UNIVERSIDADE FEEVALE

DIEGO GIROTTO ARIGONY

DESENVOLVIMENTO DE UM CLASSIFICADOR DE CLICKBAIT BASEADO EM MACHINE LEARNING

Novo Hamburgo

2020

DIEGO GIROTTO ARIGONY

DESENVOLVIMENTO DE UM CLASSIFICADOR DE CLICKBAIT BASEADO EM MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

Orientador: Rodrigo Rafael Villarreal Goulart

Novo Hamburgo

2020

DIEGO GIROTTO ARIGONY

DESENVOLVIMENTO DE UM CLASSIFICADOR DE CLICKBAIT BASEADO EM MACHINE LEARNING

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do grau de Bacharel em Ciência da Computação pela Universidade Feevale

APROVADO EM: ___ / ___ / _____

RODRIGO RAFAEL VILLARREAL GOULART
Orientador – Feevale

GABRIEL DA SILVA SIMÕES
Examinador interno – Feevale

JULIANO VARELLA DE CARVALHO
Examinador interno – Feevale

Novo Hamburgo
2020

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial: Aos amigos e às pessoas que convivem comigo diariamente, minha gratidão, pelo apoio emocional - nos períodos mais difíceis do trabalho.

RESUMO

Clickbait se refere a chamadas curtas de notícia formuladas com o objetivo de induzir os leitores a clicar em um link. O objetivo dessa prática é gerar maior tráfego de acessos a páginas de um site e, consequentemente, aumentar a renda a partir de publicidades na página. Isto se faz por meio de mensagens que podem ser vagas e informações exageradas ou errôneas, provocando curiosidade suficiente para o acesso. Em geral, o resultado desta prática vem ao custo de clareza e objetividade na informação. A Bauhaus-Universität Weimar propôs, em 2017, um desafio para o desenvolvimento de um classificador que define o conteúdo em uma das quatro classes: não *clickbait*, levemente *clickbait*, consideravelmente *clickbait* ou fortemente *clickbait*. Baseado neste desafio, o presente trabalho busca avaliar o estado da arte na área, propondo um classificador utilizando conceitos de Doc2Vec e comparando seus resultados aos submetidos ao *Clickbait Challenge*. O modelo criado atingiu métricas competitivas comparado aos trabalhos submetidos ao desafio, porém não houve melhoria na métrica primária comparado ao código base utilizado no desenvolvimento.

Palavras-chave: *Clickbait*. Classificadores. *Machine Learning*. PLN.

ABSTRACT

Clickbait refers to news teasers made with the aim of inducing readers to click on a link. The purpose of this practice is to generate more traffic from accessing pages on a website and, consequently, increase income from advertisements on the page. This is done through messages that may be vague, exaggerated or erroneous information, causing enough curiosity for access. Usually, the result of this practice comes at the cost of clarity and objectivity in the information. Bauhaus-Universität Weimar proposed, in 2017, a challenge for the development of a classifier that defines the content in one of the four classes: not clickbait, slightly clickbait, considerably clickbait or strongly clickbait. Based on this challenge, the present project seeks to assess the state of the art in the area, proposing a classifier using the concepts of Doc2Vec and comparing its results to those submitted to the Clickbait Challenge. The created model achieved competitive metrics compared to the works submitted to the challenge, however there was no improvement in the primary metric compared to the base code used in the development.

Keywords: Clickbait. Classifiers. Machine Learning. NLP.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AUC	*Area under the ROC Curve*
CEO	*Chief Executive Officer*
Doc2Vec	*Document to Vector*
GloVe	*Global Vectors*
GRU	*Gated Recurrent Unit*
JSON	*JavaScript Object Notation*
LSTM	*Long Short-Term Memory*
MSE	*Mean Squared Error*
PLN	Processamento de Linguagem Natural
PV-DBOW	*Bag of Words version of Paragraph Vector*
PV-DM	*Distributed Memory version of Paragraph Vector*
ROC	*Receiver Operating Characteristic Curve*
SRU	*Simple Recurrent Unit*
SVM	*Support Vector Machines*
Word2Vec	*Word to Vector*

SUMÁRIO

1 INTRODUÇÃO

Atualmente, a internet é uma fonte de conhecimento dos mais variados assuntos. Oferecendo acesso fácil a blogs pessoais, notícias nacionais e internacionais, conteúdo técnico dos mais diversos assuntos e fóruns de discussão, a oferta de informações é praticamente inesgotável.

Na atualidade o jornalismo, por exemplo, está em grande parte no meio digital. Dito isso, esse tipo de empresa depende diretamente da renda gerada em seus meios digitais. Com a crescente oferta de informações, é de se esperar uma enorme concorrência pela atenção dos leitores na rede. Buscando aumentar métricas importantes para a receita de publicidades, como quantidade de acessos e tempo de visitação, diversos sites vem usando a estratégia de chamadas *clickbait*.

Clickbait, segundo Potthast et al. (2016), se refere a um certo tipo de propaganda de conteúdo web formuladas com o objetivo de induzir o leitor a clicar em seu link. Um resultado imediato dessa prática é que, geralmente, a formulação é realizada à custa de objetividade e clareza na informação da chamada. Se faz por meio de mensagens que podem ser vagas e informações exageradas ou errôneas, provocando curiosidade suficiente para o acesso. Exemplos de chamadas *clickbait* conforme corpus de Potthast et al. (2018b):

- *Chinese actress sued for staring too intensely.*
- *As a BASE jumper leaped, his girlfriend snapped photos. Then came confusion, hope and despair.*
- *10 books to read before you see the movies this summer.*
- *Someone put 1,000 cardboard cuttouts of Nicki Minaj's ass on the steps of a Finland Cathedral.*
- *10 years ago, production errors at the Airbus Group factory crippled the company. Today, it's a model of efficiency:*

Essa prática, além de prejudicial a todo o sistema de notícias online, pode gerar sérios problemas econômicos. Exemplo disso é o caso da Apple (HARGREAVES, 2008), quando chamadas de notícias davam a entender que Steve Jobs, CEO da empresa, havia sofrido um infarto. As ações da empresa despencaram cerca de 10% em valor na bolsa americana.

Na literatura, conforme Geçkil et al. (2020), o problema da classificação de notícias *clickbait* tem sido abordado, no geral, com o uso de métodos de Processamento

de Linguagem Natural (PLN). Potthast et al. (2016) realizou um dos primeiros e mais conhecidos trabalhos na área, propondo um modelo para a detecção *clickbait* baseado em 215 atributos, possibilitando seu classificador random forest atingir a marca de 0.79 ROC-AUC, 0,76 de precisão e 0,76 de recall. O artigo também é responsável pelo primeiro corpus (coleção de textos) *clickbait*, com um total 2992 tweets provenientes dos 20 perfis dos que mais geram conteúdo de notícias no Twitter, baseado pela influência em termos de re-tweets. Posteriormente, conforme processo descrito por Potthast et al. (2018b), uma evolução do corpus foi realizada, desta vez contendo 38,517 tweets anotados, denominado como Webis *Clickbait* Corpus 2017.

Utilizando o Webis *Clickbait* Corpus 2017, o *Clickbait Challenge*, uma competição organizada em 2017, incentivou os grupos de participantes a criar um modelo para determinar *clickbaits*. Foram submetidos 16 resultados, onde 10 possuíam o formato de artigo e 8 incluíram também o código-fonte de seu modelo. O estudo deste problema não é algo novo. Contudo, o crescimento da prática com a migração da informação para o plano digital tem impulsionado a pesquisa na área.

Baseado nessa problemática, o presente trabalho busca aprimorar técnicas de classificação utilizada nos modelos submetidos ao *Clickbait Challenge*. Dentre oportunidades encontradas, destaca-se o uso aprofundado de paragraph vector - Doc2Vec, um conceito apresentado por Le e Mikolov (2014) para criar uma representação numérica de um documento. A técnica é uma alternativa ao bag-of-words, técnica comumente utilizada em algoritmos de Processamento de Linguagem Natural.

Glenski et al. (2017), em artigo de resultado submetido ao *Clickbait Challenge*, sugere o uso de Doc2Vec na busca de similaridade entre texto da chamada (tweet) e título do artigo como uma forma de capturar a relação entre o tweet e o artigo. O conceito utilizado no modelo Doc2Vec, similar ao Word2Vec (MIKOLOV et al., 2013), consiste em um algoritmo de aprendizagem não supervisionada para gerar um vetor representativo de partes de tamanho variado de texto, como sentenças, parágrafos e documentos. Em posse desse vetor, é possível aferir a similaridade entre essas partes. O presente estudo visa desenvolver um modelo de classificação que utilize Doc2Vec, sendo treinado e avaliado de acordo com o *Clickbait Challenge*.

1.1 OBJETIVOS

1.1.1 Objetivo geral

O presente estudo tem por objetivo desenvolver um classificador de notícias *clickbait* utilizando conceitos de Doc2Vec em sua implementação.

1.1.2 Objetivos específicos

- Pesquisa sobre métodos de aprendizado de máquina adequados ao *dataset* disponibilizado no *Clickbait Challenge*.
- Pesquisa sobre a aplicação do conceito Doc2Vec na classificação.
- Desenvolver um modelo de classificador com base nas pesquisas realizadas.
- Avaliar os resultados do classificador desenvolvido e comparar aos resultados obtidos nos trabalhos relacionados.
- Apresentar os resultados obtidos.

1.2 METODOLOGIA

Este trabalho tem como objetivo desenvolver um classificador para a resolução do problema de identificação de notícias *clickbait*. Pode-se defini-lo, portanto, como um projeto de pesquisa de natureza aplicada, pois conforme Prodanov e Freitas (2013): "[...] objetiva gerar conhecimentos para aplicação prática dirigidos à solução de problemas específicos. Envolve verdades e interesses locais.".

Quanto a classificação de seus procedimentos técnicos, pode ser enquadrado como bibliográfico e experimental. É bibliográfico dada a necessidade de pesquisa acerca do material já publicado, buscando informação relevante em relação ao tema. Também é experimental, pois vai gerar experimentos procurando melhorias em comparação ao resultado de trabalhos anteriores.

A abordagem utilizada é quantitativa. A análise dos resultados obtidos nos experimentos será feita tendo em vista as mesmas métricas utilizadas no *Clickbait Challenge*: erro médio quadrático, *F1 Score*, predição, recall e acurácia.

Por fim, em relação ao objetivo do projeto, pode ser definido como exploratório. A pesquisa e desenvolvimento tem o intuito de validar a hipótese de Doc2Vec como uma alternativa de modelo de classificação para o problema destacado no *Clickbait Challenge* e enriquecer o estudo na área da classificação de notícias *clickbait*.

1.3 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado da seguinte maneira:

- No próximo capítulo são apresentados os fundamentos teóricos resultantes da pesquisa bibliográfica acerca do assunto, incluindo Processamento de Linguagem Natural, classificadores de texto e classificadores no contexto de *clickbait*.

- No Capítulo 3 são descritos trabalhos correlatos. O objetivo é apresentar o estado da arte para classificadores *clickbait*.
- O quarto capítulo faz uma análise do *dataset* e do código reproduzido e utilizado como base, o desenvolvimento da hipótese sugerida, assim como apresenta os resultados obtidos com o modelo desenvolvido, comparando-o ao código base e demais submissões ao *Clickbait Challenge*.
- Por fim, o último capítulo apresenta a conclusão do trabalho realizado, pontuando o resultado obtido pelo trabalho como um todo e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO

O problema de classificação *clickbait* é razoavelmente recente, estando relacionado a diversos assuntos atuais da área de aprendizado de máquina. Neste capítulo serão apresentado tópicos chave relacionados aos conhecimentos utilizados para o desenvolvimento do presente trabalho.

2.1 CLICKBAIT

Para editores de conteúdo online, o modelo de receita mais popular é publicidade. Publicidades são exibidas ao longo do conteúdo das páginas e os editores são remunerados por cada visualização de publicidade feita por visitantes destas páginas (POTTHAST et al., 2018b). Para trazer mais visualizações a seus sites (e, consequentemente, mais renda), grande parte dos editores faz publicidade (*teasers* de conteúdo) de seus artigos em redes sociais. Conforme Potthast et al. (2018b), na perspectiva de um editor, esses *teasers* devem revelar o mínimo suficiente para que o público alvo (ou melhor ainda: qualquer um) seja tentado a visitar a página. De forma inversa, na perspectiva ideal para o leitor, o *teasers* deve conter um resumo bem construído do artigo, assim a visita à página somente será feita se houver interesse real no resto do conteúdo. Os ideais são, portanto, naturalmente contrários.

Figura 1 – Exemplo de chamada clickbait em vídeo no YouTube

Fonte: do autor (2020)

Blom e Hansen (2015) afirmam que jornalistas, por tradição, usam uma série de estratégias para fazer com que suas manchetes capturem a atenção dos leitores. Dentre os meios para isso, pode-se destacar o uso de estilo e narrativa na criação da manchete para torná-la mais interessante (exemplificado na Figura 1), assim como o uso de sensacionalismo e outros tipos de conteúdo enfatizando sexo, escândalos e tragédia.

Conforme Chakraborty et al. (2016), *clickbaits* exploram o fenômeno cognitivo conhecido como Lacuna de Curiosidade (*Curiosity Gap*), onde as manchetes fornecem informações que gerem curiosidade suficiente para que os leitores se sintam obrigados a

clicar no link para satisfazer essa curiosidade. Por mais que essas iscas (*baits*) funcionem, a expectativa criada pelo usuário não é satisfeita.

Mark (2014) apresenta as técnicas de *clickbait* como facilitadoras de distração. A autora menciona a carga cognitiva gerada a partir das transições de um artigo para outro. Quando sobrecarregados, ficamos mais suscetíveis a realizar tarefas leves e, portanto, consumir conteúdo raso. Como resultado disso tudo, a autora cita o stress, mau humor e baixa produtividade.

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

Conforme Reshamwala et al (2013), linguagens naturais são quaisquer linguagens que humanos aprendem no meio em que estão inseridos e usam para comunicar-se entre si. Independente da forma de comunicação - escrita, falada, visual - linguagens naturais são um meio de se adquirir e transmitir informações.

O Processamento de Linguagem Natural (PLN) é uma área de pesquisa e aplicação que explora como computadores podem ser utilizados com o objetivo de compreender e manipular informações de entrada em linguagem natural para fins úteis (CHOWDHURY, 2005). A aplicação de PLN é dada em diversos campos de estudo, como sumarização de texto, classificação, tradução, reconhecimento de fala, agentes de conversação e ferramentas de extração/recuperação de informações. É uma área de grande demanda, graças à crescente produção de informação digital e a necessidade de métodos e processos que possibilitem o trabalho nestes dados.

Para que se faça possível um computador compreender e interpretar uma linguagem natural, é necessário primeiramente entender como nós realizamos essa mesma tarefa. Para isso, parte do estudo de PLN se apoia na área de pesquisa de linguística. A linguística, conforme Manning e Schütze (1999), busca caracterizar a forma de nossa escrita, conversa e outros meios de comunicação. Segundo o trabalho citado, um dos principais problemas da área reside nessa caracterização: é extremamente difícil definir características bem definidas quando as pessoas estão sempre dispostas a modificá-las a fim de satisfazer suas necessidades de comunicação.

2.2.1 Níveis na análise linguística

Segundo Liddy (2001), o método mais explicativo para representar o que acontece em um sistema de PLN é por meio de uma abordagem de "níveis da linguagem- também referenciado na literatura como modelo síncrono de linguagem, diferindo do modelo sequencial, que considera que o Processamento de Linguagem Natural é realizado um após o outro, de forma estritamente sequencial. A autora afirma que a pesquisa psicolinguística sugere um processamento muito mais dinâmico, com os níveis da análise interagindo em

variados sentidos.

Para que um computador possa realizar o objetivo do PLN - transformar linguagem natural em informações que o computador possa utilizar - é preciso que a linguagem seja analisada em vários níveis. Para que possa melhorar os níveis de compreensão em um sistema PLN, é essencial que utilize mais níveis de linguagem. Conforme Covington, Nute e Vellino (1995), estes são níveis utilizados em PLN que correspondem a subcampos da linguística:

- Fonologia: estudo relacionado ao reconhecimento dos fonemas que compõem as palavras de uma linguagem falada. Em sistemas PLN, o componente de fonologia transcreve registro de ondas sonoras em símbolos discretos (fonemas). Liddy (2001) define três tipos de regras utilizadas na análise fonológica:
  - Análise fonética: para análise do som nas palavras.
  - Análise fonêmica: para análise na variação de pronúncia em palavras faladas em conjunto. Um dos objetivos é definir quais os sons de uma linguagem que apresentam valor distintivo.
  - Análise prosódica: para análises em tensão e entoação ao longo de uma sentença.
- Morfologia: trata do estudo da composição das palavras, caracterizado por morfemas - as menores unidades que formam as palavras, como raiz, radical e afixo. Notando que o sentido de cada morfema continua o mesmo através das palavras, é possível decompor uma palavra desconhecida em seus morfemas e atribuir seu sentido a partir daí. Sistemas de PLN, portanto, podem gerar conhecimento a partir da identificação destes elementos morfológicos. Pode-se considerar um fato morfológico, por exemplo, que a maioria das palavras em português no plural terminam com a letra "s".
- Sintaxe: a análise sintática observa a da estrutura das sentenças. O objetivo dessa etapa, no PLN, é definir uma estrutura hierárquica sintaticamente decomposta de acordo com regras que regem a construção de frases em determinada linguagem.
- Semântica: semântica na linguística é o estudo do significado na linguagem. Observa a relação entre palavras, frases e símbolos para o que representam (sua denotação).
- Pragmática: é um ramo da linguística responsável pelo estudo da linguagem como um objetivo de comunicação, relacionando o uso linguagem a um contexto. Para sistemas PLN, é responsável pela resposta apropriada de acordo com o rumo de uma conversação, por exemplo.

Ainda sobre essas etapas, Covington, Nute e Vellino (1995) ainda assumem que, mesmo qualitativamente diferentes, as etapas não devem ser processadas separadamente, pois é comum que ambiguidades de uma das etapas possa ser resolvida em outra.

2.2.2 Abordagens para PLN

Assim como demais tarefas em computação, não há uma abordagem única e definitiva para executarmos tarefas de PLN. As abordagens disponíveis se encaixam em quatro categorias: simbólica, estatística, conexionista e híbrida (LIDDY, 2001). Destas, as mais referenciadas em estudos da área são simbólica e estatística, presentes desde o início das pesquisas na área. Essa seção contém uma breve descrição de cada abordagem.

A abordagem simbólica (também referenciada na literatura como abordagem "racionalista") se baseia nos estudos do campo da linguística, focando no uso de significados e regras de uma linguagem. Liddy (2001) afirma que a abordagem simbólica realiza uma análise profunda dos fenômenos linguísticos e é baseada na representação explícita de fatos linguísticos de uma linguagem. A descrição dos "Níveis na análise linguística"realizada anteriormente, por exemplo, é feita com base na visão simbólica.

Exemplos dessa abordagem são algoritmos apresentados nesse estudo como "Classificadores baseados em padrões". Aubaid (2018) define esse tipo de classificador como sistemas que se baseiam em regras para a representação de aprendizagem de informação.

Segundo Liddy (2001), sistemas simbólicos vem sendo usado por décadas em variados campos de pesquisa e aplicação, como extração de informações, categorização de texto, resolução de ambiguidades, entre outros.

Abordagens estatísticas fazem uso de técnicas matemáticas e, geralmente, fazem uso de porções de texto longos para criação de um modelo generalizado baseado nos fenômenos linguísticos obtidos neste corpus. Ao contrário da abordagem simbólica, a informação observada é utilizada como fonte primária para a evidência (LIDDY, 2001). Nesta mesma linha, Manning e Schütze (1999) definem como um método que utiliza de um corpus, procurando por padrões e associações, que podem ou não estar associados com regras sintáticas ou semânticas.

A abordagem estatística se baseia em sistemas de aprendizado supervisionado, ou seja, realizado a partir de um corpus anotado (*training set*). Manning e Schütze (1999) citam modelos de Markov (e a aplicação do algoritmo de Viterbi) como exemplo de método estatístico.

Modelos estatísticos vem sendo empregados em tarefas como reconhecimento de fala, aquisição léxica, análise, *part-of-speech tagging*, entre outros (LIDDY, 2001).

Similar à abordagem estatística, a abordagem conexionista também desenvolve modelos generalizados a partir de fenômenos linguísticos. Nela, a diferença é que os mé-

todos estatísticos são combinados com uma série de teorias de representação (LIDDY, 2001). Modelos conexionistas, na literatura, são ocasionalmente referenciados como localistas (*localist models*) e distribuídos (*distributed models*). Conforme Gasser (1990), as unidades em abordagens localistas representam um conceito único, enquanto em abordagens distribuídas os conceitos podem ser dividido em várias unidades, onde cada unidade irá participar na representação de vários conceitos.

Nota-se que as abordagens simbólica, estatística e conexionista possuem diferentes características. A escolha de abordagem para resolução de um problema de PLN deve levar isso em conta, visto que cada abordagem dispõe de meios que podem facilitar ou dificultar a execução do sistema, dados os atributos do mesmo. Klavans e Resnik (1996) afirmam ainda que não há um método puramente estatístico. Segundo os autores, o uso de estatística é baseado em um modelo simbólico. Os dois modelos não são, portanto, opostos.

2.3 CLASSIFICADORES DE TEXTO

O problema de classificadores é amplamente estudado nas comunidades de *data-mining* e recuperação de informações. Pode-se destacar algumas aplicações em que o problema está inserido:

- Classificação de conteúdo impróprio em redes sociais: métodos automatizados de detecção de conteúdo impróprio estão cada vez mais populares nas redes sociais. Um exemplo recorrente é detecção de fake-news, como o apresentado no artigo de Granik e Mesyura (2017).
- Filtragem e organização de notícias: Serviços de notícias podem trabalhar com um grande volume de conteúdo variado. Pode tornar-se difícil organizar esse conteúdo manualmente, portanto métodos automatizados para tal tarefa podem ser úteis. Lang (1995) propôs um sistema classificador de interesse de usuário em artigos.
- Organização e recuperação de documentos: classificadores podem ser utilizados de bibliotecas digitais de documentos, artigos científicos e conteúdo social.
- Classificação e filtragem de email (spam): serviços de email comumente precisam classificar emails para determinar assunto e/ou determinar emails indesejados.

Uma generalização em alto nível do problema pode ser definida da seguinte forma. Temos um conjunto de dados de treinamento $D = \{X_1, ..., X_N\}$, onde cada registro é rotulado com um valor de classe obtido de um conjunto K de possíveis classes. O conjunto de dados de treinamento é então utilizado para criação de um modelo de classificação que irá relacionar características de um dado registro a uma das classes (AGGARWAL; ZHAI, 2012).

2.3.1 Pré-Processamento do texto

O pré-processamento de informações é uma prática frequente no desenvolvimento de métodos que fazem uso de *machine-learning*. O objetivo é remover inconsistências e aumentar a qualidade dos dados. Para a classificação de textos, especialmente, é de suma importância, visto que há uma quantidade grande de características de texto e a comum existência de conteúdo irrelevante (ruído). As escolhas feitas nessa etapa afetam significativamente a eficácia do classificador.

Na pesquisa bibliográfica, encontram-se ações para o pré-processamento que são comuns a vários trabalhos:

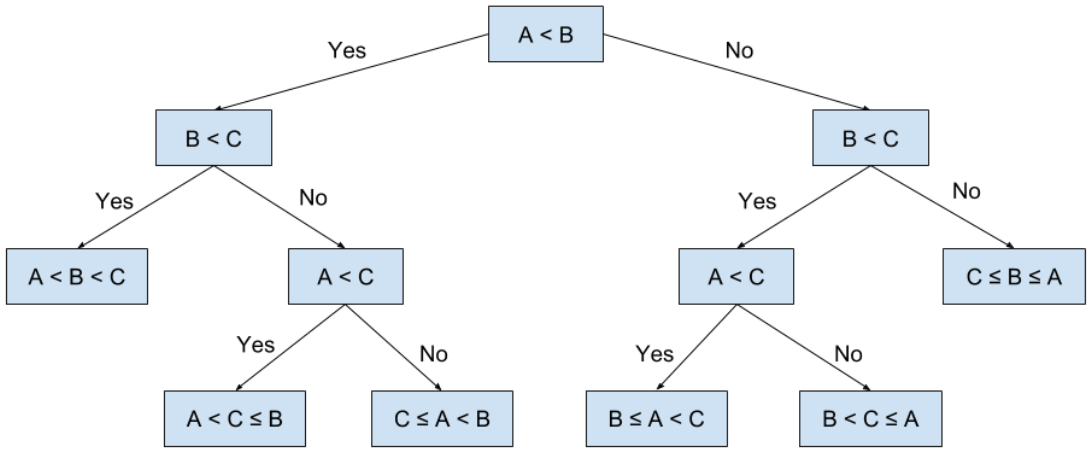
- Conversão para *lowercase*: na maioria dos contextos de classificação não há a necessidade de diferenciar as palavras maiúsculas de minúsculas.
- Remoção de *stopwords*: *stopwords* são palavras comuns a um idioma ou a um documento que não são específicas ou discriminatórias para o classificação trabalhada. Geralmente é criado um conjunto de *stopwords* e feita a busca e remoção no texto.
- Padronização *stemming*: em *stemming* (ou lematização), diferentes formas de uma mesma palavra são transformadas em uma única, como em singulares, plurais e radical de palavras. Para um classificador pode ser útil tratar palavras como afogando e afogamento como característica de uma mesma classe, consolidando então o radical "afog".
- Tokenização: é a prática de segmentar um texto em partes com significado único (*tokens*). Geralmente, a entrada inicial de texto realizada por um usuário em um sistema de PLN é dada na forma de um conjunto de caracteres (*string*). Esse conjunto de caracteres será quebrado em palavras (COVINGTON; NUTE; VELLINO, 1995). A *string* "Como vai seu dia?", por exemplo, seria transformada em uma lista de palavras *[como, vai, seu, dia]*.

2.3.2 Metodologias

Diversos métodos e algoritmos podem ser empregados em problemas na área de classificação de textos. Essa seção apresenta um resumo de algumas destas técnicas e suas utilidades para a classificação textual. (AGGARWAL; ZHAI, 2012) expõe os seguintes métodos como métodos chave nessa área: Árvores de decisão, classificadores baseados em padrões, classificadores SVM, redes neurais e classificadores bayesianos.

Árvore de decisão é uma das formas mais populares de aprendizagem supervisionada. Nela, o problema é representado em uma árvore dividida hierarquicamente de acordo com a informação de entrada e seus atributos (exemplificado na Figura 2). Para determinada instância de texto, de acordo com o caminho percorrido na árvore, pode-se

Figura 2 – Representação visual de uma árvore de decisão



Fonte: (NICULAESCU, 2018)

realizar a classificação. Neste método, procura-se obter a árvore de menor tamanho possível, com a maior acurácia possível. Para isso, um ponto de atenção é a estratégia de escolha dos atributos, buscando os mais próximos à raiz da árvore (ou seja, os atributos de importância elevada para definição de uma classe) (GONçALVES, 2007).

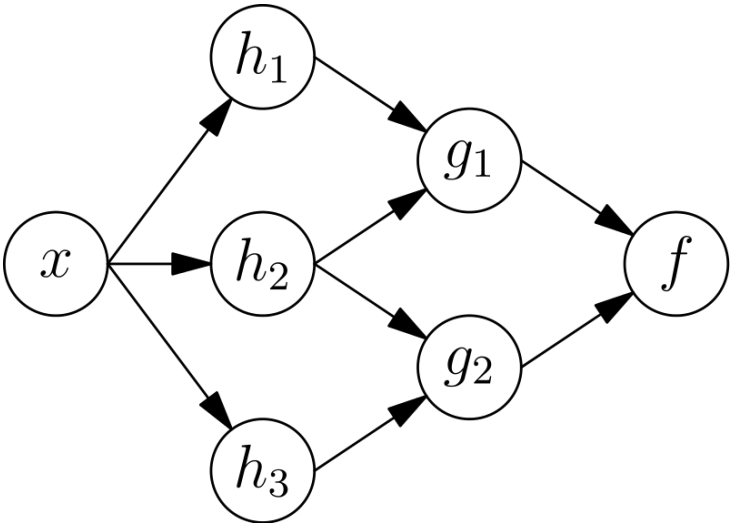
Em classificadores baseados em padrões (ou classificador baseado em regras), conforme Aggarwal e Zhai (2012), é definido um conjunto de regras de palavras que são mais prováveis de estar relacionada a uma das classes. Para Aubaid (2018), sistemas baseados em regra utilizam regras como a representação de aprendizagem para a informação. Segundo os autores, a implicação destes sistemas é a dependência em sistemas especializados, que por sua vez simulam o raciocínio humano para decifrar um problema denso em informação.

Classificadores de máquina de vetor de suporte (SVM, do inglês *Support Vector Machine*) tem o objetivo de encontrar limites ideais entre classes para posterior classificação. Os pontos mais próximos a este limite são os vetores de suporte. O SVM é considerado um método não-paramétrico, pois precisa guardar exemplos de treinamento para o funcionamento do classificador. Para classificar uma nova informação, será observado em qual lado do separador limite a informação foi alocada.

Redes neurais artificiais são utilizadas para uma variedade de domínios no contexto de aprendizado de máquina. Seu funcionamento é inspirado pelo processo de aprendizado humano. Conforme Knocklein (2019), redes neurais consiste em uma rede de nodos (em alguns casos referenciados como "neurônios") que usam entradas de informações para

produzir saída de informação para uma nova camada de nodos, até que a informação chegue ao nodo terminal e seja apresentado o resultado final do modelo (exemplificado na Figura 3). Na figura, o nodo de entrada é representado por x, tendo seu conteúdo utilizado nas camadas dos nodos h e g, finalizando em um nodo terminal f.

Figura 3 – Representação visual de uma rede neural artificial



Fonte: (Wikipedia contributors, 2020)

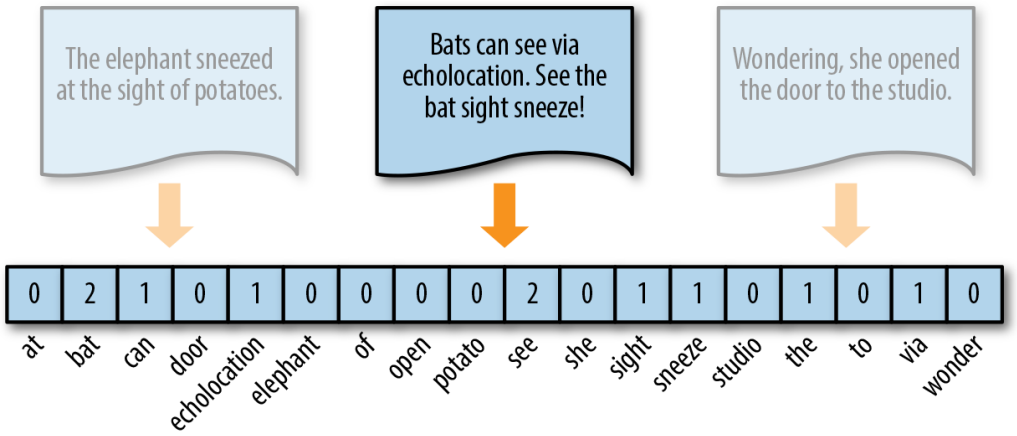
Classificadores bayesianos são considerados classificadores probabilísticos, sendo baseados no teorema de Bayes. Sua premissa é de que os atributos são independentes entre si. Para isso, é estudada a probabilidade individual de cada um deles quanto às classes. Rish (2001) demonstra que naive bayes tem seus melhores resultados em dois cenários: atributos completamente independentes (como já esperado) e atributos funcionalmente dependentes.

2.4 BAG OF WORDS

Bag of Words é uma das abordagens mais comuns para a recuperação de informações de um texto. Essa abordagem usa as palavras como indicadores e a frequências destas palavras como característica para a classificação de texto. Zhang, Jin e Zhou (2010) definem o modelo *Bag of Words* como um dos métodos de representação mais populares para a categorização de objetos. Numa representação *Bag of Words*, um vetor de dados estruturados, em modelo chave-valor, contendo as palavras que ocorrem em dado documento e sua frequência no mesmo.

Uma representação de um documento neste modelo pode ser definida como $d = (a_1, a_2, ..., a_N)$, onde d é o vetor de palavras do documento e a_n corresponde ao n-ésimo termo no documento com sua cardinalidade.

Figura 4 – Frequência textual como vetor Bag of words



Fonte: (BENGFORT; BILBRO; OJEDA, 2018)

2.5 DOC2VEC

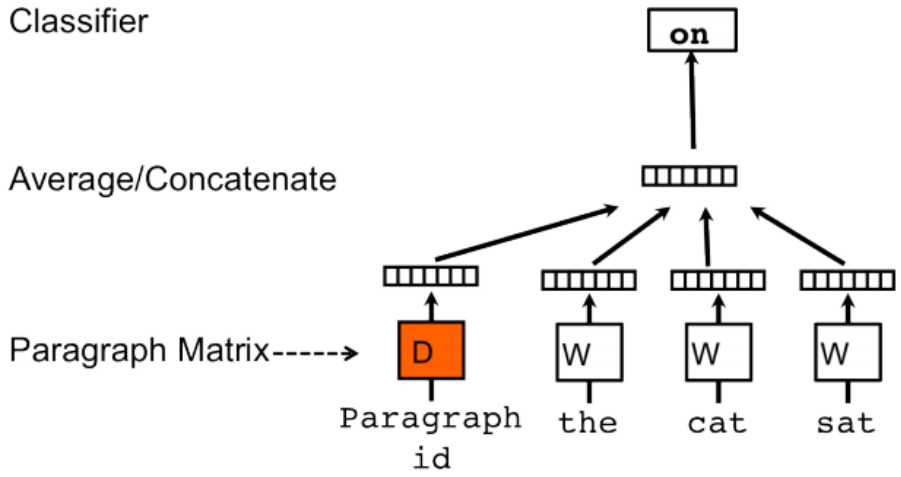
Doc2Vec (do inglês *document to vector*) é um conceito apresentado em 2014 por Quoc Le e Tomas Mikolov como "*Paragraph Vector*"(LE; MIKOLOV, 2014). Conforme os autores, diversos algoritmos de *machine learning* necessitam de uma entrada em um formato de vetor de atributos de tamanho fixo. Uma das formas mais comuns para essa tarefa é o método de saco de palavras (*bag of words*), porém dois problemas surgem nele: a perda da ordenação de palavras e a marginalização da semântica. Quanto a ordem, nota-se que pela estrutura do modelo, sentenças diferentes que possuam as mesmas palavras obtém a mesma representação, mesmo que seus sentidos sejam divergentes. Quanto a semântica, o artigo apresenta que palavras como *powerful* (poderoso/potente), *strong* (forte/vigoroso) e *Paris* estão igualmente distantes, apesar da semelhança semântica entre *powerful* e *strong*. O estudo propôs, então, um algoritmo não supervisionado que assimila uma representação de atributos de tamanho fixo a partir de textos de tamanho variado.

Os autores afirmam que o *Paragraph Vector* é inspirado em métodos anteriores de aprendizado de vetor de palavras. Segundo eles, apesar do fato de que o vetor de palavras seja iniciado randomicamente, eventualmente ele irá capturar semânticas como um resultado indireto na tarefa de predição. Portanto, essa ideia é utilizada também no algoritmo proposto por Le e Mikolov (2014).

Na estrutura do Doc2Vec (ilustrada na Figura 5), cada parágrafo é mapeado para um único vetor, representado na Figura 5 pela matriz "D". Cada palavra também é mapeada para um vetor único, representado na Figura 5 pelas matrizes "W". Desse conjunto de vetores é feita a média e concatenado para predição da próxima palavra no contexto. Esse método de vetor de parágrafo é chamado de versão de memória distribuída do vetor de parágrafo (*Distributed Memory version of Paragraph Vector*), ou PV-DM.

Outra versão do Doc2Vec é ilustrada na Figura 6. Segundo os autores, nessa versão

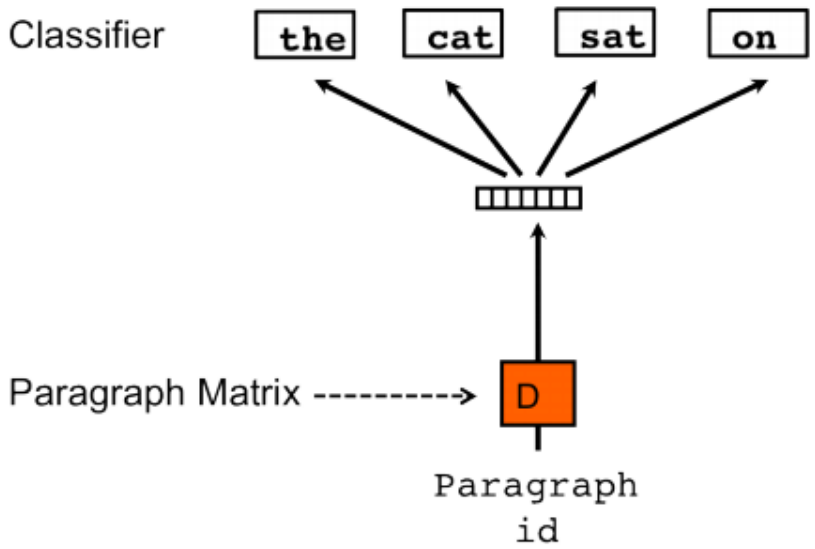
Figura 5 – Representação da estrutura para aprendizado de vetor de parágrafo, modelo PV-DM



Fonte: (LE; MIKOLOV, 2014)

o contexto de palavras na entrada é ignorado, porém o modelo é forçado a prever palavras randomicamente selecionadas do parágrafo. Esse método de vetor de parágrafo é chamado de versão de saco de palavras do vetor de parágrafo (*Bag of Words version of Paragraph Vector*), ou PV-DBOW.

Figura 6 – Representação da estrutura para aprendizado de vetor de parágrafo, modelo PV-DBOW



Fonte: (LE; MIKOLOV, 2014)

Após a etapa de treinamento, os vetores de parágrafo resultantes podem ser utilizados como atributos do parágrafo e, consequentemente, entradas para métodos convencionais de *machine learning* como regressão, SVM e *K-means*.

Como resultado do trabalho, Le e Mikolov (2014) descrevem o algoritmo como competitivo com os métodos de estado da arte na área, com os bons resultados demonstrando mérito na captura de semântica dos parágrafos. Os autores ainda destacam o potencial do algoritmo de superar fraquezas descritas nos modelos *bag of words*.

2.6 MÉTRICAS PARA AVALIAÇÃO DE DESEMPENHO DE CLASSIFICADORES

Uma série de métricas foram escolhidas para avaliar o desempenho dos classificadores submetidos ao *Clickbait Challenge*, como erro quadrático médio (MSE), *F1-Score*, precisão, recall e acurácia. Destas, o MSE foi definido como primário na avaliação dos resultados, ficando o resto das métricas computadas disponíveis como métricas de avaliação adicional, conforme a organização do desafio.

2.6.1 Matriz de Confusão

Para entender e interpretar as informações obtidas a partir das métricas, é obrigatório a compreensão dos conceitos empregados na Matriz de Confusão.

A Matriz de Confusão é uma tabela que indica erros e acertos a partir dos resultados de um modelo, comparando resultando previsto com o resultado esperado. Um exemplo desta matriz é apresentado na Figura 7, abaixo.

Figura 7 – Exemplo de Matriz de Confusão

		Resultado previsto	
		Sim	**Não**
Resultado esperado	**Sim**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Não	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Fonte: do autor (2020)

2.6.2 Erro quadrático médio (MSE)

Em estatística, o erro quadrático médio (MSE, de *Mean Squared Error*) calcula a média da diferença entre os valores reais e os valores previstos, ao quadrado. Em uma representação visual, pode-se considerar como o quão perto uma linha de predição está dos pontos que representam os valores reais. Portanto, quanto menor o valor de MSE, mais próximas do correto estão as predições.

Sua equação matemática é definida como $MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_i - y'_i)^2$, onde $\frac{1}{n}\Sigma_{i=1}^{n}$ corresponde à média do somatório do quadrado dos erros - descrito por $(y_i - y'_i)^2$.

2.6.3 Precisão

Também citada como valor preditivo positivo (*positive predictive values*), é a métrica que define a quantidade dos valores que foram classificados como corretos, quantos realmente eram. Pode-se dizer que uma alta precisão significa que o modelo retornou mais resultados relevantes que irrelevantes, sendo considerada uma média de exatidão, portanto.

Sua equação pode ser definida como $Precisão = \frac{VerdadeiroPositivo}{VerdadeiroPrevisto+FalsoPositivo}$.

2.6.4 Recall

O *recall* (revocação em português, também citado como sensibilidade) apresenta a frequência em que o modelo encontra corretamente os valores de uma certa classe. A métrica fornece as características de completude e quantidade, da mesma forma que Precisão apresenta para exatidão e qualidade. Um valor alto mostra que o modelo retornou a maioria dos resultados relevantes.

Sua equação pode ser definida como $Recall = \frac{VerdadeiroPositivo}{VerdadeiroPositivo+FalsoNegativo}$.

2.6.5 F1-Score

O *F1-Score* (também referenciado como *F-score* ou *F-measure*) é uma métrica da análise estatística que busca combinar os resultados de precisão e *recall* para trazer um único indicativo da qualidade geral de um modelo. Quanto maior o valor, melhor preparado está o modelo.

Sua fórmula é definida como $F1 = \frac{2 * precisão * recall}{precisão + recall}$.

2.6.6 Acurácia

A acurácia é um indicador geral de como performou o classificador - com que frequência está correto. Em termos simples, é o número de predições realizadas com sucesso, frente ao tamanho total trabalhado.

Sua equação pode ser definida como $Acurácia = \frac{VerdadeiroPositivo+VerdadeiroNegativo}{Total}$.

3 TRABALHOS CORRELATOS

O problema apresentado nesse estudo é recente, fruto do crescimento de usuários com acesso à internet e a popularização das redes sociais. Dito isso, nota-se que a maior parte da literatura disponível deste assunto em específico é posterior a 2016. Assim, existem poucos conjuntos de dados (*corpus*) disponíveis publicamente voltados para a classificação de postagens clickbait e, em sua maioria, são disponibilizados em língua inglesa.

3.1 WEBIS CLICKBAIT CORPUS

No ano de 2016 foi criado o dataset nomeado *Webis Clickbait Corpus 2016*, por Potthast et al. (2016). Nele, os autores agruparam uma série de postagens da rede social Twitter, provenientes dos 20 perfis de editores mais prolíficos, baseado em sua influência em termo de recompartilhamentos. Foram incluídos, portanto, diversos gigantes das notícias estadunidense, como New York Times e Wall Street Journal, sites de matérias variadas, como BuzzFeed e Yahoo!, entre outros. No processo de criação do dataset, cada postagem foi analisada e classificada por três assessores. A classificação foi baseada no texto da postagem e na imagem da postagem - sem acessar o link da matéria. Segundo os autores, os resultados da classificação mostraram que todos os 20 perfis selecionados faziam uso de *clickbait*. Ao todo, foram compilados 2992 na criação do dataset, dos quais 767 (aproximadamente 26%) foram classificados como *clickbait*.

O dataset ainda viria a ser expandido em Potthast et al. (2018b), com a formulação de um corpus com 38517 postagens do twitter devidamente anotadas. Desta vez, foram usados os 27 perfis com mais recompartilhamentos, cobrindo um período de 150 dias. A criação deste dataset foi, em específico, para uso no *Clickbait Challenge 2017*.

3.2 CLICKBAIT CHALLENGE

Em 2017 foi realizado o *Clickbait Challenge*, um desafio na área da detecção de postagens *clickbait* em redes sociais. A tarefa do desafio foi desenvolver um classificador que classifique o quão *clickbait* uma postagem é (*clickbait force*), com base em informações como texto de chamada na postagem e conteúdo da notícias em si. Conforme Potthast et al. (2018a), o objetivo principal do desafio foi impulsionar a pesquisa e desenvolvimento nas tarefas de detecção automática de *clickbait* nas redes sociais.

No site do desafio *Clickbait Challenge* encontra-se o resultado de 16 detectores submetidos com sucesso ao desafio. Segundo Potthast et al. (2018a), melhoras significativas foram alcançadas comparado ao anterior estado da arte, em termos de performance

na detecção. Dos 16 detectores submetidos, 10 disponibilizaram um artigo detalhando seu trabalho, assim como 8 disponibilizaram o código fonte utilizado na resolução do problema. Conforme os autores, novos registros ainda são aceitos como forma de apoio a pesquisa de melhores classificadores *clickbait*. Dos resultados submetidos com sucesso ao desafio e disponibilizados no site, os dois melhores resultados obtidos baseados na métrica definida como primária (MSE) são o de Omidvar, Jiang e An (2018), com MSE de 0.032, e Zhou (2017), com MSE de 0.033, conforme Tabela 1.

Tabela 1 – Comparativo dos detectores submetidos com sucesso ao *Clickbait Challenge*

MSE	F1	Precisão	*Recall*	Acurácia	Nome da equipe
0.032	0.670	0.732	0.619	0.855	albacore
0.033	**0.683**	0.719	0.650	**0.856**	zingel
0.034	0.679	0.717	0.645	0.855	anchovy
0.036	0.641	0.714	0.581	0.845	emperor
0.036	0.036	0.728	0.568	0.847	carpetshark
0.039	0.656	0.659	0.654	0.837	arowana
0.041	0.631	0.642	0.621	0.827	pineapplefish
0.043	0.565	0.699	0.474	0.826	whitebait
0.044	0.552	0.758	0.434	0.832	clickbait17-baseline
0.045	0.604	0.711	0.524	0.836	pike
0.046	0.654	0.654	0.653	0.835	tuna
0.079	0.650	0.530	**0.841**	0.785	torpedo
0.099	0.023	**0.779**	0.012	0.764	houndshark
0.118	0.467	0.380	0.605	0.671	dory
0.174	0.261	0.167	0.593	0.209	salmon
0.252	0.434	0.287	0.893	0.446	snapper

Fonte: Clickbait Challenge (2017)

Zhou (2017), na submissão com o segundo melhor resultado do *Clickbait Challenge* no momento da escrita deste trabalho, relata tratar o problema de regressão como um problema de multi-classificação. Como solução, foi desenvolvido um modelo que fez uso de mecanismos do tipo *self-attentive neural network*. Conforme o autor, diferente do habitual *external-attentive network*, o método utilizado não requer informações externas para aprender o vetor de contexto (*context vector*). Além da segunda colocação no desafio por uma pequena diferença do primeiro colocado na métrica primária do desafio (MSE), o classificador da equipe obteve o melhor resultado nas métricas de *F1 Score* e Acurácia, além de um ótimo tempo para execução (00:03:27).

Omidvar, Jiang e An (2018) treinaram e testaram classificadores com diferentes técnicas de *deep-learning* em seu trabalho para encontrar um modelo com resultados satisfatórios na detecção de *clickbaits*. Dentre os resultados das tentativas propostas no trabalho, o modelo que faz utilização de redes neurais recorrentes GRU (*Gated Recurrent Unit*) bi-direcionais teve o melhor resultado, obtendo inclusive a primeira colocação no *Clickbait Challenge* até o momento de escrita do presente trabalho. Conforme os autores,

o modelo proposto não depende de engenharia de atributos, significando que é capaz de aprender a representação automaticamente a fim de classificar as postagens como *clickbait* ou não *clickbait*. O modelo utiliza somente o campo "*postText*", ao contrário de outros submetidos ao desafio, que usavam mais informações (imagens, *keywords*, matérias relacionadas).

4 DESENVOLVIMENTO

O modelo de classificador desenvolvido no presente trabalho tem como base o algoritmo detector de clickbait "Albacore", submetido ao *Clickbait Challenge* no ano de 2017 e atual primeiro colocado pelas métricas do desafio, conforme Tabela 1.

Assim como no trabalho de Omidvar, Jiang e An (2018), o *dataset* utilizado para teste foi obtido a partir do *dataset* disponibilizado para treino. O *dataset* oficial para testes não está publicado abertamente, ficando disponível somente através da plataforma *tira.io* (TIRA, 2020) mediante formulário de cadastro. Foi solicitada a participação para o desenvolvimento deste trabalho, porém não houve retorno em tempo hábil.

4.1 CONFIGURAÇÃO DO AMBIENTE

As configurações de hardware e sistema da máquina utilizadas na reprodução do trabalho Albacore e desenvolvimento do experimento do artigo são apresentadas na Tabela 2. O código reproduzido e, consequentemente, o modelo desenvolvido a partir dele utiliza Python como linguagem de programação, junto de bibliotecas específicas para *machine-learning* que serão apresentadas a seguir.

Tabela 2 – Configurações de hardware e sistema da máquina utilizada para reprodução e desenvolvimento

Componente	**Descrição**
Sistema Operacional	Pop!_OS 20.10 (baseado em Ubuntu 20.10)
Processador	Intel i7-9750H
GPU	NVIDIA GeForce GTX 1660 Ti Mobile
Memória Ram	32 GB
Linguagem de Programação	Python 3.8.6

Fonte: do autor

4.2 ANÁLISE DA BASE DE DADOS

A base de dados utilizada no *Clickbait Challenge* (e consequentemente neste trabalho) é denominada *Webis Clickbait Corpus* e consiste em 38517 postagens da rede social Twitter, restritas a língua inglesa (POTTHAST et al., 2018b). Cada postagem, também chamada de *tweet*, é uma mensagem curta de até 140 caracteres que pode ser acompanhada de uma imagem e um link. Através da API do Twitter, os *tweets* foram obtidos a partir dos perfis de editoras de notícias com a maior quantidade de recompartilhamentos em um que se iniciou em 1º de Dezembro de 2016, até 30 de Abril de 2017. Conforme as informação dos datasets providos pelo desafio exibidos na Tabela 3, as 38517 postagens

foram divididas em uma dataset com 19538 postagens para treino e validação (dos quais 4761 são clickbaits).

As 18979 postagens restantes são utilizadas em teste no servidor próprio para o teste de submissões do *Clickbait Challenge*, disponível mediante cadastro. Os dados contidos nele são privados e não estão disponíveis publicamente.

Tabela 3 – Detalhes dos datasets de validação e teste do *Clickbait Challenge*

Dataset	Tweets	*Clickbait*	Não *Clickbait*
Validação	19538	4761	14777
Teste	18979	4515	14464

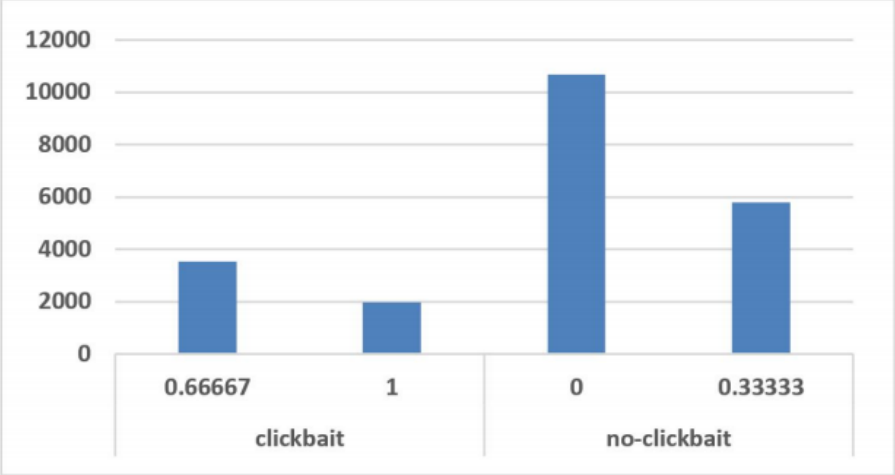
Fonte: Adaptado de (OMIDVAR; JIANG; AN, 2018)

Cada uma destas postagens foi anotada quanto a intensidade de *clickbait* atribuida por uma equipe de cinco avaliadores. Uma escala Likert de quatro pontos foi utilizada para classificar as postagens com os seguintes rótulos:

- *Não clickbait (0.0).*
- *Levemente clickbait (0.33).*
- *Consideravelmente clickbait (0.66).*
- *Fortemente clickbait (1.0).*

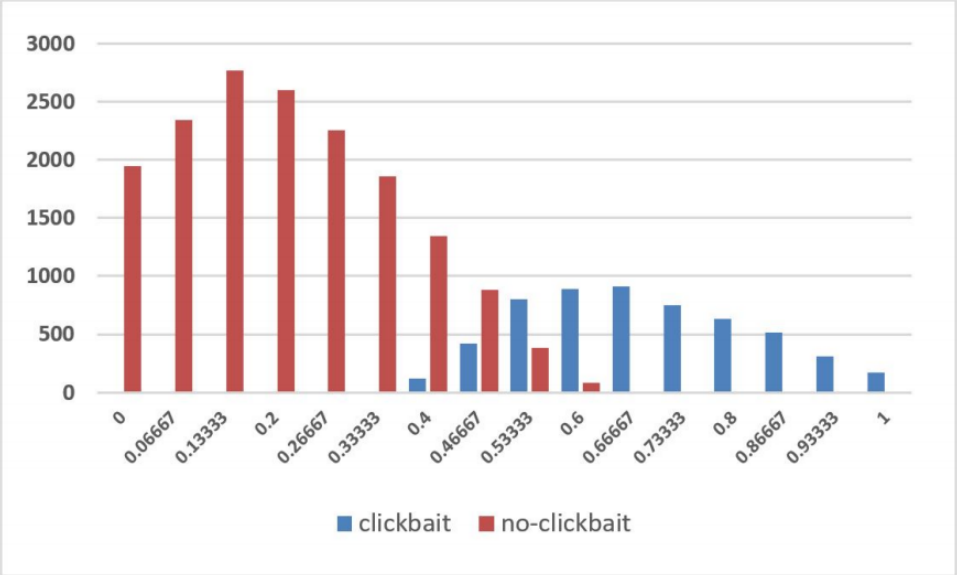
Conforme analisado por Omidvar, Jiang e An (2018), há um desequilíbrio nos *datasets* quanto a quantidade de registros *clickbait* e não *clickbait* (considerando não *clickbait* quando o valor mediano das cinco avaliações é menor ou igual a 0.33333), como pode ser verificado nas Figuras 8 e 9.

Figura 8 – Quantidade total de *tweets* baseado no valor mediano das avaliações



Fonte: (OMIDVAR; JIANG; AN, 2018)

Figura 9 – Distribuição dos *tweets* baseado no valor mediano das avaliações



Fonte: (OMIDVAR; JIANG; AN, 2018)

Os registros dos *datasets* disponibilizados publicamente estão em formato arquivos *JSON Lines* (arquivos de extensão "jsonl"). Um arquivo JSON (*JavaScript Object Notation*) é um formato compacto para armazenamento e troca de dados. *JSON Lines*, por sua vez, consiste em arquivos onde cada linha é um objeto JSON válido, separados por um caractere de nova linha (`\n`). O *dataset* disponibilizado oferece os seguintes arquivos de recurso:

- instances.jsonl: Arquivo *JSON Lines* em que cada objeto contém informações extraídas de uma postagem específica e o artigo/matéria de destino. A estrutura deste arquivo pode ser vista na Figura 10.
- truth.jsonl: Arquivo *JSON Lines* em que cada objeto contém as informações de avaliação para uma postagem específica. A estrutura deste arquivo é demonstrada na Figura 11.
- media/: Pasta contendo todas as imagens de postagem referenciadas no arquivo instances.jsonl.

Figura 10 – Amostra de objeto JSON com as informações de conteúdo de uma matéria, obtido no arquivo instances.jsonl do *dataset* disponibilizado


```
{
"id": "608999590243741697",
"postTimestamp": "Thu Jun 11 14:09:51 +0000 2015",
"postText": ["Some people are such food snobs"],
"postMedia": ["608999590243741697.png"],
"targetTitle": "Some people are such food snobs",
"targetDescription": "You'll never guess one...",
"targetKeywords": "food, foodfront, food waste...",
"targetParagraphs": [
"What a drag it is, eating kale that isn't ...",
"A new study, published this Wednesday by ...",
...],
"targetCaptions": ["(Flikr/USDA)"]
}
```


Fonte: (WEBISEVENTS, 2020)

Figura 11 – Amostra de objeto JSON com as informações de classificação de uma matéria, obtido no arquivo truth.jsonl do *dataset* disponibilizado


```
{
"id": "608999590243741697",
"truthJudgments": [0.33, 1.0, 1.0, 0.66, 0.33],
"truthMean"  : 0.6666667,
"truthMedian": 0.6666667,
"truthMode"  : 1.0,
"truthClass" : "clickbait"
}
```


Fonte: (WEBISEVENTS, 2020)

A Tabela 4 demonstra os campos disponíveis e respectiva descrição na estrutura dos recursos instances.jsonl e truth.jsonl do *dataset*.

Tabela 4 – Estrutura do dataset utilizado no *Clickbait Challenge*

Campo	Descrição
ID	Identificador único do artigo no dataset
postTimestamp	*Timestamp* (data) da postagem
postText	O texto da postagem de divulgação do artigo
postMedia	A imagem utilizada na divulgação do artigo
targetTitle	Título do artigo
targetDescription	Descrição do artigo
targetKeywords	Palavras-chave do artigo
targetParagraphs	Conteúdo do artigo
targetCaptions	Legendas do artigo
truthJudgements	Vetor com a nota das cinco avaliações de cada artigo
truthMean	Média das notas de avaliação
truthMedian	Mediana das notas de avaliação
truthClass	Classe binária indicando matéria como clickbait ou não clickbait

À parte dos aspectos técnicos dos datasets criados para o *Clickbait Challenge*, Omidvar, Jiang e An (2018) relata ainda possíveis problemas relativos a forma como foram realizadas as avaliações das postagens quanto a seu conteúdo:

> *One of the issue regarding the data set is that while the evaluators are provided with both the post text and a link to the target article, they were not obliged to read the actual article. So, we can consider the judgements are just based on the post texts.*

No mesmo artigo, os autores questionam ainda a capacidade dos cinco avaliadores para a tarefa de classificação, dado o conhecimento e *background* nos tópicos:

> *Also, the scores for the tweets are dependent to the evaluator's background, knowledge, and topics of interest. So, there should be some noises in the dataset because of the existing differences between evaluators.*

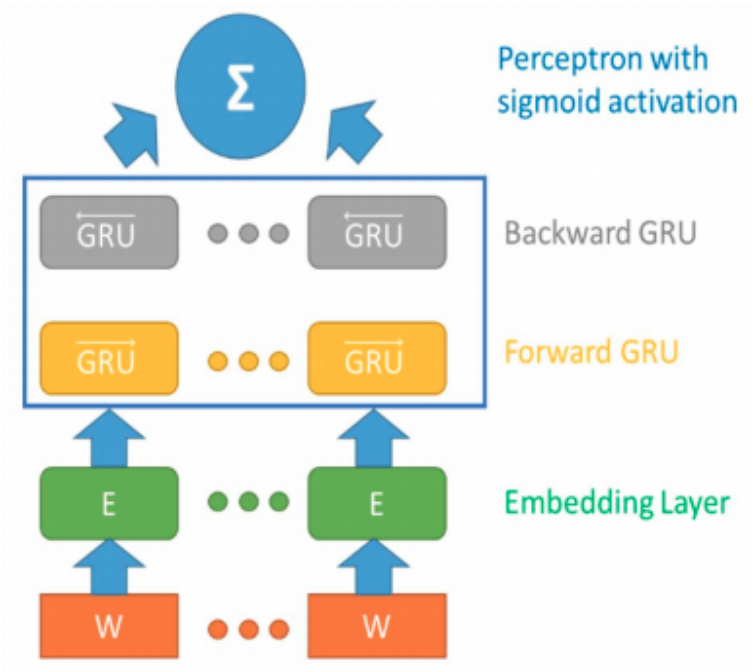
4.3 REPRODUÇÃO E ANÁLISE DO CÓDIGO BASE

No presente trabalho foi realizada a reprodução do classificador Albacore, submetido ao *Clickbait Challenge* por Omidvar, Jiang e An (2018) e primeiro colocado no desafio, considerando a métrica primária MSE. Para chegar ao classificador final submetido ao desafio, a equipe treinou e testou classificadores com diferentes técnicas de *deep-learning* e seus respectivos hiperparâmetros. Esta seção visa apresentar a abordagem final utilizada no classificador Albacore para submissão.

Conforme os autores, o *dataset* de testes foi criado a partir do *dataset* de validação, visto que o oficial não havia sido liberado publicamente. Para isso, foi utilizada a técnica de amostragem estratificada com 30% dos registros totais.

O modelo final proposto pela equipe é apresentado na Figura 12, e consiste em uma camada de *embedding*, seguida de uma camada de GRU bi-direcional, finalizando em uma camada única de modelo *Perceptron*.

Figura 12 – Abordagem final do modelo Albacore de detecção de *clickbait*



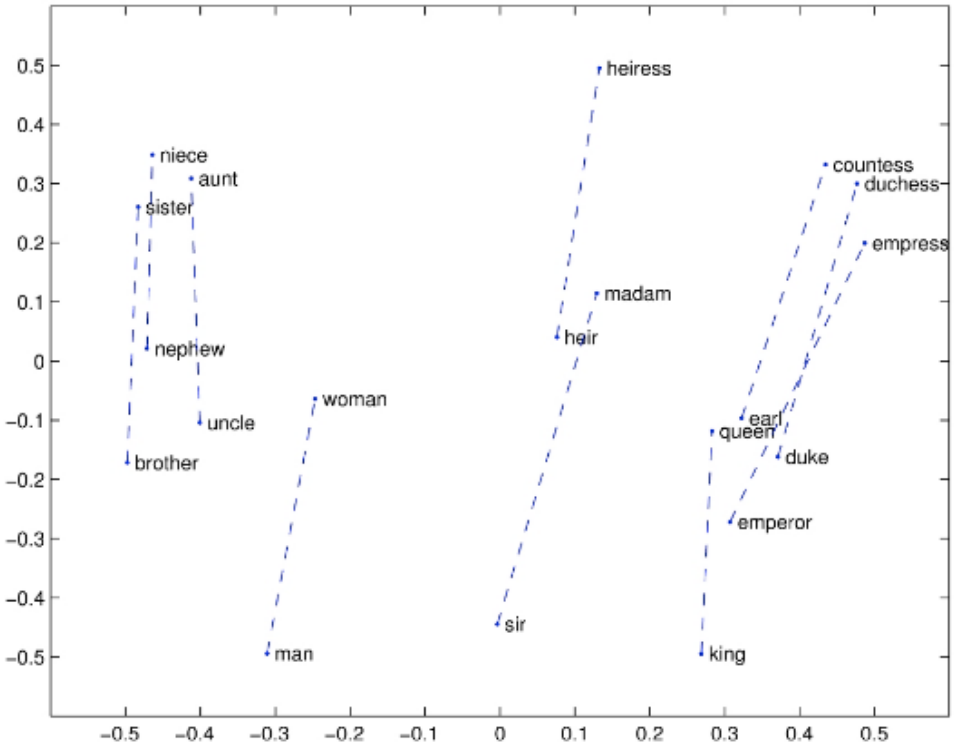
Fonte: (OMIDVAR; JIANG; AN, 2018)

4.3.1 Camada de embedding

A camada de *embedding* do modelo final do Albacore é responsável por transformar representações *one-hot* das palavras de entrada - onde cada palavra é um indíce - em representações vetoriais densas.

No modelo submetido foram utilizados os embeddings pré-treinados GloVe - *Global Vectors* (PENNINGTON; SOCHER; MANNING, 2014). GloVe é uma técnica de representação vetorial de palavras que busca mapeá-las em um espaço em que a distância entre as palavras é relacionada a sua similaridade semântica. Uma representação visual desta representação é apresentada na Figura 13.

Figura 13 – Representação visual da técnica de embedding GloVe



Fonte: (PENNINGTON; SOCHER; MANNING, 2014)

Os embeddings pré-treinados utilizados no modelo possuem uma quantidade enorme de vetores de palavras, com um total de 6 bilhões de tokens, 400 mil vocabulários, com vetores de 50, 100, 200 e 300 dimensões para cada um destes, em arquivo disponibilizado sob licença PDDL - *Public Domain Dedication and License* - para uso livre das informações contidas nele. Nos testes realizados pela equipe, o modelo com GloVe de 100 dimensões foi o que obteve o melhor resultado.

4.3.2 Camada GRU

Para a segunda camada, a equipe responsável pelo classificador Albacore realizou testes e comparou os resultados entre SRU bi-direcional (ELMAN, 1990), GRU bi-direcional e LSTM bi-direcional. Os autores afirmam então que, dentre estes, a técnica de GRU bi-direcional foi a que mostrou os melhores resultados para detecção *clickbait*.

Tanto LSTM quanto GRU são técnicas de redes neurais que buscam resolver um problema comum no aprendizado de máquina em sequências longas, que é a chamada memória de curto prazo. Em redes neurais recorrentes, a medida que o treino acontece os valores de gradiente vão sendo alterados para atualizar e guiar os pesos na rede. Ao longo do tempo, pode ocorrer a diminuição deste gradiente, fazendo com que camadas que não recebam atualizações "parem de aprender". Portanto, a RNN pode "esquecer"o que foi visto no início de uma longa sequência (DATASCIENCEACADEMY, 2019).

Para resolver este problema, GRU faz uso de dois vetores - chamados *update gate* e *reset gate* - responsáveis por decidir o que será passado para a saída da rede. O que torna esses vetores essenciais para a técnica é a capacidade de manter informações passadas sem que se dissípem com o tempo. O *reset gate* aprende quais informações/atributos precisam ser esquecidas, enquanto o *update gate* aprende quais informações salvas devem ser atualizadas a partir do *input*.

Quanto a característica de bidirecionalidade empregada no modelo, corresponde a técnica utilizada em redes neurais para que as predições em uma sequência possuam contexto tanto do passado quanto do futuro.

4.3.3 Camada Perceptron

Por fim, para determinar o quão *clickbait* é uma postagem, a equipe utilizou uma camada única Perceptron com ativação de função sigmóide.

O funcionamento desta camada consiste em um cálculo realizado com base na soma dos pesos dos vetores de entrada multiplicado pelos seus pesos, sendo este resultado utilizado na função de ativação. No trabalho em questão, a função de ativação utilizada é a sigmóide, caracterizada pela equação $f(x) = \frac{1}{1+e^{-x}}$, sendo responsável por produzir um valor final entre 0 e 1.

Vale lembrar que este valor é usado, então, para definir a classe prevista, com base nas classes disponíveis, por meio de seus valores lineares:

- *Não clickbait (0.0).*
- *Levemente clickbait (0.33).*
- *Consideravelmente clickbait (0.66).*
- *Fortemente clickbait (1.0).*

4.4 MODELO DESENVOLVIDO

Conforme mencionado na seção anterior, o modelo desenvolvido para este trabalho utiliza como base o classificador Albacore de Omidvar, Jiang e An (2018). Partindo do código base, foram realizadas alterações principalmente na camada de *embedding*. A camada de *embedding* é de grande importância para as tarefas de *machine learning*, dada a sua utilidade na criação de representações densas das informações de entrada.

A hipótese testada consiste no uso de Doc2Vec para a criação dos vetores de *embedding*. Para a utilização de Doc2Vec na camada de *embedding* do modelo proposto foi utilizado o módulo de mesmo nome da biblioteca Gensim (GENSIM, 2020). Gensim é uma

biblioteca *open-source* para modelagem não supervisionada de tópicos e Processamento de Linguagem Natural, disponível gratuitamente para Python.

Na utilização do Doc2Vec disponibilizado pela biblioteca Gensim foi necessário, inicialmente, realizar a estruturação dos dados de entrada para criação do modelo. Para isso, as informações do dataset de validação foram carregados e, utilizando do campo de texto da postagem (*postText*), foi criada uma lista nomeada *sentences*, onde cada registro é o texto de uma postagem após o processo de tokenização. Tokenização é a prática de segmentar um texto em partes com significado único. Neste processo, o texto da postagem "Como vai seu dia?", por exemplo, seria transformada em uma lista de palavras *[como, vai, seu, dia]*. Para a finalização da estrutura dos dados de entrada, é preciso que os itens da lista sejam convertidos para o formato *TaggedDocument*, representando um documento junto a uma tag, formato de entrada do Doc2Vec, conforme documentação (GENSIM, 2020). A rotina de estruturação dos dados de entrada está disponível no trecho de código disponibilizado na Figura 14.

Figura 14 – Trecho de código responsável pela estruturação dos dados de entrada para o Doc2Vec no classificador desenvolvido

```
# Estruturação e tokenization dos documentos
sentences = []
vocab = Counter()
files = ["data/clickbait17-validation"]
for each_fp in files:
    with open(os.path.join(each_fp, 'instances.jsonl'), 'rb') as f:
        for each_line in f:
            each_item = json.loads(each_line.decode('utf-8'))
            for each_sentence in each_item["postText"]:
                aux = tokenise(each_sentence)
                sentences.append(aux)
                vocab.update(aux)

# Convertendo os documentos tokenizados para o formato "gensim tagged data"
trainingData = list(tagged_document(sentences))
```

Fonte: do autor

Após isso, pode-se iniciar o processo de criação do modelo Doc2Vec propriamente dito. Seguindo os padrões do desenvolvimento realizado no código base por Omidvar, Jiang e An (2018), foram testados diversos hiperparâmetros nos testes realizados com o Doc2Vec. Conforme Figura 15, foram mantidos os hiperparâmetros que obtiveram o melhor valor na métrica MSE nos testes realizados. Os seguintes parâmetros foram testados para se obter o melhor resultado:

- *dm*: Define o algoritmo, sendo "1"para PV-DM (Distributed Memory version of Paragraph Vector) ou "0"para PV-DBOW (Bag of Words version of Paragraph Vector).

- *vector_size*: Dimensionalidade do vetor de atributos.
- *epochs*: Quantidade de iterações sobre o corpus.

Figura 15 – Trecho de código responsável pela inicialização do modelo Doc2Vec no classificador desenvolvido

```
# Treinando o modelo Doc2Vec
model = Doc2Vec(trainingData, dm=1, vector_size=20, window=2, workers=4, epochs = 100)

# Salvando o modelo gerado para uso posterior
model.save("s_clickbait.100_doc2vec.model")

# Carregando o modelo salvo
doc2vecmodel = Doc2Vec.load("s_clickbait.100_doc2vec.model")
```

Fonte: do autor

Por fim, a Figura 16 contém o trecho de código central do funcionamento do classificador desenvolvido. O trecho, num contexto geral, pouco difere do apresentado inicialmente pela equipe de Albacore. O fluxo consiste na criação de uma pilha linear (método *Sequential*). Nela são adicionadas as camadas de *embedding*, GRU bidirecional e ativação sigmóide, estrutura descrito previamente neste trabalho.Para isso, o código do classificador utiliza as funcionalidades da ferramenta Keras (KERAS, 2020) na criação de suas redes neurais.

Figura 16 – Trecho principal do código do classificador desenvolvido

```
model = Sequential()
model.add(Embedding(input_dim = nb_words,
                    output_dim = WV_DIM,
                    mask_zero = False,
                    weights = [wv_matrix],
                    input_length = MAX_SEQUENCE_LENGTH+1
                    ,trainable = False))
model.add(Dropout(dropout_embedding))
model.add(Bidirectional(GRU(512, dropout=0.2, recurrent_dropout=0.5)))
model.add(Dense(1))
model.add(Activation('sigmoid'))
model.compile(metrics=['mean_squared_error'],
              loss='mse',
              optimizer='rmsprop')

batch_size = 64
earlystop_cb = keras.callbacks.EarlyStopping(monitor='mean_squared_error', patience=7, verbose=1, mode='auto')
model.fit(X_train, y_train, batch_size=batch_size, epochs=20,
          validation_split=0.1, callbacks=[earlystop_cb])

petruth_means = model.predict(X_test)
```

Fonte: do autor

Além da alteração na camada de *embedding*, foram necessárias outras alterações e ajustes para que tanto o código base quanto o modelo desenvolvido executasse com sucesso, visto que houve mudanças nas versões de diversas das bibliotecas utilizadas inicialmente na submissão ao desafio *Clickbait Challenge*. Ademais, o fluxo da arquitetura utilizado na abordagem adaptada deste trabalho segue a mesma abordagem já exemplificada na Figura 12.

4.5 RESULTADOS

Nesta seção são apresentados os resultados obtidos a partir do modelo desenvolvido, comparando-o primeiramente com o modelo utilizado como base, Albacore, e posteriormente com os demais resultados submetidos ao *Clickbait Challenge*.

Para avaliação do modelo desenvolvido são utilizados as mesmas métricas do *Clickbait Challenge*, já descritas anteriormente: MSE como métrica primária, seguido de *F1-Score*, Precisão, Recall e Acurácia como métricas secundárias para avaliação adicional.

Conforme detalhado na Tabela 5, comparado ao classificador base, o modelo desenvolvido obteve métricas semelhantes, porém levemente inferiores. Dentre as métricas, houve um aumento somente em uma delas, sendo a de precisão. A métrica primária MSE piorou de 0.0319, da reprodução do modelo, para 0.0428 no modelo desenvolvido.

Tabela 5 – Comparativo dos detectores utilizados de base para o trabalho e detector desenvolvido

MSE	**F1**	**Precisão**	***Recall***	**Acurácia**	**Nome da equipe**
0.0319	0.656	0.718	0.604	0.843	albacore - reprodução
0.0320	**0.670**	0.732	**0.619**	**0.855**	albacore
0.0428	0.617	**0.744**	0.589	0.713	**Modelo desenvolvido**

Fonte: do autor, adaptado de Clickbait Challenge (2017)

Com os resultados obtidos, o modelo não é capaz de disputar como melhor classificador dada a métrica MSE. Entretanto, o modelo utilizando *embeddings* Doc2Vec ainda assim obteve resultados competitivos em relação a modelos desenvolvidos e submetidos previamente ao *Clickbait Challenge*, demonstrando ser um estudo válido para a área. Um comparativo com os demais classificadores pode ser visto na Tabela 6.

É importante mencionar que, por mais próximos que possam ter sido os resultados, o resultado real somente poderia ser aferido ao submeter o classificador desenvolvido para o ambiente fechado tira.io, onde ocorre o desafio compartilhado do *Clickbait Challenge*, fazendo uso dos *datasets* oficiais para o teste, bem como estrutura controlada. Conforme mencionado anteriormente, foi feita a tentativa de cadastro via formulário, porém não houve retorno em tempo hábil.

Tabela 6 – Comparativo dos detectores submetidos com sucesso ao *Clickbait Challenge* e detector desenvolvido

MSE	**F1**	**Precisão**	***Recall***	**Acurácia**	**Nome da equipe**
0.032	0.670	0.732	0.619	0.855	albacore
0.033	**0.683**	0.719	0.650	**0.856**	zingel
0.034	0.679	0.717	0.645	0.855	anchovy
0.036	0.641	0.714	0.581	0.845	emperor
0.036	0.036	0.728	0.568	0.847	carpetshark
0.039	0.656	0.659	0.654	0.837	arowana
0.041	0.631	0.642	0.621	0.827	pineapplefish
0.042	0.617	0.744	0.589	0.713	**Modelo desenvolvido**
0.043	0.565	0.699	0.474	0.826	whitebait
0.044	0.552	0.758	0.434	0.832	clickbait17-baseline
0.045	0.604	0.711	0.524	0.836	pike
0.046	0.654	0.654	0.653	0.835	tuna
0.079	0.650	0.530	**0.841**	0.785	torpedo
0.099	0.023	**0.779**	0.012	0.764	houndshark
0.118	0.467	0.380	0.605	0.671	dory
0.174	0.261	0.167	0.593	0.209	salmon
0.252	0.434	0.287	0.893	0.446	snapper

Fonte: do autor, adaptado de Clickbait Challenge (2017)

5 CONCLUSÃO

Conforme constatado na revisão bibliográfica realizada neste estudo, a área de classificação de postagens realizadas em redes sociais quanto às suas características *clickbait* é relativamente nova. Sendo um campo de pesquisa atual, diversas propostas de melhoria e adaptações na resolução do problema vem sendo desenvolvidos. Uma iniciativa interessante, visando impulsionar esses estudos, é o *Clickbait Challenge*, um desafio na área da detecção de postagens *clickbait* em redes sociais realizado em 2017, ainda ativo para novas submissões.

O desafio *Clickbait Challenge* impulsionou o estudo na área de detecção *clickbait*, organizando e disponibilizando publicamente as submissões obtidas. Dentre os trabalhos disponíveis, o modelo de Omidvar, Jiang e An (2018), primeiro colocado até o momento da escrita, foi escolhido para replicação e adaptações no presente trabalho. Dentre oportunidades possíveis, foi optado por trabalhar com a camada de *embedding*, implementando uma versão com uso de Doc2Vec do modelo.

A reprodução do modelo Albacore realizada neste trabalho obteve valores próximos aos divulgados pela equipe para o *Clickbait Challenge*. As diferenças obtidas nas métricas podem ser facilmente justificáveis pela diferença de ambiente, visto que houveram atualizações em bibliotecas utilizadas nos códigos desde sua submissão, assim como diferenças no *dataset* utilizado na etapa de teste.

Os resultados obtidos a partir do uso de Doc2Vec atestam sua validade como uma alternativa de *embedding* na tarefa de classificação *clickbait*. O modelo desenvolvido se manteve competitivo em relação as submissões existentes no desafio, mesmo que não tenha alcançado as marcas do código base, atual primeiro colocado.

Visando trabalhos futuros para a área, alguns aspectos podem ser interessantes de se abordar:

- Realizar um estudo para trabalhar com o atributo de imagem das disponivel nas postagens disponibilizadas no *dataset*.
- Realizar testes com alternativas de otimizadores para a rede neural utilizada.
- Testar o uso de modelos pré-treinados na implementação do Doc2Vec.
- Aprofundar o estudo nas demais camadas do modelo, propondo técnicas alternativas às usadas.
- Ampliar dataset utilizado, assim como melhorar a forma de criação do mesmo. Omidvar, Jiang e An (2018) propõem pontos interessantes quanto a isso, em seu

artigo.

REFERÊNCIAS

AGGARWAL, C. C.; ZHAI, C. A survey of text classification algorithms. In: *Mining Text Data*. [S.l.: s.n.], 2012. p. 163–222. Citado 3 vezes nas páginas 19, 20 e 21.

AUBAID, A. M. A. M. Text classification using word embedding in rule-based methodologies: A systematic mapping. In: *in TEM Journal. Volume 7, Issue 4*. [S.l.: s.n.], 2018. p. 902–914. Citado 2 vezes nas páginas 18 e 21.

BENGFORT, B.; BILBRO, R.; OJEDA, T. *Applied Text Analysis with Python: Enabling Language-Aware Data Products with Machine Learning*. [S.l.]: "O'Reilly Media, Inc.", 2018. Citado na página 23.

BLOM, J. N.; HANSEN, K. R. Click bait: Forward-reference as lure in online news headlines. *Journal of Pragmatics*, Elsevier, v. 76, p. 87–100, 2015. Citado na página 15.

CHAKRABORTY, A. et al. Stop clickbait: Detecting and preventing clickbaits in online news media. In: IEEE. *2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. [S.l.], 2016. p. 9–16. Citado na página 15.

COVINGTON, M. A.; NUTE, D.; VELLINO, A. *Prolog Programming in Depth*. 1. ed. The University of Georgia, Athens, Georgia 30602–7415 U.S.A.: [s.n.], 1995. Citado 3 vezes nas páginas 17, 18 e 20.

DATASCIENCEACADEMY. *Deep Learning Book*. 2019. Disponível em: <http:/-/www.deeplearningbook.com.br/>. Acesso em: 20 out. 2020. Citado na página 36.

ELMAN, J. L. Finding structure in time. *Cognitive science*, Wiley Online Library, v. 14, n. 2, p. 179–211, 1990. Citado na página 36.

GASSER, M. Connectionism and universals of second language acquisition. In: . [S.l.]: Cambridge University Press, 1990. v. 12, n. 2, p. 179–199. Citado na página 19.

GEÇKIL, A. et al. Detecting clickbait on online news sites. In: *Putting Social Media and Networking Data in Practice for Education, Planning, Prediction and Recommendation*. [S.l.]: Springer, 2020. p. 199–211. Citado na página 11.

GENSIM. *Gensim: Topic modelling for humans*. 2020. Disponível em: <https:/-/radimrehurek.com/gensim/>. Acesso em: 12 out. 2020. Citado 2 vezes nas páginas 37 e 38.

GLENSKI, M. et al. Fishing for clickbaits in social images and texts with linguistically-infused neural network models. *arXiv preprint arXiv:1710.06390*, 2017. Citado na página 12.

GONçALVES, E. C. *Extração de Árvores de decisão com a ferramenta de data mining weka*. 2007. Disponível em: <http://www.devmedia.com.br/ extracao-de-arvores-de-decisao-com-a-ferramenta-de-data-mining-weka/>. Acesso em: 30 mai. 2020. Citado na página 21.

GRANIK, M.; MESYURA, V. Fake news detection using naive bayes classifier. In: *2017 IEEE First Ukraine Conference on Electrical and Computer Engineering (UKRCON)*. [S.l.: s.n.], 2017. Citado na página 19.

HARGREAVES, S. *Apple's stock hit by Web rumor: Online report that CEO Steve Jobs suffered a heart attack was false. SEC said to be investigating*. 2008. Disponível em: <https://money.cnn.com/2008/10/03/technology/apple/>. Acesso em: 4 abr. 2020. Citado na página 11.

KERAS. *Keras: the Python deep learning API*. 2020. Disponível em: <https://keras.io/>. Acesso em: 13 out. 2020. Citado na página 39.

KLAVANS, J. L.; RESNIK, P. *The balancing act: combining symbolic and statistical approaches to language*. [S.l.]: MIT press, 1996. Citado na página 19.

KNOCKLEIN, O. *Classification Using Neural Networks*. 2019. Disponível em: <https://towardsdatascience.com/classification-using-neural-networks-b8e98f3a904f>. Acesso em: 30 mai. 2020. Citado na página 21.

LANG, K. Newsweeder: Learning to filter netnews. In: *in Proceedings of the 12th International Machine Learning Conference (ML95)*. [S.l.: s.n.], 1995. Citado na página 19.

LE, Q.; MIKOLOV, T. Distributed representations of sentences and documents. In: *International conference on machine learning*. [S.l.: s.n.], 2014. p. 1188–1196. Citado 4 vezes nas páginas 12, 23, 24 e 25.

LIDDY, E. D. Natural language processing. In: *In Encyclopedia of Library and Information Science*. [S.l.: s.n.], 2001. Citado 4 vezes nas páginas 16, 17, 18 e 19.

MANNING, C. D.; SCHÜTZE, H. *Foundations of statistical natural language processing*. [S.l.]: MIT press, 1999. Citado 2 vezes nas páginas 16 e 18.

MARK, G. *Click bait is a distracting affront to our focus*. 2014. Disponível em: <https://www.nytimes.com/roomfordebate/2014/11/24/you-wont-believe-what-these-people-say-about-click-bait/click-bait-is-a-distracting-affront-to-our-focus>. Acesso em: 30 mai. 2020. Citado na página 16.

MIKOLOV, T. et al. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. Citado na página 12.

NICULAESCU, O. *Classifying data with decision trees*. 2018. Disponível em: <https://elf11.github.io/2018/07/01/python-decision-trees-acm.html>. Acesso em: 30 mai. 2020. Citado na página 21.

OMIDVAR, A.; JIANG, H.; AN, A. Using neural network for identifying clickbaits in online news media. In: SPRINGER. *Annual International Symposium on Information Management and Big Data*. [S.l.], 2018. p. 220–232. Citado 9 vezes nas páginas 28, 30, 31, 32, 34, 35, 37, 38 e 42.

PENNINGTON, J.; SOCHER, R.; MANNING, C. D. Glove: Global vectors for word representation. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. [S.l.: s.n.], 2014. p. 1532–1543. Citado 2 vezes nas páginas 35 e 36.

POTTHAST, M. et al. The clickbait challenge 2017: Towards a regression model for clickbait strength. *CoRR*, abs/1812.10847, 2018. Disponível em: <http://arxiv.org/abs-/1812.10847>. Citado na página 27.

POTTHAST, M. et al. Crowdsourcing a large corpus of clickbait on twitter. In: *Proceedings of the 27th International Conference on Computational Linguistics*. [S.l.: s.n.], 2018. p. 1498–1507. Citado 5 vezes nas páginas 11, 12, 15, 27 e 30.

POTTHAST, M. et al. Clickbait detection. In: SPRINGER. *European Conference on Information Retrieval*. [S.l.], 2016. p. 810–817. Citado 3 vezes nas páginas 11, 12 e 27.

PRODANOV, C. C.; FREITAS, E. C. de. *Metodologia do trabalho científico: métodos e técnicas da pesquisa e do trabalho acadêmico-2ª Edição*. [S.l.]: Editora Feevale, 2013. Citado na página 13.

RISH, I. An empirical study of the naive bayes classifier. In: *IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*. [S.l.: s.n.], 2001. Citado na página 22.

TIRA. *TIRA - Evaluation as a Service*. 2020. Disponível em: <https://www.tira.io/>. Acesso em: 5 out. 2020. Citado na página 30.

WEBISEVENTS. *Clickbait Detection*. 2020. Disponível em: <https://events.webis.de-/clickbait-challenge>. Acesso em: 2 out. 2020. Citado na página 33.

Wikipedia contributors. *Artificial neural network — Wikipedia, The Free Encyclopedia*. 2020. `https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=960600716`. [Online; accessed 7-June-2020]. Citado na página 22.

ZHANG, Y.; JIN, R.; ZHOU, Z.-H. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, Springer, v. 1, n. 1-4, p. 43–52, 2010. Citado na página 22.

ZHOU, Y. Clickbait detection in tweets using self-attentive network. *arXiv preprint arXiv:1710.05364*, 2017. Citado na página 28.