UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO

BRUNO SIQUEIRA FERREIRA

DESENVOLVENDO UM CLASSIFICADOR DE CLICKBAIT PARA TWEETS COM WORD EMBEDDINGS

FLORIANÓPOLIS

2019

BRUNO SIQUEIRA FERREIRA

DESENVOLVENDO UM CLASSIFICADOR DE CLICKBAIT PARA TWEETS COM WORD EMBEDDINGS

Trabalho apresentado como requisito parcial para a obtenção do título de Bacharel no curso de Sistemas de Informação do Departamento de Informática e Estatística da Universidade Federal de Santa Catarina.

Orientador: Prof Renato Fileto, MSc

FLORIANÓPOLIS

2019

Bruno Siqueira Ferreira
Desenvolvendo um classificador de clickbait para tweets com word embeddings/ Bruno Siqueira Ferreira. – Florianópolis, 2019-
34 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof Renato Fileto, MSc

Monografia – Universidade Federal de Santa Catarina, 2019.

1. detecção de clickbait. word embeddings. semântica. classificadores.
I. Orientador: Prof Renato Fileto, MSc.
II. Universidade Federal de Santa Catarina
III.
IV. Desenvolvendo um classificador de clickbait para tweets com word embeddings

CDU 02:141:005.7

RESUMO

Clickbaits são uma forma de título de notícia vago, porém intrigante, com objetivo de fazer o leitor clicar na notícia e acessar algum site. Com a propagação recente deste tipo de manchete, uma busca por uma maneira automática de detecta-los vem se tornando cada vez mais relevante. A tarefa compartilhada Clickbait Challenge ajudou a avançar os estudos desta área, com diversos trabalhos competindo para obter os melhores resultados para um conjunto de dados fornecido. Em um desses, foram utilizados word embeddings para realizar a classificação. Este TCC faz um estudo de propostas para classificação de clickbaits e propõe melhorias no trabalho do Clickbait Challenge que usa word embedding, usando Short Semantic Patterns num modelo de machine learining treinado com regressão linear. Nosso modelo atinge um F1 score de 0,793, melhor que o modelo base, e tem um erro médio quadrático de 0,113, melhor que o modelo base sobre a mesmo subconjunto de dados utilizados. Em conclusão, o modelo descrito neste trabalho comprova que características (features) extraídas mediante análise semântica, tais como padrões SSP, contribuem para a melhoria dos resultados do classificador de clickbaits

Palavras-chaves: detecção de clickbait. word embeddings. semântica. classificadores.

ABSTRACT

Clickbaits are a type of headlines that are empty but intriguing, with the objective of making the reader click on the article and access some website. With the recent propagation of this headlines, a search for some way of identifying them has been becoming more relevant. The shared task of the Clickbait challenge has helped advance the studies on this area, with many works competing to obtain the best results for a data set provided. In one of those, word embeddings are utilized to make the classifier. This thesis studies proposals for clickbait classification and proposes improvements on the Clickbait Challenge work that uses word embeddings, using the text's semantics on a machine learning model trained with linear regression. Our model reached a F1 Score of 0,793, better than the base model, and has a MSE of 0,113, better than the base model over the same subset of data. In conclusion, the model described in this work proves that features extracted from semantic analysis, like SSP patterns, contribute to the improvement of results of clickbait classifiers.

Key-words: clickbait detection, word embeddings, semantics, classifiers

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Numa época onde os leitores têm acesso a sites e notícias gerais, nacionais e internacionais, blogs que contêm informações detalhadas sobre determinados tópicos, e muitas manchetes o tempo inteiro na sua tela, chamar a atenção se torna uma tarefa importante. Assim, escritores tiveram de se adaptar para sobreviver a este novo ambiente onde disputam entre si a atenção do internauta, buscando vizualições em suas páginas para conseguirem receitas de propagandas. Uma estratégia comumente usada é chamada de *clickbait*.

Clickbaits, segundo Hurst (2016), são uma forma de título de nóticia com proposito de ser vago, mas intrigante o suficiente para fazer o leitor clicar na notícia na sua rede social e ir para a página do artigo, onde ele consegue mais informações. Esses títulos são sensacionalistas por natureza e são feitos para chamar atenção e incitar curiosidade sobre suas histórias sem revelar informações importantes.

Detecção automática de clickbaits é um objetivo recente que vem se tornando cada vez mais relevante, conforme a frequência de clickbaits em redes sociais vem aumentando. Em seu artigo, Potthast et al. (2018) indica todos os datasets com categorização de notícias em clickbaits ou não que conseguiu encontrar, num total de 6 datasets incluindo o seu próprio criado para o artigo, o primeiro desses formado em 2016. Neste curto período de tempo foram criados alguns métodos diferentes para detectar clickbaits, com prevalência de algoritmos utilizando redes neurais e *deep learning*. Este trabalho propõe uma extensão para um método já existente, fazendo a utilização de *Short semantic patterns* (SSP) para gerar uma melhoria de resultados.

Normalmente, em uma classificação de texto, é comum usar abordagens como *Bag-of-words* para recuperar informações de um texto. Essas abordagens usam palavras como indicadores e frequências de palavras como características para efetuar classificação de texto. Assim, os algoritmos de classificação acabam trabalhando apenas com terminologia para achar padrões, enquanto padrões conceituais ficam ignorados, tendo problemas com expressões, palavras sinônimas, palavras com múltiplos significados e problemas de generalização (BLOEHDORN; HOTHO, 2006). Estes são problemas que o uso da semântica pode resolver.

Segundo Gentry (2015), semântica trata do significado e da interpretação de palavras e frases. É esta que determina como nos comunicamos, como lemos e entendemos uns aos outros, e como interpretamos palavras. SSP é uma técnica desenvolvida por Sorato e Fileto (2019) que serve para minerar padrões semânticos em textos curtos, como os tweets. Com o uso destes padrões, é esperado gerar uma

melhora na classificação e identificação de clickbaits.

O classfiicador de clikbaits desenvolvido neste trabalho é treinado e avaliado usando o dataset do *Clickbait Challenge*[1], uma tarefa compartilhada em busca de um classificador de clickbait criado por Potthast et al. (2018). Para a avaliação final deste trabalho, os resultados obtidos serão comparados com resultados obtidos por outros participadores do Clickbait Challenge.

O dataset contém informações de tweets que possuem links para notícias ou artigos, e também contém infomações sobre estes artigos. Este já vem previamente particionado em dois subconjuntos, um para treino e outro para testes.

Primeiramente é realizado um processo de limpeza nos dados. Este procedimento é importante para remover inconsistências e erros e aumentar a qualidade dos dados. Então são aplicados algoritmos de tokenização e lemmatização. Tokenização é separar um texto em suas palavras, enquanto lemmatização serve para transformar palavras flexionadas em suas versões deflexionadas.

Após estes processos, são gerados os SSPs para melhor auxiliar o treinamento deste classificador. Com os padrões prontos, o classificador é treinado e testado utilizando um modelo de aprendizado de máquina baseado em regressão linear, e por fim os resultados obtidos são comparados com os de outros trabalhos utilizando o mesmo dataset do Clickbait Challenge.

1.1 OBEJTIVO GERAL

Propor, Implementar e testar um detector de clickbaits utilizando word embeddings

1.1.1 OBJETIVOS ESPECÍFICOS

- Analisar trabalhos correlatos à literatura sobre detecção automática de clickbaits, para entender o estado da arte na área.;

- Desenvolver um classificador que usa word embeddings para detectar clickbaits;

- Realizar um análise do desempenho do classificador produzido e comparar com outros classificadores;

1.2 METODOLOGIA

O método científico utilizado neste trabalho é o método indutivo. São identificados e analisados alguns classificadores de clickbait e é desenvolvido um novo

[1] https://www.clickbait-challenge.org/

classificador com uma proposta diferente que é comparado aos demais.

Do ponto de vista da sua natureza, este trabalho pode ser classificado com uma pesquisa aplicada, buscando desenvolver este classificador para diminuir a ocorrência e influência dos clickbaits.

A abordagem do problema será quantitativa, tendo em vista que o classificador será analisado em base de seus resultados quanto a classificação do dataset. As medidas utilizadas para esta análise são as mesmas usadas pelo Clickbait challenge para em sua tabela comparativa de trabalhos, erro médio quadrático, F1 Score, predição, recall e acurácia.

Quanto ao seu objetivo, este pode ser classificado como pesquisa exploratória, pois criando o classificador ele busca maior familiaridade com o problema dos clickbaits. Ainda, será realizada a avaliação do classificador, visando identificar seus pontos positivos e oportunidades de melhoria.

Por fim, em relação aos procedimentos técnicos do trabalho, este trabalho é classificado como bibliográfico e experimental. Bibliográfico pois tem como objetivo identificar padrões e maneiras de tratá-los, mas também experimental porque tenta verificar estas soluções em comparação com outras.

1.3 ESTRUTURA DO TRABALHO

O trabalho está estruturado da seguinte maneira:

- No capítulo 2 são definidos fundamentos utilizados durante o desenvolvimento do trabalho;
- No capítulo 3 são descrito trabalhos correlatos afim de apresentar o estado da arte, além de uma tablea comparativa entre os trabalhos desenvolvidos sobre o Clickbait Challenge;
- No capítulo 4 são realizadas análises sobre o dataset escolhido para o trabalho, e sobre o código utilizado como base para o desenvolvimento do modelo;
- No capítulo 5 é apresentado o desenvolvimento do modelo;
- No capítulo 6 são apresentados os resultados obtidos com o modelo e comparados aos obtidos por outros trabalhos, para ver as melhorias geradas;
- No capítulo 7, por fim, é realizada a conclusão do trabalho, destacando as melhorias encontradas, e são listados possíveis trabalhos futuros.

2 FUNDAMENTOS

Este capítulo descreve o problema de detecção de clickbaits e alguns conceitos e técnicas utilizados para resolvê-lo. A ênfase é nos conceitos e técnicas utilizados em nossa solução e necessários para o entendimento deste trabalho.

2.1 DETECÇÃO AUTOMÁTICA DE CLICKBAITS

Clickbait é uma estratégia utilizada para chamar atenção onde a notícia ou artigo tem um título demasiadamente vago, porém interessante o suficiente para induzir curiosidade, para conseguir cliques que levem à página com o artigo em si, conseguindo mais visualizações e maior renda de propagandas.A palavra clickbait tem sua origem no inglês, sendo uma junção dos termos *click*, significando clique, e *bait*, significando isca. Este tipo de notícia é frequentemente visto em redes sociais, como Facebook e Twitter, lugares em que conseguem atrair o maior número de pessoas e podem ser facilmente compartilhados a um número ainda maior (HURST, 2016). São exemplos de clickbaits:

- *10 things Apple didn't tell you about the new iPhone*
- *What happened next will surprise you*
- *This is what the actor/actress from 90s looks like now*
- *What did Donald Trump just say about Obama and Clinton*
- *9 things you must have to be a good data scientist*
- *How owning an iPhone boosts up your sex life*

Segundo Chen, Conroy e Rubin (2015), *clickbaits* são frequentemente enganosos e possuem informações não verificadas, sendo grandes contribuidores do crescente número de nóticias falsas, as *fake news*, na internet. Reportagens como estas podem gerar consequencias reais, socialmente e até economicamente. Por exemplo, um rumor em 2008 que dizia que Steve Jobs havia sofrido um ataque cardíaco fez com que as ações da Apple caissem 10% em valor.

Alem disso, estudos cognitivos indicam que *clickbaits* provocam distração. Quanto mais os leitores continuam acessando novos artigos após serem atraidos pelos títulos, as trocas constantes levam a sobrecarga cognitiva, desencorajando leitores de ler artigos informativos e mais completos (CHAKRABORTY et al., 2016).

Com *clickbaits* em vigor, seus problemas vão se acentuando cada vez mais. Em vista disto, este trabalho irá analisar e propor um método de classificação com o propósito de identificar notícias e artigos que são *clickbaits* daqueles que não são.

2.2 PROCESSAMENTO DE LINGUAGEM NATURAL

Natural language processing (NLP) é o procedimento de automatizar computadores para o entendimento e manipulamento de textos e falas Chowdhury (2003). Algumas técnicas são frequentemente utilizadas para realização deste procedimento, tais como tokenização, *Part-of-speech tagging*, e lematização.

Tokenização consiste em segmentar um texto em partes com um significado único (*tokens*). É um processo importante por ser uma etapa inicial para outras ações em NLP. O objetivo da tokenização é a exploração das palavras em um texto.

Part-of-speech tagging (POS-Tagging) trata-se da identificação da classe gramatical dos tokens de um texto. COmo uma parte importante do processo de NLP, POS-Tagging é um campo muito estudado da área, com ferramentas do estado-da-arte sendo capazes de alcançar mais de 97% de acurácia na classificação de tokens (DERCZYNSKI et al., 2013).

Lematização, segundo De Lucca e Nunes (2002), "consiste em reunir todas as ocorrências da mesma palavra sob uma única forma, o lema, como acontece num dicionário, em vez de apresentá-las tal como aparecem nos textos, com variações no gênero, no número ou na grafia". Este método causa uma menor variação de palavras, tornando possível um maior número de conexões entre sentenças e auxiliando na criação de word embeddings.

2.3 WORD EMBEDDINGS E SHORT SEMANTIC PATTERNS (SSP)

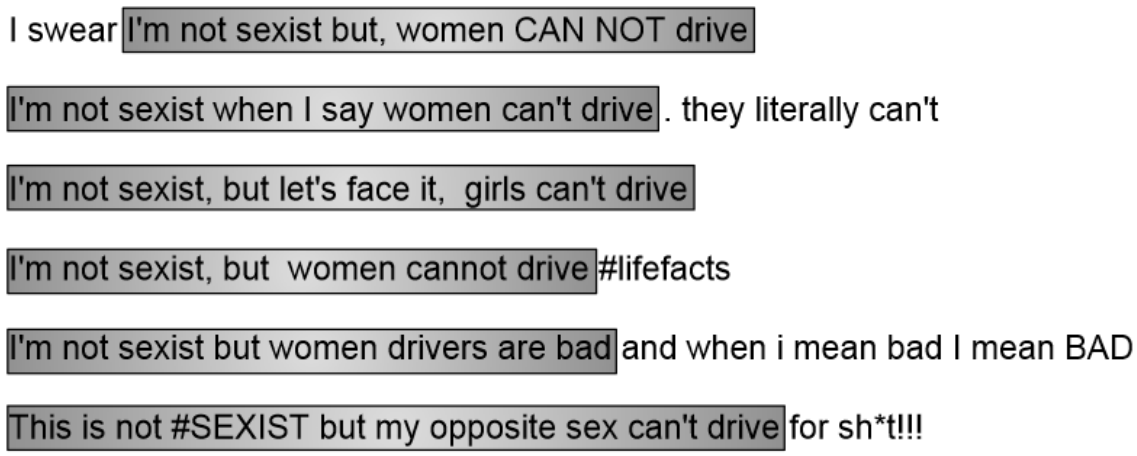
Word embeddings são vetores dimensionais que relacionam uma palavra e um determinado contexto. Estes vetores otimizam o processamento computacional, tornando ações complexas de palavras em operações com matrizes mais simples (LEVY; GOLDBERG, 2014). A ideia por trás dessa técnica é que palavras num mesmo contexto tendem a ter significados semelhantes, postando-se então próximas num gráfico vetorial.

Exemplificando o uso de word embeddings, as frases "Tenha um bom dia" e "Um ótimo dia para você" tem significados extremamente parecidos. Porém, para um sistema que só considera léxicos sem se preocupar com sua semântica, as frases podem ser um tanto diferentes, já que palavras como "bom" e "ótimo" são interpretadas como palavras, sem se preocupar com sua proximidade semântica. Os word embeddings servem como uma maneira de reduzir essa diferença, colocando palavras em espaços

vetoriais onde seus significados podem melhor ser representados e comparados pelas máquinas (KARANI, 2018).

Short Semantic Patterns (SSP) é uma técnica desenvolvida por Sorato e Fileto (2019), que faz o uso de word embeddings para buscar semelhanças semânticas em blocos curtos de texto. Um SSP se referre a uma coleção de sequências de palavras de textos distintos, as quais podem ter tamanhos diferentes e léxicos distintos, mas carregam um significado similar. Um exemplo de SSP é ilustrado na Figura 1, através de algumas de suas instâncias que foram automaticamente mineradas em torno do word embedding referente à palavra sexismo usando o método proposto por Sorato e Fileto.

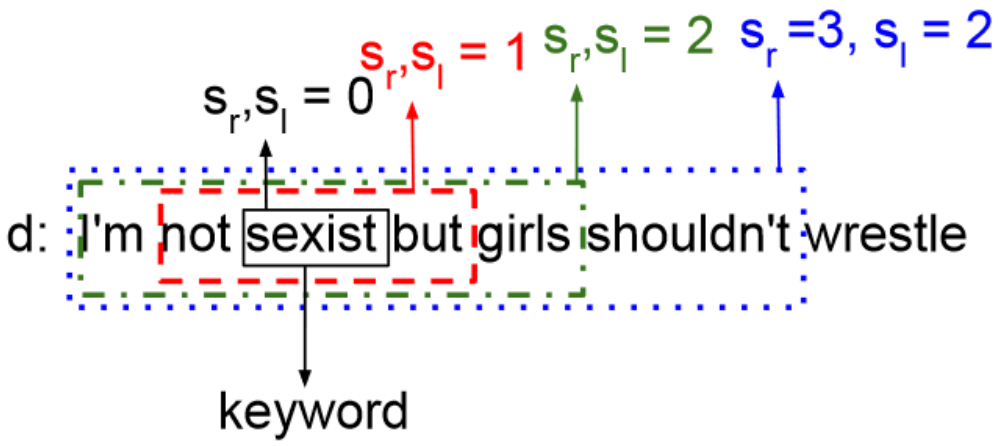
FIGURA 1 – EXEMPLOS DE INSTÂNCIAS DE UM SSP REFERENTE A SEXISMO



FONTE: Sorato e Fileto (2019)

A mineração de SSPs começa buscando por palavras ou expressões semanticamente similares (segundo os respectivos word embeddings) a alguma palavra chave de um conjunto-fornecido (e.g. sexismo, misoginía) fornecido como foco para minerar SSP em documentos de texto. A partir de cada palavra ou expressão similar a alguma palavra-chave dada como foco, vai sendo expandida uma janela de contexto, ou seja, as palavras adjacentes vão sendo acrescentada à instância do padrão enquanto for mantida similaridade semântica dentro de um certo patamar com blocos de outro texto (e.g. outro tweet). A Figura 2 apresenta uma amostra das janelas de contexto em um documento.

FIGURA 2 – EXEMPLO DE JANELA DE CONTEXTO EM UM DOCUMENTO



FONTE: Sorato e Fileto (2019)

Este trabalho minera SSPs em textos referentes a clickbaits e não clickbaits, como proposto por Sorato e Fileto. As instâncias destes SSPs então são utilizadas como features adicionais para o classificador de clickbaits.

2.4 CLASSIFICADORES AUTOMÁTICOS BASEADOS EM APRENDIZADO DE MÁQUINA

Segundo Kotsiantis, Zaharakis e Pintelas (2007), existem várias aplicações para *Machine learning*, a mais significante destas é data mining. Pessoas normalmente cometem erros durante análises ou ao tentar encontrar conexões entre múltiplas características. O aprendizado de máquina pode ser utilizado nesses problemas, melhornado a eficiência dos sistemas e o design das máquinas.

Quando usado para classificação, o aprendizado de máquina envolve analisar um conjunto de dados e retornar um resultado, que pode ser uma classe ou um valor contínuo, para cada dado do conjunto. Porém, para que a máquina consiga analisar e aprender sobre os dados, é comumente realizado um pré processamento destes dados, com a intenção de facilitar o aprendizado e diminuir a necessidade de processamento por parte da máquina.

O aprendizado de máquina pode ser classificado em algumas categorias, de acordo com os dados a ser analisados e a maneira com se dá o treinamento. Caso as instâncias tenham alguma classe para categoriza-las, o aprendizado é chamado de supervisado, onde a máquina será avaliada por seus acertos e erros em relação as classes. Por outro lado, quando as instâncias não possuem uma classe definida, o aprendizado é definido como não supervisionado, onde a máquina não é usada para

predição de classes, mas sim para a possível geração de novas classes, conforme são observadas as separações formadas pela máquina. Por fim, aprendizado de reforço é quando a máquina não é dita o que fazer com os dados, sendo forçada a descobrir que medidas tomar, enquanto recebe um sinal de reforço que representa o quão boa foi sua operação (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

Este trabalho trata de um aprendizado de máquina supervisionado, onde será aplicado o modelo de regressão linear. Regressão é uma técnica que permite a inferência de variáveis dependentes baseada em variáveis independentes. A regressão linear é assim chamada pois a inferência é realizada a partir de uma função linear criada a partir das variáveis disponibilizadas. O aprendizado de máquina tem então o trabalho de, no treinamento, produzir a função linear que dará base para as respostas.

2.5 MEDIDAS DE DESEMPENHO DE CLASSIFICADORES

Para efetivar os resultados obtidos, são utilizadas uma série de medidas de desempenho, essas formalizadas pelo Clickbait Challenge como medidas para comparações entre os trabalhos. Estas medidas são: Precisão, Recall, F1 Score, Acurácia e Erro médio quadrático (*Mean Squared Error*, MSE).

Precisão diz respeito ao número de positivos corretamente previstos comparado ao total de positivos, ou seja, é o valor que representa quantos dos valores de cada tipo foram corretamente previstos, enquanto *Recall* é o número de positivos previstos comparado ao número de caso preditos positivos, isto é, indica quantos dos valores preditos ser de um determinado tipo foram preditos corretamente. Podem ser ditos como média de exatidão e média de completude, respectivamente.

Para exemplificar, diga-se que um classificador de imagens utilizado para reconhecimento de cães identifica em um vídeo 7 cachorros, onde na verdade existem 9 cães e alguns gatos, se dos 7 cães identificados 3 eram gatos, então o classificador possui uma precisão de 4/7 e um recall de 4/9. Um bom classificador terá ambos os dados o mais perto de 1. Nesse caso, os 4 cachorros acertados são verdadeiros positivos, enquanto os 3 gatos ditos como cães são falso positivos e os 5 cães não preditos são verdadeiros negativos.

F1 Score como medida se baseia na precisão e no recall do modelo, sendo este uma média do valor dos outros dois.

A acurácia do modelo é dada simplemente pelo número de predições corretas comparado ao documento inteiro. Com o dataset dividido ao meio em relação a clickbaits e não-clickbait, quanto maior for essa acurácia, melhor é o modelo.

MSE é considerado pelo Clickbait Challenge como o valor mais importante destes. É a média da soma dos quadrados dos erros de predição. Este representa a

variância existente no modelo, isto é, o quão longe o predição fica do valor real em média. O valor é melhor quanto mais perto de 0.

A Tabela 1 apresenta as formulas de cada uma destas medidas.

TABELA 1 – TABELA DE MEDIDAS

Medida	Fórmula
MSE	$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2.$
Precisão	$Precision = \frac{truepositives}{truepositives + falsepositives}$
Recall	$Recall = \frac{truepositives}{truepositives + falsenegatives}$
F1 Score	$F1 = 2 \times \frac{precision \times recall}{precision + recall}$
Acurácia	$Accuracy = \frac{truepositives + truenegatives}{totalexamples}$

FONTE: https://skymind.ai/wiki/accuracy-precision-recall-f1 e https://en.wikipedia.org/wiki/Mean_squared_error

LEGENDA: I: valor sendo predito; Y: valor correto de i; Ŷ: valor predito para i pelo classificador;

3 TRABALHOS RELACIONADOS

Detecção automática de *clickbaits* é um problema recente que começou com a grande popularização da Internet na última década. Assim, existem poucos datasets disponíveis publicamente para treinar e avaliar soluções. Com isso, um número pequeno de pesquisas foram levantadas sobre este assunto, tendo enfoque principalmente em metodologias de classificação e gerando sua própria base de dados. Em seu trabalho, Chakraborty et al. (CHAKRABORTY et al., 2016) cria seu dataset com 15000 artigos, sendo 7500 *clickbaits*, extraídos manualmente de sites que diz publicarem esses tipos de notícias e classificados manualmente como clickbaits ou não-clickbaits, com aqueles considerados clickbaits sendo analisados por seis voluntários para garantir relevância. Após uma análise comparativa entre títulos de clickbaits e não-clickbaits, foram desenvolvidos 3 classificadores utilizando modelos preditivos diferentes, sendo que o classificador usando *Support Vector Machines* (SVM) com núcleo de *Radial Bass Function* (RBF) se saiu melhor, obtendo uma acurácia de 93%.

Alguns trabalhos controem datasets públicos. É o caso do Potthast et al. (POTTHAST et al., 2018), que formou um dataset com 19538 *tweets*, manualmente classificados entre 4 categorias, "*not click baiting*", "*slightly click baiting*", "*considerably click baiting*"e "*heavily click baiting*", tendo a classificação sido conferida por 5 verificadores. Com este dataset, foi criado também o Clickbait Challenge, uma tarefa compartilhada com o objetivo de melhorar e evoluir o estado da arte.

Até o momento da escrita deste trabalho, haviam sido submetidos 16 trabalhos ao Clickbait Challenge, 10 destes com um artigo apresentado. Os dois que se sairam melhor, conforme as métricas do desafio, foram os classificadores do Omidvar, Jiang e An (2018) e do Zhou (2017).

O trabalho de Omidvar, Jiang e An (2018) utiliza-se de *deep learning* para criação de um modelo de rede neural bidirecional com unidades recorrentes bloqueadas (*Gated recurring units*, GRU). Para geração de features foram utilizados word embeddings pré-treinados provenientes da GloVe, de Pennington, Socher e Manning (2014).

Zhou (2017), por sua vez, fez o uso de rede neural auto-atenciosa (*Self-attentive network*). Diferente de uma rede neural com atencão externa, a rede auto-atenciosa consegue realizar a geração de features sozinha, sem necessitar de informações adicionais externas.

Grigorev (2017) utilizou o dataset disponibilizado pelo Clickbait Challenge, e também gerou um dataset próprio através de grupos do Facebook, onde existiam em maior parte postagens com clickbaits, para aumentar a quantidade de dados e gerar um classificador com maior nível de confiança. Aplicou um modelo de regressão linear em tipo de dado textual para geração de features e, com os resultados, gerou um segundo modelo para classificação.

No trabalho de Glenski et al. (2017), o modelo criado utiliza-se de todos os dados disponíveis no dataset, incluindo as imagens. Para usufruir de todos os dados, foram monta-

das duas arquiteturas de redes neurais linguisticamente infusas (*Linguistically-infused neural network*), utilizando uma camada de neurônios convolucionais (*Convolutional Neural Network* (CNN)) e outra fazendo o uso de *Long Short-Term Memory* (LSTM). Thomas (2017) também faz o uso da rede neural LSTM em seu trabalho, obtendo resultados parecidos.

Cao, Le et al. (2017) utiliza machine learning para criar seu modelo. São formuladas diversas novas características, entre elas nível de similaridade entre o tweet e o artigo, entre o tweet e o título do artigo e entre o tweet e as palavras-chaves do artigo. Além dessas, características discutidas em trabalhos anteriores foram também analisadas, chegando a um total de 331 features, das quais foram selecionadas 180 para evitar o sobreajuste (*overfitting*), e essas por sua vez foram avaliadas e reduzidas as 60 mais relevantes.

O trabalho de Gairola et al. (2017) propõe um modelo de redes neurais que consiste de três partes: LSTM bidirecional com atenção, rede neural siamesa com text embeddings, e rede neural siamesa com visual embeddings.

Por fim, o trabalho de Indurthi e Oota (2017) é a base deste trabalho. Utiliza os word embeddings pré-treinados da GloVe, com 300 dimensões, e outras features geradas manualmente: presença de perguntas, presença de números no começo do tweet, presença de gerúndio no verbo e presença de adjetivos superlativos. Com estas features, é treinado um modelo de aprendizado de máquina com regressão linear.

Este trabalho foi o escolhido por ser um modelo simples, o qual já fazia a utilização de word embeddings, tornando o desenvolvimento mais fácil e fazendo a comparação ter mais relevância. Nosso trabalho irá alterá-lo de maneira a utilizar os SSPs para obter um resultado melhor. Acredita-se que a semântica representa um fator importante na detecção de clickbaits, e as SSPs podem validar isso.

A Tabela 2 faz uma análise comparativa entre os modelos desenvolvidos para o Clickbait Challenge aqui citados.

TABELA 2 – Tabela comparativa de trabalhos correlatos

Autor	Modelo de classificação	Features usadas
Omidvar, Jiang e An (2018)	Deep learning GRU bidimensional	**Word embeddings** da GloVe
Zhou (2017)	Rede neural autoatenciosa	Features geradas automaticamente pela rede
Grigorev (2017)	Modelo de regressão linear SVM	Acoplado de diferentes modelos, estes treinados com base em uma das características de cada tweet
Glenski et al. (2017)	Linguistically-infused neural network com LSTM e CNN	Presença de verbos assertivos, factiveis, restritivos, ou implicativos, além de image embeddings
Thomas (2017)	Rede neural LSTM	Image embeddings
Cao, Le et al. (2017)	Machine learning com Random forest regression	60 features selecionadas manualmente, entre elas números de tokens, similaridade entre tweet e título do artigo e similaridade entre tweet e corpo do artigo
Gairola et al. (2017)	LSTM bidimensional com atenção	**Word embeddings** e Image embeddings
Indurthi e Oota (2017)	Aprendizado de máquina com regressão linear	**Word embeddings** da GloVe
Este TCC	Aprendizado de máquina com regressão linear	**Word embeddings** GloVe e SSPs

4 ANÁLISES DE ARTEFATOS EMPREGADOS NA CLASSIFICAÇÃO

Neste capítulo são apresentadas duas análises, uma dos dados disponibilizados pelo Clickbait Challenge e outra do código utilizado como baseline para o desenvolvimento deste trabalho.

4.1 ANÁLISE DOS DADOS DO CLICKBAIT CHALLENGE

O dataset disponibilizado pelo Clickbait Challenge possui tweets que contêm algum link redirecionando para uma notícia. É comum que noticiarios utilizem-se do Twitter para publicar suas notícias e chamar atenção. Os posts estão guardados em arquivos JSON que são estruturados conforme a Tabela 3.A Figura 3 representa um exemplo de um destes documentos JSON. São disponibilizados 3 datasets, dois previamente avaliados e um a ser avaliado. Os dados do dataset 3 não foram utilizados por não possuirem uma nota. Os dados dos datasets 1 e 2 foram combinados para formar o dataset 4. Informações correspondentes ao tamanho de cada dataset estão presentes na Tabela 4. Também são oferecidas as imagens que haviam para os tweets que contiam imagens, mas essas não serão analisadas neste trabalho.

TABELA 3 – Estruturação dos dados no dataset

Campo	Descrição
ID	Um identificador único para cada tweet
postText	O texto da postagem
postTimestamp	A data da postagem
postMedia	A imagem que foi publicada junto ao tweet
targetTitle	O título do artigo direcionado
targetDescription	Descrição do artigo
targetKeywords	Palavras-chaves do artigo
targetParagraphs	O conteúdo do artigo
targetCaptions	As legendas que tem no artigo
truthJudgements	As 5 notas dadas pelos avaliadores
truthMean	A média das notas
truthMedian	A mediana das notas
truthClass	Uma classe binária que indica se o post é clickbait ou não

Os tweets foram julgados por 5 avaliadores humanos que deram uma de 4 notas para tal:

- Not click baiting (Nota 0.0)
- Slightly click baiting (Nota 0.33)

FIGURA 3 – EXEMPLO DE UM DOCUMENTO JSON COM OS DADOS


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


FONTE: `https://www.clickbait-challenge.org/`

- considerably click baiting (Nota 0.66)
- clickbait(Nota 1.0)

A Tabela 2 nos mostra um imbalanceamento no número de clickbaits comparado ao número de não-clickbaits, tendo 2,9 não-clickbaits para cada nóticia clickbait.

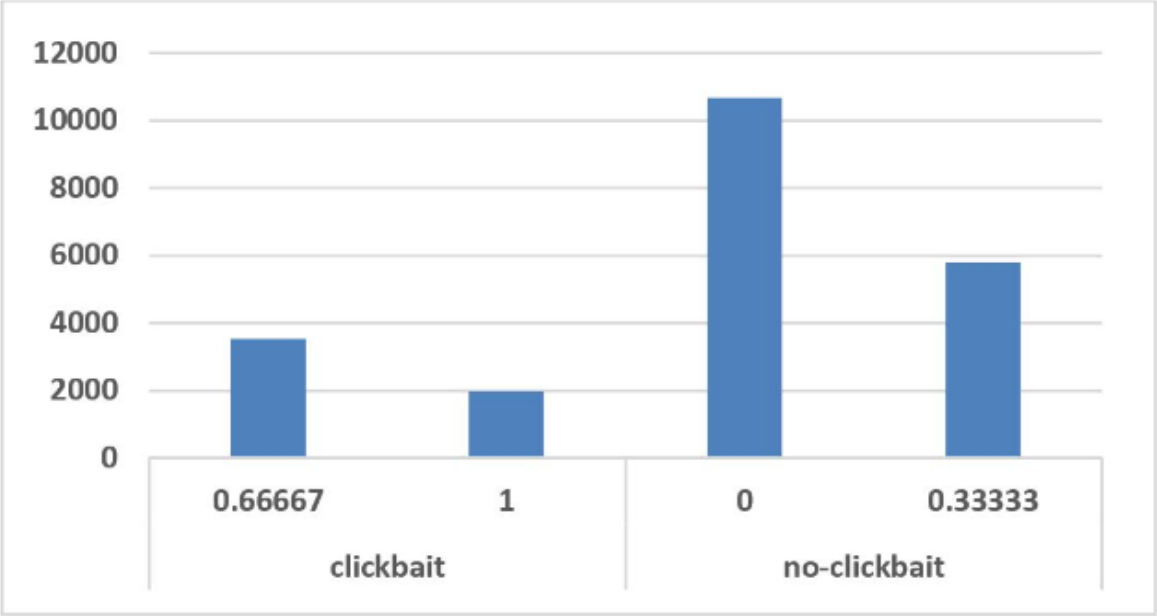
TABELA 4 – Dados de cada dataset

Dataset	Total de dados	Clickbaits	Não-clickbaits
Dataset 1	2495	762	1697
Dataset 2	19538	4761	14777
Dataset 3	80012	?	?
Datasets 1+2	21997	5523	16474

Omidvar, Jiang e An (2018) analisam os dados relacionados às notas dadas a cada tweet. A Figura 4 mostra sua análise baseada na média de votos que os posts receberam. Percebe-se que posts onde a média terminou entre "clickbait" e "considerably click baiting" são considerados clickbaits, enquanto aqueles entre "slightly click baiting" e "not clickbaiting" não são. Isto mostra que se a soma de "slightly click baiting" e "not clickbating" for maior que a soma

das outras duas opções, o tweet é categorizado como não clickbait, enquanto no contrário, o tweet é dado como clickbait.

FIGURA 4 – GRÁFICO SOBRE A VARIÁVEL TRUTHMEAN

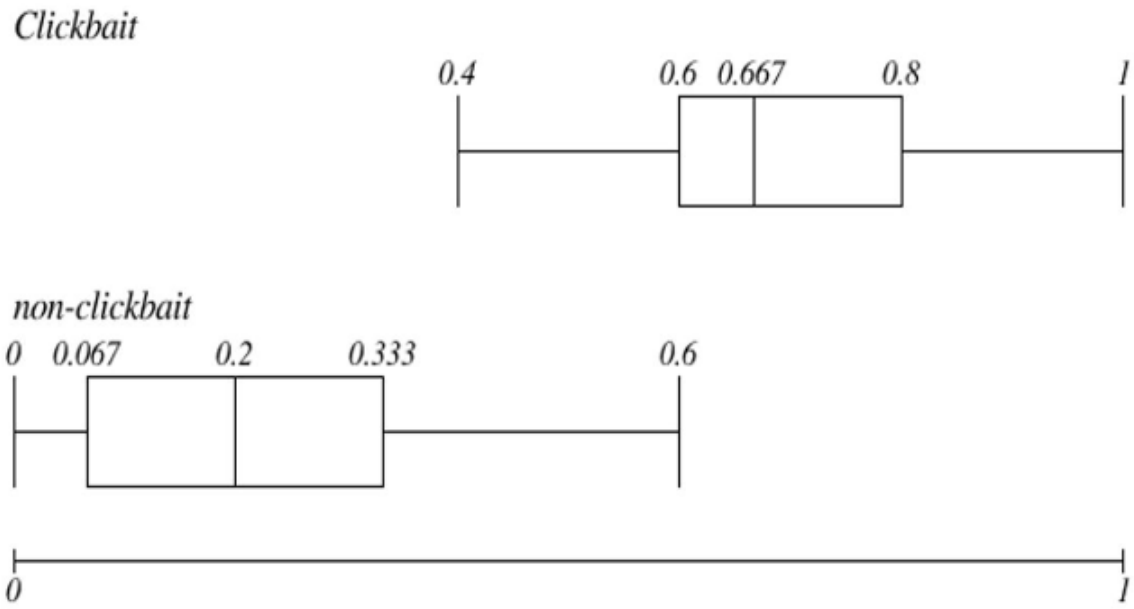


FONTE: Omidvar, Jiang e An (2018)
LEGENDA: Eixo Y - número de tweets com aquela média

Ainda na análise de Omidvar, Jiang e An (2018), a Figura 5 nos apresenta um *boxplot* que mostra a distribuição de votos por parte dos analistas. Nota-se que o máximo de voto de um não-clickbait foi 0.6, enquanto os votos para clickbait vão de 0.4 a 1, mostrando que há uma sobreposição nos votos de ambas as categorias. A Figura 6 pode comprovar este fato, porém mostra também que essa sobreposição não é tão alta.

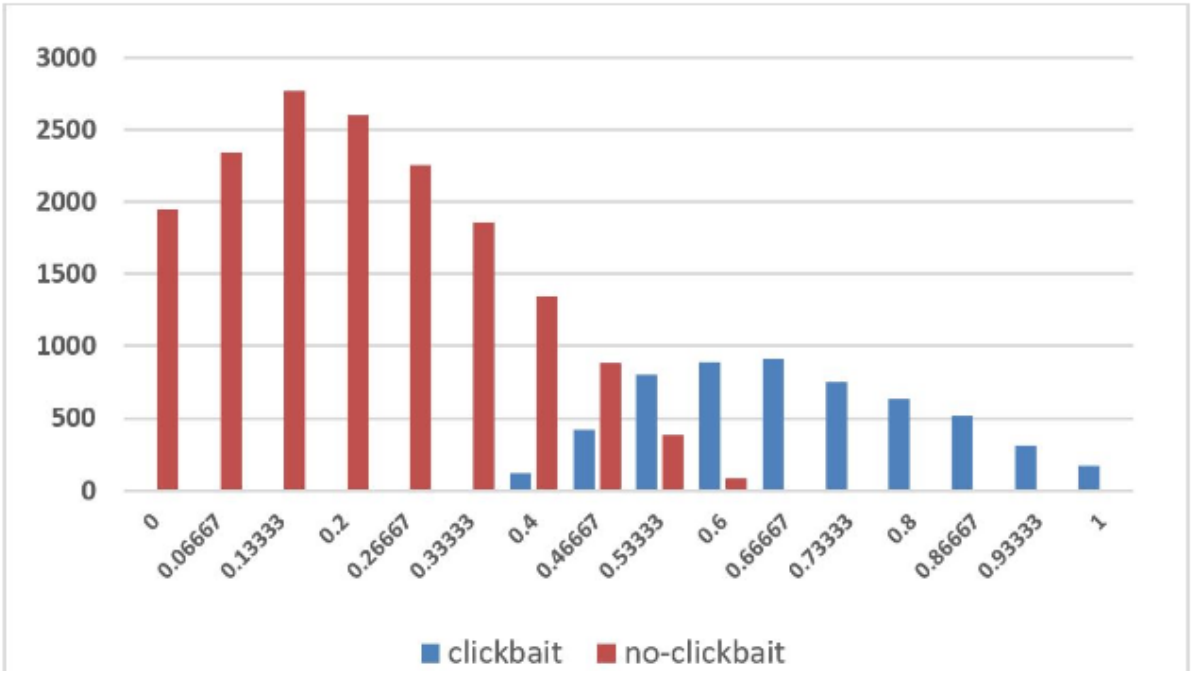
A Figura 4 apresenta um gráfico contendo a distribuição dos posts por seu número de caracteres por categoria, respeitando o número de tweets existentes em cada classe. É possível notar uma diferença entre o número de caracteres de posts classificados como clickbait e posts não-clickbaits, principalmente no número de posts com poucos caracteres.

FIGURA 5 – BOXPLOT DEMONSTRANDO VOTOS DE ANALISTAS



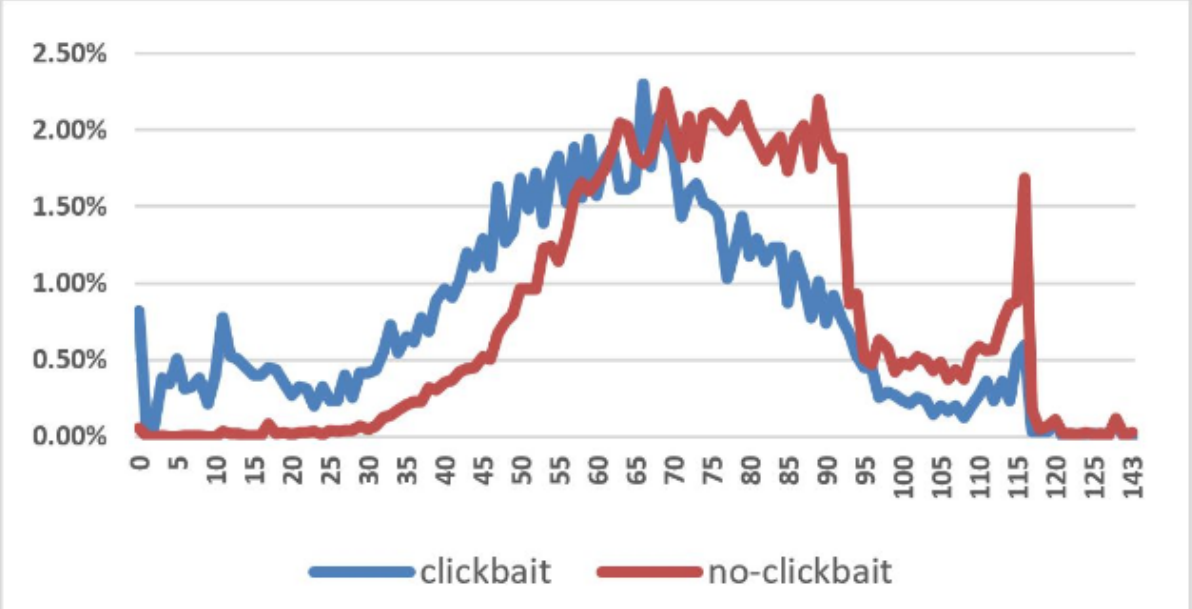
FONTE: Omidvar, Jiang e An (2018)

FIGURA 6 – GRÁFICO DE BARRAS SOBRE VOTOS DE ANALISTAS BASEADO NO VALOR MÁXIMO DA VARIÁVEL TRUTHJUDGEMENT



FONTE: Omidvar, Jiang e An (2018)
LEGENDA: Eixo Y - número de tweets que receberam aquela nota

FIGURA 7 – DISTRIBUIÇÃO DE TWEETS PELO NÚMERO DE CARACTÉRES



FONTE: Omidvar, Jiang e An (2018)
LEGENDA: Eixo X - número de caractéres por tweet; Eixo y - porcentagem de tweets

4.2 ANÁLISE DE CÓDIGO DA BASELISE

O trabalho de Indurthi e Oota (2017) apresenta um código com muitas oportunidades de melhoramento. Porém, foram realizadas poucas alterações no dataset disponibilizado, e é utilizado apenas o postText de cada dado para geração de features. São feitas features manuais além de word embeddings.

Para geração de word embeddings, Indurthi e Oota (2017) utiliza os embeddings pré treinados de GloVe (PENNINGTON; SOCHER; MANNING, 2014). O GloVe possui um armazenamento enorme de vetores pré treinados de palavras, um total de 6 bilhões de tokens e 400 mil vocabulários, e vetores de 50, 100, 200 e 300 dimensões para cada um destes.

Para a geração das outras features, é utilizada a ferramenta spaCy. spaCy[1] é uma biblioteca para NLP utilizada mundialmente e aclamada por sua rapidez e simplicidade de uso. É normalmente utilizada para tokenização, reconhecimento de entidades, *POS Tagging*, entre outras funcionalidades. Com ela são geradas as seguintes features:

- Número de palavras;
- Número de *stop words* (palavra vazia, palavras que não adicionam ao contexto);
- Tamanho médio das palavras;
- Presença de perguntas;

[1] https://spacy.io/

- Presença de números no começo da frase;
- Presença de gerúndio;
- Presença de superlativo nos adjetivos.

Com as features geradas, o dataset foi separado em suas classes (clickbait e não-clickbait) para que o número de não-clickbait fosse reduzido e a quantidade de cada classe no modelo fosse a mesma. Após isto, o dataset foi aleatoriamente dividido em duas partes, uma com 80% dos dados para treinamento do modelo e outra com 20% dos dados para testes. É então treinado e testado o modelo de machine learning utilizando regressão linear como função, tendo como objetivo realizar a previsão das notas obtidas pelos tweets. A tarefa é vista como um problema de regressão, com a nota de clickbait dos tweets é a variável dependente. Com isso, o modelo desenvolvido neste trabalho também visa o problema como um problema de regressão.

5 O MODELO DESENVOLVIDO

Neste capitúlo é apresentado como foi o desenvolvimento do modelo, além de apresentar o ambiente de desenvolvimento e ferramentas utilizadas.

5.1 AMBIENTE

O ambiente onde foi feito o modelo de classificação e realizado os experimentos é um computador rodando Linux Ubuntu 19.04, com processador Intel Core i5 5ª geração, 3,8 GB de memória RAM.

Para desenvolver tal modelo foi utilizado a linguagem Python3.7 com os seguintes módulos:

- numpy 1.15.4 para operações matemáticas;
- nltk 3.4 para processamento de dados;
- spaCy 2.0.16 para geração de word embeddings;
- scikit-learn 0.20.1 para apredizado de máquina;

5.2 DESENVOLVIMENTO

Para contruir o novo modelo, o dataset foi filtrado previamente. Os 21997 posts foram tokenizados, e ocorreu a remoção de *stop words* daqueles que as tinham. Após esses processos, é realizado a lematização desses documentos, tornando as palavras flexionadas em seus radicais, para melhor comparação e similarização. Nesta ação, todos os números encontrados em textos foram transformados em um token "[n]" para melhor representação e similaridade.

Para melhorar o modelo base, foram formuladas features adicionais baseadas nas semânticas dos textos. Tais features consistem de instâncias de Short Semantic Patterns (SSP) minerados automaticamente em textos como proposto por Sorato e Fileto (2019).

Como clickbaits em geral não possuem um tema único, podendo ser variados tipos de notícias, é difícil saber qual palavra chave utilizar para a extração de SSPs. Então, foi realizada uma pré analise para encontrar o token mais utilizada pelos tweets. A Tabela 5 mostra as 8 palavras mais frequentemente encontradas nos documentos, sem serem stop words. Como é visto, o token "[n]" foi a palavra mais utilizada, ou seja, os números são as "palavras" mais recorrentes nos posts. Então, será utilizado o token "[n]" para produzir os SSPs.

Para preparar o dataset para extração de SSPs, foram retirados todos os tweets que não haviam a ocorrência de algum número. Isto reduz o tamanho do dataset para um total de 3922 documentos, sendo 2336 não-clickbaits e 1586 clickbaits.

TABELA 5 – tokens com maior número de aparições

Palavra	Número de aparições
"[n]"	3922
"trump"	2173
"new"	1231
"says"	777
"via"	653
"people"	547
"president"	489

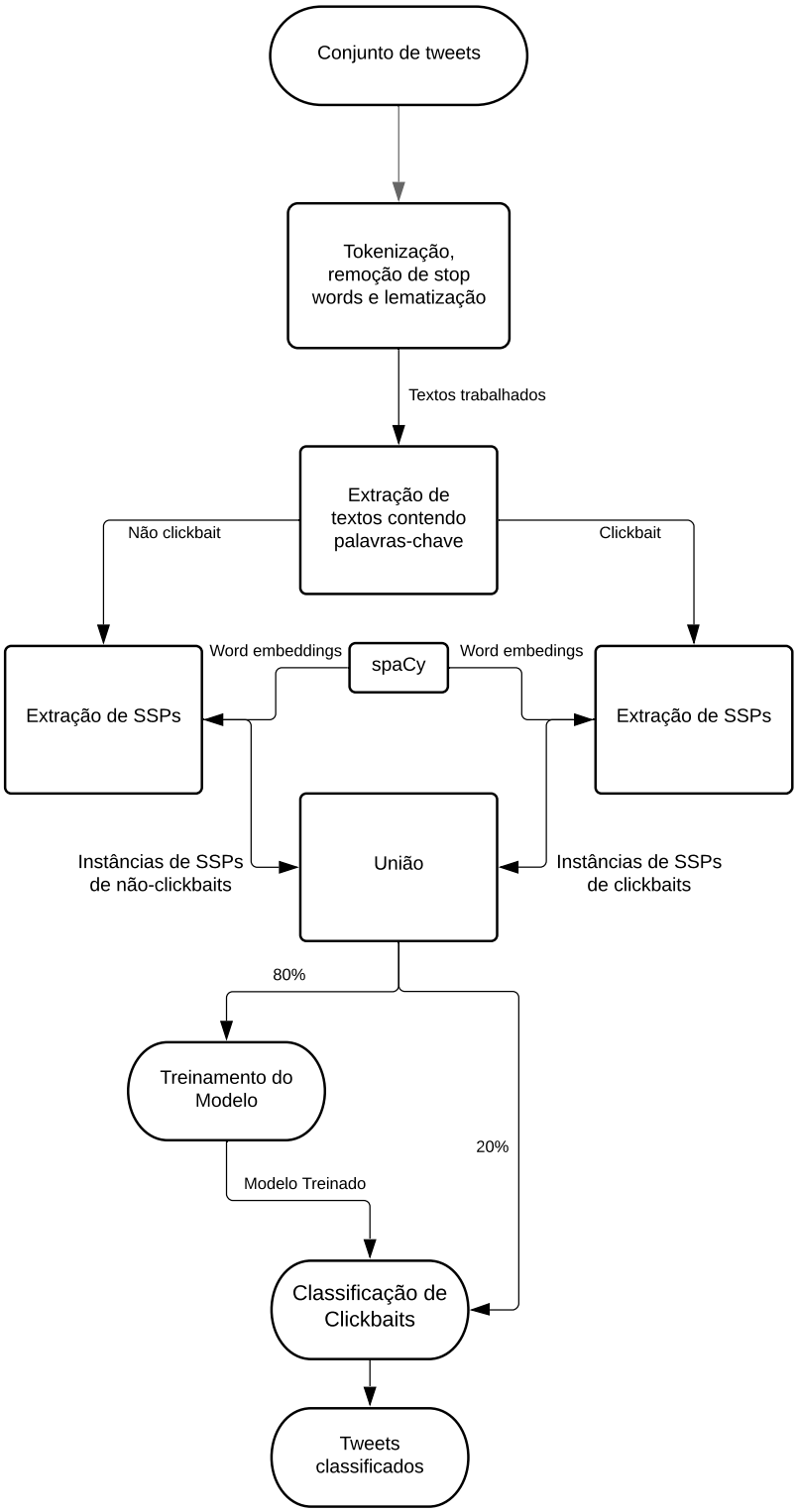
Depois da filtragem, os dados foram divididos em suas respectivas classes para realizar a extração de SSPs, com a intenção de encontrar SSPs diferentes em não-clickbaits e clickbaits. Por tanto, utilizando o número em cada documento como palavra chave, estes foram comparados entre si, com sequencias de até 2 palavras antes ou depois do número, para buscar uma semelhança de até 22,5%, utilizando os word embeddings para encontrar esta semelhança. Caso exista semelhança com algum outro documento, o item sendo analisado é guardado para processamento, com sua instância SSP sendo sua feature. Caso contrário, o item é descartado.

Com não-clickbaits e clickbaits filtrados por SSPs, sobram no final um total de 1964 não-clickbaits e 881 clickbaits. Assim como no trabalho base, os não-clickbaits são reduzidos ao número de clickbaits de maneira aleatória, afim de produzir um resultado mais significativo, levando o total de documentos analisados para 1762. É então feita a partição deste dataset em 80%, num total de 1409, para treino do modelo e 20%, total de 353, para teste e é gerado o treinamento e predições. A Figura 8 mostra como ficou a arquitetura do modelo. Visto que o dataset final é diferente do dataset usado no código baseline, foi reproduzido o teste no código baseline usando o novo dataset também para fins de comparação.

O modelo desenvolvido é o mesmo utilizado pelo algoritmo base, aprendizado de máquina com regressão linear para predição da nota de cada tweet. O código do protótipo desenvolvido e uma cópia dos dados utilizados para treinar os modelos estão disponíveis no GitHub[1].

[1] https://github.com/Brusifer/classification-project

FIGURA 8 – ARQUITETURA DO MODELO DESENVOLVIDO



FONTE: https://github.com/Brusifer/classification-project

NOTA: Utilizado Ferramenta Lucid Chart(https://www.lucidchart.com) para fazer este desenho

6 RESULTADOS

Neste capítulo são apresentados os resultados das avaliações feitas sobre o código desenvolvido, primeiramente comparando ao resultado do código base quando é utilizado o mesmo dataset, depois comparando com os outros resultados do Clickbait Challenge, inclusive o trabalho base original.

O método utilizado para realizar a avaliação do modelo é oferecido pelo Clickbait Challenge[1]. Este utiliza as métricas do módulo scikit-learn para recuperar os resultados.

Como mostra a Tabela 6, o modelo desenvolvido atingiu resultados melhores que o modelo base todas categorias de avalização por margens significantes, quando é utilizado o mesmo dataset. F1 Score, precisão, recall e acurácia aumentaram em 0,20 no mínimo, enquanto o MSE foi diminuido pela metade.

Com o conjunto de dados sendo reiterado e filtrado para caber no modelo desenvolvido, é de se esperar que o nosso modelo se saia melhor. Por isso é importante realizar a comparação com outros modelos e com resultados obtidos sobre todo o dataset.

TABELA 6 – Comparação entre o modelo desenvolvido e o modelo base

Modelo	**MSE**	F1	Precisão	Recall	Acurácia
Modelo desenvolvido	**0,113**	**0,793**	**0,788**	**0,797**	**0,796**
Modelo de Indurthi e Oota (2017)	**0,282**	0,555	0,588	0,526	0,529

Mesmo comparando com os outros resultados do Clickbait Challenge, é possível notar, na Tabela 7, que o modelo desenvolvido neste trabalho possui seus méritos. Este se destaca por obter a maior precisão e maior F1 Score entre todos os resultados obtidos. Além disso, observa-se que o recall do modelo base é o melhor dentre todos, sendo quase uma anomalia no meio dos outros, com o produzido ficando logo atrás.

Porém onde o modelo perde é no MSE. Com resultado de 0,113, este não é capaz de disputar com os modelos mais bem avaliados, ficando até 0,08 pontos atrás. A acurácia também não é párea aos superiores. Contudo, o modelo apresentou resultados competitivos em relação a outros modelos desenvolvidos com tecnologias superiores, podendo ser considerado um sucesso.

[1] https://www.clickbait-challenge.org/eval.py

TABELA 7 – RESULTADOS DO CLICKBAIT CHALLENGE

Modelo	MSE	F1	Precisão	Recall	Acurácia
Modelo desenvolvido	0,113	**0,793**	**0,788**	0,797	0,796
Indurthi e Oota (2017)	0,079	0,650	0,530	**0,841**	0,785
Omidvar, Jiang e An (2018)	**0,032**	0,670	0,732	0,619	0,855
Zhou (2017)	0,033	0,683	0,719	0,650	**0,856**
anchovy*	0,034	0,679	0,717	0,645	0,855
emperor*	0,036	0,641	0,714	0,581	0,845
Grigorev (2017)	0,036	0,036**	0,728	0,568	0,847
arowana	0,039	0,656	0,659	0,654	0,837
Glenski et al. (2017)	0,041	0,631	0,642	0,621	0,827
Thomas (2017)	0,043	0,565	0,699	0,474	0,826
Cao, Le et al. (2017)	0,045	0,604	0,711	0,524	0,836
Gairola et al. (2017)	0,046	0,654	0,654	0,653	0,835
houndshark*	0,099	0,023	0,779	0,012	0,764
dory*	0,118	0,467	0,380	0,605	0,671

FONTE: `https://www.clickbait-challenge.org/`

NOTA: *: Modelos que não contém artigos, sendo impossível recuperar seus nomes, então foram representados por seus nomes de times no Clickbait Challenge;

**: Provavelmente um erro por parte do Clickbait Challenge

7 CONCLUSÃO E TRABALHOS FUTUROS

Clickbait é uma estratégia empregada por escritores ao escreverem suas notícias para chamar a atenção de leitores e induzi-los a acessarem algum site. É utilizada principalmente na internet, onde estas notícias tem de dividir o espaço e a atenção do leitor com tantas outras. Porém, para o leitor, essas notícias acabam deixando a desejar, tendo pouco conteúdo e sendo desinteressante. Com esta técnica se tornando cada vez mais comum, foi inicada uma busca por uma maneira de detectar notícias que se utilizam desta estratégia, com o objetivo de reduzir a quantidade de clickbaits existentes na internet.

Este trabalho apresentou o desenvolvimento de um modelo para classificação de clickbaits baseado em short semantic patterns aparentes em tweets, usufruindo do dataset disponibilizado pelo clickbait challenge. Foi desenvolvido um modelo de classificação linear baseado no modelo de Indurthi e Oota (2017) fazendo a utilização de short semantic patterns como feature para treinamento, os resultados obtidos foram comparados com os de outros modelos por sobre o dataset do clickbait challenge.

Durante o trabalho foi discutido outros trabalhos relacionados, principalmente aqueles desenvolvidos com base no dataset do Clickbait Challenge. Foram apresentadas as diferentes maneiras utilizadas para realizar a detecção dos clickbaits e montada uma tabela comparativa entre estes trabalhos. Também foram explicados conceitos importantes para odesenvolvimento do modelo, como word embeddings, short semantic patterns e processamento de linguagem natural.

Os resultados obtidos pelo modelo comprovam que o uso de features baseadas em semântica, tais como embeddings e padrões SSP, é relevante na busca por clissificação de clickbaits. Enquanto outros modelos possuem melhor desempenho em algumas medidas, o classificador desenvolvido melhora muito o modelo base e alcança resultados competitivos, principalmente em termos de precisão e medida F1, contra modelos que utilizam tecnologias mais avançadas.

Como trabalhos futuros:

- Melhorar o desenvolvimento do modelo, adicionando novas features, utilizando os outros dados disponíveis e refinando a extração de SSP;
- Realizar a extração de semelhança em imagens, disponibilizadas no dataset;
- Aplicar o modelo em outro dataset, com uma maior quantidade de dados e mais bem direcionado para extração de SSPs.

REFERÊNCIAS

BLOEHDORN, Stephan; HOTHO, Andreas. Boosting for Text Classification with Semantic Features. In: MOBASHER, Bamshad et al. (Ed.). **Advances in Web Mining and Web Usage Analysis**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 149–166. ISBN 978-3-540-47128-8. Citado 1 vez na página 8.

CAO, Xinyue; LE, Thai et al. Machine learning based detection of clickbait posts in social media, 2017. Citado 2 vezes nas páginas 18, 19, 30.

CHAKRABORTY, A. et al. Stop Clickbait: Detecting and preventing clickbaits in online news media. In: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). [S.l.: s.n.], ago. 2016. p. 9–16. DOI: 10.1109/ASONAM.2016.7752207. Citado 2 vezes nas páginas 11, 17.

CHEN, Yimin; CONROY, Niall J.; RUBIN, Victoria L. Misleading Online Content: Recognizing Clickbait As "False News". In: PROCEEDINGS of the 2015 ACM on Workshop on Multimodal Deception Detection. Seattle, Washington, USA: ACM, 2015. (WMDD '15), p. 15–19. ISBN 978-1-4503-3987-2. DOI: 10.1145/2823465.2823467. Disponível em: <http://doi.acm.org/10.1145/2823465.2823467>. Citado 1 vez na página 11.

CHOWDHURY, Gobinda G. Natural language processing. **Annual Review of Information Science and Technology**, Wiley Online Library, v. 37, n. 1, p. 51–89, jan. 2003. ISSN 1550-8382. DOI: 10.1002/aris.1440370103. Disponível em: <https://doi.org/10.1002/aris.1440370103>. Citado 1 vez na página 12.

DE LUCCA, JL; NUNES, Maria das Graças Volpe. Lematização versus Stemming. **USP, UFSCar, UNESP, São Carlos, São Paulo**, 2002. Citado 1 vez na página 12.

DERCZYNSKI, Leon et al. Twitter Part-of-Speech Tagging for All: Overcoming Sparse and Noisy Data. In: PROCEEDINGS of the International Conference Recent Advances in Natural Language Processing RANLP 2013. Hissar, Bulgaria: INCOMA Ltd. Shoumen, BULGARIA, set. 2013. p. 198–206. Disponível em: <https://www.aclweb.org/anthology/R13-1026>. Citado 1 vez na página 12.

GAIROLA, Siddhartha et al. A neural clickbait detection engine. **arXiv preprint arXiv:1710.01507**, 2017. Citado 2 vezes nas páginas 18, 19, 30.

GENTRY, Angela. **What is semantics?** [S.l.: s.n.], 2015. https://study.com/academy/lesson/what-is-semantics-definition-examples-quiz.html. Accessed: 2019-06-28. Citado 1 vez na página 8.

GLENSKI, Maria et al. Fishing for clickbaits in social images and texts with linguistically-infused neural network models, 2017. Citado 2 vezes nas páginas 17, 19, 30.

GRIGOREV, Alexey. Identifying clickbait posts on social media with an ensemble of linear models, 2017. Citado 2 vezes nas páginas 17, 19, 30.

HURST, Nathan. **To clickbait or not to clickbait? an examination of clickbait headline effects on source credibility**. 2016. Tese (Doutorado) – University of Missouri–Columbia. Citado 2 vezes nas páginas 8, 11.

INDURTHI, Vijayasaradhi; OOTA, Subba Reddy. Clickbait detection using word embeddings, 2017. Citado 5 vezes nas páginas 18, 19, 24, 29–31.

KARANI, Dhruvil. **Introduction to Word Embedding and Word2Vec**. [S.l.: s.n.], 2018. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. Accessed: 2019-11-01. Citado 1 vez na página 13.

KOTSIANTIS, Sotiris B; ZAHARAKIS, I; PINTELAS, P. Supervised machine learning: A review of classification techniques. **Emerging artificial intelligence applications in computer engineering**, v. 160, p. 3–24, 2007. Citado 2 vezes nas páginas 14, 15.

LEVY, Omer; GOLDBERG, Yoav. Dependency-based word embeddings. In: PROCEEDINGS of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). [S.l.: s.n.], 2014. v. 2, p. 302–308. Citado 1 vez na página 12.

OMIDVAR, Amin; JIANG, Hui; AN, Aijun. Using Neural Network for Identifying Clickbaits in Online News Media. In: SPRINGER. ANNUAL International Symposium on Information Management and Big Data. [S.l.: s.n.], 2018. p. 220–232. Citado 5 vezes nas páginas 17, 19, 21–24, 30.

PENNINGTON, Jeffrey; SOCHER, Richard; MANNING, Christopher D. GloVe: Global Vectors for Word Representation. In: EMPIRICAL Methods in Natural Language Processing (EMNLP). [S.l.: s.n.], 2014. p. 1532–1543. Disponível em: <http://www.aclweb.org/anthology/D14-1162>. Citado 2 vezes nas páginas 17, 24.

POTTHAST, Martin et al. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. **CoRR**, abs/1812.10847, 2018. arXiv: 1812.10847. Disponível em: <http://arxiv.org/abs/1812.10847>. Citado 3 vezes nas páginas 8, 9, 17.

SORATO, Danielly; FILETO, Renato. Linguistic Pattern Mining for Data Analysis in Microblog Texts Using Word Embeddings. In: PROCEEDINGS of the XV Brazilian Symposium on Information Systems. Aracaju, Brazil: ACM, 2019. (SBSI'19), 19:1–19:8. ISBN 978-1-4503-7237-4. DOI: 10.1145/3330204.3330228. Disponível em: <http://doi.acm.org/10.1145/3330204.3330228>. Citado 3 vezes nas páginas 8, 13, 14, 26.

THOMAS, Philippe. Clickbait identification using neural networks, 2017. Citado 2 vezes nas páginas 18, 19, 30.

ZHOU, Yiwei. Clickbait detection in tweets using self-attentive network, 2017. Citado 3 vezes nas páginas 17, 19, 30.

APÊNDICES

Apêndice A – Artigo do trabalho

Desenvolvendo um classificador de clickbait para tweets com word embeddings

Bruno S. Ferreira[1]

[1]Departamento de Informática e Estatística – Universidade Federal de Santa Catarina(UFSC)

bruno.s.ferreira@grad.ufsc.br

Abstract. *Clickbaits are a type of headlines that are empty but intriguing, with the objective of making the reader click on the article and access some website. With the recent propagation of this headlines, a search for some way of identifying them has been becoming more relevant. The shared task of the Clickbait challenge has helped advance the studies on this area, with many works competing to obtain the best results for a data set provided. This thesis studies proposals for clickbait classification and proposes improvements on the Clickbait Challenge work that uses word embeddings, using the text's semantics on a machine learning model trained with linear regression.*

Resumo. *Clickbaits são uma forma de título de notícia vago, porém intrigante, com objetivo de fazer o leitor clicar na notícia e acessar algum site. Com a propagação recente deste tipo de manchete, uma busca por uma maneira automática de detecta-los vem se tornando cada vez mais relevante. A tarefa compartilhada Clickbait Challenge ajudou a avançar os estudos desta área, com diversos trabalhos competindo para obter os melhores resultados para um conjunto de dados fornecido. Este TCC faz um estudo de propostas para classificação de clickbaits e propõe melhorias no trabalho do Clickbait Challenge que usa word embedding, usando Short Semantic Patterns num modelo de machine learining treinado com regressão linear.*

1. Introdução

Numa época onde os leitores têm acesso a sites e notícias gerais, nacionais e internacionais, blogs que contêm informações detalhadas sobre determinados tópicos, e muitas manchetes o tempo inteiro na sua tela, chamar a atenção se torna uma tarefa importante. Assim, escritores tiveram de se adaptar para sobreviver a este novo ambiente onde disputam entre si a atenção do internauta, buscando vizualições em suas páginas para conseguirem receitas de propagandas. Uma estratégia comumente usada é chamada de clickbait.

Clickbaits, segundo Hurst (2016), são uma forma de título de nóticia com proposito de ser vago, mas intrigante o suficiente para fazer o leitor clicar na notícia na sua rede social e ir para a página do artigo, onde ele consegue mais informações. Esses títulos são sensacionalistas por natureza e são feitos para chamar atenção e incitar curiosidade sobre suas histórias sem revelar informações importantes.

Detecção automática de clickbaits é um objetivo recente que vem se tornando cada vez mais relevante, conforme a frequência de clickbaits em redes sociais vem

aumentando. Em seu artigo, Potthast et al. (2018) indica todos os datasets com categorização de notícias em clickbaits ou não que conseguiu encontrar, num total de 6 datasets incluindo o seu próprio criado para o artigo, o primeiro desses formado em 2016. Neste curto período de tempo foram criados alguns métodos diferentes para detectar clickbaits, com prevalência de algoritmos utilizando redes neurais e deep learning. Este trabalho propõe uma extensão para um método já existente, fazendo a utilização de Short semantic patterns (SSP) para gerar uma melhoria de resultados.

Normalmente, em uma classificação de texto, é comum usar abordagens como Bag-of-words para recuperar informações de um texto. Essas abordagens usam palavras como indicadores e frequências de palavras como características para efetuar classificação de texto. Assim, os algoritmos de classificação acabam trabalhando apenas com terminologia para achar padrões, enquanto padrões conceituais ficam ignorados, tendo problemas com expressões, palavras sinônimas, palavras com múltiplos significados e problemas de generalização (BLOEHDORN; HOTHO, 2006). Estes são problemas que o uso da semântica pode resolver.

Segundo Gentry (2015), semântica trata do significado e da interpretação de palavras e frases. É esta que determina como nos comunicamos, como lemos e entendemos uns aos outros, e como interpretamos palavras. SSP é uma técnica desenvolvida por Sorato e Fileto (2019) que serve para minerar padrões semânticos em textos curtos, como os tweets. Com o uso destes padrões, é esperado gerar uma melhora na classificação e identificação de clickbaits.

O classificador de clikbaits desenvolvido neste trabalho é treinado e avaliado usando o dataset do Clickbait Challenge, uma tarefa compartilhada em busca de um classificador de clickbait criado por Potthast et al. (2018). Para a avaliação final deste trabalho, os resultados obtidos serão comparados com resultados obtidos por outros participadores do Clickbait Challenge.

2. Fundamentação

2.1. Clickbaits

Clickbait é uma estratégia utilizada para chamar atenção onde a notícia ou artigo tem um título demasiadamente vago, porém interessante o suficiente para induzir curiosidade, para conseguir cliques que levem à página com o artigo em si, conseguindo mais visualizações e maior renda de propagandas. A palavra clickbait tem sua origem no inglês, sendo uma junção dos termos click, significando clique, e bait, significando isca. Este tipo de notícia é frequentemente visto em redes sociais, como Facebook e Twitter, lugares em que conseguem atrair o maior número de pessoas e podem ser facilmente compartilhados a um número ainda maior (HURST, 2016). São exemplos de clickbaits:

- 10 things Apple didn’t tell you about the new iPhone
- What happened next will surprise you
- 9 things you must have to be a good data scientist
- This is what the actor/actress from 90s looks like now

Segundo Chen, Conroy e Rubin (2015), clickbaits são frequentemente enganosos e possuem informações não verificadas, sendo grandes contribuidores do crescente número de nóticias falsas, as fake news, na internet. Reportagens como estas podem

gerar consequencias reais, socialmente e até economicamente. Por exemplo, um rumor em 2008 que dizia que Steve Jobs havia sofrido um ataque cardíaco fez com que as ações da Apple caissem 10% em valor.

Alem disso, estudos cognitivos indicam que clickbaits provocam distração. Quanto mais os leitores continuam acessando novos artigos após serem atraidos pelos títulos, as trocas constantes levam a sobrecarga cognitiva, desencorajando leitores de ler artigos informativos e mais completos (CHAKRABORTY et al., 2016).

2.2. Word embeddings e Short Semantic Patterns

Word embeddings são vetores dimensionais que relacionam uma palavra e um determinado contexto. Estes vetores otimizam o processamento computacional, tornando ações complexas de palavras em operações com matrizes mais simples (LEVY; GOLDBERG, 2014). A ideia por trás dessa técnica é que palavras num mesmo contexto tendem a ter significados semelhantes, postando-se então próximas num gráfico vetorial.

Exemplificando o uso de word embeddings, as frases "Tenha um bom dia" e "Um ótimo dia para você" tem significados extremamente parecidos. Porém, para um sistema que só considera léxicos sem se preocupar com sua semântica, as frases podem ser um tanto diferentes, já que palavras como "bom" e "ótimo" são interpretadas como palavras, sem se preocupar com sua proximidade semântica. Os word embeddings servem como uma maneira de reduzir essa diferença, colocando palavras em espaços vetoriais onde seus significados podem melhor ser representados e comparados pelas máquinas (KARANI, 2018).

Short Semantic Patterns (SSP) é uma técnica desenvolvida por Sorato e Fileto (2019), que faz o uso de word embeddings para buscar semelhanças semânticas em blocos curtos de texto. Um SSP se referre a uma coleção de sequências de palavras de textos distintos, as quais podem ter tamanhos diferentes e léxicos distintos, mas carregam um significado similar. Um exemplo de SSP é ilustrado na Figura 1, através de algumas de suas instâncias que foram automaticamente mineradas em torno do word embedding referente à palavra sexismo usando o método proposto por Sorato e Fileto.

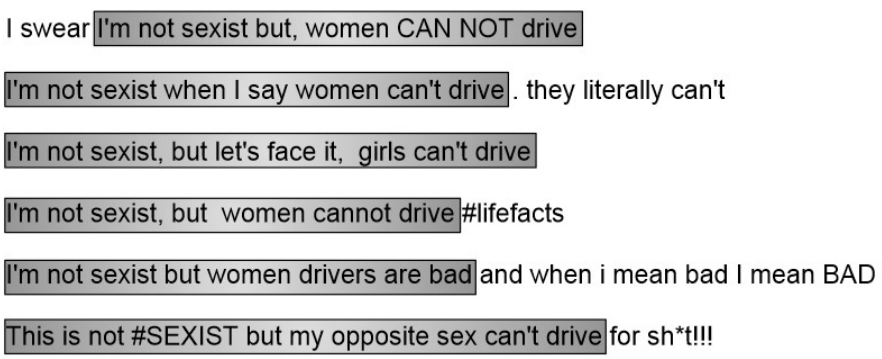


Figura 1. Exemplos de Instâncias de SSP referente a sexismo

A mineração de SSPs começa buscando por palavras ou expressões semanticamente similares (segundo os respectivos word embeddings) a alguma palavra-chave de um conjunto-fornecido (e.g. sexismo, misoginía) fornecido como foco para minerar SSP em documentos de texto. A partir de cada palavra ou expressão similar a

alguma palavra-chave dada como foco, vai sendo expandida uma janela de contexto, ou seja, as palavras adjacentes vão sendo acrescentada à instância do padrão enquanto for mantida similaridade semântica dentro de um certo patamar com blocos de outro texto (e.g. outro tweet). A Figura 2 apresenta uma amostra das janelas de contexto em um documento.

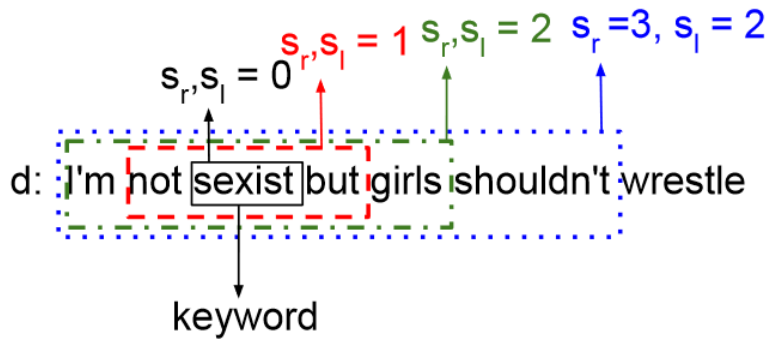


Figura 2. Exemplo de janela de contexto em um documento

3. Trabalhos Relacionados

Até o momento da escrita deste trabalho, haviam sido submetidos 16 trabalhos ao Clickbait Challenge, 10 destes com um artigo apresentado. Os dois que se sairam melhor, conforme as métricas do desafio, foram os classificadores do Omidvar, Jiang e An (2018) e do Zhou (2017).

O trabalho de Omidvar, Jiang e An (2018) utiliza-se de deep learning para criação de um modelo de rede neural bidirecional com unidades recorrentes bloqueadas (Gated recurring units, GRU). Para geração de features foram utilizados word embeddings pré-treinados provenientes da GloVe, de Pennington, Socher e Manning (2014).

Zhou (2017), por sua vez, fez o uso de rede neural auto-atenciosa (Self-attentive network). Diferente de uma rede neural com atencão externa, a rede auto-atenciosa consegue realizar a geração de features sozinha, sem necessitar de informações adicionais externas.

Grigorev (2017) utilizou o dataset disponibilizado pelo Clickbait Challenge, e também gerou um dataset próprio através de grupos do Facebook, onde existiam em maior parte postagens com clickbaits, para aumentar a quantidade de dados e gerar um classificador com maior nível de confiança. Aplicou um modelo de regressão linear em tipo de dado textual para geração de features e, com os resultados, gerou um segundo modelo para classificação.

No trabalho de Glenski et al. (2017), o modelo criado utiliza-se de todos os dados disponíveis no dataset, incluindo as imagens. Para usufruir de todos os dados, foram montadas duas arquiteturas de redes neurais linguisticamente infusas (Linguistically-infused neural network), utilizando uma camada de neurônios convolucionais (Convolutional Neural Network(CNN)) e outra fazendo o uso de Long Short-Term Memory (LSTM). Thomas (2017) também faz o uso da rede neural LSTM em seu trabalho, obtendo resultados parecidos.

Cao, Le et al. (2017) utiliza machine learning para criar seu modelo. São formuladas diversas novas características, entre elas nível de similaridade entre o tweet

e o artigo, entre o tweet e o título do artigo e entre o tweet e as palavras-chaves do artigo. Além dessas, características discutidas em trabalhos anteriores foram também analisadas, chegando a um total de 331 features, das quais foram selecionadas 180 para evitar o sobreajuste (overfitting), e essas por sua vez foram avaliadas e reduzidas as 60 mais relevantes.

O trabalho de Gairola et al. (2017) propõe um modelo de redes neurais que consiste de três partes: LSTM bidirecional com atenção, rede neural siamesa com text embeddings, e rede neural siamesa com visual embeddings.

Por fim, o trabalho de Indurthi e Oota (2017) é a base deste trabalho. Utiliza os word embeddings pré-treinados da GloVe, com 300 dimensões, e outras features geradas manualmente: presença de perguntas, presença de números no começo do tweet, presença de gerúndio no verbo e presença de adjetivos superlativos. Com estas features, é treinado um modelo de aprendizado de máquina com regressão linear.

Este trabalho foi o escolhido por ser um modelo simples, o qual já fazia a utilização de word embeddings, tornando o desenvolvimento mais fácil e fazendo a comparação ter mais relevância. Nosso trabalho irá alterá-lo de maneira a utilizar os SSPs para obter um resultado melhor. Acredita-se que a semântica representa um fator importante na detecção de clickbaits, e as SSPs podem validar isso.

4. O modelo desenvolvido

4.1. Ambiente

O ambiente onde foi feito o modelo de classificação e realizado os experimentos é um computador rodando Linux Ubuntu 19.04, com processador Intel Core i5 5ª geração, 3,8 GB de memória RAM.

Para desenvolver tal modelo foi utilizado a linguagem Python 3.7 com os seguintes módulos:

- numbpy 1.15.4 para operações matemáticas;
- nltk 3.4 para processamento de dados;
- spaCy 2.0.16 para geração de word embeddings;
- scikit-learn 0.20.1 para aprendizado de máquina.

4.2. Desenvolvimento

Para contruir o novo modelo, o dataset foi filtrado previamente. Os 21997 posts foram tokenizados, e ocorreu a remoção de stop words daqueles que as tinham. Após esses processos, é realizado a lematização desses documentos, tornando as palavras flexionadas em seus radicais, para melhor comparação e similarização. Nesta ação, todos os números encontrados em textos foram transformados em um token "[n]" para melhor representação e similaridade.

Para melhorar o modelo base, foram formuladas features adicionais baseadas nas semânticas dos textos. Tais features consistem de instâncias de Short Semantic Patterns (SSP) minerados automaticamente em textos como proposto por Sorato e Fileto (2019).

Como clickbaits em geral não possuem um tema único, podendo ser variados tipos de notícias, é difícil saber qual palavra chave utilizar para a extração de SSPs. Então, foi realizada uma pré analise para encontrar o token mais utilizada pelos tweets.

O token "[n]" foi a palavra mais utilizada, ou seja, os números são as "palavras" mais recorrentes nos posts. Então, será utilizado o token "[n]" para produzir os SSPs.

Para preparar o dataset para extração de SSPs, foram retirados todos os tweets que não haviam a ocorrência de algum número. Isto reduz o tamanho do dataset para um total de 3922 documentos, sendo 2336 não-clickbaits e 1586 clickbaits.

Depois da filtragem, os dados foram divididos em suas respectivas classes para realizar a extração de SSPs, com a intenção de encontrar SSPs diferentes em não-clickbaits e clickbaits. Por tanto, utilizando o número em cada documento como palavra chave, estes foram comparados entre si, com sequencias de até 2 palavras antes ou depois do número, para buscar uma semelhança de até 22,5%, utilizando os word embeddings para encontrar esta semelhança. Caso exista semelhança com algum outro documento, o item sendo analisado é guardado para processamento, com sua instância SSP sendo sua feature. Caso contrário, o item é descartado.

Com não-clickbaits e clickbaits filtrados por SSPs, sobram no final um total de 1964 não-clickbaits e 881 clickbaits. Assim como no trabalho base, os não-clickbaits são reduzidos ao número de clickbaits de maneira aleatória, afim de produzir um resultado mais significativo, levando o total de documentos analisados para 1762. É então feita a partição deste dataset em 80%, num total de 1409, para treino do modelo e 20%, total de 353, para teste e é gerado o treinamento e predições. A Figura 8 mostra como ficou a arquitetura do modelo. Visto que o dataset final é diferente do dataset usado no código baseline, foi reproduzido o teste no código baseline usando o novo dataset também para fins de comparação.

O modelo desenvolvido é o mesmo utilizado pelo algoritmo base, aprendizado demáquina com regressão linear para predição da nota de cada tweet.

5. Resultados

O método utilizado para realizar a avaliação do modelo é oferecido pelo Clickbait Challenge. Este utiliza as métricas do módulo scikit-learn para recuperar os resultados.

Como mostra a Tabela 1, o modelo desenvolvido atingiu resultados melhores que o modelo base todas categorias de avalização por margens significantes, quando é utilizado o mesmo dataset. F1 Score, precisão, recall e acurácia aumentaram em 0,20 no mínimo, enquanto o MSE foi diminuido pela metade.

Com o conjunto de dados sendo reiterado e filtrado para caber no modelo desenvolvido, é de se esperar que o nosso modelo se saia melhor. Por isso é importante realizar a comparação com outros modelos e com resultados obtidos sobre todo o dataset.

Modelo	MSE	F1	Precisão	Recall	Acurácia
Modelo desenvolvido	0,113	0,793	0,788	0,797	0,796
Modelo de Indurthi e Oota (2017)	0,282	0,555	0,588	0,526	0,529

Tabela 1. Comparação entre o modelo desenvolvido e o modelo base

Mesmo comparando com os outros resultados do Clickbait Challenge, o modelo desenvolvido neste trabalho possui seus méritos. Este se destaca por obter a maior precisão e maior F1 Score entre todos os resultados obtidos. Além disso, observa-se que o recall do modelo base é o melhor dentre todos, sendo quase uma anomalia no meio dos outros, com o produzido ficando logo atrás.

Porém onde o modelo perde é no MSE. Com resultado de 0,113, este não é capaz de disputar com os modelos mais bem avaliados, ficando até 0,08 pontos atrás. A acurácia também não é párea aos superiores. Contudo, o modelo apresentou resultados competitivos em relação a outros modelos desenvolvidos com tecnologias superiores, podendo ser considerado um sucesso.

6. Conclusão e trabalhos futuros

Este trabalho apresentou o desenvolvimento de um modelo para classificação de clickbaits baseado em short semantic patterns aparentes em tweets, usufruindo do dataset disponibilizado pelo clickbait challenge. Foi desenvolvido um modelo de classificação linear baseado no modelo de Indurthi e Oota (2017) fazendo a utilização de short semantic patterns como feature para treinamento, os resultados obtidos foram comparados com os de outros modelos por sobre o dataset do clickbait challenge.

Os resultados obtidos pelo modelo comprovam que o uso de features baseadas em semântica, tais como embeddings e padrões SSP, é relevante na busca por clissificação de clickbaits. Enquanto outros modelos possuem melhor desempenho em algumas medidas, o classificador desenvolvido melhora muito o modelo base e alcança resultados competitivos, principalmente em termos de precisão e medida F1, contra modelos que utilizam tecnologias mais avançadas.

Como trabalhos futuros:

- Melhorar o desenvolvimento do modelo, adicionando novas features, utilizando os outros dados disponíveis e refinando a extração de SSP;
- Realizar a extração de semelhança em imagens, disponibilizadas no dataset;
- Aplicar o modelo em outro dataset, com uma maior quantidade de dados e mais bem direcionado para extração de SSPs.

Referências

BLOEHDORN, Stephan; HOTHO, Andreas. Boosting for Text Classification with Semantic Features. In: MOBASHER, Bamshad et al. (Ed.). Advances in Web Mining and Web Usage Analysis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 149–166. ISBN 978-3-540-47128-8.

CAO, Xinyue; LE, Thai et al. Machine learning based detection of clickbait posts in social media, 2017.

CHAKRABORTY, A. et al. Stop Clickbait: Detecting and preventing clickbaits in online news media. In: 2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). [S.l.: s.n.], ago. 2016. p. 9–16. DOI:10. 1109/ASONAM.2016.7752207.

CHEN, Yimin; CONROY, Niall J.; RUBIN, Victoria L. Misleading Online Content: Recognizing Clickbait As "False News". In: PROCEEDINGS of the 2015 ACM on Workshop on Multimodal Deception Detection. Seattle, Washington, USA: ACM,

2015. (WMDD'15), p. 15–19. ISBN 978-1-4503-3987-2. DOI: 10.1145/2823465.2823467. Disponível em: <http://doi.acm.org/10.1145/2823465.2823467>.

GAIROLA, Siddhartha et al. A neural clickbait detection engine. arXiv preprint arXiv:1710.01507, 2017.

GENTRY, Angela. What is semantics? [S.l.: s.n.], 2015. https://study.com/academy/

lesson/what-is-semantics-definition-examples-quiz.html. Accessed: 2019-06-28.

GLENSKI, Maria et al. Fishing for clickbaits in social images and texts with linguistically infused neural network models, 2017.

GRIGOREV, Alexey. Identifying clickbait posts on social media with an ensemble of linear models, 2017.

HURST, Nathan. To clickbait or not to clickbait? an examination of clickbait headline effects on source credibility. 2016. Tese (Doutorado) – University of Missouri Columbia.

INDURTHI, Vijayasaradhi; OOTA, Subba Reddy. Clickbait detection using word embeddings, 2017.

KARANI, Dhruvil. Introduction to Word Embedding and Word2Vec. [S.l.: s.n.], 2018. https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa. Accessed: 2019-11-01.

LEVY, Omer; GOLDBERG, Yoav. Dependency-based word embeddings. In: PROCEEDINGS of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). [S.l.: s.n.], 2014. v. 2, p. 302–308.

OMIDVAR, Amin; JIANG, Hui; AN, Aijun. Using Neural Network for Identifying Clickbaits in Online News Media. In: SPRINGER. ANNUAL International Symposium on Information Management and Big Data. [S.l.: s.n.], 2018. p. 220–232.

POTTHAST, Martin et al. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. CoRR, abs/1812.10847, 2018. arXiv:1812.10847. Disponível em: <http://arxiv.org/abs/1812.10847>.

SORATO, Danielly; FILETO, Renato. Linguistic Pattern Mining for Data Analysis in Microblog Texts Using Word Embeddings. In: PROCEEDINGS of the XV Brazilian Symposium on Information Systems. Aracaju, Brazil: ACM, 2019. (SBSI'19), 19:1–19:8. ISBN 978-1-4503-7237-4. DOI: 10.1145/3330204.3330228. Disponível em: <http://doi.acm.org/10.1145/3330204.3330228>.

THOMAS, Philippe. Clickbait identification using neural networks, 2017.

ZHOU, Yiwei. Clickbait detection in tweets using self-attentive network, 2017.

Apêndice B – Código fonte

O código fonte também pode ser encontrado em https://github.com/Brusifer/classification-project.

```
import json
import sys
import os
import libspacy
import string
# import libgrams
import numpy as np
from math import *
from tqdm import tqdm
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn import linear_model
from sklearn.utils import shuffle
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet
from nltk.corpus import stopwords
from collections import Counter
#from sklearn.manifold import TSNE
import libglove

lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))
# exclude = set(string.punctuation)

def main():
  train_dir1 = 'clickbait17-train-170331'
  instances_filename = 'instances.jsonl'
  truths_filename = 'truth.jsonl'
  train_dir2 = 'clickbait17-train-170630'
  instances_filename = 'instances.jsonl'
  truths_filename = 'truth.jsonl'
  raw_data={} #Data set indexed by the id, id is a string
  raw_truths={} #Truths indexed by the id, id is a string


  abs_path = os.path.join(train_dir1, instances_filename)
  fp = open(abs_path)
  for line in fp:
    json_obj = json.loads(line)
    #print json_obj['postText']
    item_id = json_obj['id']
    raw_data[item_id]=json_obj

  #print raw_data

  fp.close()
  abs_path = os.path.join(train_dir2, instances_filename)
  fp = open(abs_path)
  for line in fp:
    json_obj = json.loads(line)
```

```
  #print json_obj['postText']
  item_id = json_obj['id']
  raw_data[item_id]=json_obj


abs_path = os.path.join(train_dir1, truths_filename)
fp = open(abs_path)
for line in fp:
  json_obj = json.loads(line)
  item_id = json_obj['id']
  raw_truths[item_id]=json_obj

abs_path = os.path.join(train_dir2, truths_filename)
fp = open(abs_path)
for line in fp:
  json_obj = json.loads(line)
  item_id = json_obj['id']
  raw_truths[item_id]=json_obj

feature = 'postText'

raw_cb = []
y_cb=[]
y_labels=[]
item_ids=[]
cb=[]
nocb=[]
item_ids_cb=[]
item_ids_nocb=[]
all_words=[]
for item_id in tqdm(raw_data):
  rating = raw_truths[item_id]['truthMean']
  postText = raw_data[item_id].get('postText','')
  postText = clean_str(postText[0].strip())
  postText = clean_title(postText)
  label=raw_truths[item_id]['truthClass']
  #sys.exit()

  item = raw_data[item_id]
  item['rating'] = rating
  item['postText']=postText
  item['id']=item_id

  postWords = word_tokenize(postText)
  item['postWords']=postWords

  postWords = [w for w in postWords if not w in stop_words]
  # postWords = [w for w in postWords if not w in exclude]

  pos = nltk.pos_tag(postWords)
  item['pos']=pos

  lemmatized = []

  for word, tag in pos :
    if word == 'http' or word == ':' or word == '//t': continue
    if word.isdigit() :
      lemmatized.append('[n]')
      continue
```

```
      if tag[0] == 'J':
        part = wordnet.ADJ
      elif tag[0] == 'N':
        part = wordnet.NOUN
      elif tag[0] == 'V':
        part = wordnet.VERB
      elif tag[0] == 'R':
        part = wordnet.ADV
      else :
        part = wordnet.NOUN

      lemmatized.append(word)

    item['lemmatized'] = lemmatized

    all_words.extend(lemmatized)

    item['lemmaSent'] = ' '.join(lemmatized)

    if '[n]' not in item['lemmaSent']: continue

    raw_cb.append(item)
    if label=='clickbait':
      cb.append(item)
    else:
      nocb.append(item)
    label = rating_to_class(rating)
    y_cb.append(rating)
    y_labels.append(label)
    item_ids.append(item_id)
  fp.close()

  # print(Counter(all_words).most_common(10))

  nocb_filtered = []
  cb_filtered = []

  print('Filtering nocb')
  for item in tqdm(nocb):
    sent_array = item['lemmaSent'].split()
    if len(sent_array) < 2: continue
    word_idx = sent_array.index('[n]')
    for comparison in nocb:
      if item['postText'] == comparison['postText']: continue
      comparison_sent_array = comparison['lemmaSent'].split()
      if len(comparison_sent_array) < 2: continue
      comparison_idx = comparison_sent_array.index('[n]')
      if(word_idx > len(sent_array)-2 and comparison_idx < 1) or
(word_idx < 1 and comparison_idx > len(comparison_sent_array) -2):
continue
      if(word_idx > len(sent_array)-2 or comparison_idx >
len(comparison_sent_array) -2):
        word_vec = libspacy.get_vector(' '.join([item['lemmaSent']
[word_idx-2], item['lemmaSent'][word_idx-1], item['lemmaSent']
[word_idx]]))
        comparison_vec = libspacy.get_vector('
'.join([comparison['lemmaSent'][comparison_idx-2],
comparison['lemmaSent'][comparison_idx-1], comparison['lemmaSent']
```

```
[comparison_idx]]))
        difference = sqrt(sum(pow(a-b,2) for a, b in zip(word_vec,
comparison_vec)))
      elif (word_idx < 1 or comparison_idx < 1):
        word_vec = libspacy.get_vector(' '.join([item['lemmaSent']
[word_idx], item['lemmaSent'][word_idx+1], item['lemmaSent']
[word_idx+2]]))
        comparison_vec = libspacy.get_vector('
'.join([comparison['lemmaSent'][comparison_idx],
comparison['lemmaSent'][comparison_idx+1], comparison['lemmaSent']
[comparison_idx+2]]))
        difference = sqrt(sum(pow(a-b,2) for a, b in zip(word_vec,
comparison_vec)))
      else:
        word_vec = libspacy.get_vector(' '.join([item['lemmaSent']
[word_idx -1], item['lemmaSent'][word_idx], item['lemmaSent']
[word_idx+1]]))
        comparison_vec = libspacy.get_vector('
'.join([comparison['lemmaSent'][comparison_idx-1],
comparison['lemmaSent'][comparison_idx], comparison['lemmaSent']
[comparison_idx+1]]))
        difference = sqrt(sum(pow(a-b,2) for a, b in zip(word_vec,
comparison_vec)))
      if difference > 22.5 :
        nocb_filtered.append(item)
        item_ids_nocb.append(item['id'])
        item['ssp'] = word_vec
        break

  print('Filtering cb')
  for item in tqdm(cb) :
    sent_array = item['lemmaSent'].split()
    if len(sent_array) < 2: continue
    word_idx = sent_array.index('[n]')
    for comparison in cb:
      if item['postText'] == comparison['postText']: continue
      comparison_sent_array = comparison['lemmaSent'].split()
      if len(comparison_sent_array) < 2: continue
      comparison_idx = comparison_sent_array.index('[n]')
      if(word_idx > len(sent_array)-2 and comparison_idx < 1) or
(word_idx < 1 and comparison_idx > len(comparison_sent_array) -2):
continue
      if(word_idx > len(sent_array)-2 or comparison_idx >
len(comparison_sent_array) -2):
        word_vec = libspacy.get_vector(' '.join([item['lemmaSent']
[word_idx-2], item['lemmaSent'][word_idx-1], item['lemmaSent']
[word_idx]]))
        comparison_vec = libspacy.get_vector('
'.join([comparison['lemmaSent'][comparison_idx-2],
comparison['lemmaSent'][comparison_idx-1], comparison['lemmaSent']
[comparison_idx]]))
        difference = sqrt(sum(pow(a-b,2) for a, b in zip(word_vec,
comparison_vec)))
      elif (word_idx < 1 or comparison_idx < 1):
        word_vec = libspacy.get_vector(' '.join([item['lemmaSent']
[word_idx], item['lemmaSent'][word_idx+1], item['lemmaSent']
[word_idx+2]]))
        comparison_vec = libspacy.get_vector('
'.join([comparison['lemmaSent'][comparison_idx],
```

```
comparison['lemmaSent'][comparison_idx+1], comparison['lemmaSent']
[comparison_idx+2]]))
        difference = sqrt(sum(pow(a-b,2) for a, b in zip(word_vec,
comparison_vec)))
      else:
        word_vec = libspacy.get_vector(' '.join([item['lemmaSent']
[word_idx -1], item['lemmaSent'][word_idx], item['lemmaSent']
[word_idx+1]]))
        comparison_vec = libspacy.get_vector('
'.join([comparison['lemmaSent'][comparison_idx-1],
comparison['lemmaSent'][comparison_idx], comparison['lemmaSent']
[comparison_idx+1]]))
        difference = sqrt(sum(pow(a-b,2) for a, b in zip(word_vec,
comparison_vec)))
      if difference > 22.5 :
        cb_filtered.append(item)
        item_ids_cb.append(item['id'])
        item['ssp'] = word_vec
        break


  print('Number of cb: ', len(cb_filtered), 'Number of nocb: ',
len(nocb_filtered))
  cb=shuffle(cb_filtered, random_state=0)
  nocb=shuffle(nocb_filtered, random_state=0)
  nocb=nocb[:len(cb)]
  print ("CB=", len(cb), "NOCB=", len(nocb), "ITEM_IDS_CB",
len(item_ids_cb), "ITEM_IDS_NOCB", len(item_ids_nocb))
  item_ids_nocb=item_ids_nocb[:len(cb)]
  y_cb = [0]*len(nocb)+[1]*len(cb)
  raw_cb = nocb + cb
  item_ids = item_ids_nocb + item_ids_cb
  (raw_cb, y_cb, item_ids) = shuffle(raw_cb, y_cb, item_ids,
random_state=0)


  # (raw_cb, y_cb, item_ids) = shuffle(raw_cb, y_cb, item_ids,
random_state=0)

  #create the dataset for subba
  fa=open('clickbait_titles.txt','w')
  fb=open('clickbait_ratings.txt','w')
  for (cb, rating) in zip(raw_cb, y_cb):
    fa.write(cb['postText']+'\n')
    fb.write(str(rating)+'\n')
  fa.close()
  fb.close()


  #(X, Y) = make_scatter(raw_cb, y_cb)
  train_percent=0.8
  train_size=int(len(raw_cb)*train_percent)
  X_raw_train = raw_cb[:train_size]
  y_train = y_cb[:train_size]
  train_ids = item_ids[:train_size]


  X_raw_test = raw_cb[train_size:]
  y_test = y_cb[train_size:]
```

```
  test_ids = item_ids[train_size:]

  #create test annotations
  fp=open("test_annotations.jsonl",'w')
  for item_id in test_ids:
    json_obj = raw_truths[item_id]
    json_str = json.dumps(json_obj)
    fp.write(json_str+'\n')
  fp.close()

  print ("X_raw_train, y_train", len(X_raw_train), len(y_train))
  print ("X_raw_test, y_test", len(X_raw_test), len(y_test) )
  X_train=[]
  X_test=[]

  print("Extracting features from train")
  for item in X_raw_train:
    # raw_title = item['postText']
    # vectors = libspacy.get_vector(raw_title)
    features = np.append(item['ssp'],[len(item['postWords'])])
    X_train.append(features)

  print( "Extracting features from test")
  for item in X_raw_test:
    # raw_title = item['postText']
    # vectors = libspacy.get_vector(raw_title)
    features = np.append(item['ssp'],[len(item['postWords'])])
    X_test.append(features)

  num_features = len(features)
  print( "Size of train, test", len(X_train), len(X_test))
  print( "Size of  labels train, test", len(y_train), len(y_test))
  print( "#features=", num_features)

  print("Try linear regression")
  model = linear_model.LinearRegression()
  #model = svm.SVR(C=1.0, epsilon=0.2)
  model.fit(X_train, y_train)
  print("Mean squared error test: %.4f" %
np.mean((model.predict(X_test) - y_test) ** 2))
  print("Mean squared error train: %.4f" %
np.mean((model.predict(X_train) - y_train) ** 2))

  print("Minor improvements")
  y_pred = model.predict(X_test)
  y_pred = [ 0 if i < 0 else i for  i in y_pred]
  y_pred = [ 1 if i > 1 else i for  i in y_pred]
  y_pred = np.array(y_pred)
  print("Mean squared error test: %.4f" % np.mean((y_pred - y_test) **
2))
  #print y_pred
  #y_pred = np.random.rand(len(X_test)) #Uncomment this line to check
with random guesses
  create_predictions("test_predictions", y_pred, test_ids)
  create_predictions("test_truths", y_test, test_ids)
  #Print those instances where the prediction varies by a threshold

  fp=open('max_errors.txt','w')
  for (y_p, y_real, test_id) in zip(y_pred, y_test, test_ids):
```

```
    if abs(y_p - y_real) > 0.4:
      line ='%s %f %f' % (raw_data[test_id][feature], y_p, y_real)
      fp.write(line+'\n')
  fp.close()
  #print(model.coef_)
  #print(sorted(model.coef_.tolist()))
  os.system('python eval.py test_annotations.jsonl test_predictions
outfile')
  print (model.coef_, model.intercept_)

#End of main


def create_raw_file(filename, data):
  fd = open(filename,'w')
  for row in data:
    fd.write(row+'\n')
  fd.close()

def rating_to_class(rating):
  if rating > 0.75:
    return 3
  if rating > 0.5:
    return 2
  if rating > 0.25:
    return 1
  return 0

def create_predictions(filename, predictions, item_ids):
  fd = open(filename, 'w')
  for (item_id, prediction) in zip(item_ids, predictions):
    json_obj={"id":item_id, "clickbaitScore":float(prediction)}
    json_str = json.dumps(json_obj)
    fd.write(json_str+'\n')

  fd.close()

def clean_title(title):
  title=title.replace("'", " ")
  title=title.replace("  ", " ")
  words = title.lower().split(' ')
  words = [ w for w in words if not w.startswith('@')]
  words = [ w for w in words if not w.startswith('#')]
  words = [ w for w in words if not w.startswith('rt')]
  #words = [ w for w in words if len(w) > 1 and not w[0].isdigit()]

  return ' '.join(words)

def clean_str(sentence):
  sentence = sentence.replace('\n',' ')
  return ''.join([c if ord(c) < 128 else ' ' for c in sentence])


if __name__ == "__main__":
  main()
```