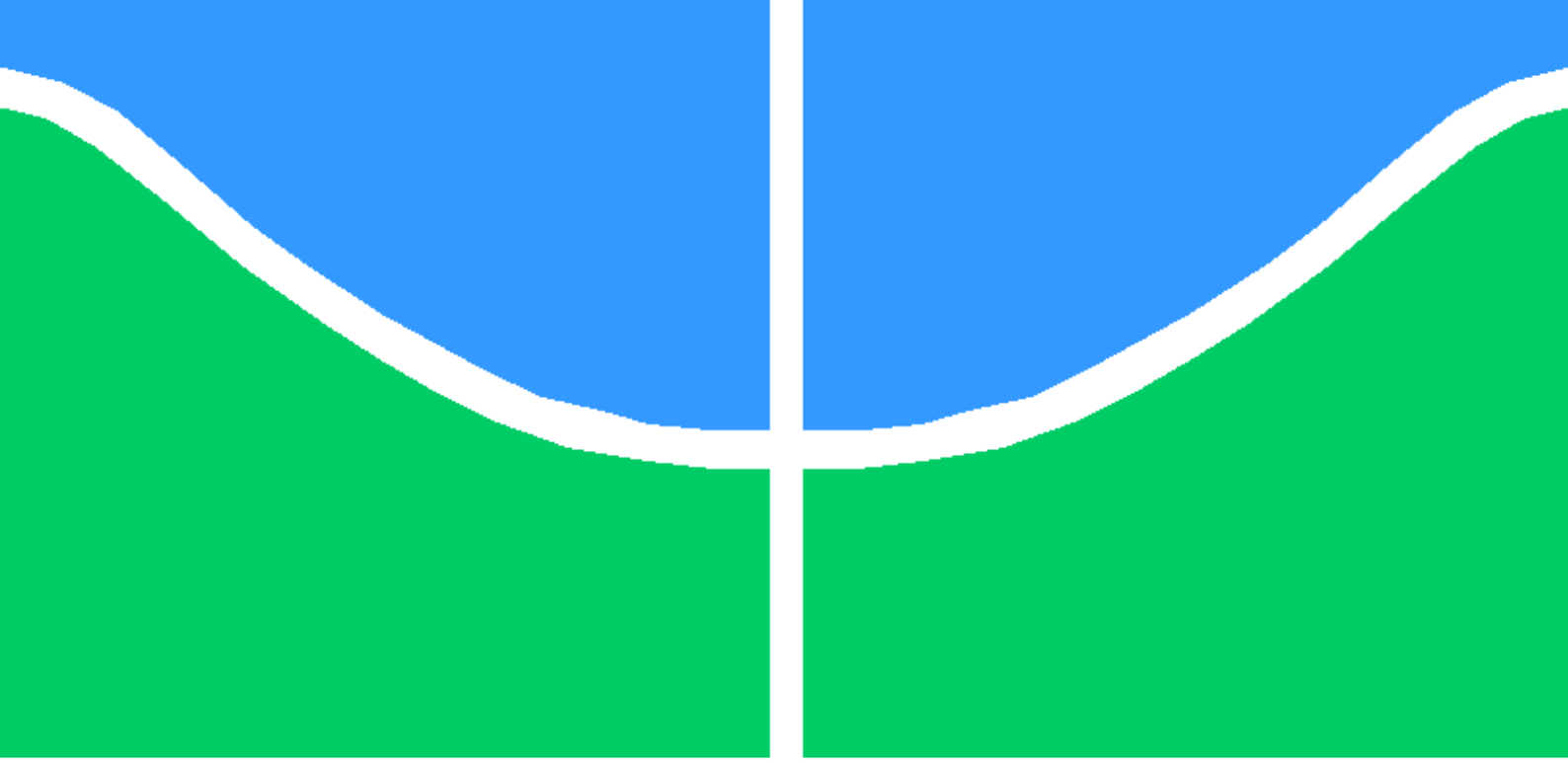

PROJETO FINAL DE GRADUAÇÃO

Análise de modelos de aprendizado de máquina para detecção de notícias falsas

João Lucas Vale Assis

Brasília, Maio de 2022

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASILIA
Faculdade de Tecnologia

PROJETO FINAL DE GRADUAÇÃO

Análise de modelos de aprendizado de máquina para detecção de notícias falsas

João Lucas Vale Assis

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheiro de Redes de Comunicação

Banca Examinadora

Prof. Mylène C.Q. Farias, ENE/UnB
Orientador

Prof. Daniel Guerreiro e Silva, ENE/UnB
Examinador Interno

Prof. Joao Luiz Carvalho, ENE/UnB
Examinador Interno

"Em tempos de engano universal, falar a verdade torna-se um ato revolucionário."

George Orwell

Agradecimentos

Por muitas vezes na nossa vida a visão que temos do nosso caminho é ofuscada. Ofuscada por pensamentos, medos, dúvidas ou pela falta de iluminação, alguma luz que nos mostre uma indicação de onde estamos e para onde queremos ir. Eu demorei um tempo para aprender que para onde queremos ir, é para os braços de quem nós somos no fundo. E o caminho até lá são todas essas situações que nos tiram da inércia e nos levam a ser um pouco mais do que somos hoje. A graduação na UnB definitivamente foi uma parte nesse longo caminho, uma longa curva, muitas vezes sinuosa que me aproximou de quem eu verdadeiramente sou. E também me aproximou de quem é luz na minha vida. A essas pessoas dedico esse trabalho e toda minha gratidão por ter partilhado da jornada. Aos meus pais João Marcos e Regina, tão verdadeiros e amorosos, por terem me dado a liberdade de acertar e de principalmente errar várias vezes, mas sempre compartilhando muito amor nesse processo. Aos meu irmãos Ana Clara e Marcos Vitor, verdadeiros companheiros e cúmplices do árduo trabalho de acordar cedo e ir estudar. A todos meus familiares, como a Tuta, que acreditam em mim e são presentes de alguma forma nos meus dias. À minha querida namorada e amiga Gabriela, que na reta final do trabalho partilhou do esforço comigo e se mostrou presente com muito carinho e amor. Aos meu queridos amigos, que desde a época da escola compartilham comigo momentos de crescimento. Igor Gustavo, Igor Mineiro, Gabriel Mendes, Lucas Alves, Thawan Guedes e João Gabriel, a amizade de vocês é muita preciosa para mim. Ao Gabriel Lins, meu querido amigo e hoje também colega de trabalho, que suportou junto a mim as tormentas e também alegrias, mas principalmente tormentas, da graduação e do trabalho. Sua amizade é muita preciosa pra mim. A todos amigos, não citados aqui, que fiz durante minha vida e compartilharam comigo uma risada, uma lágrima ou uma conversa. Aos excelentes professores da UnB e da minha vida profissional que já se iniciou.
À todos vocês que, do meu caminho, de alguma forma já fizeram ou ainda fazem parte.

João Lucas Vale Assis

RESUMO

Informação é um componente crucial hoje para se viver e se localizar na sociedade atual. Um mundo cada vez mais globalizado exige o acesso simplificado e ágil a notícias e acontecimentos em qualquer parte do planeta. Tais informações ajudam a moldar e formar opiniões que movem e influenciam diretamente as atitudes dos seres humanos. Hoje, de maneira idealizada, quem está sintonizado com a realidade exposta pelos grandes meios de comunicação se encontra a par de tomar decisões que vão de acordo com o momento atual. Desde o início dos anos 1990, uma quantidade sem precedentes de informações são divulgadas pelo mundo, propulsionadas pelas tecnologias criadas com o advento da Internet. Quantificando essa informação, em 2020, 64,2 *Zettabytes* de dados foram criados, capturados, copiados e consumidos globalmente, segundo previsões calculados pelo Statista [1]. Tamanha quantidade de informações levantam questões essenciais para a análise de quem as consome, como por exemplo: como saber que dada informação é verdadeira? É possível confiar na análise racional humana, alinhada a ferramentas que nos permitem comparar uma informação com fatos já registrados pelos jornais, artigos científicos, livros e muitos outros meios de registros confiáveis de informações sobre o mundo. Mas seria possível confiar que todo e qualquer ser humano tem tais meios de verificar e checar por si a veracidade de uma informação?

O presente trabalho possui o objetivo de analisar a possibilidade da utilização de aprendizado de máquina como ferramenta acessível aos usuários brasileiros de redes sociais para que possam filtrar e classificar notícias como verdadeiras ou falsas. Como uma vertente do estudo em inteligência artificial, o aprendizado de máquina parte do princípio de programar um computador com o objetivo de agilizar e otimizar uma dada tarefa automatizada, no nosso caso, a classificação de notícias. Notícias podem ser classificadas utilizando a lógica humana alinhada a conhecimentos históricos, mas esse tipo de abordagem se torna muito complexa para um algoritmo computacional realizar. Nesse caso, a análise realizada foi feita de forma a identificar padrões na escrita ou no viés das palavras utilizadas nas notícias incluídas no *dataset*. Esse é um tipo de abordagem que poderia processar volumes muito grandes de dados em pouco tempo e ajudar usuários a identificarem prováveis notícias falsas. Foi analisado o desempenho de diversos algoritmos já existentes de aprendizado de máquina e escolhido o modelo *PAC - Passive Agressive Classifier* como ênfase do estudo. Os motivos da escolha são: a sua baixa complexidade computacional e o fato de ter obtido as melhores métricas de performance (92% de acurácia). Utilizando de dados originários de bancos de dados de notícias de cunho jornalístico e também bancos de dados de boatos e notícias falsas compartilhadas em redes sociais como WhatsApp e Facebook, analisaremos o panorama Brasileiro de compartilhamento de notícias online e a possibilidade de mitigação do impacto causado pela

[1] https://www.statista.com/statistics/871513/worldwide-data-created/

desinformação fazendo uso de inteligência artificial.

ABSTRACT

Information is a crucial component today for living and locating in today's society. An increasingly globalized world requires simplified and agile access to news and events anywhere on the planet. Such information helps to shape and form opinions that move and directly influence the attitudes of human beings. Today, in an idealized way, those who are in tune with the reality exposed by the mass media are aware of making decisions that are in line with the current moment. Since the early 1990s, an unprecedented amount of information has been disseminated around the world, driven by technologies created with the advent of the Internet. Quantifying this information, in 2020, 64.2 Zettabytes of data were created, captured, copied and consumed globally, according to forecasts calculated by Statista [2]. Such amount of information raises essential questions for the analysis of who consumes it, such as: how to know that given information is true? It is possible to rely on human rational analysis, aligned with tools that allow us to compare information with facts already recorded in newspapers, scientific articles, books and many other means of reliable records of information about the world. But would it be possible to trust that each and every human being has such means of verifying and checking for himself the veracity of information?

The present work has the objective of analyzing the possibility of using machine learning as a tool accessible to brazilian users of social networks so that they can filter and classify news as true or false. As an aspect of the study of artificial intelligence, machine learning starts from the principle of programming a computer with the objective of speeding up and optimizing a given automated task, in our case, the classification of news. News can be classified using human logic aligned with historical knowledge, but this type of approach becomes too complex for a computational algorithm to perform. Thus, the analysis performed was carried out in order to identify patterns in the writing or in the bias of the words used in the news included in the dataset. This is a type of approach that could process very large volumes of data in a short time and help users to identify likely fake news. The performance of several existing machine learning algorithms was analyzed and the PAC - Passive Aggressive Classifier model was chosen as the emphasis of the study. The reasons for choosing it are: its low computational complexity and the fact that it obtained the best performance metrics (92% accuracy). Using data from journalistic news databases and also databases of rumors and fake news shared on social networks such as WhatsApp and Facebook, we will analyze the Brazilian panorama of online news sharing and the possibility of mitigating the impact caused by disinformation by making use of artificial intelligence.

[2] https://www.statista.com/statistics/871513/worldwide-data-created/

SUMÁRIO

LISTA DE FIGURAS

LISTA DE ABREVIATURAS

Acrônimos

PA	*Passive Agressive Classifier*
t-SNE	*t-distributed Stochastic Neighbor Embedding*
PLN	Processamento de Linguagem Natural
DL	*Deep Learning*
ML	*Machine Learning*
TI	Tecnologia da Informação
UnB	Universidade de Brasília

1

Introdução

Nunca foi tão fácil se informar como no presente momento. O advento da Internet e o desenvolvimento da nanotecnologia permitiram que seres humanos, utilizando-se de seus dispositivos eletrônicos conectados a Internet, como celulares e computadores, se mantenham sempre a um toque ou clique de distância de informações relevantes sobre o seu país e o mundo. E dentro das inúmeras possibilidades da rede mundial de computadores, existem hoje uma das principais fontes de informações: redes sociais. Redes sociais são utilizadas por mais de um terço da população mundial (GALLAGHER, 2017). Tais portais permitem a conexão e proximidade social entre pessoas que possuam algum tipo de relacionamento pré-estabelecido cara-a-cara ou que muitas vezes nunca tenham se visto fora de uma tela. Além de conexões sociais entre pessoas, redes sociais permitem o compartilhamento de notícias em um volume sem precedentes, mudando completamente a forma como notícias são produzidas, disseminadas e consumidas. Com um novo jeito de se disseminar informação também se iniciou uma verdadeira guerra digital, em que temos campanhas de desinformação sendo propagadas com o objetivo de reduzir a credibilidade de agências oficiais de notícias e também influenciar na opinião e decisão de leitores. O poder de influenciar a opinião de milhões de pessoas através da desinformação se torna algo atrativo nas mãos de quem tem os meios para tal e que possui interesses nos resultados de tais campanhas (JR., 2021).

Em um mundo cada vez mais conectado, velocidade é priorizada em relação à verdade. Dessa forma, os meios de divulgação de notícias possuem um poder considerável de atingir um número muito grande de pessoas em pouco tempo. A veracidade de tais notícias só pode ser questionada e, talvez provada, após um longo período de disseminação digital.

Olhando o espalhamento de notícias no mundo digital através de um ótica epidemiológica, tal disseminação de notícias possui um taxa de reprodução bastante alta dependendo do momento

específico em que elas são compartilhadas e dada a relevância de seu conteúdo. Utilizando modelos epidêmicos podemos caracterizar a taxa de reprodução básica (R_0) de um tipo de notícia em diversas redes sociais. Por exemplo, foi verificado que durante o período inicial da pandemia de COVID-19, entre 1º de Janeiro de 2020 e 14 de Fevereiro de 2020, as taxas de reprodução R_0 de notícias a respeito da pandemia atingiram os valores de 2,25 e 1,84 no Instagram e no Twitter, respectivamente (CINELLI et al., 2020). Esses números significam a média de usuários que irão começar a publicar notícias online sobre COVID-19 a partir do contato com um usuário que já postou alguma notícia sobre o mesmo assunto. Se esses mesmos valores fossem observados em um espalhamento de vírus biológicos no mundo real, teríamos valores atingidos superiores a qualquer epidemia já observada, ressaltando que valores de $R_0 > 1$ indicam a possibilidade de uma pandemia. Essa comparação do espalhamento de notícias utilizando modelos de espalhamento de vírus biológicos pode ser bastante útil para entender a capacidade de se propagar informações na Internet. Tendo em mente tamanha capacidade reprodutiva das notícias pela Internet, cada vez mais se vê necessário a implementação de técnicas de classificação automáticas de notícias, com o objetivo de alertar os usuários sobre possíveis notícias falsas e assim suprimir o impacto causado pela desinformação no mundo digital.

1.1 Motivação

A partir de um espectro voltado ao panorama brasileiro, em que temos um grande número de usuários brasileiros se informando por meio da Internet, cada vez mais se vê necessário o uso de ferramentas que auxiliem no combate ao problema da desinformação no meio digital. Já é do conhecimento de muitos desses usuários a existência de agências checadoras da veracidade de notícias, as chamadas agências de *fact-checking*. Tais agências executam um papel importantíssimo na classificação de notícias compartilhadas em redes sociais, mas tal papel depende da execução manual e humana de uma tarefa de verificação, o que requer tempo e também engajamento de usuários para que o resultado da checagem seja compartilhada e atinja usuários que tiveram contato com uma determinada notícia. Técnicas de aprendizado de máquina permitem classificar automaticamente um grande volume de notícias, para que os usuários tenham a transparência de verificar a probabilidade de uma notícia ser falsa antes mesmo de que alguma agência de *fact-checking* invista tempo e esforços em verifica-la. Como muitas das notícias falsas possuem fortes padrões de escrita que seguem um enviesamento ideológico (muitas vezes buscando um convencimento baseado em sentimentos do leitor), um algoritmo consegue identificar tais padrões e calcular uma probabilidade de que a notícia seja verdadeira, utilizando de treinamentos realizados com bancos de dados de notícias passadas. Técnicas de aprendizado de máquina têm apresentado bastante sucesso no reconhecimento de padrões, por esse motivo elas estão sendo cada vez mais utilizadas para a análise de texto de notícias falsas (SILVA et al., 2020).

Em um país em que a opinião pública é amplamente influenciada por mensagens compartilhadas em redes sociais (DATASENADO, 2019), é cada vez mais necessário a disponibilização de ferramentas de checagem da veracidade dessas mensagens, com o objetivo de mitigar o impacto da desinformação. A utilização da classificação automática de *fake news* baseada em aprendizado de

máquinas alinhada com a atuação manual de agências de *fact-checking* é uma sólida ferramenta no combate a desinformação *on-line*, em que os modelos ajudam as agências a agirem com rapidez ao selecionarem quais notícias realmente valem a pena serem investigadas.

1.2 Objetivo Geral

Tendo como motivação o problema da desinformação propagada nas redes sociais no Brasil atualmente, o presente trabalho estabelece como meta analisar o desempenho de modelos de aprendizado de máquina na classificação automática de notícias brasileiras. Além disso, foram desenvolvidas técnicas com o objetivo de analisar a estrutura linguística dessas mesmas notícias. Essa etapa de análise é útil para buscar pistas sobre quais são os padrões estruturais que compõem as principais *fake news* compartilhadas no Brasil atualmente e, assim, validar se os modelos de aprendizado de máquina utilizados estão fazendo uso de tais padrões para classificar os dados. Com o objetivo de treinar e testar os modelos escolhidos, foram construídos um dataset de treinamento e um de teste. Ambos serão compostos de notícias verdadeiras e falsas obtidas a partir de técnicas de *web scraping* realizadas em portais de notícias. O dataset de treinamento é composto por notícias verdadeiras e falsas obtidas a partir de um banco de notícias conhecido como *Fake.br Corpus* [1] construído por pesquisadores brasileiros (SILVA et al., 2020) fazendo *web scraping* de portais de notícias. O dataset de teste será composto por notícias falsas obtidas no site *boatos.org* [2], conhecido por desmentir boatos e notícias falsas compartilhadas nas redes sociais *WhatsApp* [3] e *Facebook* [4]. Já as notícias verdadeiras serão obtidas a partir do site *g1.globo.com* [5], um dos portais de notícias mais acessados no Brasil e que possui credibilidade já confirmada por grande parte da população, pois independente da existência de um possível enviesamento ideológico em suas notícias, existe uma tendência de portais como esse em relatar a verdade presenciada em acontecimentos no Brasil e no mundo, pois isso gera confiança e credibilidade por parte de grande parte dos leitores. Os modelos treinados e testados foram o CPA (Classificador Passivo Agressivo), CFA (Classificador Floresta Aleatória), SVM (Support Vector Machine), CAD (Classificador de Árvore de Decisão) e Regressão Logística. Com tais dados em mãos, fazemos uma análise estrutural de *features* do dataset de treinamento utilizando o modelo t-SNE (t-Distributed Stochastic Neighbor Embedding).

1.3 Justificativa

Uma análise sobre o desempenho de modelos selecionados de classificação baseados em *NLP - Natural Language Processing* (Processamento de Linguagem Natural), pode nortear possíveis esforços na construção de ferramentas que consigam verificar em tempo real notícias compartilhadas em redes sociais. Além disso, vê-se necessário uma análise de fatores que influenciam na tomada de

[1] https://github.com/roneysco/Fake.br-Corpus
[2] https://www.boatos.org/
[3] https://www.whatsapp.com/?lang=pt$_br$
[4] https://www.facebook.com/
[5] https://g1.globo.com/

decisão no momento da classificação por tais modelos, dessa forma, tendo melhorias na construção de tais modelos com o objetivo de ter valores de acurácia e desempenho melhores para tal tarefa.

Naturalmente surgem perguntas quando se realiza tal tarefa de classificação automática de *fake news*: Quais fatores são cruciais para a tomada de decisão dos classificadores? Seria possível identificar padrões claros em *fake news*? O contexto socio-político do momento em que foi escrito dada notícia pode ser de importância para um classificador? Não é possível garantir a resposta a todas essas perguntas no dado trabalho, mas talvez seja possível visualizar pistas.

2

Fundamentação teórica

Observando tamanha velocidade e dinamismo do mundo de compartilhamento de notícias em redes sociais, é cada vez mais necessário ter ferramentas que possam verificar a veracidade das notícias compartilhadas, o quanto antes. A desinformação possui grande capacidade de propagação digital e consequências desastrosas no mundo real, já que é muito difícil contar com a verificação instantânea da veracidade do enorme volume de notícias compartilhadas em um dia utilizando apenas técnicas manuais.

2.1 Panorama nacional das fake news no Brasil

Ilustrações claras do uso de *fake news* já foram presenciadas e comprovadas como ferramenta política em diversos casos no mundo real, vide exemplo a eleição do ex-presidente norte-americano Donald Trump em 2016, em que ferramentas analíticas foram usadas durante sua campanha para mapear perfis estratégicos de usuários de redes sociais e exibir para tais usuários sequências de notícias falsas com o objetivo de gerar desinformação e, consequentemente, vantagem eleitoral (BOVET; MAKSE, 2019). Trazendo tal discussão para o Brasil, o foco do presente trabalho, temos que 134 milhões de brasileiros tem acesso à internet, isso equivale a 3/4 da população do país, segundo levantamento feito pela TIC Domicílios 2019 (CGI.BR/NIC.BR, 2020). Além disso, de acordo com um estudo da Avaaz, 9 em cada 10 brasileiros entrevistados no país viram pelo menos uma notícia falsa sobre a COVID-19 em 2020. Aliado a isso, 7 em cada 10 brasileiros entrevistados acreditavam em pelo menos uma notícia falsa sobre a pandemia (AVAAZ, 2020). Uma preocupação real da Organização Mundial da Saúde durante a pandemia de COVID-19 é, além da doença em si, a rapidez do espalhamento de desinformações sobre a doença, que pode

estar se espalhando mais rápido que o próprio vírus. Utilizando da pandemia como panorama para entender a dinâmica de *fake news*, é possível notar suas reais consequências na sociedade, em que são vistas informações falsas sendo criadas e espalhadas intencionalmente de acordo com agendas políticas que atendem o interesse de grupos visando algum benefício da situação de desinformação coletiva.

A Reuters Institute, aliada à Universidade de Oxford, realiza anualmente um estudo revelando dados a respeito do consumo digital de notícias a partir de um questionário feito pela YouGov, uma empresa internacional de pesquisa de mercado na internet. No ano de 2021, 92.000 internautas em 46 países, incluindo o Brasil, foram entrevistados. Nesse estudo foi possível obter dados importantes sobre o consumo de notícias de forma digital no Brasil (NEWMAN et al., 2021), o que agrega bastante na análise do panorama nacional. Um detalhe inicial e importante fica visível na Figura 2.1, em que podemos visualizar um decaimento ao longo dos últimos anos no consumo de notícias a partir de meios tradicionais, como no caso, o jornal impresso, tendo um decaimento mais expressivo após o início da pandemia de COVID-19. Mídias tradicionais estão rapidamente perdendo espaço para as mídias digitais, o que fortalece o argumento de que é importante focar esforços no estudo da dinâmica dessas recentes mídias.

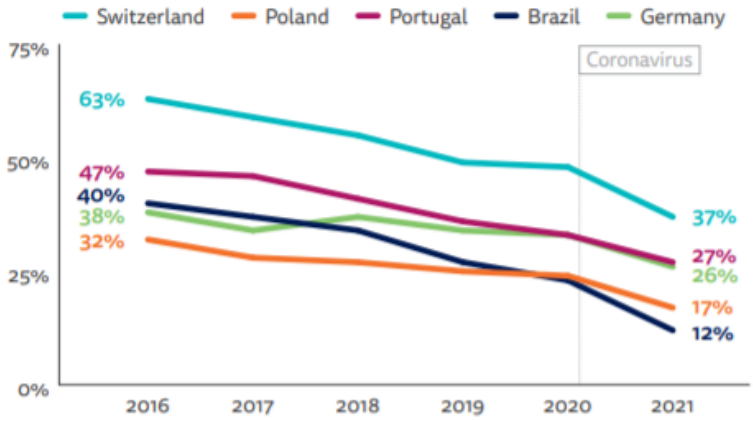

Figura 2.1: Proporção de pessoas que consumiram notícias de jornais impressos na última semana (2016-21). - *Retirado de The Reuters Institute Digital News Report, 2021.*

No Brasil, a pesquisa foi realizada com um grupo de 2.009 pessoas e, em termos de penetração da Internet, observou-se que 71% da população brasileira (211 milhões de pessoas), têm acesso à internet. Dessas pessoas, 82% possuem preocupações em relação às notícias falsas. Em comparação, na Alemanha, essa porcentagem foi de 37%. Alinhado a essa preocupação com notícias falsas, no Brasil, 54% das pessoas acreditam que podem confiar na maioria das notícias na maior parte do tempo. Apesar de tal confiança, ainda existe uma preocupação considerável com as informações propagadas por políticos, e 41% das pessoas se preocuparam com o comportamento de políticos em relação ao espalhamento de notícias falsas a respeito do coronavírus. Isso fica bastante evidente quando em 17 de Maio de 2021 foi observado pela agência independente de *fact-checking* Aos Fatos[1]

[1]https://www.aosfatos.org/noticias/bolsonaro-acumula-3000-declaracoes-falsas-ou-distorcidas-desde-o-inicio-do-mandato/

que 3000 declarações falsas ou imprecisas foram propagadas pelo presidente Jair Bolsonaro desde a sua posse até o momento da reportagem. Outro dado importante, ilustrado na Figura 2.2, é que 35% das pessoas consideraram no Brasil o *WhatsApp* a plataforma mais preocupante em relação a fake news sobre COVID-19, em segundo lugar ficou o Facebook com 18%. Esses dados sobre plataformas serão de bastante relevância no presente trabalho, pois os datasets de notícias falsas foram construídos com dados obtidos das plataformas WhatsApp e Facebook, as duas plataformas onde circulam informações que mais geram desconfiança para o usuário brasileiro, como mostrou a pesquisa.

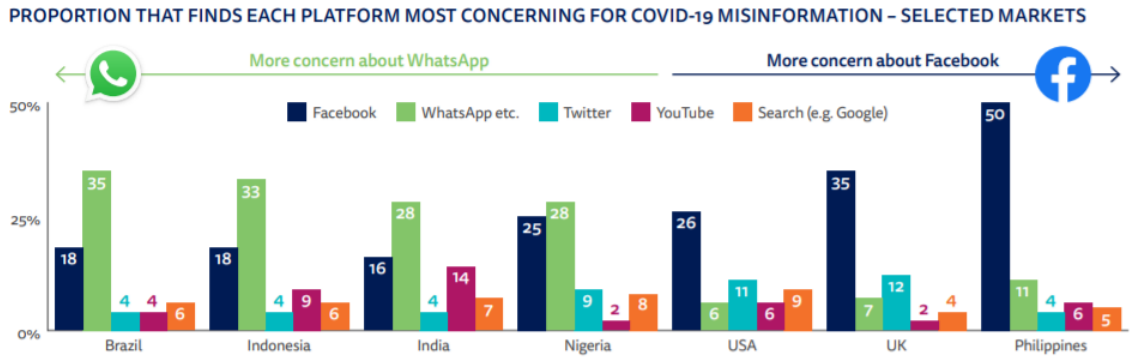

Figura 2.2: Proporção de pessoas que consideram cada plataforma mais preocupante em relação à desinformação sobre a COVID-19. - *Retirado de The Reuters Institute Digital News Report, 2021.*

2.1.1 Como o aprendizado de máquina pode auxiliar no combate as fake news

Campanhas de desinformação afetam a credibilidade de toda uma estrutura jornalística. Trazendo impactos negativos diretamente a vida de quem consome notícias, a falta de estratégias ágeis de verificação e sinalização de notícias falsas é bastante preocupante. Como já discutido, a rápida disseminação de notícias falsas é amplamente utilizada por indivíduos que esperam tirar vantagem econômica ou política da desinformação coletiva. São inúmeros os casos de golpes compartilhados em redes sociais que induzem seus usuários a compartilharem informações pessoais (*phishing*), em que golpistas visam tirar alguma vantagem econômica de suas vítimas. E, mais recentemente, a construção da injúria política baseada em falácias sobre candidatos ou partidos políticos com objetivo de ganhar vantagem eleitoral. Outro tipo de *fake news* bastante compartilhada recentemente são as notícias baseadas em ideologias conspiratórias contra determinado movimento, vide exemplo a campanha anti-vacina duranta a pandemia de COVID-19. Infelizmente, o impacto negativo da desinformação hoje vai além de golpes e narrativas políticas e resulta na perda de vidas humanas, o que nos mostra a urgência de se buscar soluções para tal problema. A agência de notícias BBC estima que, entre janeiro e março de 2020, pelo menos 800 pessoas tenham morrido em todo o mundo em decorrência de notícias falsas sobre o coronavírus[2].

Soluções automáticas para a detecção de notícias falsas poderiam ser usadas por plataformas de redes sociais com o objetivo de classificar uma grande massa de desinformação. Além disso, esse

[2]https://www.bbc.com/news/world-53755067

tipo de tecnologia pode ajudar agências de *fact-checking* a identificarem rapidamente notícias com alta probabilidade de serem falsas, deixando assim seus esforços para notícias mais desafiadoras de serem investigadas. No mundo todo atualmente, a academia e a indústria buscam como combater *fake news* de forma automática dada a sua rápida velocidade de espalhamento no mundo digital. O presente trabalho irá analisar a possibilidade da implementação de Aprendizado de Máquina como técnica de detecção automática de *fake news* utilizando o modelo Passivo Agressivo e outros. Os *datasets* serão compostos completamente por notícias compartilhadas no Brasil para que possamos concentrar os esforços no preocupante cenário brasileiro e também analisar a construção de *features* (atributos) que compõem tais notícias e como os modelos de aprendizado podem identificar padrões nas *fake news*.

2.2 Aprendizado de máquina

Avanços na computação nos permitem hoje a obtenção e armazenamento de volumes gigantes de dados (ALPAYDIN, 2004). Todos esses dados, quando analisados e transformados em informação útil de fato, permitem que algoritmos consigam, por exemplo, fazer predições. Essas predições são possíveis porque tais dados possuem padrões que podem ser analisados, quantificados e incluídos em um sistema preditivo. Para que um computador comece a identificar padrões é preciso ensiná-lo o que cada conjunto de dados representa. Por exemplo, hoje é possível ensinar um algoritmo a processar uma foto de um gato andando no parque e identificar que ali existe um conjunto de pixels que pode ser chamado de "gato". Ele consegue fazer isso pois previamente ele mesmo processou milhares de fotos de gatos em várias situações. Com isso, esse algoritmo começou a associar que um conjunto de pixels com orelhas pontudas, focinho curto e bigodes possui uma chance alta de ser um gato, ou melhor, a palavra "gato" foi o que o programador disse para o algoritmo sobre o que era esse conjunto de pixels. Ainda assim, esse algoritmo não consegue identificar gatos sempre e em qualquer situação, mas ele consegue fazer uma boa e útil aproximação. É isso que o aprendizado de máquina permite: identificar padrões de um processo e usar esses padrões para fazer predições, assumindo que no futuro próximo os dados não serão tão diferentes assim dos dados do passado que ele usou para aprender aqueles padrões, e com isso fazer uma aproximação. A Figura 2.3 ilustra um exemplo de classificação e localização de imagens utilizando um modelo de aprendizado de máquina para detecção de uso incorreto de máscaras.

Dentre as várias aplicações de *machine learning* existe o método conhecido como data mining (mineração de dados), que consiste na obtenção de estruturas de dados a partir de uma fonte, com o objetivo de processar esses dados e obter padrões dos mesmos. No nosso caso, o processo de data mining consistirá na obtenção de milhares de notícias que serão processadas e, a partir delas serão obtidas informações sobre padrões que compõem notícias classificadas como verdadeiras ou falsas e que assim seja possível classificar, com uma boa acurácia, notícias que o algoritmo nunca "viu". Mas apenas ter uma coleção de dados não significa muita coisa para um algoritmo de aprendizado, é necessário ter uma inteligência. É por isso que aprendizado de máquina é parte do estudo de *Inteligência Artificial*, pois o algoritmo precisa aprender padrões e se adaptar a mudanças no ambiente, de forma que o administrador do sistema não precise prever problemas e providenciar

soluções para o modelo.

Utilizando estatística como base teórica para construir modelos matemáticos, o objetivo principal do aprendizado de máquina é fazer inferências a partir de amostras. Para se alcançar tal objetivo temos que desenvolver duas etapas gerais: primeiramente, o treinamento do algoritmo em que parâmetros do modelo são otimizados ao processar uma quantidade grande de dados passados (o chamado *dataset* de treinamento). Após o treinamento, com um modelo pronto para realizar inferências, devemos instruí-lo a processar uma dada entrada de dados para que ele possa realizar uma predição. Nesse caso, a predição vai ser apontar se uma dada entrada é classificada como uma notícia verdadeira ou falsa. Em alguns tipos de aplicações, essa etapa de inferência deve ser eficiente, ou seja, sua complexidade computacional pode ser tão importante quanto o resultado de sua acurácia.

No geral, o aprendizado pode ser supervisionado, ou seja, cada entrada nos dados de treinamento são rotulados com as respostas corretas, ou não supervisionado, que é o caso em que os dados não são rotulados e o processo de aprendizagem tenta reconhecer padrões automaticamente (NADKARNI; OHNO-MACHADO; CHAPMAN, 2011). Além disso, modelos de aprendizado de máquina podem ser classificados como generativos ou discriminativos. Métodos generativos buscam criar modelos que consistem em distribuições probabilísticas que permitem "gerar" dados sintéticos. Já métodos discriminativos estimam probabilidades posteriormente, baseados em observações. Um exemplo de abordagem para cada método consiste no problema de identificar uma língua desconhecida falada por um indivíduo. Métodos generativos para resolver tal problema consistiriam em aplicar conhecimentos profundos sobre diversas línguas até encontrar uma língua que responda a questão. Métodos discriminativos iriam consistir em etapas menos dependentes de conhecimentos prévios, utilizando-se de diferenças entre as línguas já conhecidas para encontrar a resposta. Regressão logística e campos condicionais aleatórios são exemplos de métodos discriminativos, enquanto classificadores como *Naive Bayes* e modelos de *Markov* são exemplos de métodos generativos.

Um algoritmo de aprendizado de máquina deve ser capaz de aprender, a partir de uma experiência passada, a realizar um grupo de tarefas e com uma certa medida de performance, se a sua performance para realizar tais tarefas é melhorada com a experiência, temos um algoritmo de aprendizado (GOODFELLOW; BENGIO; COURVILLE, 2016). Tarefas em aprendizado de máquina geralmente são descritas em termos de como o sistema deve processar um exemplo. Basicamente, um exemplo é uma coleção de features, que são unidades de informação mensuráveis e relevantes para se ajustar um algoritmo de aprendizado de máquina. No caso do presente trabalho, um exemplo é uma notícia, e uma feature é uma palavra existente nessa notícia. Tipicamente, um exemplo é representado por um vetor $x \in \Re^n$, aonde cada entrada x_i do vetor é uma outra feature. Além disso, no nosso caso a tarefa empregada será a classificação.

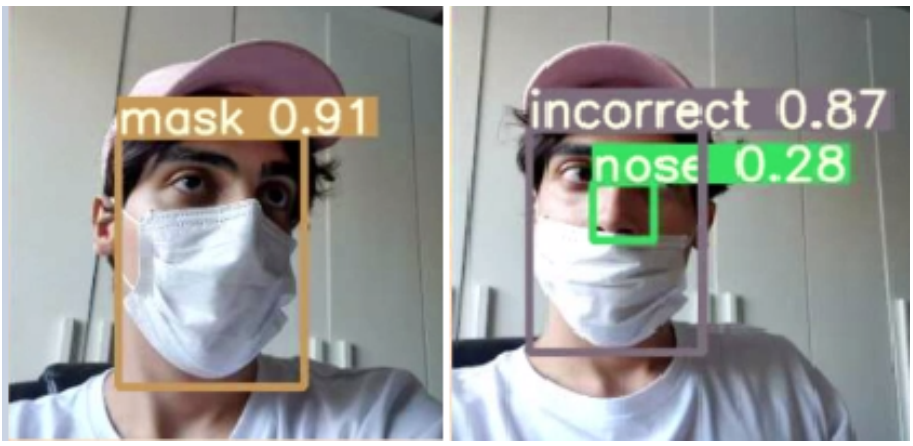

Figura 2.3: Exemplo de reconhecimento de padrões em imagens para detectar o uso incorreto de máscaras.

2.2.1 Classificação

Para aprendizado de máquina, um problema de classificação acontece quando temos duas ou mais possíveis classes e o algoritmo deve tomar a decisão de atribuir uma das possíveis classes a uma dada entrada (ALPAYDIN, 2004). Classificação é um problema de *aprendizado supervisionado*, ou seja, o objetivo é aprender o mapeamento que conecta uma entrada A a uma saída B, ambas conhecidas na etapa de treinamento. Nesse tipo de problema, como cada exemplo está associado à uma classe, essa tarefa é possível de ser realizada, já que através da tarefa de "supervisão" mostramos para o algoritmo o que fazer ou quais são os resultados esperados. O algoritmo, após processar informações relevantes sobre as classes, deve ajustar um modelo de treinamento com os dados do passado e ser capaz de calcular a probabilidade de atribuir corretamente uma classe a um novo exemplo. No caso do presente estudo, o texto das notícias e se ela é uma notícia verdadeira ou não são as informações relevantes que o modelo de treinamento considera na hora de ajustar as duas classes existentes: verdadeira ou falsa (Figura 2.4).

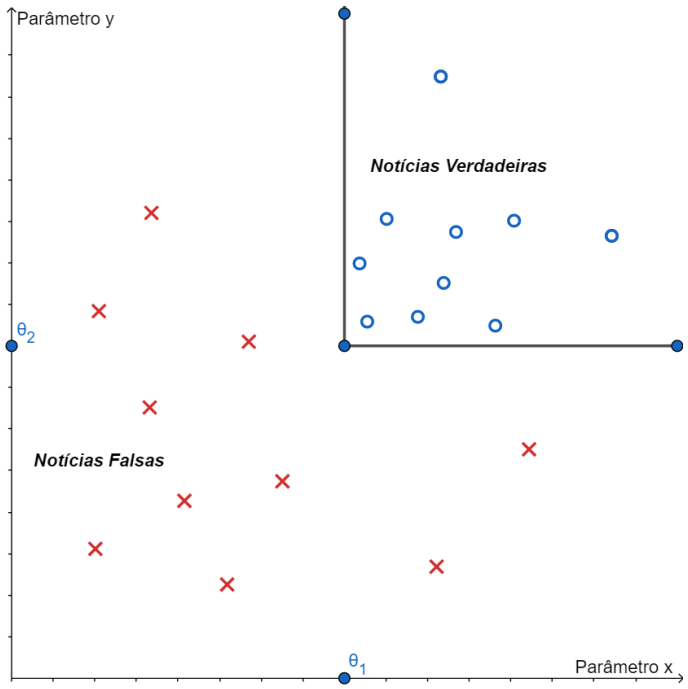

Figura 2.4: Exemplo de um dataset de treinamento onde cada ponto no gráfico representa um exemplo (notícia), sendo que o símbolo de cada notícia representa a classe, 'x' para falsa e 'o' para verdadeira. O eixo y representa uma determinada *feature* y e o eixo x uma determinada *feature* x. Existe um discriminante que separa os dois possíveis tipos de exemplos.

Após treinar com os dados do passado, um exemplo de regra de classificação criada para o modelo seria do tipo:

$$SE\ x\ >\ \theta_1\ E\ y\ >\ \theta_2\ ENTAO\ verdadeiro\ SE\ NAO\ falso \tag{2.1}$$

Para valores adequados de θ_1 *e* θ_2 como discriminantes separando exemplos de diferentes classes.

Com essa regra geral, é possível criar uma aplicação de predição tendo um modelo que ajusta seus parâmetros com base em dados passados. Caso os dados futuros sejam similares aos do passado, então conseguimos fazer predições para novas entradas. No caso, utilizando como entrada notícias nunca antes vistas pelo modelo, ele conseguirá fazer predições com uma determinada probabilidade calculada sobre a veracidade dessas notícias. Em especial, interessa-nos o cálculo da probabilidade de uma notícia ser verdadeira ou falsa, ou seja, $P(B|A)$, em que A são as *features* da notícia relevantes ao modelo e B uma *label* que pode assumir dois valores, 0 ou 1 para falso ou verdadeiro, respectivamente. Dessa forma, fica evidente que a tarefa de classificação é aprender uma associação entre A e B. Para um dado $A = x$, se temos que $P(B = 0|A = x) = 0,9$, podemos afirmar que para uma dada notícia existe a probabilidade de 90% de ser falsa e, de forma equivalente, 10% de ser verdadeira.

O problema fundamental da classificação de textos está na dificuldade de representar um dado texto a nível numérico, ainda que tenhamos várias técnicas disponíveis (Term Frequency - Inverse Document Frequency, *word2vec*, *glove*) que serão abordadas posteriormente. A limitação dessas

técnicas se encontram quanto se tenta ir além da representação semântica. Um texto não são apenas palavras soltas, o sentido de um texto está atrelado à ordem que as palavras se encontram e para se ter uma classificação eficiente é necessário buscar técnicas que consigam interpretar contextos de disposição de palavras, como por exemplo técnicas mais avançadas como redes neurais convolucionais e redes neurais recorrentes, ou técnicas clássicas como classificação.

Redes neurais envolvem processos complexos de Deep Learning que vão além do escopo do presente trabalho. Tarefas de classificação podem ser implementadas com algoritmos mais simples, como os modelos do tipo *online-learning*, que serão abordados posteriormente.

2.2.2 Medidas de desempenho

Uma métrica relevante para tarefas de classificação é a *acurácia* do modelo. A acurácia é a proporção de exemplos para qual o modelo produziu saídas corretas. De forma complementar, nós também podemos obter informações equivalentes medindo a *taxa de erro*, que mostra a proporção de exemplos para qual o modelo produziu saídas incorretas. Tanto a acurácia quando a taxa de erro são consideradas boas métricas apenas quando temos um *dataset* de treinamento balanceado, que contenha quantidades parecidas de diferentes classes, de forma que se esse não for o caso, o modelo pode se tornar enviesado. Estamos interessados em quão bem os modelos de aprendizado se comportam com dados nunca antes vistos, para termos uma ideia de uso no mundo real. Por isso o *dataset* de teste é composto inteiramente de notícias que o modelo nunca processou durante o treinamento. Assim conseguimos descrever cada resultado da classificação da seguinte forma, como definida por Alpaydin (ALPAYDIN, 2004):

- Falso Positivo (FP): label negativo classificado como positivo;
- Verdadeiro Positivo (VP): label positivo classificado como positivo;
- Falso Negativo (FN): label positivo classificado como negativo;
- Verdadeiro Negativo (VN): label negativo classificado como negativo;

Com esses resultados, conseguimos definir métricas quantitativas para entender o desempenho de cada modelo. Primeiramente, a *acurácia* (equação 2.2), já discutida, que consiste no número de acertos (VP e VN), dentro o número de exemplos totais (N). Além da acurácia, é importante considerar a precisão (equação 2.3), que mede o número de verdadeiros positivos (VP) dentre todas as predições positivas (VP e FP). Existe também a *sensibilidade* (equação 2.4), *recall* em inglês, que consiste no número de verdadeiros positivos (VP) em relação a todas os exemplos com labels positivos (VP e FN). Por último, existe uma métrica mais assertiva que une a precisão e a sensibilidade, o *F1 score* (equação 2.5), que basicamente é uma média harmônica dessas duas métricas e leva em conta diferentes aspectos dos erros e acertos das predições, resultando em uma métrica possivelmente menos enviesada quando comparada à acurácia.

$$\text{Acuracia } = \frac{(VP + VN)}{N} = \frac{(VP + VN)}{FP + VP + FN + VN} \tag{2.2}$$

$$\text{Precisao} = \frac{(VP)}{VP + FP} \tag{2.3}$$

$$\text{Sensibilidade} = \frac{(VP)}{VP + FN} \tag{2.4}$$

$$\text{F1score} = \frac{2}{\frac{1}{Sensibilidade} + \frac{1}{Precisao}} = \frac{2.(Sensibilidade.Precisao)}{Sensibilidade + Precisao} \tag{2.5}$$

Além disso, utilizaremos um artifício conhecido como matriz de confusão para analisar graficamente o desempenho dos modelos treinados após serem testados com o mesmo *dataset* de teste. A matriz de confusão consiste em uma tabela que mostra as frequências de classificação para cada classe de um modelo. No caso de uma classificação binária, consistindo em apenas duas diferentes classes, teremos uma matriz de confusão 2x2, como mostra a Figura 2.5.

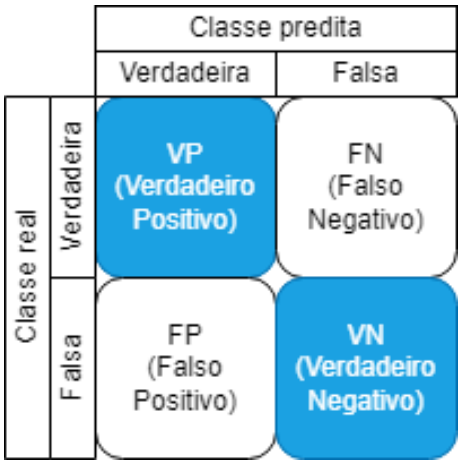

Figura 2.5: Exemplo de matriz de confusão para classificação binária.

2.3 *NLP - Natural Language Processing*

Antes de alimentar modelos de aprendizado com dados a serem utilizados no treinamento de predições, é necessário entender como esses modelos computacionais entendem dados que representam um texto escrito em uma linguagem humana. PLN - Processamento de Linguagem Natural (*NLP - Natural Language Processing*, em inglês) é o termo cunhado para definir a vertente do estudo de inteligência artificial que se preocupa com a capacidade e limitações de um algoritmo computacional no entendimento da linguagem dos seres humanos, pois entendendo tal linguagem é possível processar textos. A implementação do PLN permite que hoje seja viável a utilização de assistentes digitais por comando de voz como a *Alexa*[3] e a *Siri*[4], por exemplo. Para entender

[3]https://oglobo.globo.com/economia/tecnologia/amazon-lanca-caixa-de-som-inteligente-que-responde-comandos-de-voz-14486970

[4]https://g1.globo.com/tecnologia/noticia/apple-apresenta-ios-11-nova-siri-e-sistemas-operacionais-para-macs-e-relogios-inteligentes.ghtml

um comando como "ligar as luzes do quarto", o modelo utilizado pelas assistentes digitais deve ser capaz de destrinchar todos os termos que compõem tal sentença para que, processando cada palavra, seja possível entender o contexto inteiro e, assim, o comando desejado. Além das estruturas sintática, semântica, lexical e morfológica, um algoritmo ainda tem a tarefa de tentar entender conceitos mais ambíguos como reconhecimento de contexto, tonalidade de voz, duplos-sentidos, analisar sentimentos e tarefas que, apesar de triviais para seres humanos, se tornam complexas para um computador. Algumas tarefas que o PLN permite serem realizadas incluem: identificação de erros de gramática, reconhecimento de entidades nomeadas e a categorização das mesmas, determinação de relação entre entidades ou eventos e extração de informações. Atualmente, várias técnicas de estatística e aprendizado de máquina estão em desenvolvimento para abordar as tarefas descritas e muitas outras. Geralmente, essas técnicas consistem na aprendizagem de padrões obtidos no treinamento que permitem uma generalização sobre as saídas futuras do algoritmo.

Apesar dos desafios, o processamento de linguagem natural é uma ferramenta que estabelece uma interface entre humano e computador, que possibilita, dentre muitas possibilidades, a automação de tarefas que comumente seriam trabalhosas para seres humanos, como, por exemplo a classificação de notícias. Para ajustar um modelo de classificação é necessário antes tomar alguns passos, esses são: limpeza de dados, pré-processamento e *"tokenização"* (PINHEIRO, 2021). O termo *"tokenização"* é uma adaptação traduzida de forma livre do termo em inglês *tokenization*, que consiste no processo de se obter sequências de caracteres em um documento em particular, agrupados como uma unidade semântica útil para o processamento. O presente trabalho irá utilizar a linguagem de programação *Python* e a biblioteca de código-aberto *NLTK*[5] (*Natural Language Toolkit*) para realizar as etapas de limpeza, pré-processamento e *"tokenização"* dos *datasets* utilizados pelos modelos.

2.4 Modelos utilizados

Para conseguir realizar as classificações de notícias iremos recorrer a utilização de modelos de aprendizado de máquina já concebidos, ajustados com os dados de treinamento. Como estamos trabalhando em um caso em que conhecemos as entradas, chamadas de *features*, e também as saídas, que nosso caso serão as *labels fake* ou *true* para uma notícia falsa ou verdadeira, respectivamente, iremos recorrer a modelos de aprendizado de máquina supervisionado. Basicamente, um modelo de aprendizado de máquina é um algoritmo que irá ser ajustado a dados de treinamento, para que assim reconheça padrões nesses dados de entrada. No nosso caso, os modelos irão, de forma iterativa, buscar padrões nas notícias verdadeiras ou falsas. A forma como esses modelos buscam esses padrões é o que os distinguem. O modelo escolhido como ênfase do presente trabalho foi o classificador Passivo Agressivo, para efeitos de comparação, outros modelos foram testados nas etapas de implementação.

[5] https://www.nltk.org/

2.4.1 *Passive agressive classifier*

Um classificador PA (Passivo Agressivo) é um classificador utilizado em aplicações de *big data*, quando se têm grandes sequências de entrada de dados (CRAMMER et al., 2006). Além disso, ele é um algoritmo do tipo *online-learning*, em que os dados de entradas chegam em ordem sequencial e o modelo de aprendizado de máquina é atualizado constantemente, ao contrário do que acontece com modelos de aprendizado em *batch*, em que o *dataset* inteiro de treinamento é utilizado de uma vez. Esse funcionamento faz do classificador Passivo Agressivo um algoritmo bastante eficiente, mesmo ao se processar grandes *datasets*. Esse tipo de algoritmo é bastante útil em uma situação em que se deve consumir dinamicamente novos dados de um meio sempre em atualização como, por exemplo, as redes sociais. O funcionamento básico do algoritmo consiste em se manter passivo caso ele faça alguma classificação correta e agressivo se houver algum resultado incorreto. Dessa forma, o algoritmo PA basicamente obtém um documento, atualiza seu classificador e descarta esse documento, de forma que ele sempre estará correto para o último documento visto (GUPTA; MEEL, 2021). A Figura 2.6 ilustra o antigo peso do classificador (w) sendo ajustado para um novo valor de peso (w_{novo}) por conta de uma classificação errada feita para um documento (d). O valor de (w_{novo}) foi ajustado por uma perda (L) que move o vetor peso do classificador para se adaptar à última classificação realizada. As cruzes vermelhas representam notícias classificadas como falsas e o círculos azuis representam notícias classificadas como verdadeiras. O objetivo dele é corrigir os erros no vetor de peso, resultando em pequenos melhoramentos nele. Além disso, o algoritmo tenta sempre reduzir a perda acumulada, de forma que se evite alterações muito grandes, realizadas de uma só vez, no peso do vetor de classificações (CRAMMER et al., 2006).

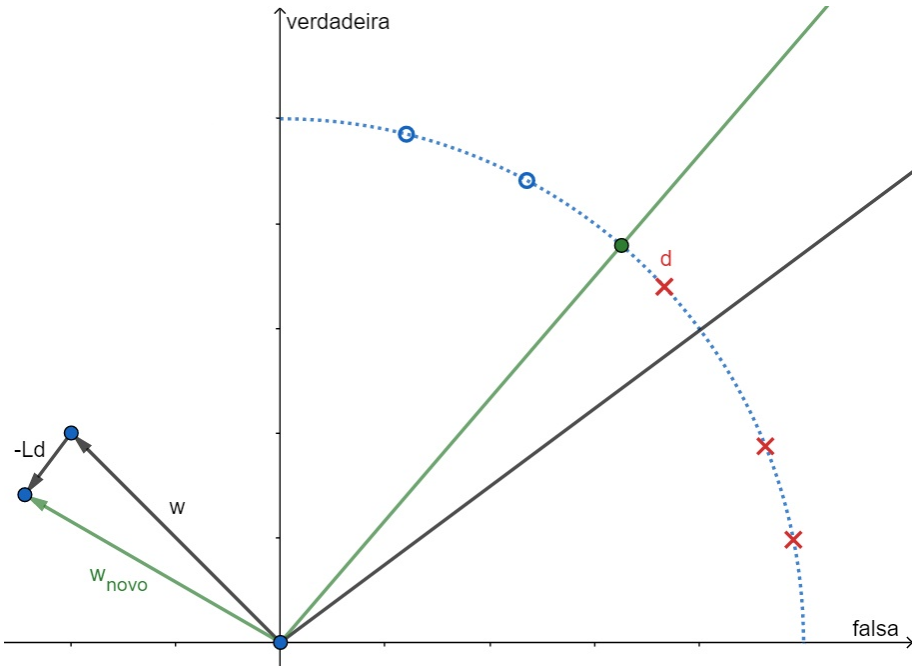

Figura 2.6: Funcionamento do treinamento do classificador passivo agressivo no caso em que uma classificação foi feita errada. O documento d foi classificado erroneamente, resultando em uma atualização no vetor peso w. As cruzes vermelhas representam notícias classificadas como falsas e o círculos azuis representam notícias classificadas como verdadeiras.

Seu funcionamento básico consiste em:

1. Inicializar o vetor peso $\vec{w} = (0, ..., 0)$ como um vetor nulo
2. Monitorar uma sequência de documentos na entrada
3. Receber um novo documento $\vec{d} = (d_1, ..., d_v)$
4. Aplicar etapas de pré-processamento, normalização $\|\vec{d}\| = 1$ e tokenização
5. Prever como positivo se $d^T w > 0$, em que d^T é o vetor transposto do documento de entrada
6. Observar a verdadeira classe y, *sendo que* $y = \pm 1$
7. O esperado é ter:
   - $d^T w \geqslant +1$ *se positivo* $(y = +1)$
   - $d^T w \leqslant -1$ *se negativo* $(y = -1)$

   que é o mesmo que $y(d^T w) \geqslant 1$
8. Calcular a perda L, sendo que $L = max(0, 1 - y(d^T w))$
9. Atualizar o vetor peso do classificador $w_{novo} = w + yLd$

Na Figura 2.7 é exibido um fluxograma das etapas do funcionamento do algoritmo classificador Passivo Agressivo.

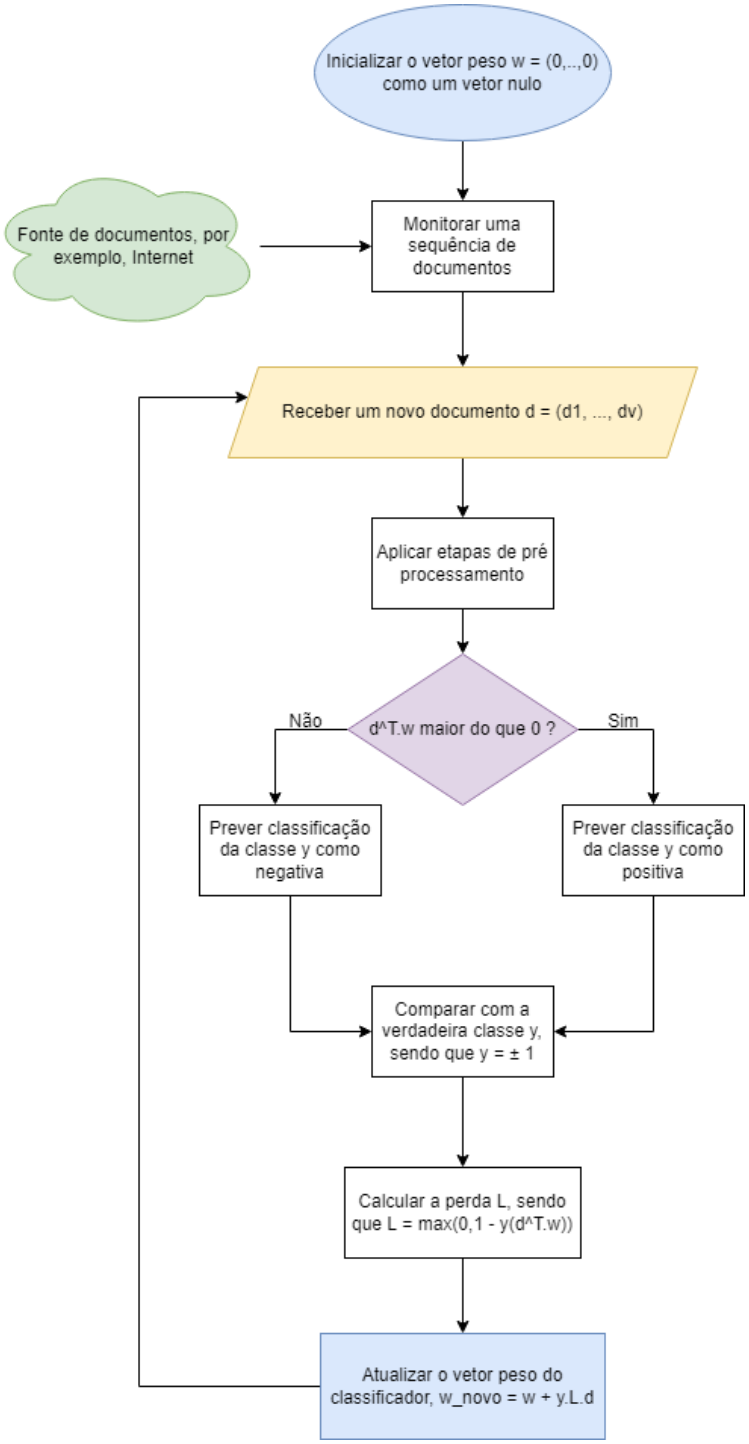

Figura 2.7: Fluxograma do funcionamento do classificador Passivo Agressivo.

2.4.2 *Redução de dimensionalidade com Distributed Stochastic Neighbor Embedding (t-SNE)*

Aprendizado de máquina com PLN envolve geralmente número enormes de *features* que representam um dado documento (uma notícia, por exemplo). Cada *feature* é uma dimensão utilizada para descrever um ponto localizado em um espaço. Suponha que um projeto envolve a extração de 2.000 *features* ($p = 2.000$) de um dataset de textos extraídos da Internet. Cada palavra (*feature*) desse *dataset*, é uma dimensão para cada texto representado por um ponto X localizado em um espaço $\Re^p$. Visualizar informações envolvendo muitas dimensões é bastante complexo para um ser humano interpretar, dessa forma, a viabilidade de se extrair informações de um *dataset* de textos exige a diminuição de dimensões. A técnica conhecida como *t-SNE* consiste na redução de dimensionalidade de dados com o objetivo de plotar dados em um gráfico de dispersão de duas ou três dimensões. Essa é uma técnica não linear e não supervisionada que é uma variação do algoritmo *Stochastic Neighbor Embedding (SNE)*. O SNE tem como princípio a conversão de distâncias Euclidianas entre pontos de dados de dimensões maiores, em probabilidades condicionais de dimensões menores que representam similaridades entre esses pontos(MAATEN; HINTON, 2008). A *t-SNE* consiste na resolução dos problemas do *SNE* de forma que a função custo utilizada por ela difere da usada pela *SNE* de duas formas:

- Usando uma versão simétrica da função de custo da *SNE* com gradientes mais simples;
- Usando uma distribuição *t-Student* ao invés de uma distribuição Gaussiana para computar as similaridades entre dois pontos em um espaço de poucas dimensões.

Assim, o algoritmo *t-SNE* calcula a similaridade entre pares de dados em um espaço de muitas dimensões em um espaço de poucas dimensões. Em seguida, o algoritmo tenta otimizar essas duas medidas de similaridades usando uma função de custo.

3

Metodologia

Nesta pesquisa, foi definida como ênfase do estudo a possibilidade de se classificar automaticamente notícias falsas com base em algoritmos de aprendizado de máquina e extrair conhecimento do corpus. Para implementar tal objetivo foi escolhido a linguagem *Python* para se executar as etapas necessárias. Na Figura 3.1 é possível observar os passos necessários no processo de classificação de notícias.

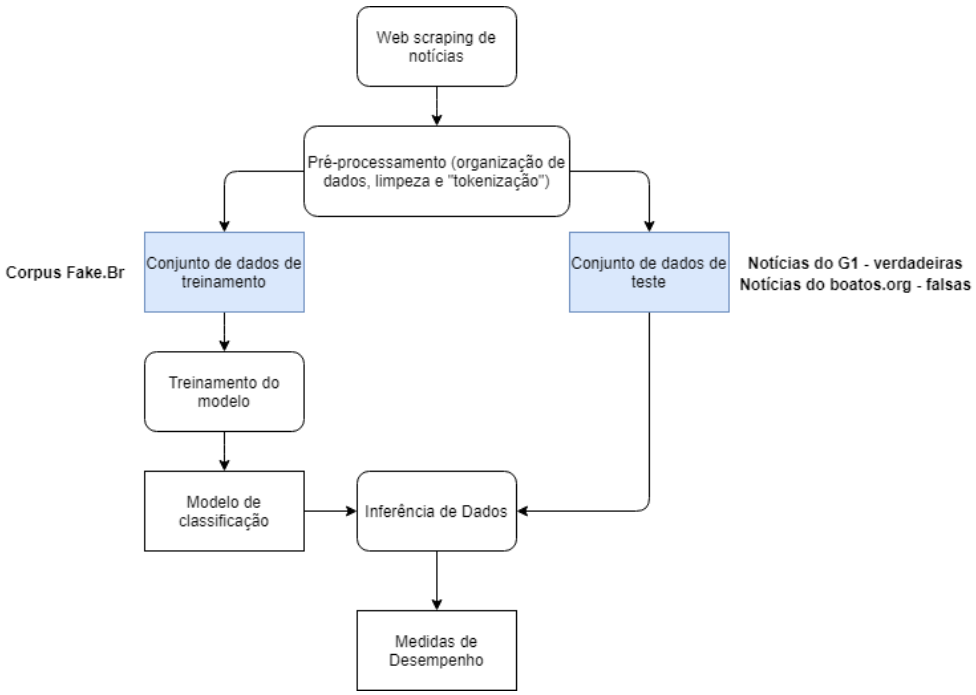

Figura 3.1: Fluxograma de etapas para a tarefa de classificação.

O primeiro passo no experimento é a obtenção (*web scraping*) de notícias verdadeiras e falsas para composição dos *datasets*. Com o objetivo de simular uma utilização em um cenário real, os *datasets* de treinamento e teste são compostos de origens diferentes, de forma que possuem pouca ou quase nenhuma similaridade. Assim, com os dados obtidos passa-se para a etapa de pré-processamento, em que são removidos acentos, caracteres especiais e outras "sujeiras" que possam atrapalhar na tarefa de classificação, permitindo assim a etapa seguinte de extração de *features*. Obtido as *features*, o *dataset* de treinamento é utilizando para treinar o modelo escolhido, que então permite a tarefa de inferência de dados, tendo como entrada o *dataset* de teste. O resultado desse processo é um conjunto de dados inferidos, com resultados da classificação realizada pelo modelo treinado, assim, a partir desses dados é possível obter métricas de desempenho do modelo.

3.1 Construção dos Datasets

O *dataset* de treinamento é composto pelas notícias disponibilizados pelo *Fake.br-Corpus*[1]. O estudo realizado consistiu na análise de diversos modelos de aprendizado de máquina treinados com um conjunto de documentos (corpus) construído para a tarefa específica de classificação de fake news (SILVA et al., 2020). Esse corpus, como sugerido por trabalhos prévios (RUBIN; CHEN; CONROY, 2015), consiste em notícias falsas complementadas por suas correspondentes notícias verdadeiras (o que é uma tarefa desafiadora de ser completada). Um *dataset* composto dessa forma permite que seja identificado pelo modelo, padrões e regularidades em ambos os casos de notícias verdadeiras e falsas. Esse *corpus* é composto pro 7200 notícias (3600 falsas e 3600 verdadeiras)

[1]https://github.com/roneysco/Fake.br-Corpus

sendo que as notícias foram publicadas entre Janeiro de 2016 e Janeiro de 2018. O processo de construção do corpus é apresentado na Figura 3.2. A Figura 3.2 ilustra o processo de construção do *Fake.br-Corpus*, em que notícias classificadas como meia verdade foram excluídas do corpus. Além disso, através de palavras chave das notícias falsas, foi possível obter notícias verdadeiras equivalentes em portais de notícias confiáveis.

O *dataset* de teste, por sua vez, foi obtido pela realização de *web scraping* (Anexo I) em alguns dos principais portais de notícias da internet brasileira. As notícias verdadeiras foram obtidas[2] no portal G1[3], pois, como elucidado no Capítulo 1, é um dos portais mais acessados pelos brasileiros, possuindo bastante credibilidade. Levando em consideração a já crescente preocupação de boatos e notícias compartilhadas nas redes sociais *Facebook* e *WhatsApp* no Brasil, como mostrado na Seção 2.1, as notícias falsas foram obtidas no portal Boatos.org[4] que concentra um extenso número de boatos e notícias falsas compartilhadas nessas redes sociais. O processo de coletar notícias dos portais para compor o *dataset* de teste consistiu primeiramente na obtenção de uma série de links dentro dos portais. Para isso então, foram listado uma série de páginas dentro dos portais contendo links para notícias. Então, para cada link, foi obtido a notícia dentro desse link e o devido processamento foi realizado para obter somente o texto base da notícia. Sendo assim, links externos, propagadas e outros conteúdos irrelevantes para o processamento foram removidos do texto utilizando a biblioteca *BeautifulSoup*. O *dataset* de teste foi composto por um total de 9.440 notícias (4.711 falsas e 4.729 verdadeiras), escritas e publicadas entre Janeiro de 2015 até Setembro de 2021.

O objetivo principal de se construir os *datasets* dessa forma, é possibilitar um experimento simulando um caso real em que notícias e/ou boatos publicados posteriormente as notícias usadas para treinar o modelo, também sejam classificadas. Dessa forma, obtendo boas métricas, pode ser viável que um modelo não precise ser treinado sempre com as notícias mais recentes para que consiga classificar notícias verdadeiras ou falsas com um boa acurácia.

[2] *web scraping* do site do G1 realizado com base no código encontrado em https://www.kaggle.com/diogocaliman/minerador-de-not-cias-web-scraping/notebook, com adaptações

[3] https://g1.globo.com/

[4] https://www.boatos.org/

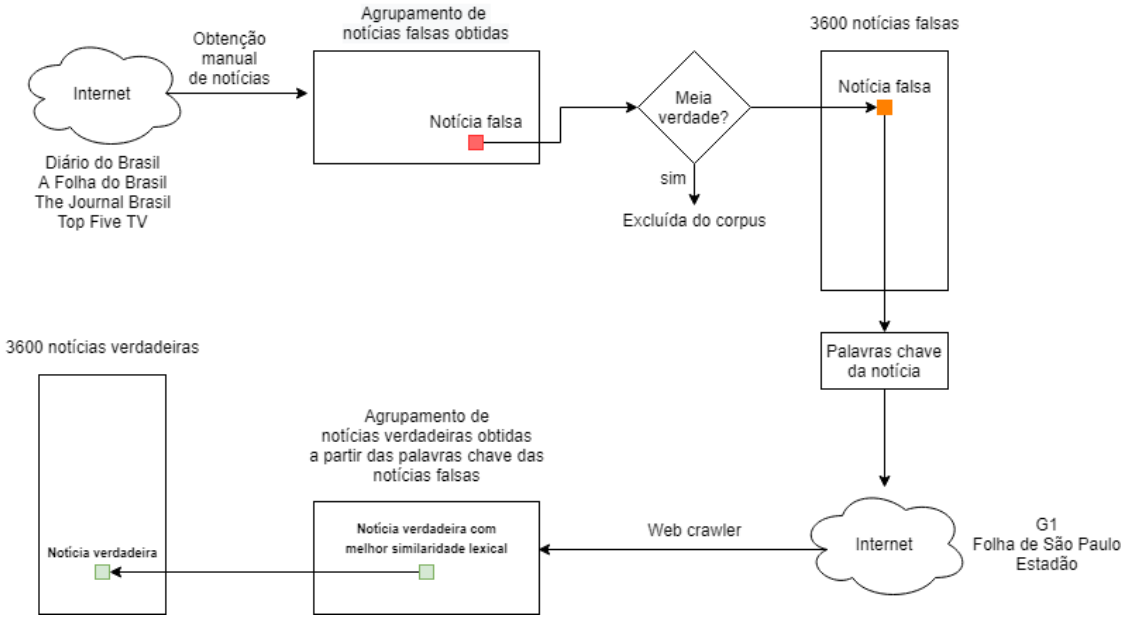

Figura 3.2: Fluxograma de etapas para a construção do corpus Fake.br, utilizado como dataset de treinamento. - *Adaptado de (SILVA et al., 2020)* .

3.2 Etapas de Pré Processamento

As etapas de pré processamento consistem na organização dos dados, limpeza de caracteres e *"tokenização"*. Após a etapa de *web scraping* das notícias, foi utilizado a biblioteca *Pandas*[5] para manipular e organizar o conjunto de dados de milhares de notícias em *dataframes*. Com essa biblioteca foi possível organizar todos os *dataframes* em um padrão de colunas que segue a seguinte ordem: *index*, *link*, *timestamp*, notícia e *label*, como mostra a Figura 3.3.

	link	timestamp	noticia	label
0	https://piaui.folha.uol.com.br//2019/10/16/ver...	2019-10-16T19:10:31-03:00	"Na Holanda, a ministra da Saude trabalha duas...	fake
1	https://piaui.folha.uol.com.br//2018/08/27/ver...	2018-08-27T15:00:38-03:00	"Pacaraima nao tinha um homicidio ha tres anos...	fake
2	https://piaui.folha.uol.com.br//2021/04/23/ver...	2021-04-23T11:16:22-03:00	"Agua da torneira testa positivo para Covid-19...	fake
3	https://piaui.folha.uol.com.br//2019/10/10/gil...	2019-10-10T07:01:44-03:00	"Que ministro do Supremo depois fez isso [entr...	fake
4	https://piaui.folha.uol.com.br//2021/05/19/ver...	2021-05-19T18:25:42-03:00	Bom dia seguidores da pagina Viva Israel, esto...	fake

Figura 3.3: Organização padrão dos *dataframes*.

3.2.1 Limpeza de dados/pré-processamento

Uma etapa prévia de limpeza e pré-processamento é importante no processamento de modelos *NLP* pois podemos normalizar e preparar os nossos *datasets* com o objetivo de otimizar tal processo. Dessa forma, deixamos apenas dados relevantes de fato, para a máquina entender uma dada tarefa. Essa etapa consiste em reduzir o vocabulário e tornar os dados menos esparsos, tarefa conveniente

[5]https://pandas.pydata.org/

para um algoritmo que processa uma grande quantidade de dados. No caso do processamento de notícias, existem muitos exemplos no *dataset* que possuem pouca relevância para a tarefa de classificação, vide exemplo, estruturas gramaticais como artigos e conjunções, pois são *features* que se repetem por todo o *dataset* e possuem pouca relevância nos pesos que classificam uma notícia como verdadeira ou falsa. Tais dados são chamados de *stop-words* e no início do pré-processamento geralmente já são removidas do *dataset*. Além disso, existem *features*, que não são artigos e nem conjunções, mas, por exemplo, verbos que se repetem bastante e são comuns em todo o *dataset*, possuindo assim pouca relevância na tarefa de classificação, como por exemplo o verbo "foi". Veremos que na etapa de *"tokenização"* escolhida no presente trabalho, tais palavras foram naturalmente consideras irrelevantes para o algoritmo. Além disso, como parte da limpeza de dados, também é importante remover caracteres especiais, como acentos e outros símbolos.

3.2.2 *"Tokenização"*

Para que um algoritmo computacional entenda palavras escritas em um determinado conjunto de dados, é necessário transformar esse texto em uma informação numérica, já que um texto não é um tipo de dado estruturado. Existem várias técnicas para se alcançar tal objetivo, entre os principais: *bag of words*, *TF-IDF* e *word embeddings* (BIRD; KLEIN; LOPER, 2009). Iremos aqui fazer uso do *TF-IDF* e do *word embeddings*, ignoraremos o bag of words pois seu funcionamento não vai de acordo com a tarefa de classificação proposta. Isso se dá pelo fato do bag of words consistir na representação numérica de uma palavra apenas indicando se ela está presente ou não em um documento (notícia), além disso é possível representar com ele o número de ocorrências de uma palavra em um dado documento. Como já explicado, os nossos modelos estão interessados em palavras que possuem relevância no *corpus* (*dataset* inteiro). Somente a existência ou o número de ocorrências de uma palavra no *corpus* não indica exatamente que essa é uma palavra relevante. No nosso caso em que analisamos notícias, muitas palavras são comuns em qualquer notícia, verdadeiras ou falsas, como por exemplo, os dias da semana. Partindo desse princípio, o nosso dataset será *"tokenizado"* utilizando a técnica *TF-IDF (Term Frequency Inverse Document Frequency)*. O TF-IDF consiste em avaliar a frequência de aparição de um dado *feature* em um dado documento (TF - term frequency), incorporando no cálculo o inverso da frequência geral desse mesmo *feature* em relação a todos os documentos do *corpus*, de forma que termos muito comuns e repetitivos no *corpus* sejam interpretados como menos relevantes. Assim, palavras únicas e que aparecem poucas vezes no *corpus* ganham um peso vetorial mais relevante.

A técnica *word embeddings* possui uma estratégia diferente. Ela consiste em vetores contendo números reais que representam palavras (*features*) ou até documentos inteiros em um espaço vetorial. O mapeamento de palavras para um vetor é ajustado por uma rede neural, de forma que durante essa etapa, as palavras são treinadas e definidas de acordo com o contexto em que se encontram inseridas e de não de forma isolada. O resultado desse tipo de processamento é a possibilidade de preservar informações morfológicas, sintáticas e semânticas, diferente dos outros métodos discutidos aqui. Com essa capacidade de capturar regularidades linguísticas, temos a possibilidade de localizarmos palavras próximas no espaço vetorial usando aritméticas simples. Então temos que

palavras semelhantes e próximas, no contexto das notícias, são colocadas próximas umas das outras em um espaço vetorial de duas dimensões, por exemplo. Esse tipo de técnica se mostrará de suma importância para entendermos relações de proximidade entre palavras que caracterizam notícias verdadeiras e falsas, nos permitindo extrair conhecimento a respeitos de padrões linguísticos nos textos. Uma última e importante etapa de pré-processamento é a limitação do dicionário criado após a etapa de *"tokenização"*. Assim, podemos selecionar os termos mais relevantes pro modelo e otimizar sua complexidade computacional nas etapas de treinamento e inferência.

3.2.3 Aplicação do pré processamento

Após organizar os dados, foi realizada uma limpeza inicial dos *dataframes*, que consistiu na remoção de números, caracteres especiais e *URLs*. Além disso, mais uma etapa necessária de limpeza foi realizada: a remoção de *stopwords*. *Stopwords* consistem basicamente de palavras usadas frequentemente em dada língua, no nosso caso, da língua portuguesa. Logo, artigos, preposições, pronomes, advérbios e verbos auxiliares foram todos removidos dos *dataframes* a partir de uma lista importada da biblioteca *Python NLTK*[6], que define quais são essas *stopwords* na língua portuguesa.

Para cada modelo selecionado, foi então realizada a tarefa de *"tokenização"* dos *dataframes* de treinamento e de teste. O processo escolhido foi o *TF-IDF (Term Frequency - Inverse Document Frequency)*, que basicamente consiste em calcular o peso, ou importância, de uma dada palavra (*token composto por n-words*) no contexto de um *dataframe* construído a partir de vários documentos (*noticias*) como já visto na Seção 2.3. Separando o conceito em duas partes, temos primeiro o *TF (Term Frequency)*, que define a frequência de aparição de um determinado termo em um documento. Essa frequência pode ser definida de várias formas:

- Número de vezes que uma palavra aparece em um documento
- Frequência de um termo ajustado para o tamanho do documento (somatório de ocorrência dividido pelo número total de palavras em um documento)
- Frequência em escala logarítmica, por exemplo: $log(1 + n)$ em que n é a contagem de ocorrências de um termo
- Frequência booleana (1 se houver ocorrência do termo, ou 0 se não houver ocorrência do termo)

Já o termo *IDF (Inverse Document Frequency)* diz respeito ao quão comum (ou incomum) é uma palavra dentre o todos os documentos do corpus. Ele é calculado em escala logarítmica como mostra a Equação 3.1, onde t é o termo (palavra) que queremos saber o quão comum é dentre um número N de documentos d em um corpus D. O denominador é simplesmente o número de documentos em que aparece o termo t.

[6] https://www.nltk.org/

$$idf(t, D) = \log\left(\frac{N}{count(d \in D : t \in d)}\right) \tag{3.1}$$

Para evitar a situação de divisão por 0 caso um termo não apareça no corpus, bibliotecas Python como o *scikit-learn* implementam a Equação 3.1 da forma mostrada na Equação 3.2.

$$idf(t) = \log\left(\frac{1+N}{1+df(t)}\right) \tag{3.2}$$

Em que $df(t)$ é o número de documentos em que aparece o termo t. Com o *IDF* conseguimos minimizar a relevância de termos frequentes e valorizar termos menos frequentes. Em suma, o *TF-IDF* mostra que a importância de um termo é inversamente proporcional à frequência com a qual esse termo aparece em todos os documentos. Ou seja, *TF* nos dá informação sobre o quão frequente um termo aparece em um documento e *IDF* nos dá informação sobre a raridade relativa de um termo no contexto de uma coleção de documentos. Multiplicando esses dois valores temos o valor de *TF-IDF*, como mostra a Equação 3.3.

$$tfidf(t, d, D) = tf(t, D) \cdot idf(t, D) = tf(t, D) \cdot \log\left(\frac{1+N}{1+df(t)}\right) \tag{3.3}$$

Quanto maior o valor de *TF-IDF* mais importante ou relevante é um termo, sendo que se um termo se torna menos relevante, seu valor de *TF-IDF* se aproxima de 0. Cada token $t_j, j = 0, 1, 2, ..., T-1$, do vocabulário de tamanho T, pertence ao vetor representado na Equação 3.4.

$$vocab = [t_0, t_1, ..., t_j, ..., t_{T-1}] \tag{3.4}$$

Ao final da etapa de *"tokenização"* obtemos *dataframes* representados por matrizes de dados como a ilustrada na Figura 3.4, em que temos os *tokens* associados a um peso atribuído pelo cálculo do *TF-IDF*.

Utilizar o vocabulário inteiro pode se tornar bastante custoso para o processamento do modelo que desejamos obter. Com isso em mente, um truncamento do vocabulário foi realizado após a *"tokenização"*, de forma a obtermos *dataframes* com uma dimensionalidade inferior, de 20.000 (vinte mil) *features*, composto por aqueles *features* que foram mais relevantes para a modelagem e descartando aqueles que foram pouco relevantes. O valor de 20.000 (vinte mil) foi escolhido pois observou-se que valores maiores do que esse levavam a uma porcentagem maior de *tokens* com pesos nulos, que não contribuíam para a acurácia do modelo. Já número menores do que esse apenas reduziam a acurácia do modelo por conta do deficit de *tokens* necessários para o treinamento.

Com *dataframes* compostos por palavras e pesos relacionados, é possível treinar os algoritmos de aprendizado de máquina, em que cada um produziu como saída um modelo capaz de classificar notícias como verdadeiras ou falsas.

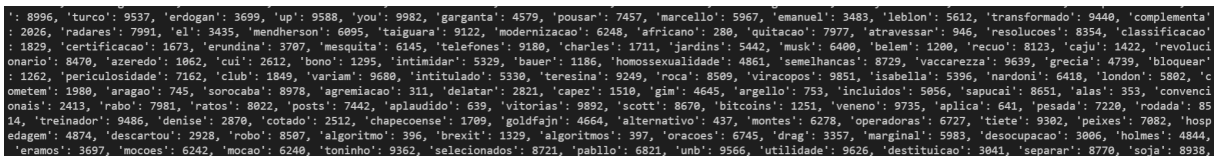

Figura 3.4: Exemplo de *dataframe* pós etapa de *"tokenização"*.

3.3 Treinamento e inferência

Com os dados prontos e organizados, é possível então iniciar o processo mais custoso computacionalmente, o treinamento. O treinamento consiste em ajustar um modelo a partir de uma extensiva e repetitiva tarefa de aprendizagem, em que o algoritmo em questão por vezes sabe as classes de cada entrada (aprendizado supervisionado) ou não conhece nenhuma classe e tenta aprender as diferenças entre as entradas por si (aprendizado não supervisionado). Em resumo, o treinamento consiste em adaptar um modelo utilizando-se de dados passados, para que assim ele consiga processar e inferir classes para dados futuros. Já na inferência, é fornecido como entrada os dados de teste para um modelo previamente treinado, com o objetivo de obter resultados de previsões futuras que nos permitirá calcular métricas de performance, como visto na Seção 2.2. Além de métricas, a inferência permite a observação da classificação de dados futuros a partir de experiências passadas.

3.4 Extração de Conhecimento e Visualização de Features

Após o treinamento dos modelos e a obtenção de métricas de desempenho, será utilizado a técnica de redução de dimensionalidade *Distributed Stochastic Neighbor Embedding (t-SNE)* para obter informações importantes das *features* extraídas dos *datasets* e, assim, extrair conhecimento. No contexto de aprendizado de máquina, extrair conhecimento consiste no processo de aplicar os resultados obtidos com as saídas dos modelos, em problemas práticos que buscam uma solução. No caso de algoritmos de classificação em que se analisam textos, como notícias, o objetivo em suma é entender quais padrões linguísticos o algoritmo em questão está levando em consideração para realizar sua classificação. Esses padrões podem ser úteis para entender a formação dos textos e identificar caminhos para se resolver uma tarefa. No caso da classificação de notícias, o objetivo é entender um pouco mais a estrutura que diferencia as notícias verdadeiras das falsas e conseguir pistas que possam melhorar os algoritmos de aprendizagem, além de, talvez identificar padrões de escrita em notícias falsas brasileiras.

O algoritmo *t-SNE* será alimentado com *features* representados por vetores que predizem o contexto de uma palavra, dado a própria palavra, o modelo então é treinado com *skip-grams*, que são *n-grams* (agrupamentos de n *tokens*) que permitem palavras serem ignoradas. O contexto de uma palavra pode ser representado assim através de um agrupamento de pares de *skip-grams* do

tipo:

$$(palavra\_alvo, palavra\_contexto) \tag{3.5}$$

Onde "palavra_contexto" aparece no contexto da vizinhança de "palavra_alvo".

Essa técnica é conhecida como *word2vec* e, diferente da técnica de *"tokenização" TF-IDF*, ela tem como objetivo a representação de palavras de um *corpus* por vetores únicos que as identificam por contextos, dessa forma conseguimos visualizar palavras que com frequência aparecem em um mesmo contexto por conta da proximidade dos vetores que as representam. Com isso, aliada ao algoritmo *t-SNE*, é possível plotar visualizações gráficas semelhantes às da Figura 3.5, que representa um corpus de imagens de dígitos manuscritos organizados em um gráfico de duas dimensões, assim conseguimos observar que as entradas representadas por vetores únicos ficam agrupadas por proximidade vetorial, em que cada grupo representa um algarismo de 0 a 5.

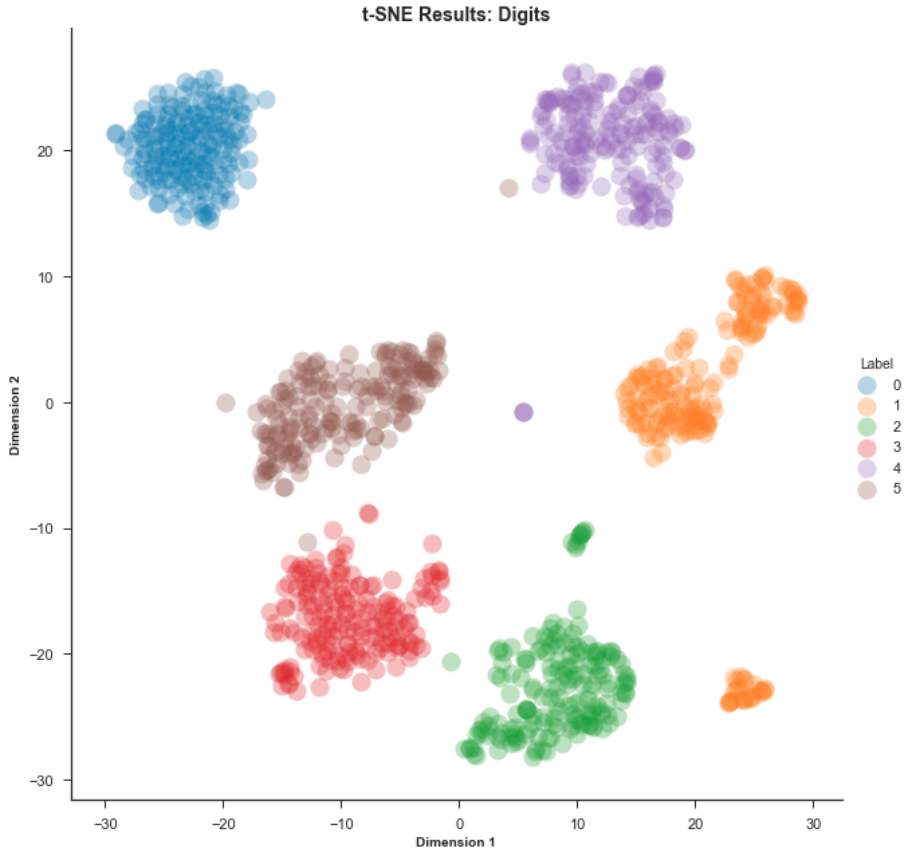

Figura 3.5: Exemplo de gráfico t-SNE com dimensões vetoriais reduzidas para agrupar imagens de dígitos manuscritos. - Obtido em https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1, acessado em 19/04/2022.

4

Resultados

Tendo em mãos as técnicas de implementação discutidas no Capítulo 3, torna-se possível obter os resultados de desempenho e compara-los para entender qual modelo teve o melhor desempenho na tarefa de classificação de notícias. É importante notar que o objetivo do estudo é discutir o desempenho de modelos de classificação testados com *datasets* de notícias que foram escritas posteriormente às notícias do treinamento, notícias essas que nunca foram "vistas" pelo modelo treinado. Podemos então fazer a seguinte pergunta: seria possível que um modelo treinado com um *dataset* confiável consiga classificar notícias em casos reais de compartilhamentos em redes sociais? Dessa forma conseguiremos formular hipóteses para responder a próxima pergunta: um modelo de classificação de notícias leva mais em consideração o conteúdo da notícia em si ou o contexto com o qual suas palavras se encontram?

4.1 Modelo Passivo Agressivo

O modelo selecionado como ênfase do estudo foi o *Passivo Agressivo* (descrito na Seção 2). Foram então realizados vários treinamentos com ele para medir a acurácia e *F1score* com diferentes quantidades de *features*, obtendo-se os valores mostrados na Tabela 4.1. Para cada treinamento com um número diferente de *features* foi também calculado a densidade (porcentagem de *features* com peso não-nulo para o classificador), para que assim fosse possível selecionar o melhor modelo resultante. Outra razão que motivou a redução de *features* (truncamento) foi a possibilidade de reduzir a complexidade e tempo de processamento do algoritmo.

Tabela 4.1: Tabela com valores de densidade (porcentagem de *features* com peso do vetor de classificação não-nulo) para o algoritmo Passivo Agressivo.

Nº de features	Acurácia	F1Score	Densidade
10000	91%	0,914	99%
15000	91%	0,914	99%
20000	92%	0,920	98%
25000	91%	0,916	97%
30000	91%	0,916	96%

Como temos o objetivo de obter as melhores métricas possíveis, selecionamos a quantidade de *features* que nos deu a melhor acurácia e *F1score* possíveis, que foi o valor de 20.000 (vinte mil) *features*. O treinamento do modelo *Passivo Agressivo* foi extremamente rápido, sendo o processo todo executado em 0,2 segundos. Realizando a etapa de teste obtivemos então a matriz de confusão exibida na Figura 4.1 e as métricas na Tabela 4.2.

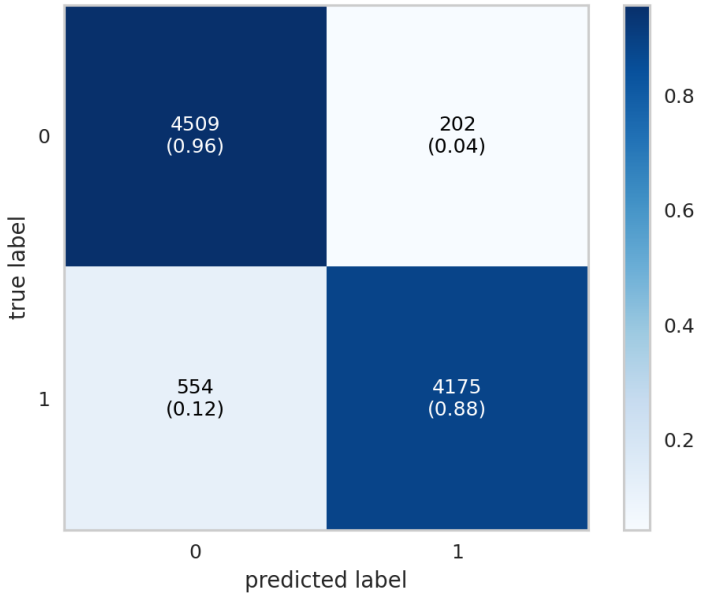

Figura 4.1: Matriz de confusão do teste com o algoritmo Passivo Agressivo. Para notícias falsas *true label* é igual a 0 e para notícias verdadeiras é igual a 1. Essa matriz mostra que 96% das notícias falsas foram classificadas como falsas e 88% das verdadeiras foram classificadas como verdadeiras.

Tabela 4.2: Métricas obtidas no teste do algoritmo Passivo Agressivo utilizando modelo treinado com 20.000 *features*.

Acurácia	Precisão	Recall	F1score
92%	95%	88%	0,920

Analisando a matriz de confusão da Figura 4.1, podemos notar que 96% das notícias que realmente eram falsas foram classificadas como falsas e 88% das notícias verdadeiras foram classificadas como verdadeiras. Isso nos mostra que o algoritmo conseguiu identificar com mais facilidade notícias falsas. Além disso, as métricas de desempenho foram bastante satisfatórias, destacando aqui a acurácia de 92% e o *F1score* de valor 0,917.

Nota-se que para o valor de 20.000 (vinte mil) *features* obtivemos uma densidade, representada como a quantidade de *features*, com peso não-nulo para o classificador, de 98% (ver Tabela 4.1), isso nos mostra que apenas 2% das palavras escolhidas não contribuem para o modelo de classificação. Essa redução de complexidade no treinamento do modelo diminui a complexidade da tarefa e garante que a maioria dos *features* utilizados realmente contribuam para o ajuste do modelo. A Figura 4.2 mostra a dimensionalidade do treinamento e a distribuição de pesos. É importante frisar que pesos (coeficiente do vetor w) positivos contam mais para a tarefa de classificação positiva (verdadeira) e pesos negativos contam mais para a tarefa de classificação negativa (falsa) do algoritmo. Em ambos gráficos da Figura 4.2 no eixo y temos o valor do peso das *features*. Já no gráfico da esquerda o eixo x representa cada *feature* (dentro dos 20.000) e no gráfico da direita o eixo x representa um somatório de quantidade de *features* para cada valor de peso.

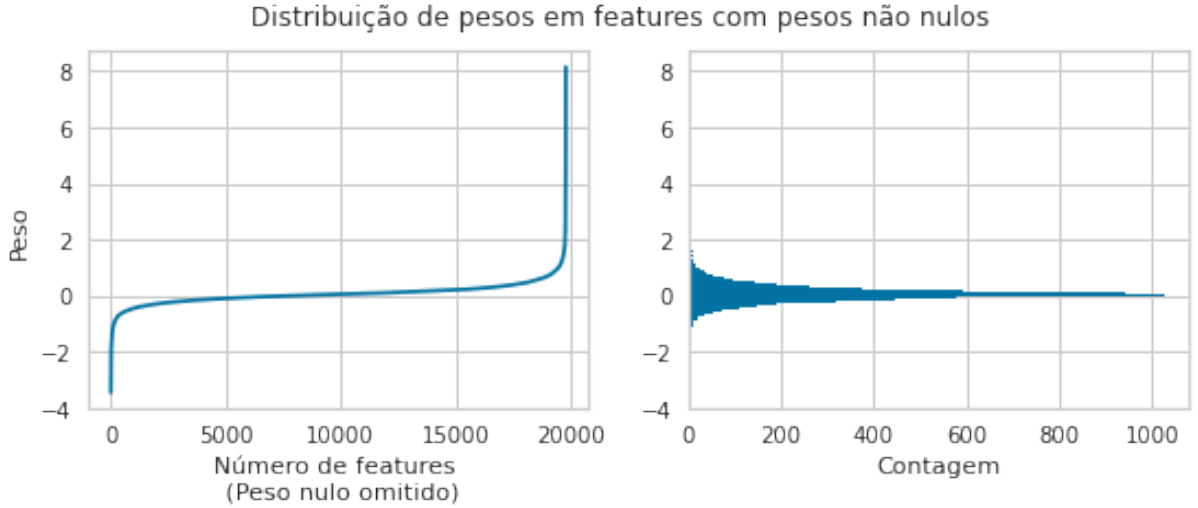

Figura 4.2: Distribuição de pesos em *features* com pesos não nulos no treinamento do algoritmo Passivo Agressivo com 20.000. Em ambos gráficos no eixo y temos o valor do peso das *features*. Já no gráfico da esquerda o eixo x representa cada *feature* (dentro dos 20.000) e no gráfico da direita o eixo x representa um somatório de quantidade de *features* para cada valor de peso.

Para começar a entender as informações extraídas do *t-SNE*, pode ser útil visualizar alguns *tokens* relevantes na classificação de notícias falsas e notícias verdadeiras, que são apresentadas nas Tabelas 4.3 e 4.4, respectivamente. Nestas tabelas, são apresentados os 20 primeiros *tokens* com os menores e maiores pesos (coeficientes do vetor de classificação) retirados do *corpus* de treinamento.

Tabela 4.3: Ranking de palavras (tokens) de notícias falsas e seus respectivos pesos (coeficientes do vetor de classificação) no treinamento do modelo PA

Token	Peso
hoje	-3.418
atraves	-3.345
podera	-2.837
friboi	-2.651
reais	-2.479
dilma	-2.440
materia	-2.270
folha	-2.090
leia	-1.989
estao	-1.980
alegou	-1.913
barragens	-1.816
brasil	-1.801
esposa	-1.796
palavras	-1.784
informacoes	-1.775
urgente	-1.747
antagonista	-1.746
possui	-1.700
revista	-1.699

Tabela 4.4: Ranking de palavras (tokens) de notícias verdadeiras e seus respectivos pesos (coeficientes do vetor de classificação) no treinamento do modelo PA.

Token	Peso
nesta	7.658
segundo	6.545
afirmou	4.557
sobre	3.723
entao	3.316
tribunal	2.999
desta	2.906
diz	2.762
falou	2.628
primeira	2.592
passado	2.573
afirma	2.551
sabado	2.519
regiao	2.499
entanto	2.447
meio	2.416
apos	2.312
passada	2.285
ano	2.243
leitores	2.231

Observando os maiores e menores pesos dos *tokens* na tarefa de classificação, conseguimos notar algumas informações úteis. Muitas notícias falsas tiveram pesos bastantes significativos em referências diretas a figuras políticas e organizações industriais ou de mídia, como por exemplo: “dilma”, “friboi”, “bolsonaro” e “folha”. No caso das notícias verdadeiras, existe uma importância dada a contrações e preposições como “nesta” e “segundo”. O motivo disso pode ser a preocupação com textos escritos de maneira formal, característica essa de textos jornalísticos. Talvez o que influencia uma notícia a ser classificada como falsa ou verdadeira não são as palavras que compõem o texto mas o contexto onde elas aparecem. Desta forma, é necessário fazer uma análise do todo em volta de uma palavra específica. Por exemplo, quais outras palavras geralmente estão próximas do *token* “dilma” dado um documento que retrata uma notícia falsa? É importante frisar que esses pesos vetoriais do classificador foram calculados baseados em *tokens* extraídos com a técnica *TF-IDF*, que leva em consideração a relevância de uma palavra dado o escopo de um *corpus* inteiro de documentos. Então não é o fato de palavras apenas aparecerem múltiplas vezes que as tornam relevantes. Por fim, palavras isoladas pode nos dar pistas, mas não explicita o contexto onde essas palavras aparecem. Para isso, é necessário fazer uma análise de proximidade de palavras utilizando técnicas como o *t-SNE*.

Outro ponto importante a ser considerado é o fato dos modelos terem sido treinados com notícias de um período de tempo delimitado entre os anos de 2015 a 2018 e as notícias de teste englobarem notícias de 2015 até 2021. Isto nos leva a seguinte pergunta: um modelo de classificação de notícias leva mais em consideração a contextualidade das palavras para um dado momento do tempo ou padrões lexicais que possam existir em notícias verdadeiras e em notícias falsas? E, se existem padrões lexicais nos textos, esses padrões mudam com o passar do tempo e da forma de escrita humana? Com que velocidade? Respostas a essas perguntas podem ser cruciais para se construir algoritmos de aprendizagem que cada vez mais consigam detectar com precisão desinformação nas redes sociais.

4.2 Comparação com outros modelos

Para critérios de comparação, foram realizados testes com os seguintes modelos de aprendizagem:

- Classificador de Árvore de Decisão: consiste na função de particionar recursivamente um conjunto de treinamento até que cada subconjunto obtido desse particionamento contenha casos de uma única classe (BARBOSA; CARNEIRO; TAVARES, 2012). Árvores de decisão tomam como entrada um documento descrito por um conjunto de *features* e retorna uma decisão (nesse caso, verdadeira ou falsa).
- Classificador de Floresta Aleatória: parte do princípio de se ter um número grande de árvores de decisão trabalhando em agrupamento, de forma que cada árvore de decisão representa um modelo de classificação que irá produzir uma saída de predição (BREIMAN, 2001). Assim, o algoritmo fica encarregado de escolher uma das possíveis predições, levando em consideração a que possui mais "votos" de cada árvore de decisão.
- SVM (Support Vector Machine): o objetivo de uma SVM é encontrar um hiperplano em um espaço dimensional p (em que p representa a quantidade de *features*) que classifique dados de forma distinta (LORENA; CARVALHO, 2007). Existem muitos hiperplanos possíveis para separar duas classes de dados. O objetivo do SVM é encontrar o plano que possui a maior distância entre os pontos de dados no espaço para as duas classes.
- Regressão Logística: utiliza funções lineares para prever valores em uma classificação binária ou multinomial. Esse algoritmo mede a relação entre uma variável dependente e uma ou mais variáveis independentes, estimando as probabilidades usando uma função logística (DREISEITL; OHNO-MACHADO, 2002).

O treinamento acontecerá de forma em que possamos analisar o desempenho do classificado Passivo Agressivo e comparar seu desempenho com outros modelos de classificação. Para a etapa de treinamento temos na Tabela 4.5 os parâmetros de treinamento, para todos os modelos que serão analisados.

Tabela 4.5: Tabela com parâmetros de treinamento para os modelos analisados.

Dataset	Nº de features	Nº de epochs (apenas para o PA)	Nº de notícias verdadeiras	Nº de notí
Fake.Br Corpus	20.000	1.000	3.600	3.6

Durante o treinamento, o número de iterações, conhecido como *epochs*, consiste na quantidade de vezes que um algoritmo irá repetir a etapa de treinamento, ajustando seus pesos, com o objetivo de melhorar o seu desempenho. A partir do momento em que essas repetições deixam de agregar melhorias significativas no desempenho, o algoritmo termina o treinamento. Para a etapa de inferência de dados temos os parâmetros descritos na Tabela 4.6.

Tabela 4.6: Tabela com parâmetros de inferência para os modelos analisados.

Dataset	Nº de notícias verdadeiras	Nº de notícias falsas
G1 (true) e Boatos.org (fake)	4.729	4.711

A Tabela 4.7 mostra os valores de Acurácia, Precisão, Recall e F1score para cada modelo.

Tabela 4.7: Métricas obtidas para treinamentos com outros modelos, utilizando o mesmo *dataset* de treinamento, 20.000 *features* e *TF-IDF* como técnica de *"tokenização"*.

Modelo	Acurácia	Precisão	Recall	F1score
PassiveAggressiveClassifier	**92%**	95%	**88%**	**0,917**
DecisionTreeClassifier	85%	90%	79%	0,844
RandomForestClassifier	79%	97%	59%	0,741
SVM (LinearSVC)	85%	**97%**	73%	0,839
LogisticRegression	84%	97%	71%	0,827

Observando a Tabela 4.7, conseguimos notar que o desempenho do método Passivo Agressivo foi melhor do que o dos outros modelos testados. A única exceção é na métrica precisão, que se mede o número de verdadeiros positivos dentre todas as predições positivas. Essa informação pode ser útil para entender o funcionamento de outros algoritmos, como o SVC que trabalha de forma semelhante ao Passivo Agressivo.

Podemos também fazer uma comparação do algoritmo Passivo Agressivo, um algoritmo do tipo *online-learning*, com o outro, o Classificador de Floresta Aleatória, que é um algoritmo do tipo árvore de decisão. Como podemos notar, na Tabela 4.7, o Classificador de Floresta Aleatória teve o pior F1score. Apesar disso, na matriz de confusão (Figura 4.3), podemos notar que o algoritmo teve um excelente desempenho para notícias falsas (98%), mas teve desempenho razoável apenas na classificação de notícias verdadeiras (60%). O que pode servir de motivação para a utilização

do classificador PA para classificar notícias.

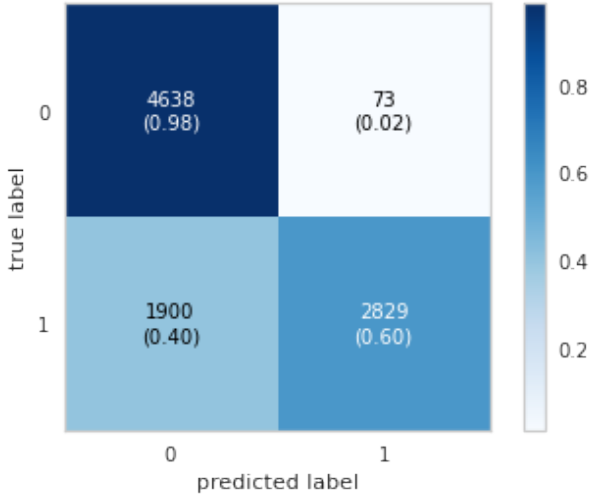

Figura 4.3: Matriz de confusão do Classificador de Floresta Aleatória. Para notícias falsas *true label* é igual a 0 e para notícias verdadeiras é igual a 1.

4.3 Estrutura das notícias

Como já discutido, apenas analisar palavras isoladas pode ser útil, mas não nos dá uma ideia de contexto e proximidade de palavras para termos uma noção da lógica utilizada pelo algoritmo na hora de fazer uma classificação. Para conseguir visualizar graficamente e entender o contexto em que as palavras se encontram no *dataset* de treinamento e como essas palavras se organizam em *clusters*, utilizamos a técnica *Word2Vec* com o algoritmo não supervisionado *t-SNE*. Primeiramente, aplicamos a técnica *t-SNE* no *dataset* de *tokens* obtidos pelo *Word2Vec*, em que os pesos dos *tokens* representam cada palavra por um vetor que dita a proximidade dela com outras palavras em um contexto. Utilizando a redução de dimensionalidade do *t-SNE* para plotar tais informações em duas dimensões conseguimos então visualizar grupos de palavras (*clusters*) que podem nos indicar contextos cruciais para a tarefa de classificação. Sendo assim, uma mesma palavra, dado diferentes contextos, pode pesar mais ou menos para a classificação de uma notícia como verdadeira ou falsa. O primeiro gráfico obtido com essa técnica foi a Figura 4.4. Nela conseguimos visualizar pontos verdes e azuis que representam *tokens* classificados com peso de classificação negativo (falso) e positivo (verdadeiro), respectivamente. Nos resultados obtidos com o *t-SNE*, inicialmente, não foi possível observar visualmente com facilidade agrupamentos significativos. Dessa forma, foi realizado uma análise mais detalhada de agrupamentos por palavras em pontos específicos. Cada ponto nesse gráfico representa uma *feature* em um espaço dimensional de 2 dimensões, em que a estrutura local foi preservada como possível, ou seja, as proximidades entre *features* calculadas pelo *word2vec* em um espaço de altas dimensões foi trazida para um espaço de poucas dimensões, de forma que é possível observar as palavras próximas por contexto, como mais detalhe, como mostram as Figuras 4.5 e 4.6.

Um detalhe importante de se notar são os pontos que se agrupam. Neles nós podemos observar *clusters* de palavras que indicam algum contexto. Isso fica claro ao se observar outros gráficos t-SNE em que selecionamos alguns *tokens*. Na Figura 4.5 podemos observar alguns agrupamentos de palavras com pesos positivos que influenciam mais para a classificação positiva no modelo. Da mesma forma, na Figura 4.6 temos agrupamentos de palavras com pesos negativos que influenciam mais para a classificação negativa no modelo. Comparando ambas imagens, conseguimos notar que existem muito mais *tokens* positivos plotados no gráficos do que *tokens* negativos, uma prova disso é que foi necessário limitar a plotagem para apenas o caso de *tokens* que repetem pelo menos 500 (quinhentas) vezes no *corpus*, no caso de *tokens* positivos. Para *tokens* negativos, limitamos a plotagem para *tokens* que repetem pelo menos 100 (cem) vezes.

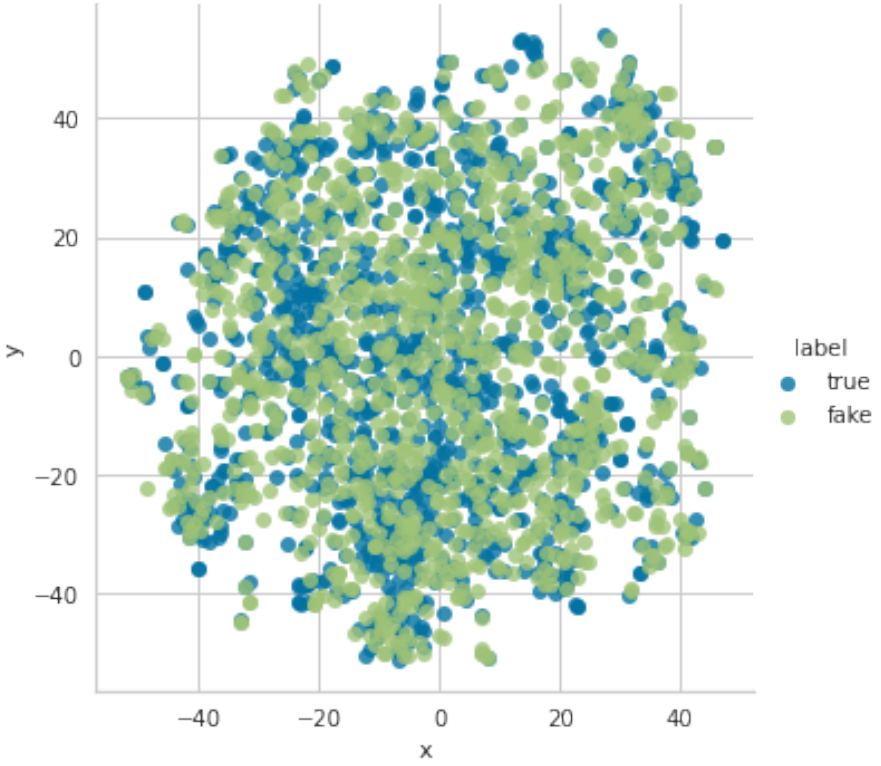

Figura 4.4: t-SNE representando o *dataset* de treinamento com classes *true* e *fake*.

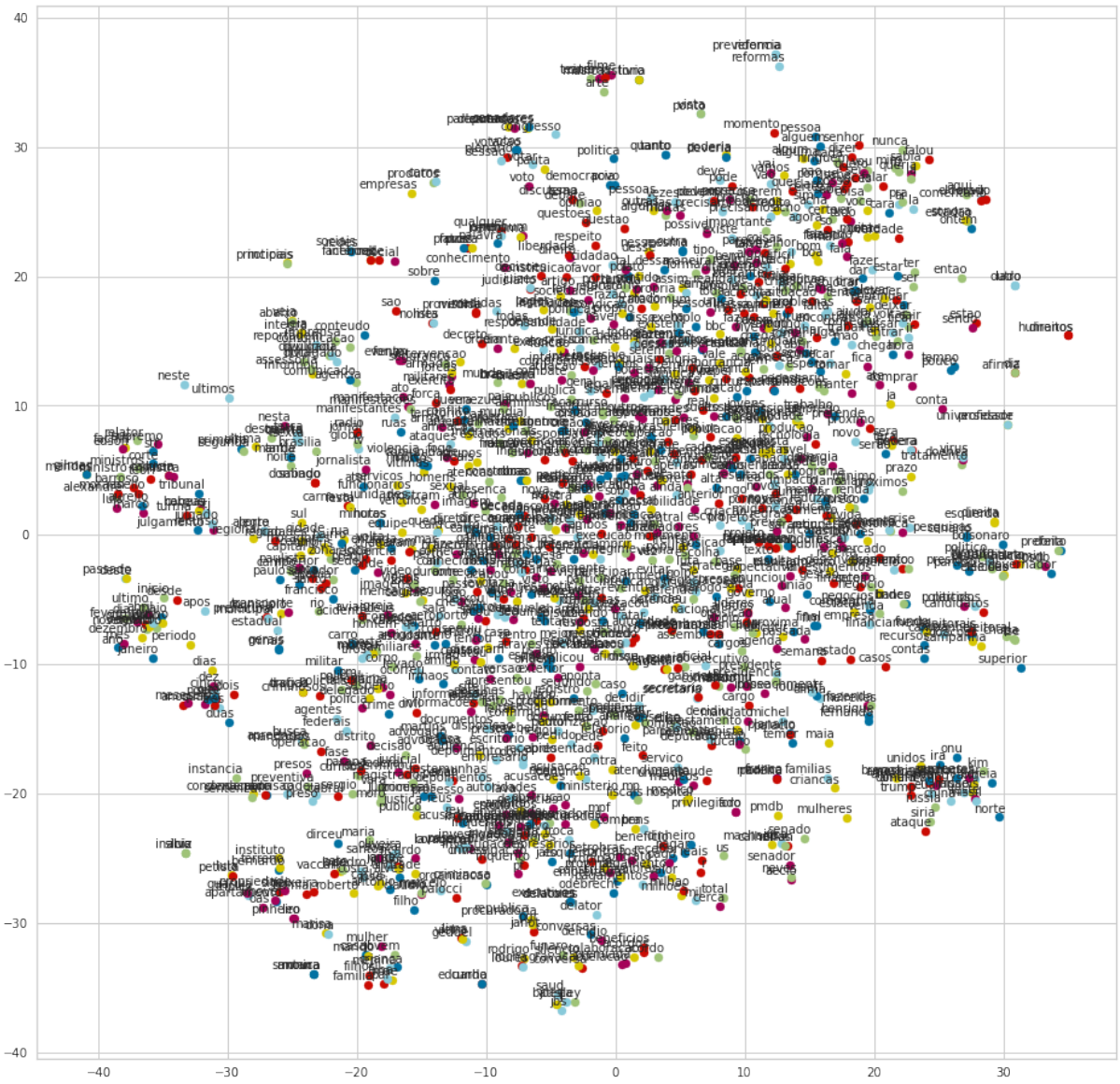

Figura 4.5: t-SNE representando o *dataset* de treinamento com apenas a classe *true*.

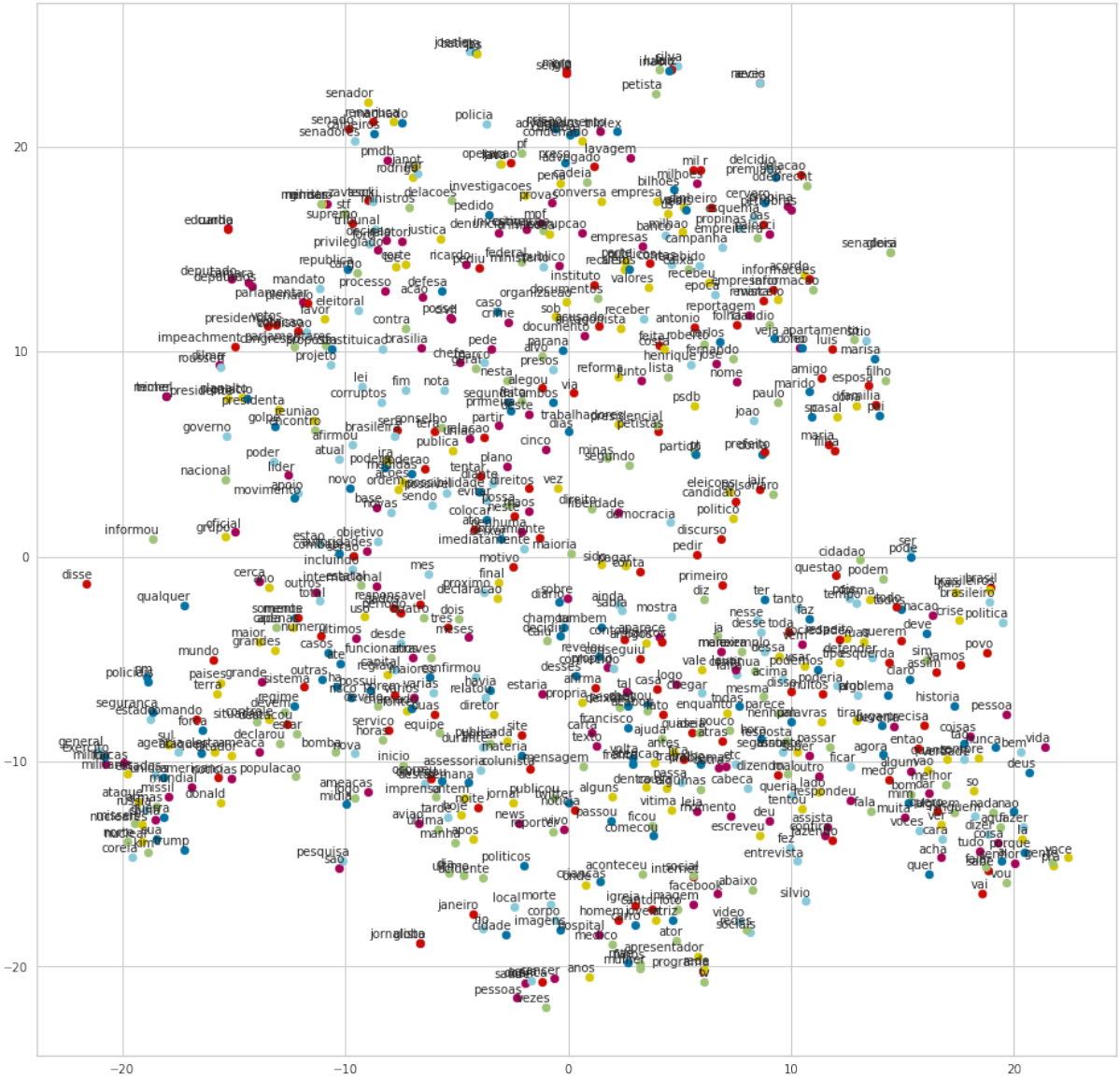

Figura 4.6: t-SNE representando o *dataset* de treinamento com apenas a classe *fake*.

Com o objetivo de ter uma melhor observação, podemos selecionar algumas palavras e analisar quais outras palavras possuem proximidade dela, dado o caso de notícias verdadeiras e notícias falsas. Com a utilização da biblioteca *Python sklearn*, através do cálculo de similaridade por cosseno conseguimos visualizar as 10 (dez) palavras mais próximas, em uma escala de 0 (zero) a 1 (um), para os exemplos “governo”, “eleicoes” e “politica”:

Tabela 4.8: Proximidades do *token* **"governo"** no caso de notícias **verdadeiras**. O valor da proximidade varia entre 0 e 1, sendo que palavras que obtiveram valores próximos de 1 indicam uma maior probabilidade de ser encontrada próxima a palavra examinada.

Token	Proximidade
economica	0,733
articular	0,667
progredir	0,637
pluripartidarismo	0,635
saida	0,633
ruralistas	0,625
planalto	0,619
aprovacao	0,616
crise	0,613
plano	0,608

Tabela 4.9: Proximidades do *token* **"governo"** no caso de notícias **falsas**. O valor da proximidade varia entre 0 e 1, sendo que palavras que obtiveram valores próximos de 1 indicam uma maior probabilidade de ser encontrada próxima a palavra examinada.

Token	Proximidade
demitir	0,958
enterrar	0,924
arminio	0,919
aliar	0,907
interino	0,906
aprove	0,905
tampao	0,901
cnt	0,901
imbassahy	0,897
acabando	0,896

Tabela 4.10: Proximidades do *token* **"eleicoes"** no caso de notícias **verdadeiras**. O valor da proximidade varia entre 0 e 1, sendo que palavras que obtiveram valores próximos de 1 indicam uma maior probabilidade de ser encontrada próxima a palavra examinada.

Token	Proximidade
eleicao	0,942
urnas	0,938
disputa	0,873
pleito	0,866
presidencial	0,854
candidaturas	0,842
presidenciais	0,841
eleitores	0,835
concorrer	0,834
disputar	0,829

Tabela 4.11: Proximidades do *token* **"eleicoes"** no caso de notícias **falsas**. O valor da proximidade varia entre 0 e 1, sendo que palavras que obtiveram valores próximos de 1 indicam uma maior probabilidade de ser encontrada próxima a palavra examinada.

Token	Proximidade
presidenciais	0,991
rejeicao	0,987
reverter	0,987
presidenta	0,985
votar	0,983
eleicao	0,983
salvar	0,982
corrupto	0,981
discurso	0,981
questionou	0,980

Tabela 4.12: Proximidades do *token* **"politica"** no caso de notícias **verdadeiras**. O valor da proximidade varia entre 0 e 1, sendo que palavras que obtiveram valores próximos de 1 indicam uma maior probabilidade de ser encontrada próxima a palavra examinada.

Token	Proximidade
politico	0,820
lideranca	0,774
legado	0,766
democracia	0,759
defendemos	0,749
liberal	0,747
esquerda	0,732
redemocratizacao	0,732
postura	0,728
papel	0,728

Tabela 4.13: Proximidades do *token* **"politica"** no caso de notícias **falsas**. O valor da proximidade varia entre 0 e 1, sendo que palavras que obtiveram valores próximos de 1 indicam uma maior probabilidade de ser encontrada próxima a palavra examinada.

Token	Proximidade
expectativa	0,983
porra	0,980
impedimento	0,977
arrogancia	0,977
volte	0,976
selecionada	0,975
reverter	0,975
disputar	0,975
democracia	0,973
quer	0,973

Algumas informações extraídas dessa análise podem ser úteis para entender possíveis padrões nas notícias falsas, com o objetivo de facilitar sua detecção:

- No caso do corpus composto de notícias falsas, os *tokens* relevantes repetem menos vezes do que no caso das notícias verdadeiras. Isso pode significar uma diferenciação crucial no estilo textual de escrita, em que textos de cunho jornalísticos, que visam informar o leitor, tendem a conter palavras relevantes para a classificação que ocorrem com maior frequência mesmo em diferentes notícias;
- Os *tokens* de notícias falsas geralmente possuem grande proximidade (valor alto de similari-

dade por cosseno) com outros *tokens*, sendo que nos exemplos expostos observamos valores acima de 0,9 para o grau de similaridade em sua grande maioria;

- Em notícias falsas foi observado *tokens* muito próximos de adjetivos, advérbios e até xingamentos, isso faz sentido com o senso comum de que notícias falsas geralmente tentam partir para uma ressonância emocional para com o leitor;
- Para os mesmos *tokens* existe muita diferença dos *tokens* mais próximos no caso de notícias verdadeiras e de notícias falsas, o que pode indicar uma diferença muito grande nas narrativas para um mesmo assunto.

Referências

ALPAYDIN, Ethem. **Introduction to Machine Learning**. [S.l.]: The MIT Press, 2004. ISBN 0-262-01211-1.

AVAAZ. **O Brasil está sofrendo uma infodemia de Covid-19**. [S.l.: s.n.], mai. 2020. Publicado online. Acessado em 10.08.2021. Disponível em: <<https://secure.avaaz.org/campaign/po/brasil_infodemia_coronavirus/>>.

BARBOSA, J.M.; CARNEIRO, T.G.S.; TAVARES, A.L. Métodos de Classificação por Árvores de Decisão. **PPGCC - Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Ouro Preto (UFOP)**, 2012. Disponível em: <<http://www.decom.ufop.br/menotti/paa111/files/PCC104-111-ars-11.1-JulianaMoreiraBarbosa.pdf>>.

BIRD, Steven; KLEIN, Ewan; LOPER, Edward. **Natural Language Processing with Python**. 1st. [S.l.]: O'Reilly Media, Inc, 2009. ISBN 978-0-596-51649-9.

BOVET, Alexandre; MAKSE, Hernán A. Influence of fake news in Twitter during the 2016 US presidential election. **Nature Communications**, v. 10, 2019. ISSN 2041-17234. DOI: <10.1038/s41467-018-07761-2>. Disponível em: <<https://doi.org/10.1038/s41467-018-07761-2>>.

BREIMAN, Leo. Random Forests. **Statistics Department, University of California**, 2001. Disponível em: <<https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf>>.

CGI.BR/NIC.BR. **TIC DOMICÍLIOS 2019**. [S.l.: s.n.], mai. 2020. Publicado online. Acessado em 10.08.2021. Disponível em: <<https://cetic.br/media/analises/tic_domicilios_2019_coletiva_imprensa.pdf>>.

CINELLI, Matteo et al. The COVID-19 social media infodemic. **Nature Scientific Reports**, p. 10, 2020. DOI: <10.1038/s41598-020-73510-5>. Disponível em: <<https://www.nature.com/articles/s41598-020-73510-5.pdf>>.

CRAMMER, Koby et al. Online Passive-Aggressive Algorithms. **Journal of Machine Learning Research 7**, School of Computer Science e Engineering - The Hebrew University, jun. 2006.

DATASENADO. **Mais de 80% dos brasileiros acreditam que redes sociais influenciam muito a opinião das pessoas**. [S.l.: s.n.], dez. 2019. Disponível em: <<https://www12.senado.leg.br/institucional/datasenado/materias/pesquisas/mais-de-80-dos-brasileiros-acreditam-que-redes-sociais-influenciam-muito-a-opiniao-das-pessoas>>.

DREISEITL, Stephan; OHNO-MACHADO, Lucila. Logistic regression and artificial neural network classification models: a methodology review. **Journal of Biomedical Informatics**, v. 35, n. 5, p. 352–359, 2002. ISSN 1532-0464. DOI: <https://doi.org/10.1016/S1532-0464(03)00034-0>. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1532046403000340>>.

GALLAGHER, Kevin. **THE SOCIAL MEDIA DEMOGRAPHICS REPORT: Differences in age, gender, and income at the top platforms**. [S.l.: s.n.], ago. 2017. Publicado online. Acessado em 10.08.2021. Disponível em: <<https://www.businessinsider.com/the-social-media-demographics-report-2017-8>>.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>.

GUPTA, Saloni; MEEL, Priyanka. Fake News Detection Using Passive-Aggressive Classifier. **Inventive Communication and Computational Technologies**, p. 155–164, 2021. DOI: <10.1007/978-981-15-7345-3_13>. Disponível em: <<https://link.springer.com/chapter/10.1007%5C%2F978-981-15-7345-3_13>>.

JR., Reynaldo Turollo. **Os (poucos) indícios de financiamento de fake news apresentados pela CPI**. [S.l.: s.n.], out. 2021. Published online. Downloaded 04.10.2022. Disponível em: <<https://veja.abril.com.br/coluna/maquiavel/os-poucos-indicios-de-financiamento-de-fake-news-apresentados-pela-cpi/>>.

LORENA, Ana Carolina; CARVALHO, André C. P. L. F. de. Uma Introdução às Support Vector Machines. **Revista de Informática Teórica e Aplicada**, v. 14, 2007. Disponível em: <<https://www.seer.ufrgs.br/index.php/rita/article/view/rita_v14_n2_p43-67>>.

MAATEN, L.J.P. van der; HINTON, G.E. Visualizing High-Dimensional Data Using t-SNE. **Journal of Machine Learning Research**, 2008. ISSN 2579-2605. Disponível em: <<https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf>>.

NADKARNI, Prakash M; OHNO-MACHADO, Lucila; CHAPMAN, Wendy W. Natural language processing: an introduction. **Journal of the American Medical Informatics Association**, v. 18, n. 5, p. 544–551, set. 2011. ISSN 1067-5027. DOI: <10.1136/amiajnl-2011-000464>. eprint: <https://academic.oup.com/jamia/article-pdf/18/5/544/5962687/18-5-544.pdf>. Disponível em: <<https://doi.org/10.1136/amiajnl-2011-000464>>.

NEWMAN, Nic et al. **Reuters Institute Digital News Report 2021**. [S.l.: s.n.], 2021. Publicado online. Acessado em 10.08.2021. Disponível em: <<https://reutersinstitute.politics.ox.ac.uk/sites/default/files/2021-06/Digital_News_Report_2021_FINAL.pdf>>.

PINHEIRO, Nina. **Introdução ao Processamento de Linguagem Natural — Natural Language Processing(NLP)**. [S.l.: s.n.], 2021. Publicado online. Acessado em 10.08.2021. Disponível em: <<https://medium.com/data-hackers/introdu%5C%C3%5C%A7%5C%C3%5C%A3o-ao-processamento-de-linguagem-natural-natural-language-processing-nlp-be907cd06c71>>.

RUBIN, Victoria L.; CHEN, Yimin; CONROY, Nadia K. Deception detection for news: Three types of fakes. **Proceedings of the Association for Information Science and Technology**, v. 52, n. 1, p. 1–4, 2015. DOI: <https://doi.org/10.1002/pra2.2015.145052010083>. eprint: <https://asistdl.onlinelibrary.wiley.com/doi/pdf/10.1002/pra2.2015.145052010083>. Disponível em: <<https://asistdl.onlinelibrary.wiley.com/doi/abs/10.1002/pra2.2015.145052010083>>.

SILVA, Renato M. et al. Towards automatically filtering fake news in Portuguese. **Expert Systems with Applications**, v. 146, p. 113199, 2020. ISSN 0957-4174. DOI: <https://doi.org/10.1016/j.eswa.2020.113199>. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S0957417420300257>>.

ANEXOS

I

Scrapper

```
import requests
from bs4 import BeautifulSoup
import unicodedata
from multiprocessing import Pool
import pandas as pd

site = "boatos.org"
search_query = '?s=%23boato'


def get_soup(url):
    page = requests.get(url)
    return BeautifulSoup(page.text, 'html.parser')


def should_ignore_paragraph(child):
    parent = child.parent
    if parent.name != 'p':
        parent = parent.parent
    p = parent.get_text()
    return ("Ps.:" in p or "PS:" in p or "PS.:" in p or "Se voc  quiser sugerir"
     in p)


def scrape_hoax(link):
    print('> Scrapping', link)
    soup = get_soup(link)

    paragraphs = [
```

```
        p.get_text() for p in soup.select('#content [style="color: #ff0000;"]')
    ]
    if len(paragraphs) == 0:
        paragraphs = [p.get_text()
                      if not should_ignore_paragraph(p) else ""
                      for p in soup.select('#content em')]

    hoax = " ".join(paragraphs)
    hoax = hoax.replace("Se inscreva no nosso canal no Youtube ", "")

    clean_hoax = unicodedata.normalize("NFKD", hoax).strip()
    timestamp = ""

    for i in soup.findAll('time'):
        if i.has_attr('datetime'):
            timestamp = i['datetime']

   # time = soup.select_one('time[datetime]')
   # timestamp = pd.to_datetime(time['datetime'], dayfirst=True)

    return {'link': link, 'timestamp': timestamp, 'hoax': clean_hoax}
   # return {'link': link, 'hoax': clean_hoax}

def find_links_from_search_page(url):
    soup = get_soup(url)
    links = [a['href'] for a in soup.select('.more-link')]

    return links


def scrape_search_for_links(page_index):
    print('Scrapping page', page_index)
    links = find_links_from_search_page('http://www.' + site + '/page/' +
                                        str(page_index) + search_query)
    print('> Found', len(links), 'links for page', page_index)
    return links


if __name__ == "__main__":
    initial_page = 1
    final_page = 110
    print('Start scrapping', site, 'from page', initial_page, 'to',
          final_page)
    with Pool(5) as p:
        all_links = p.map(scrape_search_for_links,
                          range(initial_page, final_page))
        all_links = [val for sublist in all_links for val in sublist]
        print('> Found', len(all_links), 'total')

        all_hoaxes = p.map(scrape_hoax, all_links)

        df = pd.DataFrame(all_hoaxes)
```

```
81          df = df[df['hoax'].str.len() > 100]
82          df.to_csv(site + ".csv")
83          print('df to csv')
```

Listing I.1: Scrapper - Boatos.org

II

Notebook Python

TCC_FakeNews_v3

February 25, 2022

0.0.1 NLP

0.0.2 Importando dados

[ ]:
```
# Setup básico
import os
import unicodedata
import re
import numpy as np
import pandas as pd
import csv

# Ferramentas ML
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score, confusion_matrix, f1_score
from sklearn.utils.extmath import density
from sklearn.pipeline import make_pipeline

# Visualização de dados
!pip install plotly mlxtend pandas
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.plotting import plot_confusion_matrix
import pandas as pd
import plotly.offline as py
import plotly.graph_objs as go
%matplotlib widget

# Input de dados
input_path = '/home/jlvale/TCC_fakeNews/Datasets_utilizados' # Path dos datasets
fake_and_real = pd.read_csv(os.path.
 ↪join(input_path,'pre-processed_fakeBR_semAcento.csv')) # GitHub Fake.br
boatos = pd.read_csv(os.path.join(input_path,'boatos.org_semAcento.
 ↪csv'),index_col=0) # Boatos.org
lupa = pd.read_csv(os.path.join(input_path,'piaui.folha.uol.com.br_semAcento.
 ↪csv'))# Agencia Lupa
```

```
noticias = pd.read_csv(os.path.
 ↪join(input_path,'df_final_20180101_20210901_semAcento.csv'),index_col=0) #␣
 ↪Noticias do G1 (01/2018 - 09/2021)
```

[ ]:
```
# Display dos datasets antes
display(fake_and_real.head())
display(boatos.head())
display(lupa.head())
display(noticias.head())

# Renomeando colunas
fake_and_real = fake_and_real.rename(columns={'preprocessed_news':'noticia'})
fake_and_real = fake_and_real.drop(columns=['index'])

boatos = boatos.rename(columns={'hoax':'noticia'})

lupa = lupa.rename(columns={'hoax':'noticia'})
lupa = lupa.iloc[: , 1:,]

noticias = noticias.rename(columns={'data':'timestamp','url_noticia':
 ↪'link','conteudo_noticia':'noticia'})
noticias = noticias.drop(columns=['url_noticia_curto','titulo','assunto'])

# Visualização:
print("\n")
print("Datasets reorganizados:")
display(fake_and_real.head())
display(boatos.head())
display(lupa.head())
display(noticias.head())

print('\n')
display(fake_and_real.info())
print('\n')
display(boatos.info())
print('\n')
display(lupa.info())
print('\n')
display(noticias.info())
```

[ ]:
```
# Substituindo Labels do Dataset Lupa
lupa['label'] = lupa['label'].
 ↪replace(['FALSO','EXAGERADO','CONTRADITORIO','INSUSTENTAVEL','AINDA E CEDO␣
 ↪PARA DIZER','SUBESTIMADO'],'fake')
lupa['label'] = lupa['label'].replace(['VERDADEIRO','VERDADEIRO MAS','DE␣
 ↪OLHO'],'true')
lupa = lupa.sample(frac=1).reset_index(drop=True)
```

```
lupa = lupa.dropna()
lupa = lupa.astype('str')

# Corrigindo possíveis erros nas noticias
lupa['noticia'].replace('FALSO','',inplace=True,regex=True)
lupa['noticia'].replace('EXAGERADO','',inplace=True,regex=True)
display(lupa.label.value_counts())
```

0.0.3 Limpeza/Preparação de dados

[ ]:
```
# Definindo labels. O dataset boatos é inteiramente fake e o dataset noticias é␣
 ↪inteiramente true.
boatos['label'] = 'fake'
noticias['label'] = 'true'
display(fake_and_real.head())
display(boatos.head())
display(lupa.head())
display(noticias.head())
```

[ ]:
```
# Concatenando datasets para criar dataset de TREINAMENTO
data = pd.concat([fake_and_real], axis=0) # Treinamento composto do dataset␣
 ↪Fake.br
data = data.astype('str')
data = data.sample(frac=1).reset_index(drop=True)
data["noticia"] = data["noticia"].replace(regex=r'[!/,.-]',value='')
data["noticia"] = data["noticia"].str.lower()
data["noticia"].str.normalize('NFKD').str.encode('ascii', errors='ignore').str.
 ↪decode('utf-8')
data = data.dropna()

# Visualização de dados de treinamento
display(data.head())
display(data.info())
display(data.label.value_counts())

#Downsample noticias falsas *(caso necessário)
#from sklearn.utils import resample
#data_fake = data[data.label=='fake']
#data_true = data[data.label=='true']
#data_fake_downsampled = resample(data_fake, replace=False, n_samples=2490,␣
 ↪random_state=123)
#data_downsampled = pd.concat([data_fake_downsampled, data_true])
#display(data_downsampled.label.value_counts())
```

[ ]:
```
# Separando dados de Treinamento e Teste
```

```
# Dropa NaN
data.dropna(subset = ["noticia"], inplace=True)
data.dropna(subset = ["label"], inplace=True)
data = data.reset_index()

#Dataset de Treino
X_train = data['noticia']
y_train = data['label']


# Dataset de Teste
teste = pd.concat([noticias, boatos], axis=0) #, NOTICIAS G1 e BOATOS.ORG como␣
 ↪teste SEM LUPA

# Dropa NaN
teste.dropna(subset = ["noticia"], inplace=True)
teste.dropna(subset = ["label"], inplace=True)
teste = teste.reset_index()
X_test = teste['noticia']
y_test = teste['label']

# Visualização de dados de treinamento
display(teste.head())
display(teste.info())
display(teste.label.value_counts())

print('\n')
print("\nExistem {} documentos nos dados de treinamento.".format(len(X_train)))
print('\n')
print("\nExistem {} documentos nos dados de teste.".format(len(X_test)))
```

0.0.4 Extração de Features

0.0.5 TfidfVectorizer

TF (Term Frequency): O número de vezes que uma palavra aparece em um documento é a sua Frequência do termo. Um valor mais alto significa que um termo aparece mais vezes do que outros, e assim, o documento é uma bom indicador quando tal termo é parte dos termos pesquisados.

IDF (Inverse Document Frequency): Palavras que ocorrem muitas vezes em um documento, mas também ocorrem muitas vezes em outros documentos, podem ser irrelevantes. IDF é uma medida do quão significativo um termo é dentro de corpus completo.

[ ]:
```
import nltk
import pickle
nltk.download('stopwords')
from nltk.corpus import stopwords
```

```
# Definindo stopwords em pt
stopwords=nltk.corpus.stopwords.words('portuguese')

# Extendendo as stopwords
#customList = ['folha','uol','antagonista','sabado','domingo','dezembro','veja']
#stopwords.extend(customList)

# Extração de tokens utilizando TFIDF
my_tfidf=TfidfVectorizer(analyzer='word',stop_words=stopwords,token_pattern='(?
 ↪u)\\b\\w\\w+\\b',use_idf=True, lowercase=True, ngram_range=(1, 1),␣
 ↪strip_accents=str, max_features=20000)
tfidf_train = my_tfidf.fit_transform(X_train)
tfidf_test = my_tfidf.transform(X_test)

# Exportando arquivo contendo tokens
pickle.dump(my_tfidf.vocabulary_,open("featuresTCC.pkl","wb"))
```

0.0.6 Treinamento do Modelo: Classificador Passivo-Agressivo

[ ]:
```
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score


# Passive Agressive ou RandomForestClassifier()
pa_clf = PassiveAggressiveClassifier(max_iter=1000)
pa_clf.fit(tfidf_train, y_train)
```

Inferência de dados

[ ]:
```
y_pred = pa_clf.predict(tfidf_test)

conf_mat = confusion_matrix(y_test, y_pred)
plot_confusion_matrix(conf_mat,
                      show_normed=True, colorbar=True)

accscore = accuracy_score(y_test, y_pred)
precision = precision_score(y_test,y_pred,pos_label='true')
recall = recall_score(y_test,y_pred,pos_label='true')
f1score = f1_score(y_test,y_pred,pos_label='true')


print('A acurácia da predição é {:.2f}%.\n'.format(accscore*100))
print('O valor da precisão é {:.3f}.\n'.format(precision))
print('O valor do recall é {:.3f}.\n'.format(recall))
print('O valor de F1 é {:.3f}.\n'.format(f1score))
```

Densidade

[ ]:
```
# Dimensionalidade e densidade de features

print("Dimensionalidade (i.e.,número de features): {:d}".format(pa_clf.coef_.
 ↪shape[1]))
print("Densidade (i.e., fração de elementos não nulos): {:.3f}".
 ↪format(density(pa_clf.coef_)))
```

[ ]:
```
# Pesos não nulos
weights_nonzero = pa_clf.coef_[pa_clf.coef_!=0]
feature_sorter_nonzero = np.argsort(weights_nonzero)
weights_nonzero_sorted =weights_nonzero[feature_sorter_nonzero]

# Plotar
fig, axs = plt.subplots(1,2, figsize=(9,3))

sns.lineplot(data=weights_nonzero_sorted, ax=axs[0])
axs[0].set_ylabel('Peso')
axs[0].set_xlabel('Número de features \n (Peso nulo omitido)')

axs[1].hist(weights_nonzero_sorted,
            orientation='horizontal', bins=500,)
axs[1].set_xlabel('Contagem')

fig.suptitle('Distribuição de pesos em features com pesos não nulos')

plt.show()
```

Ranking de tokens

[ ]:
```
tokens = my_tfidf.get_feature_names()
tokens_nonzero = np.array(tokens)[pa_clf.coef_[0]!=0]
tokens_nonzero_sorted = np.array(tokens_nonzero)[feature_sorter_nonzero]

num_tokens = 50
fake_indicator_tokens = tokens_nonzero_sorted[:num_tokens]
real_indicator_tokens = np.flip(tokens_nonzero_sorted[-num_tokens:])

fake_indicator = pd.DataFrame({
    'Token': fake_indicator_tokens,
    'Weight': weights_nonzero_sorted[:num_tokens]
})

real_indicator = pd.DataFrame({
    'Token': real_indicator_tokens,
    'Weight': np.flip(weights_nonzero_sorted[-num_tokens:])
})
```

```
print('Os Top {} tokens prováveis de aparecer em notícias falsas são os␣
 ↪seguintes: \n'.format(num_tokens))
display(fake_indicator)

print('\n\n...e os Top {} tokens prováveis de aparecer em notícias verdadeiras␣
 ↪são os seguintes: \n'.format(num_tokens))
display(real_indicator)
```

0.0.7 Comparação de Modelos de Classificação

Importando Libs

[ ]:
```
# Setup básico para outros modelos
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.feature_extraction.text import TfidfVectorizer
import pandas as pd
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
from mlxtend.plotting import plot_confusion_matrix
from collections import Counter
from nltk.tokenize import word_tokenize
import warnings
warnings.filterwarnings('ignore')
```

Ajustando os modelos comparativos

[ ]:
```
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

x_train=tfidf_train
x_test=tfidf_test

print(x_test.shape)
print(x_train.shape)

# Ajustando modelos
model1=PassiveAggressiveClassifier(max_iter=300)
model1.fit(x_train,y_train)
```

```
#model2=GaussianNB()
#model2.fit(x_train.toarray(),y_train)
model2=DecisionTreeClassifier()
model2.fit(x_train,y_train)
model3=RandomForestClassifier()
model3.fit(x_train,y_train)
model4=SVC()
model4.fit(x_train,y_train)
model5=LogisticRegression()
model5.fit(x_train,y_train)
```

[ ]:
```
y_pred1=model1.predict(x_test)
y_pred2=model2.predict(x_test.toarray())
y_pred3=model3.predict(x_test)
y_pred4=model4.predict(x_test)
y_pred5=model5.predict(x_test)
```

[ ]:
```
# Calculando acurácia dos modelos
acc1=accuracy_score(y_test,y_pred1)
acc2=accuracy_score(y_test,y_pred2)
acc3=accuracy_score(y_test,y_pred3)
acc4=accuracy_score(y_test,y_pred4)
acc5=accuracy_score(y_test,y_pred5)

# Calculando F1 dos modelos
f1_1=f1_score(y_test,y_pred1,pos_label='true')
f1_2=f1_score(y_test,y_pred2,pos_label='true')
f1_3=f1_score(y_test,y_pred3,pos_label='true')
f1_4=f1_score(y_test,y_pred4,pos_label='true')
f1_5=f1_score(y_test,y_pred5,pos_label='true')

# Calculando precisão dos modelos
prec1 = precision_score(y_test,y_pred1,pos_label='true')
prec2 = precision_score(y_test,y_pred2,pos_label='true')
prec3 = precision_score(y_test,y_pred3,pos_label='true')
prec4 = precision_score(y_test,y_pred4,pos_label='true')
prec5 = precision_score(y_test,y_pred5,pos_label='true')

# Calculando precisão dos modelos
rec1 = recall_score(y_test,y_pred1,pos_label='true')
rec2 = recall_score(y_test,y_pred2,pos_label='true')
rec3 = recall_score(y_test,y_pred3,pos_label='true')
rec4 = recall_score(y_test,y_pred4,pos_label='true')
rec5 = recall_score(y_test,y_pred5,pos_label='true')

labelsacc={'PassiveAggressiveClassifier':acc1,'DecisionTreeClassifier':
 ↪acc2,'RandomForestClassifier':acc3,
```

```
        'SVC':acc4,'LogisticRegression':acc5}

labelsf1={'PassiveAggressiveClassifier':f1_1,'DecisionTreeClassifier':
 ↪f1_2,'RandomForestClassifier':f1_3,
        'SVC':f1_4,'LogisticRegression':f1_5}

labelsprec={'PassiveAggressiveClassifier':prec1,'DecisionTreeClassifier':
 ↪prec2,'RandomForestClassifier':prec3,
        'SVC':prec4,'LogisticRegression':prec5}

labelsrec={'PassiveAggressiveClassifier':rec1,'DecisionTreeClassifier':
 ↪rec2,'RandomForestClassifier':rec3,
        'SVC':rec4,'LogisticRegression':rec5}

print('Valores de acurácia por modelo:')
for model,accuracy in labelsacc.items():
    print(str(model)+' : '+str(accuracy))
print('\n\n')
print('Valores de F1 por modelo:')
for model,f1score in labelsf1.items():
    print(str(model)+' : '+str(f1score))
print('\n\n')
print('Valores de precisão por modelo:')
for model,precision in labelsprec.items():
    print(str(model)+' : '+str(precision))
print('\n\n')
print('Valores de recall por modelo:')
for model,recall in labelsrec.items():
    print(str(model)+' : '+str(recall))
```

0.0.8 Vizualização de Dados e Extração de Conhecimento

[ ]:
```
# Setup básico para t-SNE
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np
import re
import nltk
nltk.download('punkt')

from gensim.models import word2vec

from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
%matplotlib inline
```

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.decomposition import TruncatedSVD
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import HashingVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import Normalizer
from sklearn import metrics

from sklearn.cluster import KMeans, MiniBatchKMeans
from yellowbrick.text import TSNEVisualizer
from yellowbrick.datasets import load_hobbies

import logging
from optparse import OptionParser
import sys
from time import time

import numpy as np

from sklearn.datasets import fetch_openml
from sklearn.manifold import TSNE
from sklearn.decomposition import PCA
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
```

[ ]:
```
# Definindo funções
STOP_WORDS = stopwords.words('portuguese')

def clean_sentence(val):
    # pre processamento
    regex = re.compile('([^\s\w]|_)+')
    sentence = regex.sub('', val).lower()
    sentence = sentence.split(" ")

    for word in list(sentence):
        if word in STOP_WORDS:
            sentence.remove(word)

    sentence = " ".join(sentence)
    return sentence

def clean_dataframe(data):

    data = data.dropna(how="any")
```

```
    for col in ['noticia']:
        data[col] = data[col].apply(clean_sentence)

    return data

def custom_tokenize(text):
    if not text:
        print('O texto a ser tokenizado é do tipo None. Definindo como uma␣
 ↪string vazia.')
        text = ''
    return word_tokenize(text)
```

Word2Vec - tSNE geral

[ ]:
```
X = []
labels = []

tokens = data['noticia'].apply(custom_tokenize)
model = word2vec.Word2Vec(tokens, vector_size=500, min_count=200)

for word in model.wv.key_to_index:
        X.append(model.wv[word])
        labels.append(word)
```

[ ]:
```
y = data.label

# PCA Antes do t-SNE
#pca = PCA(n_components=2)
#X_pca = pca.fit_transform(X)

# Modelo t-SNE
tsne = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=1000,␣
 ↪learning_rate=200)
tsne_results = tsne.fit_transform(X)

# Visualização
df_tsne = pd.DataFrame(tsne_results, columns=['x', 'y'])
df_tsne['label'] = y
sns.lmplot(x='x', y='y', data=df_tsne, hue='label', fit_reg=False)
```

Word2Vec + t-SNE detalhado

[ ]:
```
# Dataset true. Visualização de dados por proximidade vetorial dos tokens.
train_true = data[data['label'] == 'true']
train_true = clean_dataframe(train_true) # Remove as stop words
train_true.head(5)
train_tokens_true = train_true['noticia'].apply(custom_tokenize)
```

```
model_true = word2vec.Word2Vec(train_tokens_true, vector_size=100, window=20,␣
↪min_count=1, workers=4)

# Dataset fake. Visualização de dados por proximidade vetorial dos tokens.
train_fake = data[data['label'] == 'fake']
train_fake = clean_dataframe(train_fake) # Remove as stop words
train_fake.head(5)
train_tokens_fake = train_fake['noticia'].apply(custom_tokenize)
model_fake = word2vec.Word2Vec(train_tokens_fake, vector_size=100, window=20,␣
↪min_count=1, workers=4)
```

[ ]:
```
# Plotando utilizando t-SNE
def tsne_plot(model):
    "Creates and TSNE model and plots it"
    labels = []
    tokens = []

    for word in model.wv.key_to_index:
        tokens.append(model.wv[word])
        labels.append(word)

    #tsne_model = TSNE(perplexity=40, n_components=2, init='pca', n_iter=2500,␣
↪random_state=23)
    tsne_model = TSNE(n_components=2, verbose=1, perplexity=40, n_iter=1000,␣
↪learning_rate=200)
    new_values = tsne_model.fit_transform(tokens)

    x = []
    y = []
    for value in new_values:
        x.append(value[0])
        y.append(value[1])

    plt.figure(figsize=(16, 16))
    for i in range(len(x)):
        plt.scatter(x[i],y[i])
        plt.annotate(labels[i],
                     xy=(x[i], y[i]),
                     xytext=(5, 2),
                     textcoords='offset points',
                     ha='right',
                     va='bottom')
    plt.show()
```

[ ]:
```
tsne_plot(model_true)
```

[ ]:
```
tsne_plot(model_fake)
```

[ ]:
```
model_true.wv.most_similar('governo')
```

[ ]:
```
model_fake.wv.most_similar('governo')
```

[ ]:
```
model_fake.wv.most_similar('eleicoes')
```

[ ]:
```
model_true.wv.most_similar('eleicoes')
```

[ ]:
```
model_fake.wv.most_similar('politica')
```

[ ]:
```
model_true.wv.most_similar('politica')
```