Politécnico
de Viseu

Escola Superior
de Tecnologia
e Gestão de Viseu

Caracterização Anonimizada de utilizadores de Twitter com base na Análise de Sentimento

Gonçalo Sebastião Trindade Saraiva

Dissertação

Mestrado em Engenharia Informática - Sistemas de Informação

Trabalho efetuado sob a orientação de
Professor Doutor Filipe Marques da Silva Cabral Pinto
Professora Doutora Ana Cristina Wanzeller Guedes Lacerda

Dezembro de 2023

Politécnico
de Viseu

Escola Superior
de Tecnologia
e Gestão de Viseu

Caracterização Anonimizada de utilizadores de Twitter com base na Análise de Sentimento

Gonçalo Sebastião Trindade Saraiva

Dissertação

Mestrado em Engenharia Informática - Sistemas de Informação

Trabalho efetuado sob a orientação de

Professor Doutor Filipe Marques da Silva Cabral Pinto
Professora Doutora Ana Cristina Wanzeller Guedes Lacerda

Dezembro de 2023

AGRADECIMENTOS

Ao longo do meu percurso académico foram aparecendo obstáculos que me permitiram crescer e aprender como pessoa e estudante a enfrentá-los. Quero agradecer a todos os professores do curso de Mestrado em Engenharia Informática – Sistema de Informação do Instituto Politécnico de Viseu por toda a ajuda ao longo deste percurso.

Quero agradecer ao meu orientador, professor Doutor Filipe Cabral Pinto, pela disponibilidade, ajuda e paciência no decorrer do desenvolvimento deste trabalho.

Gostaria de agradecer à minha família, especialmente aos meus pais, Paulo e Paula, pois sem eles não era possível ter concluído o mestrado, à minha irmã Beatriz e à minha namorada Mariana, por toda a ajuda e por estarem presentes nos momentos mais complicados neste percurso.

Também quero agradecer aos meus colegas e amigos por todo o apoio e por estarem presentes, pois concluir o mestrado sozinho seria muito mais complicado.

RESUMO

Ao longo dos anos, impulsionada pelo avanço tecnológico e pela constante inovação de serviços, a sociedade tem intensificado substancialmente o uso das redes sociais, alcançando níveis de dependência (SNS, 2020).

A rede social X (antigo *Twitter*) é uma rede social que permite aos seus utilizadores partilhar as suas opiniões. Esta é uma das redes sociais mais utilizadas atualmente (Martin, 2023) e, quando surgiu, era permitido um máximo de 140 caracteres por *tweet*, valor que foi aumentado para 280 em 2017. Em Portugal, 2,5 milhões de pessoas usam a rede social X e, mundialmente, esta rede social conta com mais de 330 milhões de utilizadores, o que origina em média a criação de 5 787 *tweets* por segundo em todo o mundo, o que equivale a quase 500 milhões diariamente (Bacelar, 2022). Estes *tweets* geram uma quantidade significativa de dados, dados estes que estão a ser cada vez mais importantes na atualidade, pois é possível retirar informações relevantes e/ou ser capaz de identificar perfis. Isto é alcançado através da tecnologia e da descoberta de novas técnicas e algoritmos, sendo estas as principais razões para o desenvolvimento deste projeto.

O projeto teve os seguintes objetivos:

- Tratar Dados de forma a conseguir obter o melhor resultado possível.
- Determinar tópicos para conseguir perceber os assuntos.
- Classificar os *tweets* de modo a determinar se os utilizadores são a favor, contra ou neutros relativamente aos tópicos discutidos nas redes sociais.
- Recomendar conteúdos com base no sentimento detetado anteriormente, tendo em conta o tópico abordado.

Com o intuito de esclarecer os objetivos deste estudo, propõe-se o seguinte exemplo ilustrativo:

Se o Presidente da República for a entidade selecionada, procede-se, assim, à recolha de *tweets* que façam menção ao perfil "@PresidenteDaRepublica". Em seguida, serão realizadas as etapas necessárias para o **tratamento** adequado desses *tweets*, seguido pela identificação dos **tópicos** mais frequentemente discutidos em relação a essa figura pública. Isso estabelecerá a base de dados fundamental para a condução deste projeto.

Uma vez obtida a base de dados, sólida e coesa, realiza-se a **análise de sentimento** com o objetivo de identificar e classificar os perfis de utilizadores em categorias como "Positivo", "Neutro" ou "Negativo" em relação a diversos temas relacionados com a entidade. Posteriormente, será implementado um **sistema de recomendação** para enviar conteúdos diferenciados a cada grupo de utilizadores previamente identificados. Esta abordagem permitirá potencialmente influenciar as opiniões das pessoas ou auxiliá-las na formação de opiniões bem fundamentadas e construtivas, independentemente das opiniões serem positivas ou negativas a determinados assuntos.

Palavras-Chave: Análise de Sentimento, Identificação de Tópicos, *Machine Learning*, *Web Scrapping*, *Natural Language Processing*, *Recommender Systems*, *Python*, Rede Social X.

ABSTRACT

Over the years, propelled by technological advancements and continuous service innovation, society has significantly intensified the use of social media, reaching substantial levels of dependency (SNS, 2020).

The social network X (previously known as Twitter) is a social media platform that allows its users to share their opinions. It is one of the most widely used social networks today (Martin, 2023) and, when it was first introduced, it allowed a maximum of 140 characters per *tweet*, a limit that was increased to 280 characters in 2017. In Portugal, 2.5 million people use X social network, and globally, this social network boasts over 330 million users, resulting in an average of 5,787 tweets per second worldwide (Bacelar, 2022). This leads to the creation of nearly 500 million tweets daily. These data are becoming increasingly significant in contemporary society, as they provide valuable information and the ability to identify profiles within the current society, this is achieved through technology and the discovery of new techniques and algorithms, which are the main reasons for the development of this project.

The project had the following objectives:

- Clean data in order to achieve the best possible outcome.
- Determine topics to understand the subjects present in the data.
- Classify tweets in order toto determine whether users are in favor, against, or neutral regarding the topics discussed on social networks.
- Recommend content based on the detected sentiment.

To clarify the objectives of this study, consider the following example, if the President of the Republic is the selected entity, we will proceed with the collection of tweets mentioning the profile "@PresidenteDaRepublica". Subsequently, the necessary steps will be taken for the proper **treatment** of these tweets, followed by the identification of the most frequently discussed **topics** related to this public figure. This will establish the fundamental database for the execution of this project.

Once a solid and cohesive database is obtained, we will conduct **sentiment analysis** with the aim of identifying and classifying user profiles into categories such as "Positive", "Neutral," or "Negative" in relation to various topics related to the president of the republic. Subsequently, a **recommendation system** will be

implemented to deliver differentiated content to each group of previously identified users. This approach will potentially influence people's opinions or assist them in forming more well-founded and constructive opinions, regardless of whether they are positive or negative on certain subjects.

Keywords: Sentiment analysis, Topic Identification, Machine Learning, Web Scrapping, Natural Language Processing, Recommender Systems, Python, X Social Network.

ÍNDICE GERAL

ÍNDICE DE TABELAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

LISTA DE SIGLAS

API → *Application Programming Interface*

CSV → *Comma Separated Values*

DT → *Decision Tree*

FoMO → *Fear of Missing Out*

KNN → *K-Nearest Neighbor*

LDA → *Latent Dirichlet Allocation*

LogR → *Logistic Regression.*

LSA → *Latent Semantic Analysis*

MDLSA → *Multidimensional Latent Semantic Analysis*

ME → *Maximum Entropy*

ML → *Machine Learning*

NB → *Naive Bayes*

PLSA → *Probabilistic Latent Semantic Analysis*

RN → *Redes Neuronais Feedforward*

TF - IDF → *Term Frequency - Inverse Document Frequency*

WS → *Web Scrapping*

1. Introdução

1.1. Enquadramento

Nos tempos atuais, as redes sociais têm ganho cada vez mais importância na sociedade. As redes sociais são usadas diariamente e, nestas, são discutidos todos os assuntos e opiniões pessoais desde política a desporto ou apenas uma partilha de um simples encontro de amigos.

As redes sociais e as informações que estão presentes nestas são mais sérias do que aparentam uma vez que estas podem influenciar as opiniões e ações das pessoas. Um exemplo que comprova este impacto constatou-se no estudo realizado por Wilson & Wiysonge em 2020, aquando do surgimento das vacinas contra o covid-19. Através da rede social X, este estudo comprovou que existiu uma relação entre as redes sociais e dúvidas para a toma ou não da vacina (Wilson & Wiysonge, 2020).

Em Portugal, a rede social X é a 7ª rede social mais utilizada. Conta com perto de 2,5 milhões de contas ativas, e cerca de 330 milhões de utilizadores em todo o mundo (Bacelar, 2022).

Posto isto, no presente projeto foi realizado um estudo de *tweets* com base numa entidade, como por exemplo uma figura pública ou uma empresa. Isto é, foram recolhidos um determinado número de *tweets* que incluam essa mesma entidade (@) identificada.

Após a recolha de dados e tratamento destes, foi realizada uma pesquisa para identificar os tópicos mais falados e analisar o sentimento transmitido pelos utilizadores. Então foi feita uma classificação dos seus utilizadores em três grupos distintos - a favor, neutros ou contra – extrapolando-se assim uma ideia geral da opinião da sociedade. Deste modo, permite às entidades entender como estão a ser vistas pela sociedade, uma vez que se existir uma quantidade significativa de *tweets* "contra" o que pode significar uma situação de insatisfação ou discordância.

O *Machine Learning* (ML) tem recebido uma atenção considerável nos últimos tempos, devido à capacidade de prever com precisão uma grande variedade de negócios. Adicionalmente tem também conquistado a sociedade devido ao facto de conseguir extrair conhecimento através de dados (Murdoch et al., 2019). O ML pode ser utilizado em diversas áreas, como por exemplo, na medicina para ajudar a prever o

risco de pacientes vulneráveis a contraírem covid-19 (Assaf et al., 2020) ou até para ajudar a poupar energia em casas inteligentes (Machorro-Cano et al., 2020).

No contexto deste projeto o ML vai ser utilizado para inferir o sentimento dos *tweets* extraídos sobre determinada entidade, conseguindo assim desta maneira classificar como a favor, contra ou neutros. Após esta análise serão recomendados conteúdos aos utilizadores de acordo com os seus sentimentos, de forma a contribuir para uma opinião mais construtiva e real do assunto em questão.

Supondo uma entidade como uma empresa, com este tipo de conhecimento é possível saber como esta se enquadra no mercado, se está a ser referida e quais os temas que estão a ser comentados nas redes sociais. Caso o *feedback* seja positivo é possível determinar se a empresa está num bom caminho para o sucesso, caso contrário também é bom obter este tipo de conhecimento para poder melhorar os aspetos negativos. Este tipo de raciocínio pode ser aplicado às figuras públicas ajudando a melhorar assim a sua imagem perante a sociedade e o seu público-alvo.

Os objetivos principais deste projeto são: realizar o tratamento de dados para determinar tópicos; classificar *tweets* com o intuito de identificar utilizadores como positivos, negativos ou neutros; e recomendar conteúdos distintos com base nas classificações prévias.

1.2. Motivação

A sociedade atual usa cada vez mais as redes sociais, quer seja para conversar com amigos e familiares, para conhecer pessoas novas ou mesmo para partilhar as suas opiniões pessoais sobre qualquer tema.

Todos os dias são criados milhares de publicações novas, comentários e *likes*, o que proporciona milhões de dados gerados. Estes dados representam as opiniões pessoais da sociedade e as ideologias das pessoas e, sem nos apercebermos, ao colocar um simples gosto ou escrever um comentário, estamos a expor a nossa personalidade e maneira de viver (física e digital) nas redes sociais.

Assim sendo, com recurso à informação disponibilizada pelas redes sociais, obtendo os dados que com o devido tratamento, análise e a aplicação dos algoritmos adequados é possível retirar conhecimento da sociedade em geral sobre determinada entidade.

Aliando isto ao processo de classificação de utilizadores é possível entender as opiniões da sociedade sobre uma determinada entidade, podendo assim entender se esta está a ser bem vista ou não pela sociedade, ou mudar os nossos comportamentos como entidade e rumar a outras escolhas mais vantajosas, ou apenas para saber se a nossa opinião vai de encontro com as das outras pessoas. Desta forma é possível verificar uma união entre a tecnologia e a psicologia.

1.3. Definição do Problema

A exploração das redes sociais e a partilha de opiniões pessoais nestas fundamentam a motivação para este estudo, uma vez que podem possibilitar a compreensão das ideologias presentes na sociedade. Este tema é de elevada importância para figuras públicas, ou empresas, pois podemos extrair conhecimento útil das redes sociais. Por exemplo, uma empresa pode obter uma melhor compreensão do *feedback* dos seus clientes com o intuito de melhorar ou manter os seus produtos e/ou serviços. Este método pode ser mais eficaz e genuíno em comparação com a utilização de formulários que podem ser considerados entediantes de preencher, uma vez que as pessoas não sentem que estão a ser avaliadas ao partilhar as suas opiniões nas redes sociais.

O presente trabalho envolve diversos temas atuais da ciência de dados entre os quais: tratamento de dados, deteção de tópicos, análise de sentimento, identificação de perfis e sistemas de recomendação.

As fontes de dados deste projeto provêm de uma rede social e, a sua extração e tratamento carecem de alguma complexidade. Com este tipo de dados é possível identificar diversos problemas uma vez que a inserção não é controlada, ou seja, podem aparecer por exemplo, palavras abreviadas, erros ortográficos e/ou utilização de caracteres especiais no meio de palavras.

Outro fator crítico é a análise de sentimento. Os sentimentos são subjetivos, para a mesma frase, diferentes pessoas podem interpretar de maneira diferente, pelo que é possível usar os melhores modelos de ML para obter a melhor interpretação possível.

Com a utilização de um sistema de recomendação será possível enviar um novo conteúdo aos utilizadores com base no seu sentimento. Isto é, se um utilizador é completamente a favor de algo, será possível direcionar-lhe um conteúdo de caráter menos positivo de modo que este tenha um conhecimento mais abrangente e

diversificado sobre o tema e, assim, poderá construir melhor a sua opinião. O mesmo também acontece para utilizadores que expressem um sentimento negativo ou neutro.

Contudo é pouco provável conseguir mudar opiniões de alguém que já é a favor ou contra um determinado assunto, mas para utilizadores classificados como neutros ao recomendar um conteúdo positivo ou um conteúdo negativo é possível que se consiga influenciar pessoas cuja opinião ainda não está definida.

Estas técnicas já foram utilizadas em investigações anteriores (Ray et al., 2021), porém não para o mesmo objetivo. Estes são temas atuais e muito voláteis por estarem em constante melhoria devido ao facto de serem usados em diversos contextos recentes e, consequentemente, estas técnicas acabam por ser melhoradas investigações após investigações.

1.4. Objetivos de Investigação

O objetivo do presente trabalho visa identificar perfis com base na análise de sentimento. Para alcançar este objetivo propõe-se seguir os seguintes passos:

- Passo 1 → Tratar os dados recolhidos.
- Passo 2 → Detetar de tópicos abordados sobre determinada entidade.
- Passo 3 → Aplicar algoritmos de *Machine Learning* para detetar sentimento.
- Passo 4 → Classificar utilizadores como a favor, neutros ou contra.
- Passo 5 → Efetuar recomendações de conteúdo, de acordo com a sua opinião.

1.5. Estrutura da Dissertação

O presente documento encontra-se dividido em diversos capítulos, começando com o **Capítulo 1** onde é feita uma introdução que conta com a motivação, a contextualização, a definição do problema, os objetivos propostos, os resultados que se esperam obter. No **Capítulo 2** encontra-se o estado da arte, onde é feita uma apresentação dos temas e tecnologias utilizadas no contexto onde estas serão utilizadas e onde está demonstrado o trabalho efetuado por outras entidades. No **Capítulo 3** apresenta-se o caso de estudo onde se encontram os requisitos e está representada a arquitetura geral do projeto. No **Capítulo 4** é descrito como se desenvolveu a metodologia realizada neste projeto, tanto de pesquisa como de desenvolvimento. No **Capítulo 5** são apresentados e discutidos os resultados obtidos. Por fim no **Capítulo 6** é realizada uma conclusão de todo o trabalho bem como a proposta de desenvolvimento futuros.

2. Estado da Arte

2.1. Redes Sociais

As redes sociais tornaram-se inevitáveis na vida moderna, dado que além de serem usadas para passar o tempo e socializar (J. Carpenter et al., 2022), têm ganho cada vez mais importância na sociedade, tanto a nível profissional, na área de negócio, marketing e política (Bruns & Burges, 2015), como a nível pessoal. Todavia, tal como em tudo, existem aspetos negativos e positivos associados a estes meios.

Muitos líderes políticos estão preocupados com a utilização das redes sociais e desencorajam o seu uso principalmente nas escolas (J. P. Carpenter & Krutka, 2014; Warnik et al., 2016). Porém, na realidade os próprios professores usam as redes socias como um meio de partilha de conhecimento, uma vez que possibilita a criação de grupos, onde é possível debater assuntos, partilhar material escolar e exemplos práticos (Lantz-Andersson et al., 2017) que comprovam a aprendizagem dos alunos. Como é sempre preciso duvidar de tudo que se encontra na *internet*, os professores demonstram preocupação com a qualidade das fontes que escolhem e, de um modo geral, é na plataforma *Pinterest* que os professores mais recorrem para obter ideias, ferramentas e materiais (Schroeder et al., 2019).

Atualmente as redes sociais, como por exemplo, a rede social X, o *Facebook* e o *Youtube*, são usadas diariamente por praticamente toda a sociedade. Estas são uma grande fonte de dados, conhecidos como dados sociais, visto que as pessoas discutem os seus acontecimentos diários e partilham as suas opiniões nas redes sociais (H. Manguri et al., 2020). As empresas já têm vindo a utilizar as redes sociais para competir com os seus concorrentes, seja para promover produtos ou para manter a sua reputação (Arora et al., 2019).

As redes sociais têm tanta importância na sociedade que já existe relação direta com doenças mentais (Li et al., 2022). A doença mais associada às redes sociais é FoMO – *Fear of Missing Out,* que se pode definir como uma apreensão profunda de que outros possam estar a ter experiências gratificantes das quais se está ausente (Przybylski et al., 2013), deixando o utilizador com a sensação de tristeza pois sente que devia estar a aproveitar o tempo para fazer mais atividades ou aproveitar o tempo de melhor maneira.

De modo a desenvolver o presente projeto foi escolhida como fonte de dados a rede social X. Um dos motivos para esta escolha é o facto de ser a rede social mais famosa na partilha de opiniões, sendo que a sua popularidade ultrapassa qualquer tipo de *blogs* e/ou *websites* da mesma ideologia (Shepherd, 2023). Em 2021, todos os dias foram publicados quase 500 milhões de *tweets* na rede social X que originavam 8 TB[1] de dados (Bose et al., 2021). Além disso a rede social X é utilizado por diversas pessoas de diversas gerações e culturas, desde celebridades, atores, políticos, pessoas de negócio e líderes religiosos refletindo assim a opinião de todos os grupos sociais (Mandloi & Patel, 2020). Deste modo é possível considerar a rede social X como uma fonte de dados válida para identificar sentimentos de pessoas (Tyagi & Tripathi, 2019).

A rede social X conta com quatro grandes funcionalidades (Kywe et al., 2012) sendo estas:

- ***Tweet*** – Corresponde a uma mensagem de até 280 caracteres, sobre qualquer tema/conteúdo, como atividades diárias, notícias e/ou opiniões. Os *Tweets* também podem incluir *hashtags* (representado por um cardinal – #) que servem para agrupar e identificar temas, como por exemplo, #CristianoRonaldo é utilizado quando se partilha algo relacionado com o Cristiano Ronaldo.
- ***Retweet*** – É a partilha de um *tweet* de outro utilizador, podendo ou não acrescentar algo relativamente ao assunto do *tweet*.
- ***Follow*** – Significa seguir outro utilizador, ou seja, o primeiro utilizador vai receber todos os *tweets* e *retweets* do utilizador que seguiu.
- ***Mention*** – No corpo do *tweet* é(são) mencionado(s) outro(s) utilizador(es) através do prefixo "@" antes do nome do utilizador, sendo estes nomes únicos no mundo de modo a identificar o utilizador específico, como por exemplo, @cristiano que corresponde à conta da rede social X do Cristiano Ronaldo.

No presente projeto serão aplicadas técnicas de *Web Scrapping* (WS) (explicado posteriormente) para fazer o *download* de um determinado número de *tweets* dentro de um intervalo de tempo de todos os *tweets* que tenham mencionado uma determina entidade.

[1] TB - Terabyte

2.2. Recolha de Dados

A rede social X possuí uma *Application Programming Interface* (API) que permite aceder e recolher dados da plataforma. Esta API foi projetada para obter dados de acordo com as preferências de utilizador e possui duas modalidades distintas: *standard* e *premium*. Na versão *standard* é possível descarregar um limitado número de *tweets* gratuitamente dos últimos sete dias, mas como sete dias é um curto espaço de tempo não se consegue obter a quantidade de dados necessária para a identificação de perfis. Enquanto na versão *premium* já é possível obter dados em qualquer intervalo de tempo, contudo esta já tem custos monetários associados (Chen et al., 2022).

Devido às modalidades da API, para o desenvolvimento deste projeto não vai ser usada a API da rede social X, mas sim técnicas de WS. WS consiste numa API para extrair informação de uma página web (A L et al., 2020), também conhecida por *Screen Scraping* e *Web Data Extration* e, está desenvolvida para se adaptar e recolher a informação de todos os tipos de *websites* e lojas *online* (Thomas & Mathur, 2019). WS está dividido por três etapas: análise do *website*, *Website Crawling* e organização de dados. Estas três fases requerem conhecimentos de várias tecnologias sendo as principais linguagens usadas *R* ou *Python* e, como muitas vezes não é possível automatizar completamente estas etapas é necessário envolvimento humano (Krotov et al., 2020).

Com o uso de WS, podem-se colocar questões políticas, como por exemplo, ao recolher dados da rede social X pode-se ser julgado a nível de *copyright*. No presente caso, esta questão não se aplica devido ao facto de a rede social X não possuir os direitos de autor do conteúdo que é partilhado pelos seus utilizadores e, por isso é possível copiar o material e usar numa pequena escala seguindo o princípio de utilização justa. Complementarmente, ideias e/ou opiniões não sofrem riscos de *copyright*, apenas a forma específica de as representar (Krotov & Silva, 2018). Posto isto, como o objetivo do trabalho é analisar tweets para conseguir obter o sentimento que este transmite, não irá sofrer qualquer tipo de problema em relação ao *copyright*.

A prática de usar o WS com um propósito fraudulento ou ilegal é punível por diversas leis (Krotov & Silva, 2018). Um exemplo possível de demonstrar é o facto de se alguém aceder a dados confidenciais e/ou protegidos poder ser processado sob "*Computer Fraud and Abuse Act*" (surgiu em 1986, nos Estados Unidos da América

como a primeira lei federal para combater o *hacking* (*NACDL - Computer Fraud and Abuse Act (CFAA)*, 2022)). Para o presente trabalho esta lei não se aplica uma vez que os dados recolhidos serão *tweets* e estes são públicos e acessíveis por toda a gente.

Outra situação plausível de punição, é a sobrecarga ou danificação do *website* e/ou do servidor onde está alocado o *website*. Neste caso é possível ser processado por "*Trespass to chatters*" que significa danificação de bens móveis quantificáveis (Wright, 2020). No entanto, os danos têm de ser materiais e fáceis de provar em tribunal (Krotov & Silva, 2018) mas, dado as dimensões da rede social X, os seus servidores têm de estar preparados para compensar a sua quantidade de utilizadores e os milhares de dados criados diariamente e, a amostra de dados que será recolhida muito dificilmente será capaz de provocar algum dano a qualquer tipo de servidor.

O processo de recolha de dados da rede social X utilizado neste projeto será através da biblioteca *snscrape* da linguagem de programação *Python*. A biblioteca *snscrape* permite recolher dados da rede social X em *bulk* e é possível recolher dados de um utilizador específico ou de vários. Para obter os dados é necessário definir uma palavra-chave, ou seja, o nome do utilizador (entidade), definir um intervalo de tempo e ainda definir o número máximo de *tweets* que se pretendem descarregar, obtendo assim um determinado número de *tweets* para um certo intervalo de tempo e que contenham mencionado um certo utilizador (Akbari et al., 2023).

2.2.1. Preparação de Dados

Como a fonte de dados é a rede social X existem alguns métodos de limpeza de dados que são comuns (Desai & Mehta, 2016; Pradha et al., 2019):

- Na presença de *hashtag* (#), *links*, @ para mencionar outros utilizadores, *emojis*, pontuações e espaços em branco é preciso removê-los uma vez que não traduzem sentimento.
- É necessário remover palavras *stop*, isto é, palavras que não têm informações associadas, como "*the*" e "*and*".
- Ao encontrar palavras como "felizzz", que não existem no dicionário, é necessário converter estas para "feliz" de modo a serem identificadas como tendo o mesmo significado.
- Converter todo o *tweet* em maiúsculas ou minúsculas para palavras como "Obrigado" ou "obrigado" terem o mesmo significado.

Após a limpeza dos dados estar concluída corretamente, é possível avançar e começar a aplicar algoritmos para extrair sentimentos do texto em análise. Existem diversos algoritmos de *Machine Learning* que podem ser utilizados para obter análise de sentimento, como explicado posteriormente.

2.3. Modelação de Tópicos

A modelação de tópicos é uma técnica desenvolvida para produzir uma representação de documentos sob a forma de palavras-chave (Negara et al., 2019). Num modelo de tópicos, um tópico é uma lista de palavras que ocorrem juntas de forma estatisticamente significativa. Como estes modelos não conseguem entender o significado e o contexto das palavras nos documentos, estes supõem que cada parte do texto é composta por uma seleção de palavras de conjuntos prováveis em que cada conjunto corresponde a um tópico (Jelodar et al., 2019).

2.3.1. Latent Dirichlet Allocation

Ao longos dos tempos foram aparecendo diversas técnicas de deteção de tópicos. A técnica *Latent Semantic Analysis* (LSA) que era capaz de produzir uma representação de documentos na forma de coleção de palavras. Esta é o método mais conhecido por utilizar a representação do *bag-of-words* (técnica que se baseia na contagem da frequência de palavras) como representação de documentos (Landauer et al., 1998).

Como era de prever, esta técnica foi sofrendo alterações ao longo dos anos tendo surgido uma nova melhoria logo passado três anos, sendo ela denominada por *Probabilistic Latent Semantic Analysis* (PLSA) que utiliza valores probabilísticos como determinante do peso do tópico de cada assunto detetado no *dataset* (Hofmann, 2001).

Para combater os problemas e limitações do PLSA surgiu *Latent Dirichlet Allocation* (LDA) (Putri & Kusumaningrum, 2017). Este é um dos métodos escolhidos para analisar documentos de grande dimensão. O LDA pode ser utilizado para resumir, agrupar, ligar ou processar uma grande quantidade de dados, devido ao facto de este gerar uma lista de tópicos que são ponderados para cada documento (Campbell et al., 2015, p. 6). *Dirichlet* é a distribuição utilizada para descobrir tópicos no processo generativo de LDA, os resultados da distribuição são utilizados para atribuir palavras do documento a diferentes tópicos (Blei, 2012). LDA é o algoritmo mais popular e mais utilizado (Negara et al., 2019, Putri & Kusumaningrum, 2017).

2.3.2. Term Frequency – Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) é método estatístico que mostra a relevância de palavras-chave em documentos. Esta técnica tem em consideração a frequência com que uma palavra aparece num documento (*Term Frequency* (TF)) e a frequência com que a palavra aparece em todo o *corpus* e documentos (*Inverse Document Frequency* (IDF)) (Qaiser & Ali, 2018).

A formula matemática pode observar-se nas fórmulas seguintes TF (equação (1)) (C. Liu et al., 2018) em que, $n_{i,j}$ é o número de ocorrências de uma palavra no documento e $\sum_k n_{k,j}$ é o somatório de ocorrências de todas as palavras no documento.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \quad (1)$$

Para calcular o IDF (equação (2)), onde d_j é o documento, $|D|$ o número total de documentos e $|\{j: t_i \in d_j\}|$ é o número de documentos que contém a palavra.

$$idf_i = \log \frac{|D|}{|\{j: t_i \in d_j\}|} \quad (2)$$

Multiplicando as equações (1) e (2) obtemos o $tfidf_{i,j}$ (equação (3)).

$$tfidf_{i,j} = tf_{i,j} \,.\, idf_i \quad (3)$$

É possível utilizar TF-IDF para diversos fins, como por exemplo, para pesquisa de informação. Supondo um motor de pesquisa, se alguém pesquisar por *Mercedes-Benz F1 team*, os resultados vão ser apresentados com uma ordem de relevância, pelo que mais artigos relacionados com a Fórmula 1 vão ser apresentados primeiro, uma vez que TF-IDF definiu mais valor e importância a *Mercedes-Benz F1 team.* Este também pode ser usado para encontrar *Keywords* num documento, por exemplo num artigo científico as palavras com mais classificação pelo TF-IDF são as mais relevantes pelo que é possível classificar estas como *Keywords* (Stecanella, 2019).

2.4. Análise de Sentimento

Análise de sentimento ou *Opinion Mining* é o estudo computacional sobre a opinião das pessoas, eventos ou tópicos (Medhat et al., 2014). *Opinion Mining* e análise de sentimento têm apenas uma pequena diferença. *Opinion Mining* é mais focado na sociedade e destina-se a extrair e aprofundar a opinião sobre produtos, filmes ou entidades. Por outro lado, a análise de sentimento foi inicialmente formulada como uma técnica de *Natural Language Processing* (uma área da informática motivada a representar de forma automática e da melhor maneira possível a linguagem humana (Cambria & White, 2014)), de forma a ser possível extrair conhecimento de texto. Mas análise de sentimento e *Opinion Mining* enquadram-se no âmbito da análise sentimental e, como tal, podem ser abordadas como sinónimos (Tsytsarau & Palpanas, 2012).

O estudo deste tipo de análise é cada vez mais importante. Por exemplo, atualmente quando queremos comprar algum produto já não nos limitamos a perguntar opiniões apenas aos nossos familiares e amigos, pois se fizermos uma pequena pesquisa encontramos logo diversas revisões e/ou fóruns de opiniões sobre o produto em questão. Para além de que nos últimos anos é notado um grande aumento de partilhas de opiniões nas redes sociais, estas reorganizaram os negócios, influenciaram as emoções e os sentimentos do público e originaram um impacto profundo em ambientes sociais e políticos (B. Liu, 2012).

A análise de sentimento pode ser dividida em duas abordagens distintas: com base em ML e em abordagem lexical. Usando ML para detetar sentimento podemos usar por exemplo, um algoritmo de classificação (explicado posteriormente), enquanto se usarmos a abordagem lexical é utilizado um dicionário de palavras de opinião que correspondem a dados para determinar a polaridade. Deste modo são atribuídas etiquetas de sentimento (como positivo e negativo) às palavras de opinião no texto que está a ser analisado, tendo também em conta o contexto das mesmas (Aung & Myo, 2017).

Dentro da abordagem lexical existem três vertentes: a abordagem manual, a abordagem com base em dicionário e a abordagem baseada em *corpus* (Zhang et al., 2014).

- A abordagem manual é lenta quando utilizada sozinha e para combater este aspeto tem de ser combinada com outras abordagens automáticas.
- A abordagem com base em dicionário utiliza um conjunto de palavras-chave e um dicionário *online*. A essência aqui passa por recolher as palavras no texto e pesquisar o significado destas num dicionário *online* como por exemplo o *Wordnet*, algumas destas palavras podem ser associadas a conotações positivas ou negativas, no entanto, este método representa uma maneira simplificada de classificar os sentimentos, limitando-se a capturar nuances contextuais. Esta estratégia, embora eficaz para uma análise inicial, pode não abranger completamente a complexidade e subjetividade inerentes aos sentimentos expressos no texto.
- Na abordagem baseada em *corpus* são usados dados de *corpus* para identificar palavras sentimentais. Este método não é tão eficaz por si só, porque é difícil preparar um conjunto grande de palavras *corpus* para cobrir todas as possibilidades, sendo por este motivo menos eficaz que a abordagem com base em dicionário. Porém, a utilização deste é benéfica porque permite explorar dados para determinar palavras sentimentais (Nausheen & Begum, 2018).

No presente projeto vai ser usada uma análise de sentimento de *tweets* para classificar qual a percentagem de utilizadores que são a favor, contra ou neutros relativamente a uma determinada entidade. Para o objetivo proposto, que é a análise de *tweets* utilizando ML, o tipo de análise pode ser feito de duas maneiras distintas. A primeira é a classificação em três etiquetas: positivo, contra ou neutro. O segundo método é encontrar no texto palavras e identificar se estas já foram ou não classificadas sendo que, a grande dificuldade deste método é a identificação do sentido das frases, porque as frases estão relacionadas semanticamente com outras partes do texto ((Mandloi & Patel, 2020).

Antes de se aplicar qualquer tipo de algoritmo para extração de sentimento é necessário tratar os dados. Este processo é crucial pois está diretamente relacionado com o resultado final (Krouska et al., 2016).

2.4.1. Modelo Utilizado para Classificar a Análise de Sentimento

Para classificar os *tweets* foi utilizado o algoritmo *Textblob* por demonstrar uma grande exatidão quando utilizado. Outra das razões pela quais é utilizado pelos *data*

scientists é devido ao facto de ser relativamente rápido e compacto (Aljedaani et al., 2022). Este é um algoritmo bastante conhecido para análise de sentimento lexical (Sarkar, 2019).

O *Textblob* permite a tradução de *tweets* para outra linguagem, função que não vai ser utilizada pois o *dataset* será em inglês devido ao facto de existir mais conteúdo na rede social X nessa linguagem do que em português. Outro fator que influenciou a escolha deste algoritmo foi o facto deste já incluir modelos de ML pré-treinados para efetuar a análise de sentimento, o que facilita a sua implementação, pois já não é necessário voltar a treinar o algoritmo (Deitel & Deitel, 2021).

2.4.2. Machine Learning

Machine Learning (ML) é uma área da computação que estuda algoritmos e modelos estatísticos que os computadores usam para especificar uma determinada tarefa sem esta ser especificamente programada para tal fim (Mahesh, 2018). O ML é frequentemente utilizado para ensinar computadores e/ou máquinas como lidar com dados e detetar padrões e fazer previsões. Em certas ocasiões após o tratamento e visualização dos dados, não conseguimos interpretar a informação a partir dos dados, então nestas ocasiões aplicam-se algoritmos de ML. Como diz Harry Surden, ML é um ramo da inteligência artificial onde o computador e/ou máquinas têm a capacidade de aprender com a experiência, melhorando o desempenho destas ao longo do tempo. Além disso, o uso destas técnicas tem aumentado com o passar dos anos (Surden, 2014).

O ML pode ser dividido em três tipos de aprendizagem: supervisionada, não supervisionada e de reforço, como demonstra a Figura 1, sendo estes diferentes entre si (Lison, 2012). Além disso, pode-se verificar que cada um destes três tipos aprendizagem divide-se em vários subtipos, sendo de destacar os seguintes:

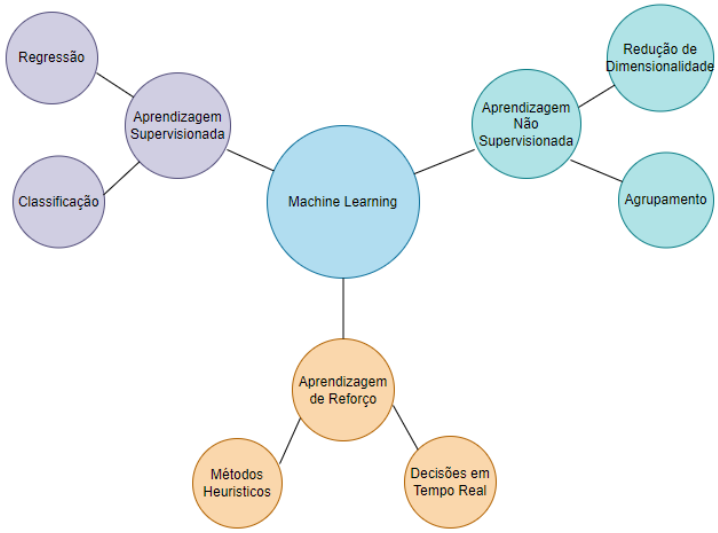


Figura 1 – Machine Learning adaptado de (Almeida et al., 2020)

Aprendizagem Supervisionada

A aprendizagem supervisionada, é utilizada quando existem exemplos de *input* e *output*, ou seja, o algoritmo aprende através de exemplos (Yilmaz, 2022) sendo o *Naive Bayes* (NB) um exemplo de um algoritmo. Esta é a aprendizagem utilizada no presente projeto uma vez que o queremos validar classificações efetuadas por modelos disponíveis.

Como dito anteriormente, a aprendizagem supervisionada é utilizada quando se conhece os dados e se sabe o resultado esperado, esta pode ser definida em duas subcategorias: regressão e classificação. Na classificação os valores utilizados são limitados a valores discretos ou categóricos (por exemplo sim ou não), enquanto os algoritmos de regressão são utilizados para valores contínuos dentro de intervalos, por exemplo para detetar uma determinada doença os valores de *input* podem ser características fisiológicas e o *output* seria a classificação de cada indivíduo (F. Posada–Quintero et al., 2021).

Naive Bayes

O algoritmo de NB é baseado no teorema de *Bayes*, este teorema consiste na abordagem lógica para atualizar probabilidades das hipóteses com base em novas evidências, desempenhando assim um papel fundamental na ciência.

Antes de se entender o teorema de *Bayes* é necessário perceber o que é a probabilidade condicionada $P(A|B)$. Esta é a determinação da probabilidade de um acontecimento A sabendo que o acontecimento B aconteceu. Matematicamente, corresponde à

probabilidade da interceção de A com B dividindo a probabilidade de B, como demonstrado na equação (4)

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad (4)$$

O teorema de *Bayes* pode ser definido como o cálculo da probabilidade de um evento acontecer com base em informações prévias ou conhecimento prévio que possa estar associado ao evento em questão.

Representado pela forma demonstrada na equação (5)

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (5)$$

Em que a $P(A)$ e $P(B)$ representam a probabilidade de um evento acontecer e têm de ser diferentes de zero e a $P(A|B)$ é a probabilidade condicionada (a $P(A)$ sabendo que B já aconteceu).

Segundo esta fórmula pode-se concluir que o Teorema de *Bayes* pode ser utilizado para obter a probabilidade de uma hipótese com base nos dados observados na equação (6):

$$P(Hipóteses|Dados) = \frac{P(Dados|Hipóteses) \cdot P(Hipóteses)}{P(Dados)} \quad (6)$$

em que $P(Dados|Hipóteses)$ é a probabilidade dos dados dada a hipótese ("se a hipótese for verdadeira, então qual é a probabilidade de observar estes dados?"), $P(Hipóteses)$ é a probabilidade prévia da hipótese ("qual é a probabilidade à priori da hipótese?"), e $P(Dados)$ é a probabilidade de observar os dados, independentemente da hipótese especificada. A probabilidade prévia é também designada por inicial. Deste modo, o modelo de NB refere-se à construção de um modelo probabilístico *Bayesiano* que atribui uma probabilidade de classe posterior a uma instância (Berrar, 2018).

Existem diversos tipos de classificadores de *Naive Bayes* (IBM, 2023b):

- *Gaussian Naive Bayes* – Esta variante de NB é utilizada com distribuições *Gaussianas*, ou seja, com variáveis contínuas. Este modelo é ajustado encontrando a média e o desvio padrão de cada classe.
- *Multinomial Naive Bayes* – Este classificador de NB é útil ao utilizar dados discretos, é utilizado, como por exemplo, em contagens de frequências e na análise de sentimento (utilizado no presente projeto).
- *Bernoulli Naive Bayes* –Outra variante de NB é utilizada com variáveis *booleanas*, ou seja, variáveis com dois valores, *True* e *False* ou, zero ou um.

Redes Neuronais

As redes neuronais, também chamadas de redes neuronais artificiais ou redes neuronais simuladas, são essencialmente a espinha dorsal dos algoritmos de *deep learning*, que por sua vez são parte do universo mais amplo do *ML*. Estas ganharam este nome e estrutura porque foram inspiradas pelo cérebro humano, imitando a forma como os neurônios biológicos se comunicam ao enviar sinais uns para os outros.

As redes neuronais são compostas por várias camadas ou nós, incluindo o nó de entrada, uma ou mais camadas ocultas e um nó de saída. Cada nó ou neurônio artificial nessas camadas está conectado a outros nós, possuindo um peso e um limite. Se a saída de um nó exceder o limite especificado este será ativado e vai transmitir dados para a próxima camada da rede. Caso contrário, nenhum dado será transmitido para a próxima camada (IBM, 2023).

Neste projeto utilizaram-se Redes Neuronais *Feedforward* (RN). Nestas a informação segue apenas numa direção na rede: dos nós de entrada, passa pelas camadas ocultas e, por fim, pelos nós de saída. Ao contrário das redes com ligações de retorno, como as que têm realimentação, nas RN a saída da rede não volta para influenciar a própria rede. Essas redes são basicamente composições de modelos mais simples, como os neurónios sigmoides, como apresenta a Figura 2 (N. Kumar, 2019).

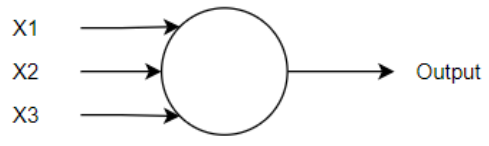


Figura 2 - Exemplo de um Sigmoide adaptado de (Narayan, 1997)

Como explicado anteriormente, numa RN a informação segue apenas numa direção e, por isso, pode ser representada como ilustrado na Figura 3.

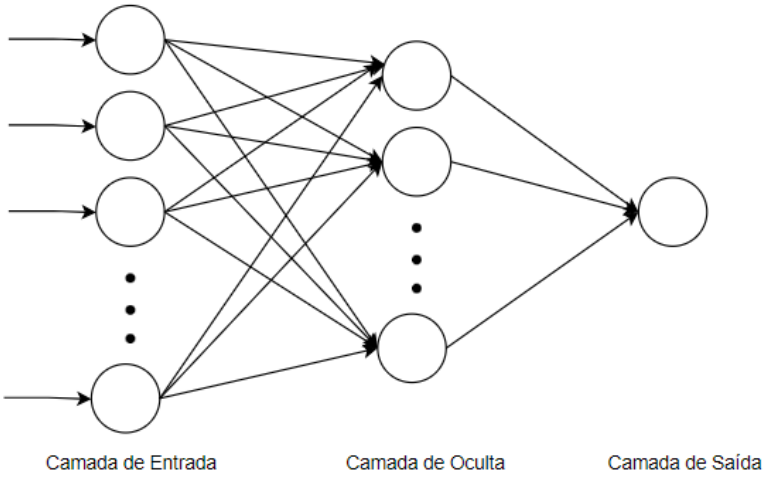


Figura 3 – Exemplo de uma RN de três camadas adaptado de (Svozil et al., 1997)

Aprendizagem Não Supervisionada

Aprendizagem não supervisionada é usada para encontrar padrões (Sathya & Abraham, 2013). Um exemplo de um algoritmo para este tipo de casos é o *K-Means*, este tem sido utilizado numa grande variedade de áreas tais como reconhecimento de padrões, mineração de dados e bioinformática, devido á sua simplicidade e eficácia (Peng et al., 2015).

Aprendizagem de Reforço

A aprendizagem de reforço é distinta dos tipos de aprendizagens anteriores, este aprende conforme o meio que se encontra e não através de dados, por exemplo um carro autónomo com o passar do tempo, é capaz de adquirir conhecimento, com objetivo de encontrar sempre a melhor opção, isto é, a ação é feita com base no ambiente que se encontra, quanto melhor for a escolha melhor é a recompensa que o ator recebe (Sutton & Barto, 2015).

Um agente que é colocado em campo, como, por exemplo, um carro autónomo com o passar do tempo, terá de ser capaz de adquirir conhecimento ao longo do tempo, com o objetivo de optar sempre pela melhor ação. Em resumo, o agente efetua uma ação com base no ambiente em que está, essa ação é analisada e, quanto melhor for a escolha, melhor recompensa o agente recebe (Sutton e Barto, 2018).

2.5. Sistemas de Recomendação

Um sistema de recomendação é um sistema inteligente que é capaz de fazer sugestões sobre algo aos utilizadores que possam ter interesse em determinado tema (Kywe et al., 2012).

Nos tempos atuais muitas lojas *online*, como por exemplo, *Amazon* ou *Ebay*, recorrem a sistemas de recomendação para assegurar os seus clientes e fazer com que estes vejam mais produtos que possam ter interesse de modo a realizarem mais vendas, aumentando assim os lucros. Hoje em dia tornou-se impensável a não utilização de sistemas de recomendação em serviços de *e-commerce* (Yu, 2012).

O presente trabalho utilizará sistemas de recomendação para enviar conteúdos aos utilizadores com base no seu sentimento, podendo fazer com que este tenha uma opinião mais construtiva e mais informada sobre determinada entidade.

Existem diversos tipos de sistemas de recomendação (Das et al., 2017) que se podem dividir em dois grupos: os personalizados e os não personalizados (Figura 4).

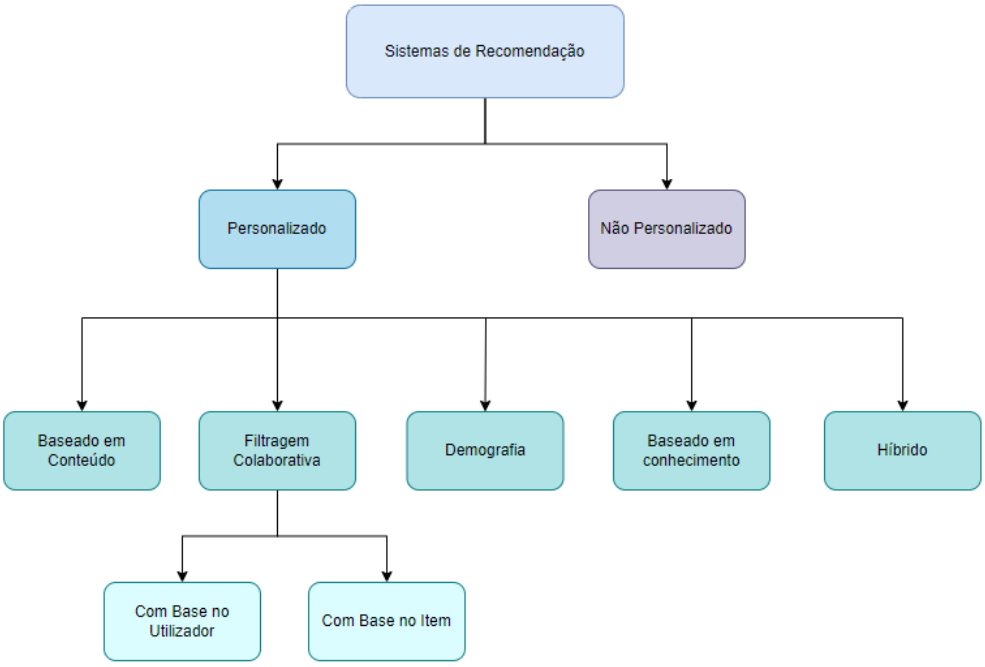


Figura 4 - Sistemas de Recomendação adaptado de (Das, Sahoo, e Datta 2017)

2.5.1. Baseado em conteúdo

Este método consiste em técnicas de filtragem com base em preferência e/ou descrição do artigo a um cliente. Em suma, estes algoritmos sugerem itens ou produtos que são

parecidos aos itens que o utilizador gostou no passado ou que está à procura no presente (Das et al., 2017).

2.5.2. Filtragem Colaborativa

A filtragem colaborativa constrói um sistema com base no comportamento passado do utilizador (a classificação é dada a itens previamente escolhidos), comparando esta decisão com a de outros utilizadores. Simplificando, se uma pessoa tem interesse em assuntos em comum com outra pessoa, é provável que algum item de interesse da segunda pessoa também interessem à primeira. Esta comparação pode ser aplicada com foco nos itens ou nos utilizadores (Das et al., 2017).

2.5.3. Demografia

A filtragem com base em demografia considera dados de utilizadores como idades, género, situação laboral e, até localização. A recomendação é feita com base nas semelhanças demográficas (Das et al., 2017).

2.5.4. Baseado em Conhecimento

Este método é apenas utilizado em domínios específicos, onde o algoritmo tem em consideração o conhecimento sobre um item e preferências do utilizador (especificamente solicitadas), sendo estes dados obrigatórios antes de qualquer tipo de recomendação. A precisão deste modelo tem em conta a utilidade da informação fornecida ao utilizador. Estes podem ser de dois tipos: *case-based* ou *constraint-based*.O método *case-based* consiste em casos centrados em itens comparáveis com base em diferentes características, já o método *constraint-based* depende de um conjunto de princípios de recomendação com características inequívocas (Das et al., 2017).

2.5.5. Híbrido

De modo a construir uma recomendação mais sólida e precisa, é possível usar o método híbrido. Este consiste em juntar diferentes sistemas de recomendação. Por exemplo, no sistema de classificação de filtragem colaborativa, o sistema falha se existirem itens sem classificação, se estiver aliado outro sistema de recomendação este problema fica ultrapassado (Das et al., 2017).

2.6. Trabalhos Relacionados

O TF-IDF tem desempenhado um papel significativo numa variedade de casos de estudo ao longo do tempo, como exemplificado no artigo de (Kim & Gil, 2019). Neste estudo, a aplicação do TF-IDF visava a deteção de artigos semelhantes. Os autores utilizam esta técnica, para identificar palavras-chave de modo a determinar o tópico de cada artigo, facilitando posteriormente o agrupamento desses artigos similares por meio do algoritmo K-means.

Além disso, outra aplicação relevante do TF-IDF é na deteção de notícias falsas, como descrito por (R. Kumar, 2020). Nesse contexto, o TF-IDF foi combinado com o algoritmo de ML *Passive Aggressive*, resultando na deteção bem-sucedida de nove em cada dez notícias falsas. Essa abordagem alcançou uma exatidão notável de 94%, conforme destacado no artigo mencionado. Estes exemplos demonstram a versatilidade e eficácia do TF-IDF em diversas aplicações, desde a identificação de tópicos até a deteção de informações fraudulentas.

Já existem alguns artigos que envolvem estas áreas, como por exemplo, o artigo de (Gujjar & Prasanna, 2021) em que se concluiu que o TextBlob é uma solução eficaz para a análise de sentimentos. Esta ferramenta revela-se em contextos empresariais, proporcionando auxílio na tomada de decisões para maximizar benefícios. Além disso, ela pode ajudar entidades a colocar anúncios específicos direcionados aos interesses dos utilizadores e, por fim, capacita as entidades a tomarem decisões estratégicas em termos de *benchmarking*.

Outro artigo importante é (Gupta et al., 2017), que compara diversos algoritmos de ML de modo a perceber o mais preciso e, demonstra áreas onde a aplicação destes algoritmos pode ser uma mais valia, como por exemplo: no comércio, na política e no desporto. O propósito do artigo mencionado é a deteção dos melhores algoritmos para cada tipo de estudo, o que no projeto proposto isso não é o pretendido, apenas se tem em consideração a utilização de algoritmos para a análise de sentimento.

Relativamente à parte de sistemas de recomendação de sistemas existem já diversos *websites* e aplicações móveis com este propósito. Um exemplo disso é o *The Fork* (*TheFork - Reserve nos melhores restaurantes da Europa*, 2023), um sistema português que surgiu em 2011 com o objetivo de ajudar a recomendar os melhores restaurantes para uma determinada localização, sendo que atualmente já é utilizado em

mais de 17 países. Este serviço utiliza como base as avaliações dos restaurantes presentes no *TripAdvisor*, a maior plataforma para ajudar a planear férias, desde marcações de viagens, hotéis e restaurantes, ajudando a comparar os serviços disponíveis com base em preços e qualidade e já contava com mais de 463 milhões de utilizadores em 2019 (*Tripadvisor*, 2019).

A dissertação de mestrado da Margarida Jerónimo (Jerónimo, 2020) elaborou um sistema de recomendação de filmes, em que o sistema aprende o padrão de consumo dos utilizadores prevendo assim desta maneira o que irá consumir no futuro. Para a realização do sistema de recomendação de filmes foram utilizadas diversas técnicas para a recomendação, entre as quais, técnicas de classificação que tinham por base demonstrações de interesse do utilizador e, também tendo em consideração técnicas de vizinhança, onde era tido em conta a similaridade de interesses com outros utilizadores Em contraste, neste trabalho foram utilizadas base de dados disponibilizadas por *GroupLens*[2], bem como informação disponibilizada *online* pelo *website* do TMDb. No presente projeto serão utilizados *tweets* retirados através de técnicas de *webscrapping* e o sistema de recomendação irá recomendar conteúdos aos grupos de utilizadores que foram classificados com a mesma opinião.

2.7. ChatGPT

O ChatGPT[3] foi desenvolvido pela OpenAI, é um modelo de linguagem desenvolvido que está na terceira versão, GPT-3.5, cada versão GPT representa um avanço relativamente à anterior, quer a nível de capacidade de entender texto quer nos seus dados.

A versão à data (GPT-3.5) conta com dados até janeiro de 2022, a sua principal função é gerar texto em forma de resposta a perguntas e estímulos fornecidos.

À data de início deste caso de estudo, a presença do ChatGPT ainda não era uma realidade, esta ferramenta surgiu apenas a meio do desenvolvimento do projeto. No entanto, devido à sua notoriedade e ampla discussão no cenário tecnológico, optou-se por utilizá-la para realizar uma comparação com os resultados. É importante notar que

[2] O GroupLens faz parte do departamento de Engenharia e Ciência da Computação da Universidade de Minnesota, dedica-se ao estudo de várias áreas, como sistemas de recomendação e comunidades online (Jerónimo, 2020)

[3] A informação deste capítulo foi obtida através de questões efetuadas ao ChatGPT.

o ChatGPT opera com seus dados "congelados", pois não possui acesso à internet. Isso implica que suas respostas podem não ser totalmente precisas em algumas situações, e sua capacidade de resposta pode variar dependendo da formulação da pergunta.

A tentativa de comparação entre os resultados do projeto e os do ChatGPT não foi bem-sucedida, enfrentando desafios por diversas razões. O ChatGPT não consegue executar todo o processo desenvolvido neste projeto de forma contínua. Portanto, foi necessário solicitar que realizasse o processo em partes distintas.

Uma limitação adicional foi identificada no tamanho do *dataset* utilizado no projeto. O computador não suporta a cópia e colagem integral do conjunto de dados devido ao seu tamanho considerável. Mas o próprio ChatGPT também não consegue processar um subconjunto de 300 registos, este gera um erro devido ao tamanho da mensagem ser demasiado grande. Isso inviabiliza a comparação direta dos resultados obtidos com o ChatGPT, uma vez que este demonstra capacidade para lidar com uma quantidade de dados significativamente menor em comparação ao *dataset* utilizado no presente projeto.

3. Caso de Uso, Requisitos e Arquitetura

O projeto teve como objetivo explorar um conjunto de *tweets* de modo a detetar tópicos, classificar os sentimentos associados e estabelecer um sistema de recomendação fundamentado no sentimento detetado.

Desta forma foi possível dividir o projeto em diversas etapas sendo a primeira a recolha de dados. De acordo com o referido anteriormente sobre o projeto, foi assim necessário recolher dados que contenham *tweets* e o nome do utilizador que escreveu esse mesmo *tweet.*

Seguiu-se então a fase de tratamento de dados. Esta é de extrema importância, uma vez que esta influencia diretamente os resultados que se irão obter, desta forma foram removidos caracteres especiais (como por exemplo números, *emojis*, #), todo o texto foi convertido para letra minúscula, foram removidas as *stopwords* (palavras sem sentimento associado) e, por fim todas as palavras foram reduzidas à sua forma canónica através do processo de lematização.

Para ser possível detetar os principais tópicos foi utilizado o TF-IDF, sendo que esta técnica permite uma representação ponderada das palavras, destacando assim aquelas que se distinguem no conjunto de dados. Deste modo, pode-se então dizer que esta técnica serve como base para a identificação de tópicos.

No que diz respeito à análise de sentimento foi usada a biblioteca *Textblob* para classificar cada *tweet* como Positivo, Negativo ou Neutro. Como esta etapa é a base do sistema de recomendação foram avaliados os resultados com o NB e as RN.

Relativamente à etapa de sistema de recomendação foi criada uma pequena interface em *Flask* onde nesta é possível escolher um tópico e um sentimento. Esta interface retornará um *tweet* sobre o mesmo tópico, mas com o sentimento contrário ao selecionado de forma a mostrar outros pontos de vista às pessoas. Porém, se a escolha for Neutro o sistema devolverá dois *tweets*, um de cada sentimento, para o utilizador obter dois pontos de vista diferentes e ganhar mais conhecimento sobre determinado tópico.

Pensado num exemplo concreto supondo que a nossa entidade é o Presidente da República, vão ser recolhidos todos os *tweets* que contenham esta conta identificada (num determinado intervalo de tempo). Seguidamente estes dados serão tratados de

modo a obter os melhores resultados possíveis. Na fase de identificação dos tópicos podíamos encontrar como por exemplo, 'Jornadas Mundiais da Juventude', 'Guerra da Ucrânia' ou 'Covid 19'. Posteriormente será feita uma análise de sentimento em cada tópico detetado e, por fim, a etapa final será a criação de um sistema de recomendação com base nos sentimentos detetados em cada tópico.

Na Tabela 1 encontra-se descrito o levantamento de requisitos

Tabela 1 - Levantamento de Requisitos

Etapa	Requisitos
Conjunto de dados	Obter um conjunto de dados com uma boa quantidade de registos
Tratamento de Dados	Tratar os dados da melhor forma de modo que os resultados obtidos sejam o mais próximo da realidade possível
Identificação de tópicos	Aplicar corretamente o TF-IDF para conseguir determinar os tópicos abordados no conjunto de dados
Análise de sentimento	Aplicar o *textblob* para obter o sentimento associado a cada *tweet*
Sistema de Recomendação	Criar um sistema de recomendação para enviar *tweet* de opiniões contrárias às escolhidas de forma a mostrar outros pontos de vista aos utilizadores

A Figura 5 representa a arquitetura do projeto identificando as suas etapas fundamentais.

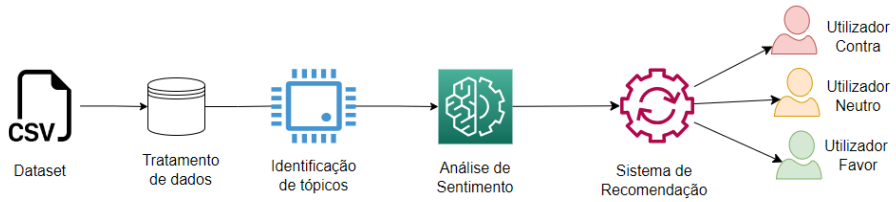


Figura 5- Arquitetura Geral do Projeto

4. Desenvolvimento

No presente capítulo será explicada a metodologia adotada no desenvolvimento do projeto, sendo constituída numa fase inicial pelo processo de pesquisa de informação e numa segunda fase pelo processo de desenvolvimento. Deste modo, para assim se obter o objetivo definido, foram usadas diversas tecnologias no desenvolvimento deste projeto.

Posteriormente à fase de pesquisa de informação, desenvolveu-se o código necessário para a realização da dissertação. Este foi desenvolvido em *Python* e foi escolhida esta linguagem de programação devido às bibliotecas que esta oferece.

Foram escolhidos dois *datasets* como base de dados para o presente projeto, os dois referem-se às eleições norte-americanas de 2020. Neste *dataset* são disponibilizados dois *Comma Separated Values* (CSV), um que contêm *tweets* com a identificação de Donald Trump, já o outro com a identificação de Joe Biden (HUI, 2020). Deste modo, todos os processos foram executados duas vezes, uma para cada candidato. Estes dados foram carregados para *dataframes* através da biblioteca *Pandas*.

Este *dataset* é de grandes dimensões e conta com 776.995 *tweets* relacionados Joe Biden e 971.097 *tweets* com Donald Trump, sendo esta razão a escolha deste *dataset* para o presente projeto.

De forma a evitar erros, facilitar a implementação dos novos processos e, evitar esperas de tempo de execução (algumas etapas são muito demoradas), os *dataframes* utilizados geram ficheiros CSV que são utilizados como fonte na etapa seguinte, conseguindo assim ter processos independentes.

Foram recolhidos todos os tempos de execução de cada etapa: no início do código era recolhido o *datetime* para uma variável e no fim da execução era guardado o *datetime* para outra variável. Com estas informações iniciais e finais tornou-se possível obter o tempo de execução de cada etapa através da subtração do tempo de fim pelo tempo de início. Todo o desenvolvimento foi realizado num computador com processador *intel* i7 da 8ª geração e com 16 GB de RAM.

4.1. Extração e Preparação de Dados

Numa fase inicial os dados foram recolhidos através da biblioteca *snscrape* de *Python* em que foram recolhidos *tweets* que continham a identificação de Donald Trump. Esta

biblioteca fazia *web-scrapping* diretamente da rede social X, contudo esta rede social mudou as suas políticas de privacidade e fechou a sua *API* a terceiros em março de 2023[4],sendo assim impossível de recolher *tweets*.

No início do desenvolvimento do projeto foram recolhidos 500 *tweets* (antes da alteração das políticas de privacidade da rede social X), de modo a testar a biblioteca, podendo-se observar o código no Apêndice A, contudo 500 *tweets* era uma amostra muito pequena para o desenvolvimento e estudo pretendido. Assim sendo, foi inevitável arranjar outra solução, utilizando-se assim um *dataset* de domínio público que contivesse *tweets* sobre as eleições norte-americanas de 2020 demonstrado um exemplo na Tabela 2.

Tabela 2 - Exemplo de Tweets do Dataset Original

Data	User	Tweet
15/10/2020 00:00	trumpytweeter	2 hours since last tweet from #Trump! Maybe he is VERY busy. Tremendously busy.
15/10/2020 00:00	Ranaabtar	You get a tie! And you get a tie! #Trump 's rally #Iowa https://t.co/jJalUUmh5D
15/10/2020 00:00	FarrisFlagg	@CLady62 Her 15 minutes were over long time ago. Omarosa never represented the black community! #TheReidOut She cried to #Trump begging for a job!

A pequena amostra de dados (registos recolhidos pelo *snscrape*) foi usada para testar todas as implementações iniciais de cada etapa uma vez que os tempos de execução eram bastante superiores devido à enorme diferença de quantidade de registos como demonstra o código no Apêndice B. Depois das alterações testadas e apenas mudando a fonte (ficheiro CSV), o código funciona para qualquer *dataset* com a mesma estrutura.

Para obter resultados mais parecidos com a realidade, os dados foram preparados da seguinte forma:

- Transformar tudo em letra minúscula de modo a analisar da mesma forma e com o mesmo sentimento palavras escritas em letra maiúscula ou minúscula. Limpar linhas vazias.
- Remover links e, identificar outros utilizadores, *hashtags*, caracteres especiais e números. Como estes não contêm qualquer tipo de sentimento associado é necessário removê-los.
- *Tokenização* do *dataset* que é um processo que consiste em dividir um texto em unidades mínimas, os *tokens*. Esta divisão foi feita com base nos espaços

[4] Esta informação foi encontrada em diferentes fóruns como por exemplo, *Reddit*, *Kaggle* e *GitHub*.

em branco entre diferentes palavras, obtendo-se assim uma lista de *tokens* em que cada token é uma palavra representada no *tweet*.

- Remoção das *stopwords*, ou seja, palavras comuns numa língua (neste caso a língua inglesa, uma vez que o texto do *dataset* se encontra em inglês) que geralmente são consideradas sem valor para a análise de texto, pois não transmitem informações específicas sobre o conteúdo do texto. Essas palavras incluem artigos pessoais, preposições e conjunções. Esta técnica é muito utilizada em tratamento de texto pois os resultados finais são significativamente melhores quando esta se aplica (Schofield et al., 2017). No caso concreto, esta lista de palavras foi manipulada uma vez que nesta se encontravam as palavras "*no" e "not"* e, ao remover estas, o sentido do *tweet* mudava completamente pelo que as mesmas não foram consideradas como *stopwords* neste trabalho.
- Lematização, de modo a ser possível classificar palavras conjugadas em diferentes tempos verbais e com o mesmo sentimento transmitido, é necessário reduzir estas à sua forma canónica.

Na Tabela 3 é possível ver as diferentes colunas desenvolvidas com as diferentes fases de limpeza de dados.

Tabela 3 - Exemplo de Tweets depois do Tratamento

Tweet	Clean	RemoveStopword	lemmatized_tokens
2 hours since last tweet from #Trump! Maybe he is VERY busy. Tremendously busy.	['hours', 'since', 'last', 'tweet', 'from', 'trump', 'maybe', 'he', 'is', 'very', 'b	['hours', 'since', 'last', 'tweet', 'trump', 'maybe', 'busy', 'tremendously', 'busy']	['hour', 'since', 'last', 'tweet', 'trump', 'maybe', 'busy', 'tremendousl
You get a tie! And you get a tie! #Trump 's rally #Iowa https://t.co/jJalUUmh5D	['you', 'get', 'a', 'tie', 'and', 'you', 'get', 'a', 'tie', 'trump', 's', 'rally', 'iowa']	['get', 'tie', 'get', 'tie', 'trump', 'rally', 'iowa']	['get', 'tie', 'get', 'tie', 'trump', 'rally', 'iowa']
@Clady62 Her 15 minutes were over long time ago. Omarosa never represented the	['her', 'minutes', 'were', 'over', 'long', 'time', 'ago', 'omarosa', 'never', 're	['minutes', 'long', 'time', 'ago', 'omarosa', 'never', 'represented', 'black', 'comm	['minute', 'long', 'time', 'ago', 'omarosa', 'never', 'represent', 'black',

Na Figura 6 é possível verificar de forma esquematizada a preparação dos dados descrita anteriormente:

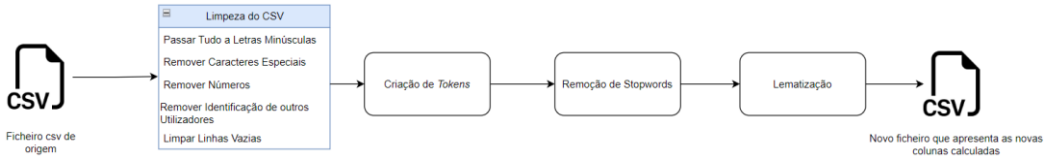


Figura 6 - Arquitetura da Limpeza de Dados

Após esta fase estar ultrapassada, é possível que o *dataset* tenha *tweets* vazios e por isso é necessário eliminar os registos onde este caso acontece.

No *dataset* existem *tweets* com palavras abreviadas e/ou com erros ortográficos que não foram corrigidas pelo processo de *limpeza* que podem afetar a veracidade dos resultados obtidos. Outro problema encontrado foi a existência de *tweets* que para

além de conterem hiperligações (que foram removidas), também continham comentários como por exemplo 'Wow' e por causa deste facto o *tweet* foi classificado como positivo posteriormente (por exemplo "*Wow https:www.examplevideo.mp4"*).

O processo de tratamento de dados, essencial para lidar com essas complexidades demorar aproximadamente 37 minutos e 31 segundos para os *tweets* relativos ao Donald Trump e 25 minutos e 52 segundos para os *tweets* de Joe Biden. Após a limpeza de dados foram criados 2 CSV com os respetivos dados (Tweets_CLEAN_JoeBiden.csv & Tweets_CLEAN_DonaldTrump.csv), para serem guardados e usados como fontes em implementações seguintes.

4.2. Aplicação de TF-IDF & Análise de Sentimento

Como referido anteriormente, o processo foi divido em etapas, então para aplicar o TF-IDF resta importar os ficheiros CSV (Tweets_CLEAN_JoeBiden.csv & Tweets_CLEAN_DonaldTrump.csv), para um *dataframe* e posteriormente aplicar o respetivo algoritmo como mostra a Figura 7.

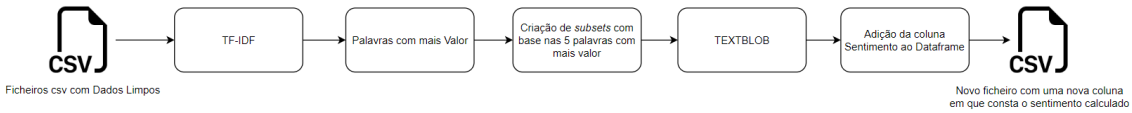


Figura 7 - Arquitetura de aplicação TF-IDF & Análise de Sentimento

4.2.1. Aplicação TF-IDF

O TF-IDF foi testado em três frases[5] de exemplo em que já se sabia o resultado esperado, e os valores obtidos foram os mesmos, o que pode servir de método para validação do código implementado, código este demostrado no Apêndice C.

Os valores obtidos podem ser baixos (em termos numéricos) devido à grande dimensão dos *datasets* (776.995 registos para Joe Biden e 971.097 registos para Donald Trump). Cada linha representa um documento e a fórmula do TF-IDF consiste, resumidamente, em dividir o número de ocorrências de cada palavra pelo número de documentos.

O TF-IDF cria uma matriz de grandes dimensões que o computador não suporta e como são muitos dados não é possível visualizar todos os valores para todas as

[5] Estas frases de exemplo foram encontradas no fórum online *kaggle* (Roan, 2020)

palavras presentes, sendo apenas possível ver um *ranking* de palavras definidos programaticamente.

Ao fim de detetar os cinco tópicos mais importantes (cinco palavras com o TF-IDF mais alto), foram criados *subsets* onde este tópico estava presente, obtendo-se assim novos *dataframes* utilizados para detetar sentimento.

4.2.2. Análise de Sentimento

De modo a detetar a análise de sentimento de cada *tweet*, foi utilizada a biblioteca *TextBlob*. Este código consiste na aplicação desta biblioteca à coluna que contém as palavras lematizadas e o algoritmo retorna valores numéricos positivos, negativos e o valor zero. Assim, para facilitar a compreensão destes valores, estes foram convertidos para 'Positivo' caso o valor obtido seja maior que zero, 'Negativo' caso o valor obtido seja menor que zero, 'Neutro' quando o valor obtido é zero e foram guardados estes registos numa coluna nova do dataframe ['Sentiment'], obtendo-se assim a classificação de cada *tweet* em cada *subset*.

Após a atribuição dos sentimentos a cada *tweet*, por meio da biblioteca *Matplotlib*, elaboraram-se gráficos que representam visualmente os valores obtidos, proporcionando uma compreensão mais intuitiva dos resultados alcançados demonstrados no Gráfico 1 da secção 5.2.2, o código encontra-se exibido no Apêndice D.

Relativamente aos tempos de execução, este processo demorou 17 minutos e 27 segundos para os *tweets* de Donald Trump e 13 minutos e 2 segundos para os de Joe Biden.

Determinação da Exatidão

Em inglês, existe uma distinção clara entre "*accuracy*" (exatidão) e "*precision*" (precisão). Entretanto, em português, esses termos muitas vezes são tratados como sinónimos. Durante a análise, calculou-se a exatidão. A exatidão, ou "accuracy", consiste em avaliar a exatidão global do modelo, obtendo assim uma visão global da quantidade de previsões corretas em todo o conjunto de dados, esta pode ser definida matematicamente pela fórmula da equação (7) (Deng et al., 2016).

$$\text{Exatidão} = \frac{\sum_{i=1}^{n} Nii}{\sum_{i=1}^{n}\sum_{j=1}^{n} Nij} \qquad (7)$$

Por outro lado, a "*precision*" é uma métrica que se refere ao cálculo da quantidade de previsões corretas para uma classificação específica, este pode ser definido pela seguinte equação (8) (Deng et al., 2016).

$$\text{Precisão} = \frac{Nii}{\sum_{k=1}^{n} Nki} \qquad (8)$$

No presente contexto, utilizou-se a métrica de exatidão ("accuracy") e matrizes de confusão.

Com o intuito de testar resultados recorreu-se a dois algoritmos diferentes para calcular a exatidão, sendo estes o NB e RN, podendo-se comparar assim os métodos estatísticos tradicionais com as redes neuronais.

Estes algoritmos partem de uma base comum (mesmo *dataset*) divergindo apenas nas suas particularidades e/ou obstáculos.

De modo a aplicar os algoritmos definiu-se a coluna X (dados de treino) onde estão representados os *tweets* já tratados, sem *stopwords* e com as palavras lematizadas do tipo *string* como se pode observar na Figura 8.

Figura 8 - Demonstração da Coluna X

Já a coluna Y (dados alvo) contem as classificações “Negativo”, “Neutro” e “Positivo” como se pode verificar Figura 9.

Sentiment
Negativo
Negativo
Neutro
Neutro
Positivo
Positivo

Figura 9 - Demonstração da Coluna Y

Após o cálculo da exatidão, optou-se por calcular a matriz de confusão, uma vez que esta proporciona *insights* cruciais sobre a quantidade de elementos corretamente classificados para cada categoria, ao mesmo tempo esta informa-nos sobre a ocorrência de falsos positivos e falsos negativos. Esta abordagem revelou-se fundamental para uma análise mais aprofundada e precisa do desempenho dos modelos.(Deng et al., 2016).

Naive Bayes

Foi escolhido o NB devido à sua eficiência, efetividade e por ser mais simples de implementar (Zhang, 2004). Na Figura 10 encontra-se a arquitetura desenvolvida, o código está representado o no apêndice E.

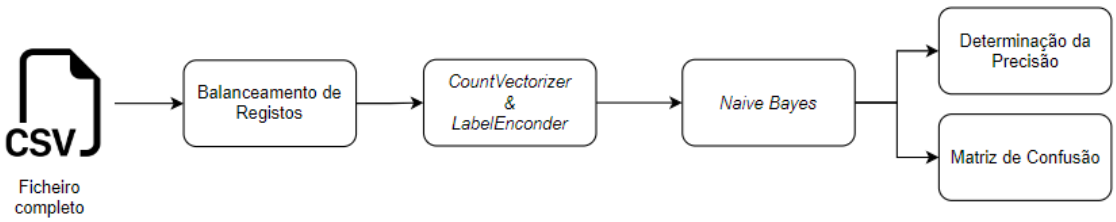


Figura 10 - Arquitetura de aplicação Naive Bayes

De forma a melhor determinar a maior exatidão escolheu-se o tópico com o número de *tweets* mais elevados, o tópico é Donald Trump no *dataset* do próprio Donald Trump. Este *dataset* contém 970 867 registos dos quais 553422 são Neutros, 254999 são Positivos e os restantes 162446 são Negativos. De maneira a calcular a exatidão da melhor forma possível, o *dataset* foi balanceado de modo a ter o mesmo registo para neutros, positivos e negativos, isto é, o algoritmo foi treinado num conjunto de dados dos quais todas as três categorias têm 162446 registos cada mostrado na Figura 11, sendo usado no modelo um total 487 338 registos.

```
Neutro      553422
Positivo    254999
Negativo    162446
Name: Sentiment, dtype: int64
---Mesmo número de Registos---
Negativo    162446
Neutro      162446
Positivo    162446
```

Figura 11 - Balanceamento dos registos NB

De forma a utilizar o NB foi necessário converter os *tweets* (já previamente tratados) para números, tendo sido este processo realizado através da classe *CountVectorizer*.

Utilizou-se o *CountVectorizer*, porque este cria uma matriz de termos-frequência, contabilizando a ocorrência de palavras em um documento. Esta representação revela-se altamente eficaz para modelos de ML tradicionais. A aplicação do *CountVectorizer* possibilita a transformação do texto em números, permitindo assim que algoritmos baseados em estatísticas, como o NB, interpretem e processem as informações de maneira eficiente. Esta abordagem de conversão, facilita a implementação de modelos, contribuindo assim para o desempenho em tarefas de classificação e análise de texto.

Na Figura 12 é possível verificar uma pequena amostra desta conversão, em que as colunas representam as diferentes palavras e as linhas representam cada *tweet*.

	0	1	2	3	4	5
0	0	0	1	2	0	0
1	0	0	0	0	0	0
2	0	0	0	0	1	0
3	0	0	0	3	0	1
4	0	1	1	0	0	0
5	1	0	0	0	0	0

Figura 12 - Conversão de Palavras para Números

Na coluna *target* utilizou-se o *LabelEncoder* para transformar os três tipos de sentimentos em dados categóricos (0,1 e 2), devido ao facto de os algoritmos ML funcionarem melhor com classes categóricas numéricas.

Na divisão dos dados de teste e treino utilizou-se o parâmetro *random_state* = 0 com a intenção que a divisão dos dados de treino e teste seja sempre a mesma em diversas execuções. A partir deste momento é possível aplicar o *Naive Bayes*.

O processo foi executado três vezes tendo tamanhos diferentes de teste 15%, 25% e 35%.

Redes Neuronais *Feedforward*

Na procura de uma abordagem alternativa para calcular a precisão, optou-se por utilizar redes neuronais do tipo *Feedforward* que tentam modelar as capacidades do cérebro humano (M. Paliwal & Kumar, 2009). Estes tipos de RN são frequentemente escolhidos para análise de texto, devido a demonstrarem bons resultados comparado a outro tipo de algoritmos. A eficácia neste contexto pode ser devido à capacidade destas redes em aprender representações complexas e hierárquicas dos dados, identificando padrões subtis que podem ser cruciais para a de análise de texto. (S. Paliwal et al., 2018) e (Kula et al., 2020)

Na Figura 13 encontra-se a arquitetura desenvolvida o código está demonstrado no apêndice E.

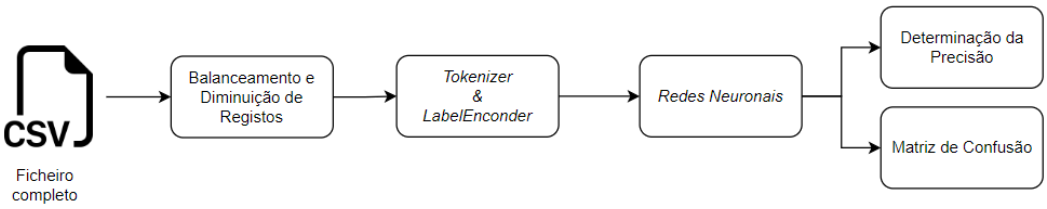


Figura 13 – Arquitetura de aplicação Redes Neuronais

Foi usado o mesmo *dataset* para treinar o modelo, os dados foram balanceados, mas devido a problemas de memória cache foi necessário reduzir a quantidade de registos do *dataset* para 9600 registos de cada sentimento demostrado na Figura 14 obtendo-se assim um total de 28800 registos. O número de 9600 registos para cada categoria foi obtido através de um processo de tentativa e erro, sendo este o número aproximado de registos que o computador consegue suportar.

```
Neutro      553422
Positivo    254999
Negativo    162446
Name: Sentiment, dtype: int64
---Mesmo número de Registos---
Negativo    9600
Neutro      9600
Positivo    9600
```

Figura 14 – Balanceamento dos Registos RN

Utilizou-se o Tokenizer para converter o texto em números. O Tokenizer consiste em converter texto para sequências de números, representados por índices, atribuindo um índice único a cada palavra. As sequências de números facilitam o treino da rede para entender a estrutura e os padrões presentes no texto, sendo assim como uma ponte

entre o texto e a RN, o que é benéfico para modelos de análise de sentimento. (*Tf.Keras.Preprocessing.Text.Tokenizer | TensorFlow v2.14.0*, 2023). Para a coluna Y, utilizou-se o novamente o *LabelEncoder*. Na divisão de dados de teste e treino também se utilizou *random_state* = 0 como no NB.

Ao aplicar o algoritmo foi definida a utilização de 10 épocas para fazer com que o modelo seja treinado 10 vezes melhorando os pesos a cada época de maneira a melhorar o desempenho de cada uma. O *bacth_size* representa a parte de dados de treino que são treinados de uma só vez, sendo este valor de 32, o que representa que são processados 32 exemplos de cada vez.

Por fim, o último parâmetro é o *validation_split* de valor de 0,1 que significa que 10% dos dados de treino serão usados para validação, que por sua vez este é usado de modo a evitar o *overfitting*. *Overfitting* acontece quando o modelo aprende demasiado bem o modelo ao ponto de afetar o desempenho negativamente (Brownlee, 2016).

A RN desenvolvida no presente projeto conta com três camadas. A primeira camada conta com uma entrada de 128 nós e uma função de ativação *ReLU* (função matemática usada com frequência as RN que ativa o nó quando a entrada é positiva). A segunda camada contém 64 nós e usa novamente a função *ReLU*. E por fim, na camada de *output* contamos com 3 classes (Positivo, Negativo, Neutro) e é utilizada a função *softmax* como função de ativação de saída (função mais frequentemente usada para problemas de classificação), sendo que nestas a saída é sempre positiva e permite determinar a probabilidade indicando a confiança dos resultados de cada classe.

Ao compilar o modelo também se recorreu a mais três parâmetros. O parâmetro loss='sparse_categorical_crossentropy', em que o *loss* representa a função de perda do modelo e o valor 'sparse_categorical_crossentropy' mede a discrepância entre as probabilidades previstas pelo modelo e os rótulos reais. O optimizer='adam', o *optimizer* faz com que os pesos sejam ajustados durante o treino e o valor 'adam' faz com que este ajustamento dos pesos seja de acordo com o valor da perda. Por fim, o último é o metrics=['accuracy'] que permite calcular a precisão do modelo.

A partir deste momento resta aplicar o algoritmo, correndo o processo 3 vezes tendo tamanhos diferentes de teste 15%, 25% e 35%.

4.3. Sistema de Recomendação

Para o sistema de recomendação foi desenvolvida uma pequena aplicação em *Flask.* Esta é uma *framework* que permite aos programadores ver o lado do utilizador e ter todo o controlo necessário sobre a aplicação tanto a nível de *backend* como *frontend* (Grinberg, 2018).

Este sistema consiste na escolha do tópico e sentimento que se pretende analisar, isto é, a escolha do tópico vai determinar o *dataframe* que será utilizado, obtendo assim os *tweets* desse tópico. Ao eleger o sentimento a *interface* retornará *tweets* de uma maneira aleatória. Existem diferentes comportamentos para os diferentes sentimentos pelo que ao escolher o sentimento 'Positivo' a interface devolverá um *tweet* aleatório classificado como negativo, ao selecionar o sentimento 'Negativo' será retornado um *tweet* aleatório positivo e, por fim, ao optar por sentimento 'Neutro' serão devolvidos dois *tweets* aleatórios, um positivo e um negativo. Na Figura 15 encontra-se esquematizada a arquitetura desta aplicação e o código desenvolvido está presente no Apêndice F.

Esta regra foi a escolhida pelo facto de o objetivo deste sistema de recomendação ser o de tentar que os utilizadores estejam o mais informado possível e, que estes percebam outros pontos de vista de modo a proporcionar opiniões mais construtivas e menos fanatismos.

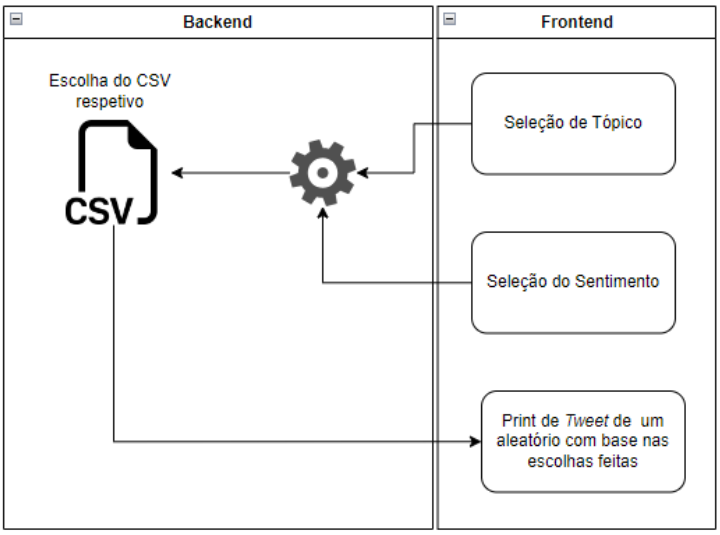


Figura 15 – Arquitetura aplicação Flask

Esta parte do caso de estudo é a mais rápida a executar são apenas de cinco segundos pois todo o processamento demorado foi executado anteriormente.

5. Análise dos Resultados

No presente capítulo serão demonstrados os resultados pelas implementações abordadas no capítulo anterior.

A fonte de dados utilizada neste projeto não incluía nenhuma coluna com o sentimento já calculado. Deste modo, como mencionado por Gujjar e Prasanna em 2021, para determinar o sentimento utilizou-se o *textblob* devido à sua eficiência (Gujjar & Prasanna, 2021). Todos os cálculos efetuados no capítulo que se segue são elaborados e avaliados com base nos resultados obtidos pelo desenvolvimento do projeto.

5.1. Resultados da Extração e Preparação dos Dados

Como referido anteriormente, através da biblioteca *sncraspe* foram recolhidos 500 registos e, nestes era guardado a data do *tweet* o nome de utilizador e o próprio *tweet.* Na Figura 16 é possível ver um enxerto dos dados obtidos.

2023-04-20 13:32:35+00:00	Flyte_Mike	@JoeBiden I miss the days when my real President @realDonaldTrump would speak and Patriots in Amer…
2023-04-20 13:32:21+00:00	miranda_bissman	@realDonaldTrump hello mr.president, when are you & the real American men going to take back A…
2023-04-20 13:32:05+00:00	scott4wa	@Susan_Dupres @SpeakerPelosi @realDonaldTrump Credit goes to @JesseKellyDC for it. It fits perfectly.

Figura 16 – Exemplo de dados obtidos

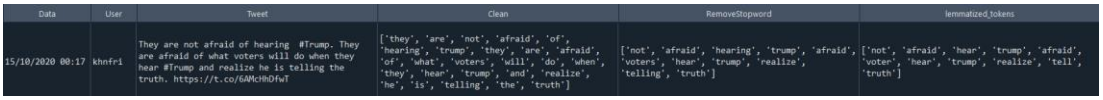
Quando esta biblioteca deixou de ser possível de ser utilizada e se recorreu a *datasets* disponibilizados, estes continham 21 colunas (*created_at, tweet_id,tweet,likes,retweet_count,source,user_id,user_name,user_screen_name,user_description,user_join_date,user_followers_count,user_location,lat,long,city,country,continent,state,state_code,collected_at*), mas destas só foram contidas em consideração a data, o utilizador e o *tweet*, (demonstrado na Figura 16) os mesmo campos que foram extraídos através do *snscrape.*

Após a aplicação do processo de limpeza, é gerado um *dataframe*, como no exemplo da Figura 17, que conta com mais colunas que aquelas que existiam inicialmente, entre as quais:

- Coluna ‘Clean’ → nesta coluna encontramos o *tweet* com a limpeza efetuada. Nesta coluna é possível observar o *tweet*, todo em letras minúsculas, sem *links*, sem outros utilizadores identificados, sem caracteres especiais e/ou números e com as palavras *tokenizadas.*
- Coluna ‘RemoveStopword’ → na presente coluna são removidas as *stopwords* presentes na coluna ‘Clean’, contudo a lista das *stopwords* foi alterada pois a lista

por defeito removia as palavras ‘no’ e ‘not’, o que não era pretendido. Deste modo a lista foi alterada para não remover estas palavras.

- Coluna ‘lemmatized_tokens’ → esta coluna apresenta as palavras no seu estado canónico, devido ao processo de lematização. Este processo foi aplicado sobre a coluna ‘RemoveStopword’.



Data	User	Tweet	Clean	RemoveStopword	lemmatized_tokens
15/10/2020 00:17	khnfri	They are not afraid of hearing #Trump. They are afraid of what voters will do when they hear #Trump and realize he is telling the truth. https://t.co/6AMcHhOfwT	['they', 'are', 'not', 'afraid', 'of', 'hearing', 'trump', 'they', 'are', 'afraid', 'of', 'what', 'voters', 'will', 'do', 'when', 'they', 'hear', 'trump', 'and', 'realize', 'he', 'is', 'telling', 'the', 'truth']	['not', 'afraid', 'hearing', 'trump', 'afraid', 'voters', 'hear', 'trump', 'realize', 'telling', 'truth']	['not', 'afraid', 'hear', 'trump', 'afraid', 'voter', 'hear', 'trump', 'realize', 'tell', 'truth']

Figura 17 – Exemplo da Limpeza de Dados

5.2. Resultados da Aplicação de TF-IDF & Análise de Sentimento

5.2.1. Resultados TF-IDF

Para facilitar a leitura visual dos valores do *TF-IDF*, estes foram arredondados. Na Figura 18 é possível ver as cinco palavras com mais valor e o seu valor e, também é possível verificar o tamanho (número de registos) do novo *dataframe* subjacente a cada tópico.

```
Word: trump -> 69.6833506713
Lenght NEW_df_trump -> 970867
Word: election -> 29.6088162575
Lenght NEW_df_election -> 271876
Word: biden -> 28.0017644066
Lenght NEW_df_biden -> 329611
Word: donaldtrump -> 22.1563541883
Lenght NEW_df_donaldtrump -> 138027
Word: vote -> 19.5596607167
Lenght NEW_df_vote -> 164763
```

Figura 18 – Exemplo de TF-IDF

Como mencionado anteriormente depois de ter os novos *dataframes* baseados nos tópicos segue-se a aplicação da análise de sentimento.

5.2.2. Resultados Análise de Sentimento

A biblioteca *textbolb* para classificar sentimento foi aplicada à coluna ‘*lemmatized_tokens’* e este valor foi guardado numa nova coluna designada de ‘*Sentiment*’. Na Figura 19 encontra-se demonstrado um exemplo do *dataframe*.

Data	User	Tweet	lemmatized_tokens	Sentiment
15/10/2020 00:39	yeahrightinc	I suspect we will never hear #Trump say much about fact his own son infected with #COVID19 due to his recklessness. Poor kid... child abuse. I am sure Trump scolds him for coughing and saying he's sick. LIke his own father, hates weakness... monster.	['suspect', 'never', 'hear', 'trump', 'say', 'much', 'fact', 'son', 'infect', 'covid', 'due', 'recklessness', 'poor', 'kid', 'child', 'abuse', 'sure', 'trump', 'scold', 'cough', 'say', 'he', 'sick', 'like', 'father', 'hate', 'weakness', 'monster']	Negativo

Figura 19 – Exemplo de Análise de Sentimento

De modo a entender os resultados obtidos, foram criados gráficos através da biblioteca *matplot* de *Python*, em que é possível verificar que relativamente ao tópico *Trump* no *dataset* do Donald Trump, a maior parte dos *tweets* foram classificados como neutros, seguidos dos *tweets* classificados como positivos e por fim os negativos.

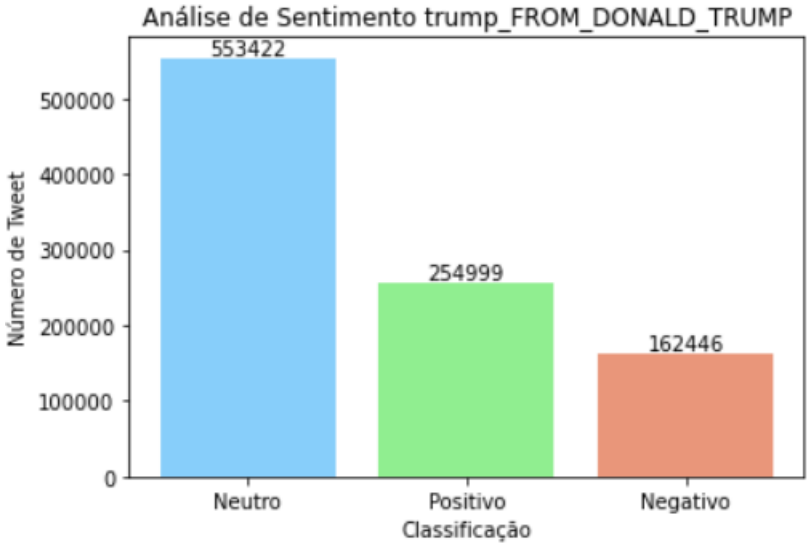


Gráfico 1 – Análise Sentimento Donald Tump tópico Trump

No gráfico de exemplo demonstrado anteriormente é possível ter uma perceção de diferentes quantidades, mas não é possível entender ao certo quanto é a diferença entre as diferentes classificações. Afim de entender melhor estas diferenças, foram calculadas as percentagens de cada sentimento detetado estando as mesmas demonstradas na Tabela 4.

Tabela 4 – Análise de Sentimento Percentagens

Neutro	Positivo	Negativo
57%	26%	17%

Determinação da Exatidão

Na presente secção serão apresentados os resultados da exatidão de respetivas matrizes de confusão para cada um dos dois algoritmos utilizados e para cada um dos tamanhos de testes distintos.

Naive Bayes

Na Tabela 5 estão representados os resultados da exatidão obtidos onde é possível observar que o valor é de 72%, o que significa o modelo conseguiu classificar 72% das vezes os sentimentos corretamente.

Tabela 5 – Resultados Naive Bayes

Testset	15%	25%	35%
Naive Bayes	0.72368 ≈ 72%	0.72276 ≈ 72%	0.72319 ≈ 72%

Nas Tabela 6, Tabela 7 e Tabela 8 é possível ver a matriz de confusão, esta apresenta a quantidade de previsões corretas e incorretas para cada categoria de sentimento no modelo de classificação. Estas tabelas fornecem uma visão detalhada do desempenho do modelo, mostrando capacidade de acertar e errar em relação a cada classe de sentimento.

Tabela 6 – Matriz de Confusão NB 15%

	0	1	2
0	21133	908	2394
1	5470	13524	5340
2	4760	1315	18257

Tabela 7 – Matriz de Confusão NB 25%

	0	1	2
0	35176	1507	4037
1	9093	22599	8809
2	7929	2145	30540

Tabela 8 – Matriz de Confusão NB 35%

	0	1	2
0	49459	2059	5500
1	12855	31434	12486
2	11473	2867	42436

Todo o processo de calcular a exatidão através do NB demorou em média 32 segundos para cada dos três tamanhos de treino distintos.

Redes Neuronais *Feedforward*

No que diz respeito aos resultados das RN pode ser possível verificar na Tabela 9 que para o tamanho de testes de 15% e 25% obteve-se uma exatidão de 87% e para o tamanho de 35% de dados de teste obteve-se 86%. Como a diferença é apenas de um por cento, podemos considerar esta diferença irrisória.

Tabela 9 – Resultados Redes Neuronais

Testset	15%	25%	35%
Redes Neuronais	0.87407 ≈ 87%	0.86708 ≈ 87%	0.86141 ≈ 86%

Da mesma forma que se observou anteriormente, nas Tabela 10, Tabela 11 e Tabela 12, é possível verificar também a matriz de confusão para cada um dos diferentes tamanhos de treino.

Tabela 10 – Matriz de Confusão RN 15%

	0	1	2
0	1188	47	179
1	315	779	363
2	317	80	1052

Tabela 11 – Matriz de Confusão RN 25%

	0	1	2
0	2009	182	171
1	93	2292	62
2	255	194	1942

Tabela 12 – Matriz de Confusão RN 35%

	0	1	2
0	2741	245	373
1	124	3147	119
2	323	213	2795

Este processo de calcular a exatidão através de RN demorou em média 10 minutos e 36 segundos para os 3 tamanhos de treino distintos demorando cerca de um minuto para cada época.

Tanto o NB como as RN alcançaram bons resultados, com taxas de exatidão de 72% e 87% respetivamente, pelo que podemos concluir que as classificações são corretas na maioria das vezes.

Essas taxas de exatidão elevadas são devidas à utilização da biblioteca *TextBlob* que já está pré-treinada antes da implementação. Os resultados obtidos confirmam que esta é uma solução eficaz para a análise de sentimentos, sugerindo que a utilização da biblioteca *TextBlob* é uma opção confiável.

5.3. Sistema de Recomendação

Como referido anteriormente, na Figura 20 é demostrada a *interface* da aplicação em que no primeiro retângulo são visíveis as escolhas selecionadas na *Dropbox*. Neste caso, o tópico escolhido é "votações" no *dataframe* de Joe Biden. No segundo retângulo é demonstrado o *tweet* recomendado pelo sistema, ou seja, no presente exemplo é devolvido o nome do utilizador (*90201: 'Pretepetals'*), o *tweet* (*90201: 'Looks like #trump is winning #Kentucky with over 70% votes counted. #ElectionDay #Election2020'*) e o sentimento classificado a este (*90201: 'Positivo'*).

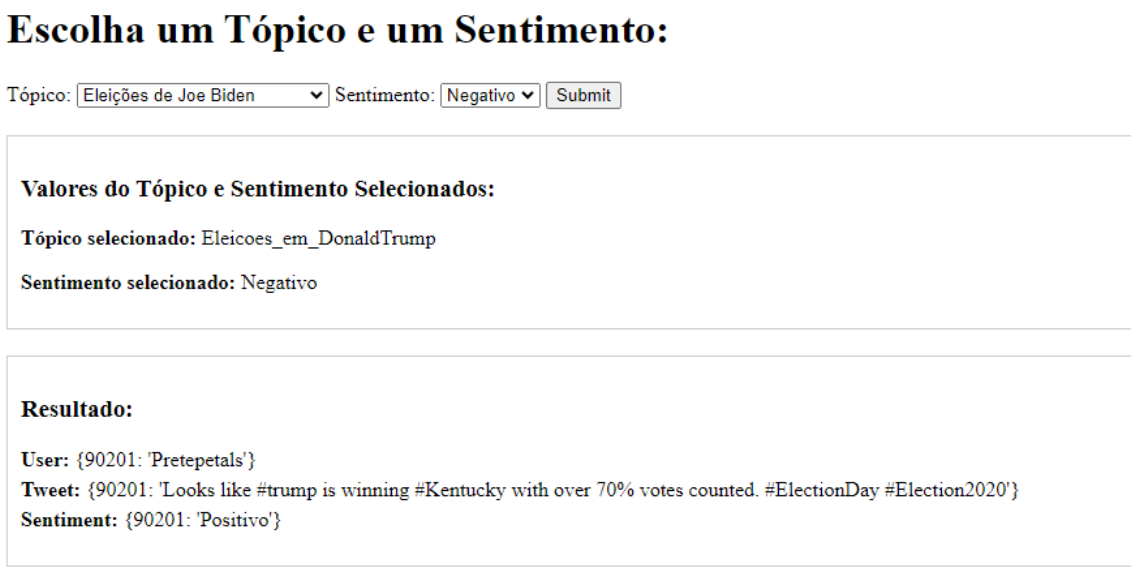


Figura 20 – Demonstração da aplicação Flask

6. Conclusões e Trabalho Futuro

O presente caso de estudo passou por diferentes etapas ao longo do seu desenvolvimento tendo estas sido planeadas e executadas com o intuito de cumprir os objetivos propostos. Esta abordagem estratégica permitiu alcançar com sucesso os objetivos estabelecidos, consolidando, assim, os resultados obtidos durante a pesquisa.

Diversas dificuldades foram encontradas no decorrer do projeto. Logo numa fase inicial foi encontrado um grande obstáculo, o facto de a rede social X ter bloqueado a sua API a terceiros devido às suas novas políticas de privacidade. Esta simples mudança teve um grande impacto no presente projeto, uma vez que, o desenvolvimento e toda a pesquisa previamente realizada para obter os dados necessários ficou sem efeito, acrescentando o facto de deixar de ser possível realizar todo o projeto de forma dinâmica, como inicialmente planeado, e o mais atual possível.

A exatidão obtida através do NB foi de aproximadamente 72% e 87% para as RN para os diferentes tamanhos de teste (15%, 25%, 35%) ambos podem ser classificados como valores altos, que se deve ao facto da biblioteca utilizada para classificar o sentimento já ser treinada previamente. Com os resultados obtidos podemos concluir que os *tweets* foram bem classificados obtendo uma percentagem elevada na utilização de RN.

Todo o código foi desenvolvido e executado num só computador (i7 da 8 geração e 16 GB de RAM) e existem tempos de execuções bastantes demorados o que complicou muitas vezes o desenvolvimento e validações dos resultados obtidos e/ou esperados, os tempos de execução foram medidos uma vez que pode existir a possibilidade de desenvolver uma aplicação com base no presente projeto. No entanto, estes tempos de execuções poderiam ser melhorados através de uma melhor otimização do código e com o recurso de máquinas mais recentes e sofisticadas.

No âmbito de trabalho futuro, existe a possibilidade de explorar modelos mais específicos, considerando a semântica do texto, em particular, há a necessidade de abordar recursos expressivos mais desafiadores, como a ironia.

Com o intuito a melhorar este trabalho, propõe-se explorar a viabilidade do desenvolvimento de uma aplicação que consolide todos os processos em um único fluxo contínuo. No cenário atual, os processos encontram-se fragmentados, exigindo

intervenções manuais e manipulação de arquivos CSV que podem facilmente ser corrompidos. Esta abordagem visa não apenas eliminar a necessidade de intervenções manuais, mas também otimizar o código, reduzindo os tempos extensos tempos de execução.

Referências Bibliográficas

A L, V., Shashikala S V, & Dr. Ravikumar G K. (2020). Information Analysis by Web Scraping Utilizing Python. *International Journal for Research in Applied Science and Engineering TechnologyTechnolog (IJRASET)*, *8*(2), 101–104. https://doi.org/10.22214/ijraset.2020.2016

Akbari, W. A., Tukino, T., Huda, B., & Muslih, M. (2023). Sentiment Analysis of Twitter User Opinions Related to Metaverse Technology Using Lexicon Based Method. *Sinkron : Jurnal Dan Penelitian Teknik Informatika*, *8*(1), Artigo 1. https://doi.org/10.33395/sinkron.v8i1.11992

Aljedaani, W., Rustam, F., Mkaouer, M. W., Ghallab, A., Rupapara, V., Washington, P. B., Lee, E., & Ashraf, I. (2022). Sentiment analysis on Twitter data integrating TextBlob and deep learning models: The case of US airline industry. *Knowledge-Based Systems*, *255*, 109780. https://doi.org/10.1016/j.knosys.2022.109780

Almeida, A., Carvalho, F., & Menino, F. (2020). *Introdução ao Machine Learning*. https://dataat.github.io/introducao-ao-machine-learning/

Arora, A., Bansal, S., Kandpal, C., Aswani, R., & Dwivedi, Y. (2019). *Measuring social media influencer index- insights from facebook, Twitter and Instagram | Elsevier Enhanced Reader*. *49*, 86–101. https://doi.org/10.1016/j.jretconser.2019.03.012

Assaf, D., Gutman, Y., Neuman, Y., Segal, G., Amit, S., Gefen-Halevi, S., Shilo, N., Epstein, A., Mor-Cohen, R., Biber, A., Rahav, G., Levy, I., & Tirosh, A. (2020). Utilization of machine-learning models to accurately predict the risk for critical COVID-19. *Internal and Emergency Medicine*, *15*(8), 1435–1443. https://doi.org/10.1007/s11739-020-02475-0

Aung, K. Z., & Myo, N. N. (2017). Sentiment analysis of students' comment using lexicon based approach. *2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS)*, 149–154. https://doi.org/10.1109/ICIS.2017.7959985

Bacelar, R. (2022, janeiro 21). *As 10 redes sociais mais usadas em Portugal em 2022—4gnews*. https://4gnews.pt/redes-sociais-mais-usadas/

Berrar, D. (2018). *Bayes' Theorem and Naive Bayes Classifier*. https://doi.org/10.1016/B978-0-12-809633-8.20473-1

Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77–84. https://doi.org/10.1145/2133806.2133826

Bose, R., Aithal, P. S., & Roy, S. (2021). Survey of Twitter Viewpoint on Application of Drugs by VADER Sentiment Analysis among Distinct Countries. *International Journal of Management, Technology, and Social Sciences*, 110–127. https://doi.org/10.47992/IJMTS.2581.6012.0132

Brownlee, J. (2016, março 20). Overfitting and Underfitting With Machine Learning Algorithms. *MachineLearningMastery.Com*. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Bruns, A., & Burges, J. (2015). Twitter hashtags from ad hoc to calculated publics. *N. Rambukkana, Ed. Hashtag Publics: The Power and Politics of Discursive Networks. New York: Peter Lang*, 13–28.

Cambria, E., & White, B. (2014). Jumping NLP Curves: A Review of Natural Language Processing Research [Review Article]. *IEEE Computational Intelligence Magazine*, *9*(2), 48–57. https://doi.org/10.1109/MCI.2014.2307227

Campbell, J. C., Hindle, A., & Stroulia, E. (2015). Chapter 6 - Latent Dirichlet Allocation: Extracting Topics from Software Engineering Data. Em C. Bird, T. Menzies, & T. Zimmermann (Eds.), *The Art and Science of Analyzing Software Data* (pp. 139–159). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-411519-4.00006-9

Carpenter, J. P., & Krutka, D. G. (2014). How and Why Educators Use Twitter: A Survey of the Field. *Journal of Research on Technology in Education*, *46*(4), 414–434. https://doi.org/10.1080/15391523.2014.925701

Carpenter, J., Tani, T., Morrison, S., & Keane, J. (2022). Exploring the landscape of educator professional activity on Twitter: An analysis of 16 education-related Twitter hashtags. *Professional Development in Education*, *48*(5), 784–805. https://doi.org/10.1080/19415257.2020.1752287

Chen, K., Duan, Z., & Yang, S. (2022). Twitter as research data: Tools, costs, skill sets, and lessons learned. *Politics and the Life Sciences*, *41*(1), 114–130. https://doi.org/10.1017/pls.2021.19

Das, D., Sahoo, L., & Datta, S. (2017). A Survey on Recommendation System. *International Journal of Computer Applications*, *160*, 6–10. https://doi.org/10.5120/ijca2017913081

Deitel, P., & Deitel, H. (2021). *Intro to Python® for Computer Science and Data Science*. Pearson Education Limited. https://www.pearson.com/en-us/subject-catalog/p/intro-to-python-for-computer-science-and-data-science-learning-to-program-with-ai-big-data-and-the-cloud/P200000003444/9780137505494

Deng, X., Liu, Q., Deng, Y., & Mahadevan, S. (2016). *An improved method to construct basic probability assignment based on the confusion matrix for classification problem—ScienceDirect*. *340–341*, Pages 250-261. https://doi.org/10.1016/j.ins.2016.01.033

Desai, M., & Mehta, M. A. (2016). Techniques for sentiment analysis of Twitter data: A comprehensive survey. *2016 International Conference on Computing, Communication and Automation (ICCCA)*, 149–154. https://doi.org/10.1109/CCAA.2016.7813707

F. Posada–Quintero, H. F., Kong, Y., & H. Chon, K. (2021). Objective pain stimulation intensity and pain sensation assessment using machine learning classification and regression based on electrodermal activity. *American Physiological Society*. https://doi.org/10.1152/ajpregu.00094.2021

Grinberg, M. (2018). *Flask Web Development: Developing Web Applications with Python*. O'Reilly Media, Inc.

Gujjar, P., & Prasanna, K. H. R. (2021). Sentiment Analysis:Textblob For Decision Making. *International Journal of Scientific Research & Engineering Trend*.

Gupta, B., Negi, M., Vishwakarma, K., Rawat, G., & Badhani, P. (2017). Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python. *International Journal of Computer Applications*, *165*(9), 29–34. https://doi.org/10.5120/ijca2017914022

H. Manguri, K., N. Ramadhan, R., & R. Mohammed Amin, P. (2020). Twitter Sentiment Analysis on Worldwide COVID-19 Outbreaks. *Kurdistan Journal of Applied Research*, 54–65. https://doi.org/10.24017/covid.8

Hofmann, T. (2001). Unsupervised Learning by Probabilistic Latent Semantic Analysis. *Machine Learning*, *42*(1), 177–196. https://doi.org/10.1023/A:1007617005950

HUI, M. (2020, outubro). *US Election 2020 Tweets*. https://www.kaggle.com/datasets/manchunhui/us-election-2020-tweets

IBM. (2023a, outubro). *O que são Redes Neurais? | IBM*. https://www.ibm.com/br-pt/topics/neural-networks

IBM. (2023b, outubro). *What are Naive Bayes classifiers? | IBM*. https://www.ibm.com/topics/naive-bayes

Jelodar, H., Wang, Y., Yuan, C., Feng, X., Jiang, X., Li, Y., & Zhao, L. (2019). Latent Dirichlet allocation (LDA) and topic modeling: Models, applications, a survey. *Multimedia Tools and Applications*, *78*(11), 15169–15211. https://doi.org/10.1007/s11042-018-6894-4

Jerónimo, M. (2020). *Sistemas de Recumendação Para conteudos e Aplicações Web* (https://repositorio.ipv.pt/). https://repositorio.ipv.pt/browse?type=author&value=Jer%C3%B3nimo%2C+Margarida+Isabel+de+Oliveira

Kim, S.-W., & Gil, J.-M. (2019). Research paper classification systems based on TF-IDF and LDA schemes. *Human-Centric Computing and Information Sciences*, *9*(1), 30. https://doi.org/10.1186/s13673-019-0192-7

Krotov, V., Johnson, L., & Silva, L. (2020). Tutorial: Legality and Ethics of Web Scraping. *Communications of the Association for Information Systems*, *47*. https://doi.org/10.17705/1CAIS.04724

Krotov, V., & Silva, L. (2018). Legality and Ethics of Web Scraping. *Emergent Research Forum*, *47*, 539–563. https://www.researchgate.net/publication/324907302

Krouska, A., Troussas, C., & Virvou, M. (2016). The effect of preprocessing techniques on Twitter sentiment analysis. *2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–5. https://doi.org/10.1109/IISA.2016.7785373

Kula, S., Choraś, M., Kozik, R., Ksieniewicz, P., & Woźniak, M. (2020). Sentiment Analysis for Fake News Detection by Means of Neural Networks. Em V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, & J. Teixeira (Eds.), *Computational Science – ICCS 2020* (pp. 653–666). Springer International Publishing. https://doi.org/10.1007/978-3-030-50423-6_49

Kumar, N. (2019, dezembro 18). Deep Learning: Feedforward Neural Networks Explained. *HackerNoon.Com*. https://medium.com/hackernoon/deep-learning-feedforward-neural-networks-explained-c34ae3f084f1

Kumar, R. (2020). Fake News Detection using Passive Aggressive and TF-IDF Vectorizer. *International Research Journal of Engineering and Technology*, *07*(12). https://doi.org/2395-0056

Kywe, S. M., Lim, E.-P., & Zhu, F. (2012). A Survey of Recommender Systems in Twitter. Em K. Aberer, A. Flache, W. Jager, L. Liu, J. Tang, & C. Guéret (Eds.), *Social Informatics* (Vol. 7710, pp. 420–433). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35386-4_31

Landauer, T. K., Foltz, P. W., & Laham, D. (1998). An introduction to latent semantic analysis. *Discourse Processes*, *25*(2–3), 259–284. https://doi.org/10.1080/01638539809545028

Lantz-Andersson, A., Hillman, T., Lundin, M., & Peterson, L. (2017). Sharing repertoires in a teacher professional Facebook group. *Learning Culture and Social Interaction*, 13. https://doi.org/10.1016/j.lcsi.2017.07.001

Li, L., Zhimin, N., Mei, S., & D. Griffiths, M. (2022). A network analysis approach to the relationship between fear of missing out (FoMO), smartphone addiction, and social

networking site use among a sample of Chinese university students |. *Computers in Human Behavior*, *128*. https://doi.org/10.1016/j.chb.2021.107086

Lison, P. (2012, outubro 3). *An introduction to machine learning*. http://home.nr.no/~plison/pdfs/talks/machinelearning.pdf

Liu, B. (2012). Sentiment Analysis and Opinion Mining. *WIREs Data Mining and Knowledge Discovery*, *8*(4). https://doi.org/10.1002/widm.1253

Liu, C., Sheng, Y., Wei, Z., & Yang, Y.-Q. (2018). Research of Text Classification Based on Improved TF-IDF Algorithm. *2018 IEEE International Conference of Intelligent Robotic and Control Engineering (IRCE)*, 218–222. https://doi.org/10.1109/IRCE.2018.8492945

Machorro-Cano, I., Alor-Hernández, G., Paredes-Valverde, M. A., Rodríguez-Mazahua, L., Sánchez-Cervantes, J. L., & Olmedo-Aguirre, J. O. (2020). HEMS-IoT: A Big Data and Machine Learning-Based Smart Home System for Energy Saving. *Energies*, *13*(5), 1097. https://doi.org/10.3390/en13051097

Mahesh, B. (2018). *Machine Learning Algorithms—A Review*. *9*(1), 7.

Mandloi, L., & Patel, R. (2020). Twitter Sentiments Analysis Using Machine Learninig Methods. *2020 International Conference for Emerging Technology (INCET)*, 1–5. https://doi.org/10.1109/INCET49848.2020.9154183

Martin, M. (2023, março 13). *29 Twitter Stats That Matter to Marketers in 2023*. Social Media Marketing & Management Dashboard. https://blog.hootsuite.com/twitter-statistics/

Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal*, *5*(4), 1093–1113. https://doi.org/10.1016/j.asej.2014.04.011

Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., & Yu, B. (2019). Interpretable machine learning: Definitions, methods, and applications. *Proceedings of the National Academy of Sciences*, *116*(44), 22071–22080. https://doi.org/10.1073/pnas.1900654116

NACDL - Computer Fraud and Abuse Act (CFAA). (2022, novembro). NACDL - National Association of Criminal Defense Lawyers. https://www.nacdl.org/Landing/ComputerFraudandAbuseAct

Narayan, S. (1997). The generalized sigmoid activation function: Competitive supervised learning. *Information Sciences*, *99*(1–2), 69–82. https://doi.org/10.1016/S0020-0255(96)00200-9

Nausheen, F., & Begum, S. H. (2018). Sentiment analysis to predict election results using Python. *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, 1259–1262. https://doi.org/10.1109/ICISC.2018.8399007

Negara, E. S., Triadi, D., & Andryani, R. (2019). Topic Modelling Twitter Data with Latent Dirichlet Allocation Method. *2019 International Conference on Electrical Engineering and Computer Science (ICECOS)*, 386–390. https://doi.org/10.1109/ICECOS47637.2019.8984523

Paliwal, M., & Kumar, U. A. (2009). Neural networks and statistical techniques: A review of applications. *Expert Systems with Applications*, *36*(1), 2–17. https://doi.org/10.1016/j.eswa.2007.10.005

Paliwal, S., Khatri, S. K., & Sharma, M. (2018). Sentiment Analysis and Prediction Using Neural Networks. *2018 International Conference on Inventive Research in Computing Applications*. https://doi.org/10.1109/ICIRCA.2018.8597358

Peng, H., Wang, J., Peréz Jimenéz, M., & Riscos-Núñez, A. (2015). *An unsupervised learning algorithm for membrane computing | Elsevier Enhanced Reader*. *3.4*, 80–91. https://doi.org/10.1016/j.ins.2015.01.019

Pradha, S., Halgamuge, M. N., & Tran Quoc Vinh, N. (2019). Effective Text Data Preprocessing Technique for Sentiment Analysis in Social Media Data. *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, 1–8. https://doi.org/10.1109/KSE.2019.8919368

Przybylski, A. K., Murayama, K., DeHaan, C. R., & Gladwell, V. (2013). *Motivational, emotional, and behavioral correlates of fear of missing out*. *29*. https://www.sciencedirect.com/science/article/pii/S0747563213000800. https://doi.org/10.1016/j.chb.2013.02.014

Putri, I. R., & Kusumaningrum, R. (2017). Latent Dirichlet Allocation (LDA) for Sentiment Analysis Toward Tourism Review in Indonesia. *Journal of Physics: Conference Series*, *801*(1), 012073. https://doi.org/10.1088/1742-6596/801/1/012073

Qaiser, S., & Ali, R. (2018). Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents. *International Journal of Computer Applications*, *181*. https://doi.org/10.5120/ijca2018917395

Roan, P. (2020). *TF-IDF [Tutorial]*. https://kaggle.com/code/paulrohan2020/tf-idf-tutorial

Sarkar, D. (2019). *Text Analytics with Python: A Practitioner's Guide to Natural Language Processing*. Apress. https://doi.org/10.1007/978-1-4842-4354-1

Sathya, R., & Abraham, A. (2013). Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification. *International Journal of Advanced Research in Artificial Intelligence*, *2*(2). https://doi.org/10.14569/IJARAI.2013.020206

Schofield, A., Magnusson, M., & Mimno, D. (2017). Pulling Out the Stops: Rethinking Stopword Removal for Topic Models. *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, 432–436. https://aclanthology.org/E17-2069

Schroeder, S., Curcio, R., & Lundgren, L. (2019). Expanding the Learning Network: How Teachers Use Pinterest. *Journal of Research on Technology in Education*, *51*(2), 166–186. https://doi.org/10.1080/15391523.2019.1573354

Shepherd, J. (2023, janeiro 3). *22 Essential Twitter Statistics You Need to Know in 2023*. The Social Shepherd. https://thesocialshepherd.com/blog/twitter-statistics

SNS. (2020, junho 30). *Redes Sociais: Usa mas não abuses!* [SNS - Serviço Nacional de Saúde]. SNS. https://www.inem.pt/2020/06/30/redes-sociais-usa-mas-nao-abuses/

Stecanella, B. (2019, maio 10). *Understanding TF-ID: A Simple Introduction*. MonkeyLearn Blog. https://monkeylearn.com/blog/what-is-tf-idf/

Surden, H. (2014). Machine Learning and Law. Em *Legal Informatics* (1.ª ed., pp. 94–98). Cambridge University Press. https://doi.org/10.1017/9781316529683.010

Sutton, R. S., & Barto, A. G. (2015). *Reinforcement Learning: An Introduction*. 352.

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, *39*(1), 43–62. https://doi.org/10.1016/S0169-7439(97)00061-0

Tf.keras.preprocessing.text.Tokenizer | TensorFlow v2.14.0. (2023, setembro 27). TensorFlow. https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/text/Tokenizer

TheFork—Reserve nos melhores restaurantes da Europa. (2023). https://www.thefork.pt/

Thomas, D. M., & Mathur, S. (2019). Data Analysis by Web Scraping using Python. *2019 3rd International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 450–454. https://doi.org/10.1109/ICECA.2019.8822022

Tripadvisor: Mais de mil milhões de avaliações e contribuições sobre hotéis, atrações, restaurantes e muito mais. (2019, novembro). Tripadvisor. https://www.tripadvisor.pt/

Tsytsarau, M., & Palpanas, T. (2012). Survey on mining subjective data on the web. *Data Mining and Knowledge Discovery*, *24*(3), 478–514. https://doi.org/10.1007/s10618-011-0238-6

Tyagi, P., & Tripathi, D. R. C. (2019). A Review towards the Sentiment Analysis Techniques for the Analysis of Twitter Data. *2nd International Conference on Advance Computing and Software Engineering*, 5. http://dx.doi.org/10.2139/ssrn.3349569

Warnik, B. R., Bitters, T. A., & Kim, S. H. (2016). *Social Media Use and Teacher Ethics* (epx.sagepub.com). *Vol. 30(5) 771 –795*, 25. https://doi.org/10.1177/0895904814552895

Wilson, S. L., & Wiysonge, C. (2020). Social media and vaccine hesitancy. *BMJ Global Health*, *5*(10), e004206. https://doi.org/10.1136/bmjgh-2020-004206

Wright, S. (2020, maio 11). Trespass to goods/chattels. *Gibbs Wright Litigation Lawyers*. https://gibbswrightlawyers.com.au/publications/trespass-goods-chattels

Yilmaz, B. (2022, dezembro 30). *Top 4 ML Applications of Sentiment Analysis in 2023*. https://research.aimultiple.com/sentiment-analysis-machine-learning/

Yu, S. J. (2012). *The dynamic competitive recommendation algorithm in social network services* (https://www.sciencedirect.com/science/article/pii/S0020025511005718). *187*, 1–14. https://doi.org/10.1016/j.ins.2011.10.020

Zhang, H. (2004). The Optimality of Naive Bayes. *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004*, *2*, 6.

Zhang, H., Gan, W., & Jiang, B. (2014). Machine Learning and Lexicon Based Methods for Sentiment Classification: A Survey. *2014 11th Web Information System and Application Conference*, 262–265. https://doi.org/10.1109/WISA.2014.55

Apêndices

Apêndice A

Código Desenvolvido para Recolha de *Tweets* com *Snscrape*

O código apresentado na Tabela 13 foi o código desenvolvido e utilizado na fase inicial do projeto para recolher os *tweets.*

Tabela 13 - Recolha de Tweets

```
import snscrape.modules.twitter as sntwitter
import pandas as pd

query ='(@realDonaldTrump) (filter:safe OR filter:unsafe)'
tweets = []
limit = 500

for tweet in sntwitter.TwitterSearchScraper(query).get_items():

    # print(vars(tweet))
    # break
    if len(tweets) == limit:
        break
    else:
        tweets.append([tweet.date, tweet.user.username,
tweet.content])

df_Tweet = pd.DataFrame (tweets,
columns=['Data','Utilizador','Tweet'])
df_Tweet.to_csv('Tweets.csv',encoding='utf-8')
```

Apêndice B

Código Utilizado para Tratamento dos dados

O código demostrado na Tabela 14 foi o código utilizado para fazer a *limpeza* do *tweet* de modo a utilizar posteriormente para determinar o sentimento.

Tabela 14 - Tratamento dos Tweets

```
import pandas as pd
import numpy as np
from datetime import datetime
```

```
# Obtém a data e hora atual
inicio = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S"
print(inicio.strftime("%Y-%m-%d %H:%M:%S"))


# df_Tweet = pd.read_csv('Dados\Tweets.csv')
df_Tweet = pd.read_csv('Dataset_Backup\hashtag_donaldtrump.csv', encoding='utf-8', delimiter=';')
# df_Tweet = pd.read_csv('Dataset_Backup\hashtag_joebiden.csv', encoding='utf-8', delimiter=';')
#-------------------------PREPARAR O TEXTO--------------------------------------------
import nltk
#Segmentação(identificar padrões onde começam e acabam frases) de dados frases e tokenização (dividar uma frase em frases mais pequenas)
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
nltk.download('omw-1.4')


#Remover caracteres especiais, números e transformar todo o texto em minúsculo.
import re # modulo para manipular strings


def clean_tweet(Tweet):
    if isinstance(Tweet, str):
        Tweet = Tweet.lower() #passar tudo para letras minusculas
        Tweet = re.sub(r'(https?:\/\/(?:www\.|(?!www))[^\s]+\.[^\s]+)','', Tweet) #remover links
        Tweet = re.sub(r'@\w+', '', Tweet) # remover outros utilizadores identificados
        Tweet = re.sub(r'[^a-z\s]', '', Tweet) # remover caracteres especiais
        Tweet = re.sub(r'\d+', '', Tweet) #remover numeros


    return Tweet


df_Tweet['Clean'] = df_Tweet['Tweet'].apply(clean_tweet)
#print(df_Tweet)
```

```
#Dividir o texto em palavras
from nltk.tokenize import word_tokenize
#Apagar nulos e passar a string para poder tokenizar as palavras
# df_Tweet = df_Tweet.dropna(subset=['Clean'])
# print(df_Tweet['Clean'].dtype)

# df_Tweet = df_Tweet[df_Tweet['Clean'].apply(lambda x: [item for
item in x if item])]
df_Tweet['Clean'] = df_Tweet['Clean'].astype(str)
df_Tweet['Clean'] = df_Tweet['Clean'].apply(word_tokenize)
df_Tweet.dropna(subset=['Clean'], inplace=True)
# print('weet')

#Palavras sem significado sentimento
from nltk.corpus import stopwords
stopwords = set(stopwords.words('english'))
# é necessário tratar as stopwords pois o df já tem as palavras em
minusculas e sem '
# pelo que é preciso fazer o mesmo às stopwords
new_stopwords = set()
for word in stopwords:
    new_word = re.sub(r'\W+', '', word)  # remove todos os caracteres
não alfanuméricos
    new_stopwords.add(new_word.lower())  # adiciona a nova palavra em
letras minúsculas ao novo conjunto de stop words

stopwords.update(new_stopwords)
# print(stopwords)
#não remover as seguintes
stopwords = [word for word in stopwords if word not in ['no', 'not']]
#Remover stopwords
def remove_stopwords(tokens):
    filtered_tokens = [token for token in tokens if token not in
stopwords]
    return filtered_tokens

df_Tweet['RemoveStopword'] =
df_Tweet['Clean'].apply(remove_stopwords)
#print(df_Tweet)

#lematização é técnica de processamento de linguagem natural usadas
```

```
para normalizar palavras em um texto, reduzindo-as a suas formas raiz
.
#from nltk.stem import SnowballStemmer
from nltk.stem import WordNetLemmatizer


lemmatizer = WordNetLemmatizer()


from nltk import pos_tag # Para obter a class gramatical de cada
palavra


def lemmatize_tokens(tokens):# primeiro elemento é a palavra e o
segundo é a tag
    lemmatized_tokens = [] #mapeia as tags para as categorias que o
lemmatizer reconhece
    for token, tag in pos_tag(tokens):
        tag = tag[0].lower() # converte a tag para o formato que o
lemmatizer espera
        tag = tag if tag in ['a', 'r', 'n', 'v'] else None #'a' para
adjetivo, 'r' para advérbio, 'n' para substantivo e 'v' para verbo
        if not tag: # se a tag não for uma das categorias acima, a
palavra não é lematizada
            lemmatized_tokens.append(token)
        else:
            lemma = WordNetLemmatizer().lemmatize(token, tag)
            lemmatized_tokens.append(lemma)
    return lemmatized_tokens


df_Tweet['lemmatized_tokens'] =
df_Tweet['RemoveStopword'].apply(lemmatize_tokens)


df_Tweet.to_csv('Tweets_CLEAN_DonaldTrump.csv',encoding='utf-8')
# df_Tweet.to_csv('Tweets_CLEAN_JoeBiden.csv',encoding='utf-8')
# df_Tweet.to_csv('Tweets_CLEAN.csv',encoding='utf-8')


# Obtém a nova data e hora atual
fim = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S" (até os segundos)
print(fim.strftime("%Y-%m-%d %H:%M:%S"))
# Calcula o intervalo de tempo entre as duas chamadas
tempo_exec = fim - inicio
# Extrai os componentes de horas, minutos e segundos do intervalo de
tempo
horas = tempo_exec.seconds // 3600
```

```
minutos = (tempo_exec.seconds % 3600) // 60
segundos = tempo_exec.seconds % 60
# Imprime o intervalo de tempo no formato "demorou x horas x minutos
e x segundos"
print(f"Demorou {horas} horas, {minutos} minutos e {segundos}
segundos.")
```

Apêndice C

Código Utilizado para Cálculo do TF-IDF & Análise de Sentimento

O código exibido na Tabela 15 foi utilizado para determinar o valor do TF-IDF e determinar o sentimento associado a cada *tweet.*

Tabela 15 - TF-IDF & Análise de Sentimento

```
from datetime import datetime
print("DONALD TRUMP")
# Obtém a data e hora atual
inicio = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S"
print(inicio.strftime("%Y-%m-%d %H:%M:%S"))

from textblob import TextBlob
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np

df_Trump = pd.read_csv('Dados\Tweets_CLEAN_DonaldTrump.csv')

# cricao do vetor tfidf
vectorizer_Trump = TfidfVectorizer()
vectors_Trump =
vectorizer_Trump.fit_transform(df_Trump['lemmatized_tokens'])
feature_names_Trump = vectorizer_Trump.get_feature_names_out()

# Calculos dos valores
tfidf_scores_Trump = np.array(vectors_Trump.mean(axis=0)).flatten()

# 5 valores mais altos do tfidf
n_top_words = 5
top_indices_Trump = np.argsort(tfidf_scores_Trump)[::-
```

```
1][:n_top_words]


# Criacao do dicionário para armazenar os substes com as palavras
principais como chaves.
subsets_dict = {}


# Calculo do sentimento
def calculate_sentiment_score(text):
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity
    if sentiment > 0:
        return 'Positivo'
    elif sentiment < 0:
        return 'Negativo'
    else: # =0
        return 'Neutro'


# Criacao dos subsets para as  top 5 palavras
for index in top_indices_Trump:
    word = feature_names_Trump[index]
    subset =
df_Trump[df_Trump['lemmatized_tokens'].str.contains(word)]
    subsets_dict[word] = subset


for index in range(n_top_words):
    word = feature_names_Trump[top_indices_Trump[index]]
    subset =
df_Trump[df_Trump['lemmatized_tokens'].str.contains(word)]
    subsets_dict[word] = subset
    weight = tfidf_scores_Trump[top_indices_Trump[index]] * 1000
    print(f"Word: {word} -> {weight:.10f}")
    print(f"Lenght NEW_df_{word} -> {subset.shape[0]}")
    # print(subset[['User', 'lemmatized_tokens']])
    # print("\n")


# Criacao de dataframes para cada subsets e utilizando a palavra com
o maior peso como nome do dataframe
for word, subset in subsets_dict.items():
    dataframe_name = f"NEW_df_{word}"
    globals()[dataframe_name] = subset
    csv_filename =
f"TFIDF_SUBSET/{dataframe_name}_FROM_DONALD_TRUMP.csv"
```

```
    # Calculate sentiment score for each row in the dataframe
    subset['Sentiment'] =
subset['lemmatized_tokens'].apply(calculate_sentiment_score)
    # Save the dataframe with the new column to a CSV file
    subset.to_csv(csv_filename, index=False)
    print('Ficheiro criado')


# Obtém a nova data e hora atual
fim = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S" (até os segundos)
print(fim.strftime("%Y-%m-%d %H:%M:%S"))
# Calcula o intervalo de tempo entre as duas chamadas
tempo_exec = fim - inicio
# Extrai os componentes de horas, minutos e segundos do intervalo de
tempo
horas = tempo_exec.seconds // 3600
minutos = (tempo_exec.seconds % 3600) // 60
segundos = tempo_exec.seconds % 60
# Imprime o intervalo de tempo no formato "demorou x horas x minutos
e x segundos"
print(f"Demorou {horas} horas, {minutos} minutos e {segundos}
segundos.")


# ----------------------------------------------------------------
print("-----------------------------------------------")
print("JOE BIDEN")
# ----------------------------------------- JOE BIDEN
from datetime import datetime
# Obtém a data e hora atual
inicio = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S"
print(inicio.strftime("%Y-%m-%d %H:%M:%S"))
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import numpy as np


df_Biden = pd.read_csv('Dados\Tweets_CLEAN_JoeBiden.csv')


# Criação do vetor
vectorizer_Biden = TfidfVectorizer()
vectors_Biden =
```

```
vectorizer_Biden.fit_transform(df_Biden['lemmatized_tokens'])
feature_names_Biden = vectorizer_Biden.get_feature_names_out()


# Calcula os valores do tfidf
tfidf_scores_Biden = np.array(vectors_Biden.mean(axis=0)).flatten()


# 5 valores mais altos do tfidf
n_top_words = 5
top_indices_Biden = np.argsort(tfidf_scores_Biden)[::-
1][:n_top_words]


# Criacao do dicionário para armazenar os substes com as palavras
principais como chaves.s
subsets_dict = {}
# Calculo do sentimento
def calculate_sentiment_score(text):
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity
    if sentiment > 0:
        return 'Positivo'
    elif sentiment < 0:
        return 'Negativo'
    else: # =0
        return 'Neutro'


# Criacao dos subsets para as  top 5 palavras
for index in top_indices_Biden:
    word = feature_names_Biden[index]
    subset =
df_Biden[df_Biden['lemmatized_tokens'].str.contains(word)]
    subsets_dict[word] = subset


# Printing the words and their corresponding subsets


for index in range(n_top_words):
    word = feature_names_Biden[top_indices_Biden[index]]
    subset =
df_Biden[df_Biden['lemmatized_tokens'].str.contains(word)]
    subsets_dict[word] = subset
    weight = tfidf_scores_Biden[top_indices_Biden[index]] * 1000
    print(f"Word: {word} -> {weight:.10f}")
```

```
    print(f"Lenght NEW_df_{word} -> {subset.shape[0]}")
    # print(subset[['User', 'lemmatized_tokens']])
    # print("\n")


# Criacao de dataframes para cada subsets e utilizando a palavra com
o maior peso como nome do dataframe
for word, subset in subsets_dict.items():
    dataframe_name = f"NEW_df_{word}"
    globals()[dataframe_name] = subset
    csv_filename =
f"TFIDF_SUBSET/{dataframe_name}_FROM_JOE_BIDEN.csv"
    # Calculate sentiment score for each row in the dataframe
    subset['Sentiment'] =
subset['lemmatized_tokens'].apply(calculate_sentiment_score)
    # Save the dataframe with the new column to a CSV file
    subset.to_csv(csv_filename, index=False)
    print('Ficheiro criado')


# Obtém a nova data e hora atual
fim = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S" (até os segundos)
print(fim.strftime("%Y-%m-%d %H:%M:%S"))
# Calcula o intervalo de tempo entre as duas chamadas
tempo_exec = fim - inicio
# Extrai os componentes de horas, minutos e segundos do intervalo de
tempo
horas = tempo_exec.seconds // 3600
minutos = (tempo_exec.seconds % 3600) // 60
segundos = tempo_exec.seconds % 60
# Imprime o intervalo de tempo no formato "demorou x horas x minutos
e x segundos"
print(f"Demorou {horas} horas, {minutos} minutos e {segundos}
segundos.")
```

Apêndice D

Código Utilizado para criar um Gráfico com Base nos Resultados da Classificação de Sentimento

O código exibido na Tabela 16 foi utilizado para ajudar na criação do gráfico com base na análise do Sentimento.

Tabela 16 - Matplot

```
import pandas as pd

df_Tweet =
pd.read_csv('TFIDF_SUBSET/NEW_df_trump_FROM_DONALD_TRUMP.csv')

import matplotlib.pyplot as plt

sentiments = df_Tweet['Sentiment'].value_counts()

plt.bar(sentiments.index, sentiments.values, color=['lightskyblue',
'lightgreen', 'darksalmon'])

plt.xlabel('Classificação')
plt.ylabel('Número de Tweet')

# Adiciona as linhas horizontais e os valores no topo de cada barra
for i, count in enumerate(sentiments.values):
    plt.text(i, count, str(count), ha='center', va='bottom')
plt.title('Análise de Sentimento trump_FROM_DONALD_TRUMP')
plt.show()

# Calcular a porcentagem de ocorrência de cada valor
percentagem_sentimentos = sentiments / len(df_Tweet) * 100

# Imprimir os resultados
print("Percentagem de ocorrência de cada valor na coluna
'Sentimento':")
print(percentagem_sentimentos)
```

Apêndice E

Código Utilizado para Calcular a Exatidão da Classificação do Sentimento

O código mostrado na Tabela 17 é o código utilizado para determinar a exatidão do Sentimento classificado com o algoritmo NB.

Tabela 17 - Cálculo da Exatidão do Algoritmo NB

```
from datetime import datetime
inicio = datetime.now()
```

```
print(inicio.strftime("%Y-%m-%d %H:%M:%S"))

# bibliotecas necessárias
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('TFIDF_SUBSET/NEW_df_trump_FROM_DONALD_TRUMP.csv')
# df = df.head(15)

contagem_sentimentos = df['Sentiment'].value_counts()
print(contagem_sentimentos)
min_registros = contagem_sentimentos.min()

df = df.groupby('Sentiment').apply(lambda x: x.sample(min_registros))
df.reset_index(drop=True, inplace=True)

print ('---Mesmo número de Registos---')
contagem_sentimentos = df['Sentiment'].value_counts()
print(contagem_sentimentos)

# Divir os dados em recursos (X) e rótulos (y)
X = df['lemmatized_tokens']
y = df['Sentiment']

# CountVectorizer para vetorizar os dados de texto
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(X)

# Use LabelEncoder para codificar os rótulos
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)

# Divisão dos dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15, random_state=0)
# X_train, X_test, y_train, y_test = train_test_split(X, y,
```

```
test_size=0.25, random_state=0)
# X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.35, random_state=0)


# modelo de classificação (Multinomial Naive Bayes)
modelo = MultinomialNB()
# Treino o modelo
modelo.fit(X_train, y_train)


# Previsões no conjunto de teste
previsoes = modelo.predict(X_test)
# Desempenho do modelo
precisao = accuracy_score(y_test, previsoes)
matriz_confusao = confusion_matrix(y_test, previsoes)


print(f' exatidão do modelo: {precisao:.5f}')
print('Matriz de Confusão:')
print(matriz_confusao)


fim = datetime.now()
print(fim.strftime("%Y-%m-%d %H:%M:%S"))
tempo_exec = fim - inicio
horas = tempo_exec.seconds // 3600
minutos = (tempo_exec.seconds % 3600) // 60
segundos = tempo_exec.seconds % 60
print(f"Demorou {horas} horas, {minutos} minutos e {segundos}
segundos.")
```

O código mostrado na Tabela 18 é o código utilizado para determinar a exatidão do Sentimento classificado com o algoritmo RN.

Tabela 18 - Cálculo da Exatidão do Algoritmo RN

```
from datetime import datetime
# Get the start date and time
inicio = datetime.now()
print(inicio.strftime("%Y-%m-%d %H:%M:%S"))


#  bibliotecas necessárias
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow import keras
from tensorflow.keras.layers import Dense
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.feature_extraction.text import CountVectorizer
#
df =
pd.read_csv('TFIDF_SUBSET/NEW_df_trump_FROM_DONALD_TRUMP.csv')


contagem_sentimentos = df['Sentiment'].value_counts()
print(contagem_sentimentos)
min_registros = contagem_sentimentos.min()
df = df.groupby('Sentiment').apply(lambda x:
x.sample(min_registros - 152846))
df.reset_index(drop=True, inplace=True)


print ('---Mesmo número de Registos---')
contagem_sentimentos = df['Sentiment'].value_counts()
print(contagem_sentimentos)
# df = df.head(10000)


# dados em  (X) e rótulos (y)
X = df['lemmatized_tokens']
y = df['Sentiment']
#LabelEncoder para passar texto a número
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)


# Divisão dos dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.15, random_state=0)
# X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=0)
# X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.35, random_state=0)


# Tokenizer para vetorizar os dados de texto em número
tokenizer = keras.preprocessing.text.Tokenizer()
```

```
tokenizer.fit_on_texts(X_train)
X_train = tokenizer.texts_to_matrix(X_train, mode='count')
X_test = tokenizer.texts_to_matrix(X_test, mode='count')


#Modelo de rede neural feedforward
model = keras.Sequential()
model.add(Dense(128, input_shape=(X_train.shape[1],),
activation='relu'))
model.add(Dense(64, activation='relu'))
model.add(Dense(len(label_encoder.classes_),
activation='softmax'))


# o modelo
model.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])


# Treino do modelo
model.fit(X_train, y_train, epochs=10, batch_size=32,
validation_split=0.1)


# previsões no conjunto de teste
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)  # Obtém as classes
previstas


# de volta para as etiquetas originais
y_pred_labels = label_encoder.inverse_transform(y_pred_classes)


print ("Número de classes:", len(label_encoder.classes_))


# Calcular a matriz de confusão
confusion =
confusion_matrix(label_encoder.inverse_transform(y_test),
y_pred_labels)
print("Matriz de Confusão:")
print(confusion)


# Calcular a  exatidão
accuracy = accuracy_score(label_encoder.inverse_transform(y_test),
y_pred_labels)
print("Exatidão:", accuracy)
```

```
fim = datetime.now()
print(fim.strftime("%Y-%m-d %H:%M:%S"))
tempo_exec = fim - inicio
horas = tempo_exec.seconds // 3600
minutos = (tempo_exec.seconds % 3600) // 60
segundos = tempo_exec.seconds % 60
print(f"Demorou {horas} horas, {minutos} minutos e {segundos}
segundos.")
```

Apêndice F

Código Utilizado para Criar a Interface e fazer o Sistema de Recomendação

O código mostrado na Tabela 19 é o *main* utilizado para executar a aplicação Flask.

Tabela 19 - Main

```
from flask import Flask, render_template, request
from recomendar_tweets_interface import Choose_topic,
recommend_system  # Importe suas funções aqui


app = Flask(__name__)


@app.route('/', methods=['GET', 'POST'])
def index():
    result = None
    final = None
    if request.method == 'POST':
        select_topic = request.form.get('selectTopic')
        select_sentimento = request.form.get('selectSentimento')


        #função Choose_topic com o valor de selectTopic
        df = Choose_topic(select_topic)


        #função recommend_system com os valores de df e
selectSentimento
        result = {
            'Tópico selecionado': select_topic,
            'Sentimento selecionado': select_sentimento,
        }


        if select_sentimento == 'Neutro':
```

```
            # Se o sentimento for neutro, obtenha uma lista de
resultados
            final = recommend_system(df, select_sentimento)
        else:
            # Caso contrário, obtenha um único resultado
            final = [recommend_system(df, select_sentimento)]

    return render_template('index.html', result=result, final=final)

if __name__ == '__main__':
    app.run(debug=True)
```

Na Tabela 20 é demonstrado o código HTML e na Tabela 21 é exibido o CSS.

Tabela 20 - Index

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="{{ url_for('static', filename=
'css/style.css') }}">
    <title>FlaskBlog</title>
</head>

<body>
    <h1>Escolha um Tópico e um Sentimento:</h1>
    <form method="POST" action="/">
        <label for="selectTopic">Tópico:</label>
        <select name="selectTopic" id="selectTopic">
            <option value=""></option>
            <option value="Votar_JoeBiden">Votar de Joe
Biden</option>
            <option value="Eleicoes_JoeBiden">Eleições de Joe
Biden</option>
            <option value="Trump_em_JoeBiden">Trump em Joe
Biden</option>
            <option value="JoeBiden_em_DonaldTrump">Joe Biden em
Donald Trump</option>
            <option value="Eleicoes_em_DonaldTrump">Eleições em
Donald Trump</option>
            <!-- Add more options as needed -->
```

```
        </select>

        <label for="selectSentimento">Sentimento:</label>
        <select name="selectSentimento" id="selectSentimento">
            <option value=""></option>
            <option value="Positivo">Positivo</option>
            <option value="Negativo">Negativo</option>
            <option value="Neutro">Neutro</option>
            <!-- Add more options as needed -->
        </select>
        <input type="submit" value="Submit">
    </form>

    {% if result %}
    <div class="result-container">
        <h3>Valores do Tópico e Sentimento Selecionados:</h3>
        <div class="topic-sentiment-values">
            <p><strong>Tópico selecionado:</strong> {{
result['Tópico selecionado'] }}</p>
            <p><strong>Sentimento selecionado:</strong> {{
result['Sentimento selecionado'] }}</p>
        </div>
    </div>

    {% if final %}
    <div class="result-container">
        <h3>Resultado:</h3>
        <ul class="result-list">
            {% if final|length > 1 %}
            {% for neutral_result in final %}
            <div class="result-container2">
                {% for key, value in neutral_result.items() if key
not in ['Tópico selecionado', 'Sentimento
                selecionado'] %}
                <li><strong>{{ key }}:</strong> {{ value }}</li>
                {% endfor %}
                {% endfor %}
                {% else %}
                {% for key, value in final[0].items() %}
                <li><strong>{{ key }}:</strong> {{ value }}</li>
                {% endfor %}
```

```
                {% endif %}
            </div>
        </ul>
    </div>
    {% endif %}
    {% endif %}
</body>
</html>
```

Tabela 21- CSS

```
.result-container {
  border: 1px solid #ccc;
  padding: 10px;
  margin-top: 20px;
}
.result-list {
  list-style-type: none;
  padding: 0;
}
.result-list li {
  margin-bottom: 5px;
}
.result-container2 {
  margin-top: 20px;
}
```

A seguinte Tabela 22 ilustra o código desenvolvido de modo a realizar o sistema de Recomendação.

Tabela 22 - Sistema de Recomendação

```
from datetime import datetime
# Obtém start date e time
# Obtém a data e hora atual
inicio = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S"
print(inicio.strftime("%Y-%m-%d %H:%M:%S"))
import pandas as pd

def Choose_topic(topico):
    print(topico)
    if topico == 'Votar_JoeBiden':
```

```
        df =
pd.read_csv('TFIDF_SUBSET/NEW_df_vote_FROM_Joe_Biden.csv')
        print(df)
        return df
    if topico == 'Eleicoes_JoeBiden':
        df =
pd.read_csv('TFIDF_SUBSET/NEW_df_election_FROM_Joe_Biden.csv')
        return df
    if topico == 'Trump_em_JoeBiden':
        df =
pd.read_csv('TFIDF_SUBSET/NEW_df_trump_FROM_Joe_Biden.csv')
        return df
    if topico == 'JoeBiden_em_DonaldTrump':
        df =
pd.read_csv('TFIDF_SUBSET/NEW_df_trump_FROM_Joe_Biden.csv')
        return df
    if topico == 'Eleicoes_em_DonaldTrump':
        df =
pd.read_csv('TFIDF_SUBSET/NEW_df_election_FROM_DONALD_TRUMP.csv')
        return df


#df = pd.read_csv('TFIDF_SUBSET/NEW_df_trump_FROM_DONALD_TRUMP.csv')
# Definir a variável de controle (escolha do Utilizador)
#controlo = 'Negativo'  # Pode ser 'POSITIVO', 'NEGATIVO' ou 'NEUTRO'


def recommend_system(df, controlo):
    #print(controlo)
    #print(df)
    if controlo == 'Positivo':
        linha_aleatoria = df[df['Sentiment'] ==
'Negativo'].sample()[['User', 'Tweet','Sentiment']]
        return linha_aleatoria.to_dict()
    elif controlo == 'Negativo':
        linha_aleatoria = df[df['Sentiment'] ==
'Positivo'].sample()[['User', 'Tweet','Sentiment']]
        return linha_aleatoria.to_dict()
    elif controlo == 'Neutro':
        resultados_positivos = df[df['Sentiment'] ==
'Negativo'].sample()[['User', 'Tweet','Sentiment']]
        resultados_negativos = df[df['Sentiment'] ==
'Positivo'].sample()[['User', 'Tweet','Sentiment']]
        return resultados_positivos.to_dict(),
resultados_negativos.to_dict()
```

```
# Obtém a nova data e hora atual
fim = datetime.now()
# Imprime o sysdate no formato "%Y-%m-%d %H:%M:%S" (até os segundos)
print(fim.strftime("%Y-%m-%d %H:%M:%S"))
# Calcula o intervalo de tempo entre as duas chamadas
tempo_exec = fim - inicio
# Extrai os componentes de horas, minutos e segundos do intervalo de tempo
horas = tempo_exec.seconds // 3600
minutos = (tempo_exec.seconds % 3600) // 60
segundos = tempo_exec.seconds % 60
# Imprime o intervalo de tempo no formato "demorou x horas x minutos e x segundos"
print(f"Demorou {horas} horas, {minutos} minutos e {segundos} segundos.")
```