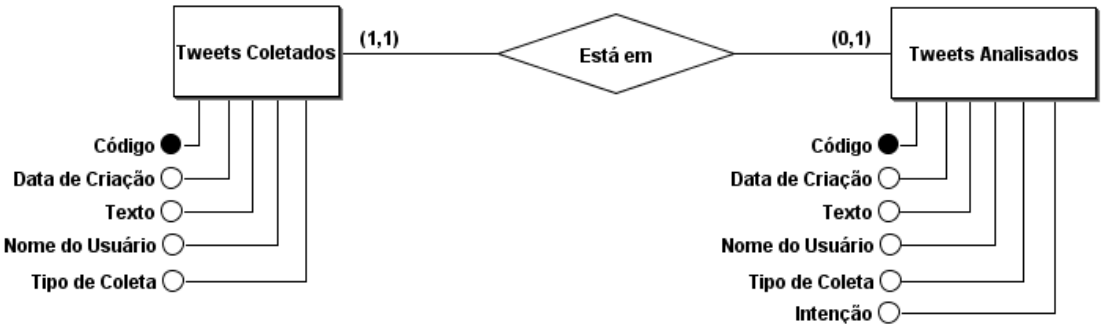
[DA13] Visualizar Gráficos



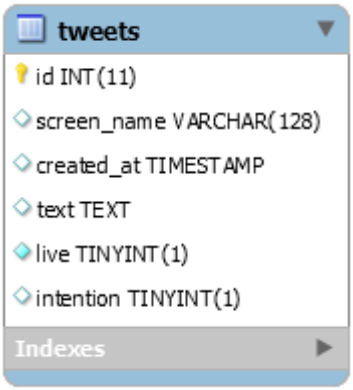
6. Modelos de dados

Nesta seção iremos mostrar os modelos de dados, que visam descrever a representação conceitual, lógica e física dos dados persistentes de nosso sistema que serão armazenados em um banco de dados SQL.

6.1 Modelo conceitual



6.2 Modelo Lógico



6.3 Modelo Físico

```
CREATE TABLE IF NOT EXISTS `intentionmining`.`tweets` (
  `id` INT(11) NOT NULL AUTO_INCREMENT,
  `screen_name` VARCHAR(128) NULL DEFAULT NULL,
  `created_at` TIMESTAMP NULL DEFAULT NULL,
  `text` TEXT NULL DEFAULT NULL,
  `live` TINYINT(1) NOT NULL,
  `intention` TINYINT(1) NULL DEFAULT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB
AUTO_INCREMENT = 172
DEFAULT CHARACTER SET = utf8;
```