Detecting Clickbaits in Tweets

Authors: Gondar, Alexandre[1]; Lopes, Miguel[2]; Magro, Helena[3]; Ribeiro, Ana[4]

Abstract: This project aims to identify whether a news post in Twitter is a clickbait or not. Clickbait is a form of false advertisement which uses text and/or images designed to attract attention and entice users to follow the link and read, view, or listen to the linked piece of online content, with a defining characteristic of being deceptive, typically sensationalized or misleading.[1] Many papers have been written about clickbait detection techniques, but many as well fail in integrating the image as an input to detect clickbaits in the posts. Vaibhaw et al (2018)[2] have created a model that not only incorporates textual features, modeled using BiLSTM and augmented with an attention mechanism, but also considers related images for clickbait detection. This article sets itself to improve this model. To do so, we have applied a bidirectional LSTM with an attention mechanism to understand the effect a word has on classifying a post as a clickbait or not; a Siamese net to capture the similarity between the text on the post and the information on the article it refers to; and a CNN-model for the images. In the end, the concatenation of the three models will serve as input to a fully connected layer. This model achieved a RSME of 0.118 and an accuracy of 83.2%.

Keywords: Clickbait, Embedding, Neural Network, LSTM, Attention-Mechanism, Siamese Network, Text Embeddings, Image Embeddings

TASK DEFINITION

Nowadays there is an increasing concern regarding the quality content on the web, such as spam or fake news. Even though clickbait may not necessarily be spam or fake news, it can be considered as a form of false advertisement.

The increasingly competitive market of the social media news, combined with the fact that Twitter's publisher's incomes are highly dependent on the clicks they generate, lead to the emergence of clickbaits. Clickbait is commonly defined as intentionally over-promising and under-delivering[1] - in a headline, on social media, in an image, or some combination of the mentioned.

[1] M20181383, m20181383@novaims.unl.pt, [2] M20180874, m20180874@novaims.unl.pt, [3] M20181404, 20181404@novaims.unl.pt, [4] M20180956, m20180956@novaims.unl.pt

Clickbait distorts the article's content and mislead the readers. For those reasons, it is very important to predict in a simple and fast manner if the post is or not a clickbait.

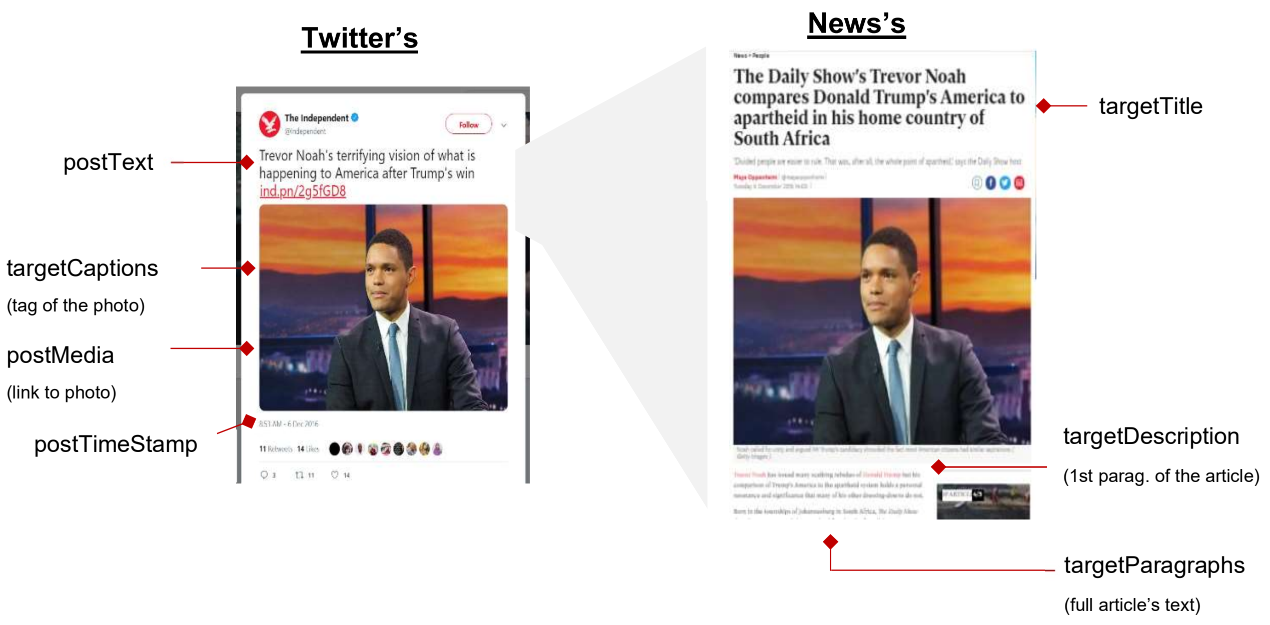
This project is based on the clickbait challenge dataset and is based on the article Vaibhaw et al (2018)[2] with some adjustments. This article presents a methodology that integrates both images and text.

DATA EXPLORATION

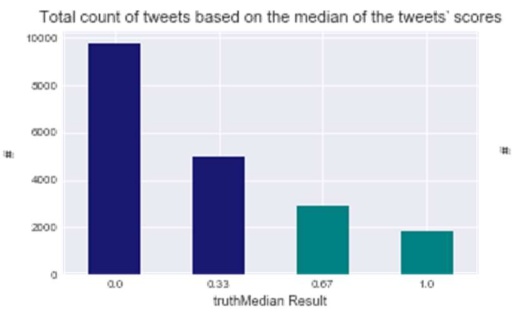
Extraction The dataset was obtained through the Clickbait Challenge3, concerning posts from the period between December 2016 and April 2017. The dataset is composed by 3 different sources:

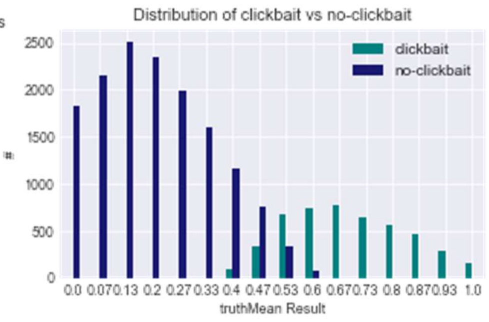
- Instances.jsonl: A line delimited JSON file. Each line is a JSON-Object containing the information that clickbait challenge team extracted for a specific post and its target article;
- Truth.jsonl: A line delimited JSON file. Each line is a JSON-Object containing the crowdsourced clickbait judgements of a specific post;
- Media/: A folder that contains all the images referenced in the instances file.

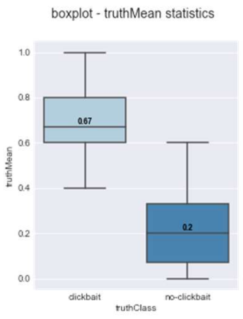
Description The clickbait challenge's dataset includes posts from Twitter and the related article it refers to. The following example is a real case of a clickbait scenario:


Twitter's
News's
The Independent
@Independent
Follow
Trevor Noah's terrifying vision of what is happening to America after Trump's win
ind.pn/2g5fGD8
The Daily Show's Trevor Noah compares Donald Trump's America to apartheid in his home country of South Africa
targetTitle
postText
targetCaptions
(tag of the photo)
postMedia
(link to photo)
postTimeStamp
targetDescription
(1st parag. of the article)
targetParagraphs
(full article's text)

Each of these posts (limited up to 140 characters) can be accompanied by an image and it is classified as a clickbait or non/clickbait, given the average score given by users.

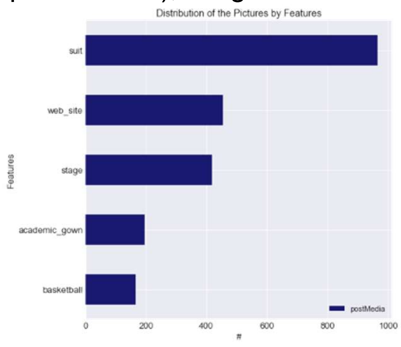


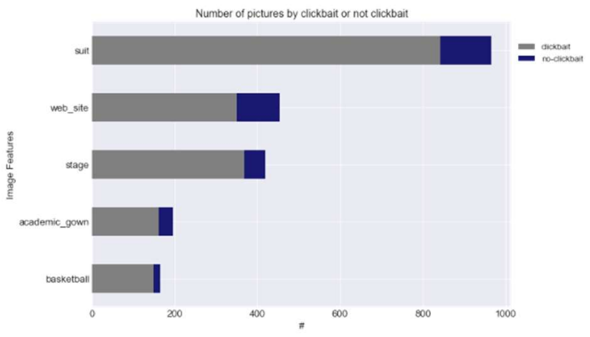




From the figures above, one can immediately infer that the dataset is slightly imbalanced, since the number of no-clickbait posts are approximately 2/3 of the all dataset. It can also be analyzed the min, quartile1, median, quartile3 and max of the score (*truthMean*) for all the tweets.

The dataset includes 11.067 images. When analyzing which image theme is more frequently posted (top 5 features), we get the following results:





For the purpose of this article, the variables that are going to be analyzed, in order to correctly categorize a post as clickbait or not, are the *postText* (text of the post with links removed), the *targetDesciption* (description tag of target article) and the *postMedia* (image associated, if exists, to the posts).

Readers are more likely to be baited into disappointing low-quality articles if a post has a misleading title (*postText*), that exaggerates or distorts the content of the news article it refers to. In the same way, many times posts use images (*postMedia*) to capture the readers' attention to the article, leading to a frustrating experience, if the latter does not correspond to the expectations created by the image.

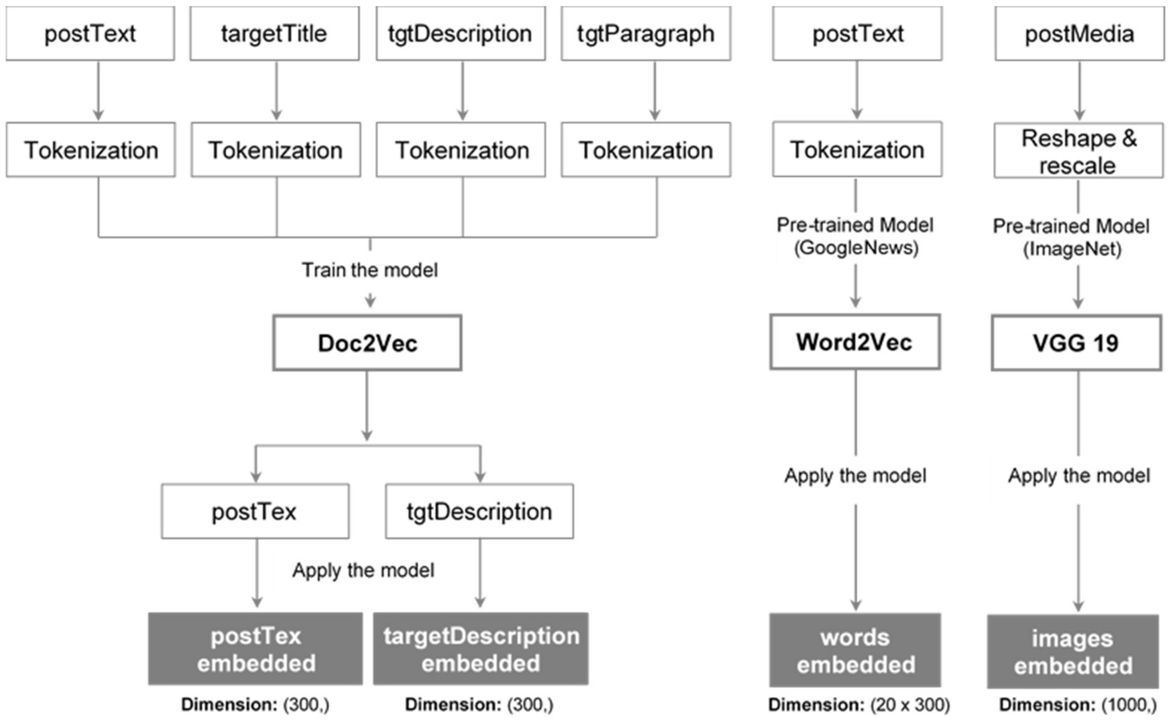
The *targetDescription*, i.e. the first paragraph of the article, generally contains the main idea that the article intends to expose. Hence, when compared with the *postText*, it immediately suggests if a post is a clickbait or not. On the other hand, the article title (*targetTitle*) was not considered as an input to our model, since it can be on its own a clickbait, not containing the real purpose of the article.

Additionally, it was performed a comparison analysis amongst the Levenshtein distances[4] between *postText* (bait field) and *targetDescription* and between *postText* and *targetTitle* and their correlation with the clickbait variable (*truthMean*). It concluded that the first distance is more correlated with the clickbait variable (*truthMean*) than the latter.

correlation	truthMean
Ld_postText_targetDescription	0.31
Ld_postText_targetTitle	- 0.17

DATA PREPROCESSING

In order to use simultaneously text and images, these needed to be imbedded. To do so, different techniques were applied, differing on the purposes of each input:



Documents Embedding For the purpose of features engineering over text sentences we used doc2vec (gensim implementation). Doc2vec is a modification of the word2vec algorithm to unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document. The goal of doc2vec is to create a vectorial representation of a document, regardless of its length.

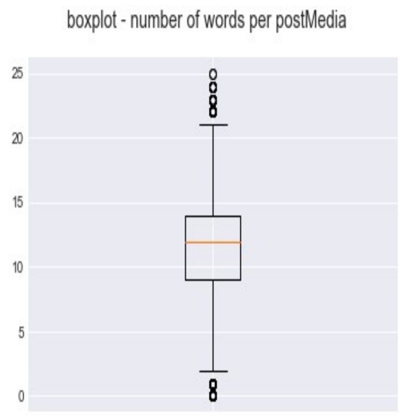
To implement Doc2Vec, it had to be previously trained. In order to do so, most of the text variables in the dataset were tokenized (*postText*, *targetDescription*, *targetTitle* and *targetParagraph*), and associated with a label in order to build the vocabulary for Doc2Vec. The alpha learning rate was optimized by 0.002 (initiated at 0.025) for each increase in the epoch number. The optimization had to be employed since the model does not do it automatically[5].

With the trained model, the *postText* and *targetDescription* were embedded into a vector o (300,). This dimension was chosen to be consistent with the dimension of word2Vec applied later in the article.

Words Embedding Embedding is the dominant technique to create a numerical representation of the semantical meaning of words. In order to do word embedding, it was applied a pre-trained 300-dimensional Word2Vec, a two-layer neural net trained over 100 billion words in Google News – aligned with the scope of this project (news post). Its purpose is to group the vectors of similar words together in a vector-space. Word2vec creates vectors that are distributed numerical representations of word features, such as the context of individual words.

It was decided that the final dimension of the words embedded would be (20 x 300) for two reasons: firstly, the maximum number of words used in the posts of our dataset was 25 (with an average of 11); secondly, the Word2Vec is already pre-trained considering a 300 dimension.

This model was applied for each word of the *postTitle*, in order to obtain its equivalent Word2Vec embedding.



Images Embedding Considering the image embedding, it was used a convolutional neural networks (CNNs) pre-trained on a large data for feature extraction. The image classification model used was VGG19, pre-trained with ImageNet, which is available in Keras and Tensorflow libraries. It was also the model used by the reference article. This model achieves 92,7% of accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1.000 classes. It was submitted to ILSVRC (ImageNet Large Scale Visual Recognition Challenge) and it is well-known for having great performance.

The first step, in the image embedding process, was to load and reshape the images to the first VGG19 convolution layer size – (1x224x224x3). Then the VGG19 model was applied. The images passed through a stack of convolution layers and three fully connected layers followed a stack of convolution layers followed by a softmax layer[6].

After running the pre-trained VGG19 model with the clickbait dataset, the most frequent features were attributed to each post in an array with a dimension of (1000,) – creating the input to the final model.

THE MODEL / APPROACH

The model created for detecting clickbait in the news post is composed by three main components: a bidirectional LSTM (biLSTM) on the words embedding, a Convolutional Neural network model on the images embedding and a Siamese neural network on document embedding. Before running the model, the dataset was split into training (56%), testing (30%) and validation (14%) data.

- **Bidirectional LSTM with Attention Mechanism on the words embedding**

For the analysis of the impact of the words in the *postText*, a bidirectional LSTM (biLSTM) was applied to the words embedded. It has been decided not to implement a standard RNN as it suffers from a problem of vanishing gradients[7] - meaning it does not efficiently model dependencies and interactions between words that are a few steps apart. Since LSTMs use gating mechanisms, they are more able to tackle this issue.

The biLSTM, contrarily to the normal LSTM model, has two networks, one that access information in forward direction and another that access in the reverse direction, thus including preceding and succeeding contexts in the generated output[8]. This feature enables, in general, biLSTMs to have better performance than LSTM when classifying a word. Additionally, an attention mechanism was employed. Attention mechanisms take two sentences, turns them into a matrix where the words of one sentence form the columns, and the words of another sentence form the rows, and then it makes matches, identifying relevant context[9]. At last, the features extracted from the attention mechanism are classified by the activation function classifier.

- **Convolutional Neural network model on the images embedding**

Convolutional Neural Networks (CNNs) is the most popular neural network model being used for image classification problem. One of the great advantages it has relatively to other models is that it allows to have fewer weights as some parameters are shared due to the coherence of local space of the inputs. This process, taking the form of convolutions, makes them especially well-suited to extract relevant information at a low computational cost[10]. The CNN applied in this article was formed by three hidden layers.

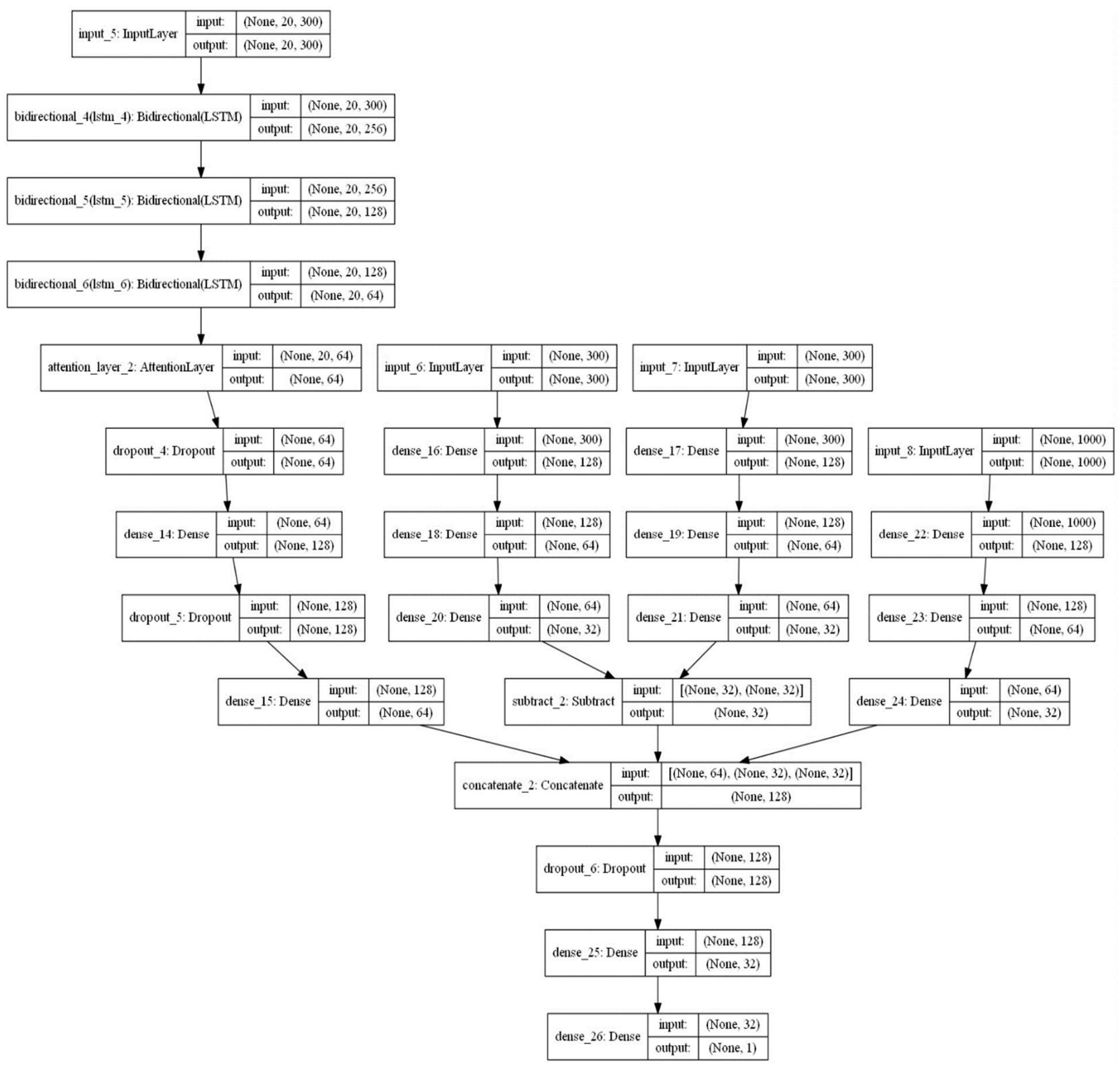
- **Siamese Neural Network on Text Embeddings**

The Siamese network is used to find and exploit the similarity between *postText* and *targetDescription*. A Siamese neural network is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. For this reason, it is widely used to verify similarity between two systems.

In this model, the Siamese Network takes as input doc2vec embeddings of the *targetDescription* and of the *postText* and runs them into parallel sequences of 3 dense layers. In the end, these two layers are compared to estimate the similarity or difference of the inputs. To compare them, it was tested both subtraction and multiplication methods.

- **Combination of the components**

At the end, the vectors that result of the three components, explained previously, are concatenated and are the input to the dropout layer, which is followed by a dense layer and finally a sigmoid layer. The operation concatenate takes as input a list of tensors, all the same shape except for the concatenation axis, and returns a single tensor: the concatenation of all inputs.

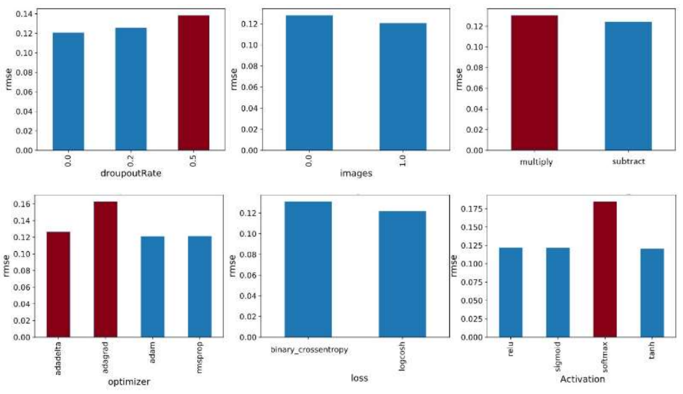

input_5: InputLayer | input: (None, 20, 300) | output: (None, 20, 300)
bidirectional_4(lstm_4): Bidirectional(LSTM) | input: (None, 20, 300) | output: (None, 20, 256)
bidirectional_5(lstm_5): Bidirectional(LSTM) | input: (None, 20, 256) | output: (None, 20, 128)
bidirectional_6(lstm_6): Bidirectional(LSTM) | input: (None, 20, 128) | output: (None, 20, 64)
attention_layer_2: AttentionLayer | input: (None, 20, 64) | output: (None, 64)
input_6: InputLayer | input: (None, 300) | output: (None, 300)
input_7: InputLayer | input: (None, 300) | output: (None, 300)
dropout_4: Dropout | input: (None, 64) | output: (None, 64)
dense_16: Dense | input: (None, 300) | output: (None, 128)
dense_17: Dense | input: (None, 300) | output: (None, 128)
input_8: InputLayer | input: (None, 1000) | output: (None, 1000)
dense_14: Dense | input: (None, 64) | output: (None, 128)
dense_18: Dense | input: (None, 128) | output: (None, 64)
dense_19: Dense | input: (None, 128) | output: (None, 64)
dense_22: Dense | input: (None, 1000) | output: (None, 128)
dropout_5: Dropout | input: (None, 128) | output: (None, 128)
dense_20: Dense | input: (None, 64) | output: (None, 32)
dense_21: Dense | input: (None, 64) | output: (None, 32)
dense_23: Dense | input: (None, 128) | output: (None, 64)
dense_15: Dense | input: (None, 128) | output: (None, 64)
subtract_2: Subtract | input: [(None, 32), (None, 32)] | output: (None, 32)
dense_24: Dense | input: (None, 64) | output: (None, 32)
concatenate_2: Concatenate | input: [(None, 64), (None, 32), (None, 32)] | output: (None, 128)
dropout_6: Dropout | input: (None, 128) | output: (None, 128)
dense_25: Dense | input: (None, 128) | output: (None, 32)
dense_26: Dense | input: (None, 32) | output: (None, 1)

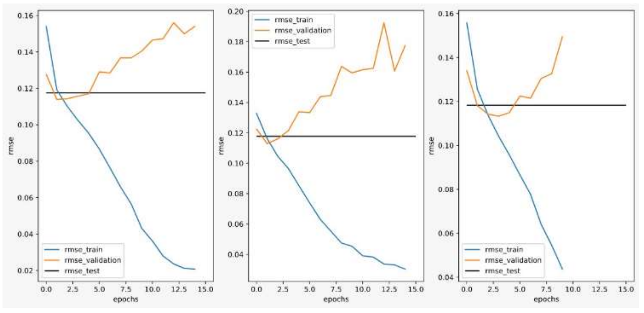
ERROR ANALYSIS AND OPTIMIZATION

Considering the hyperparameters related with the network structure, the following were optimized: dropout rate, activation function of the hidden layers and the usage or not of the images as an input. The hidden layers number was not optimized, as the network was already very complex, and it would be computationally expensive. In order to avoid overfitting, the dropout rate was optimized. When considering a large network, the performance can be better by using dropout, since the model can learn independent representations. The activation function of the hidden layers introduces nonlinearity into models, thus being important to evaluate which should be applied. It was considered that the activation function should be the same along the network[11].

Regarding the hyperparameters associated with the training algorithm, the following were optimized: the optimizer, the comparison function, the number of epochs and the loss function. The learning rate was considered in the optimizer parameter optimization (it is an optimizer argument). The learning rate controls how much to adjust the weights in the network, with respect of the loss gradient. Thus, it is one of the most important hyperparameters to optimize[12]. For the comparison functions, it was tested subtraction, a more intuitive way to compare two outputs and the multiplication, referenced in several articles concerning clickbait[2]. Epochs is the number of times that the whole training data is shown to the network while training. In order to avoid the overfitting, the epochs number must be optimized and evaluated with the train and validation results. At last, the loss function ensures the model will work as intended. This function will essentially calculate how poorly the model is performing by comparing what the model is predicting with the actual value it is supposed to output[13].

The model was run 24 times with a large range of values for each one of these hyperparameters (combination chosen randomly). Subsequently, a correlation analysis of the different possible hyperparameters with the mean square quadratic error (RMSE) was performed. The ones that presented higher values for the RMSE, were excluded from the second run. For example, the interception of the two diagrams (test / validation) tells us the ideal number of epochs to be executed (between 2 and 5).

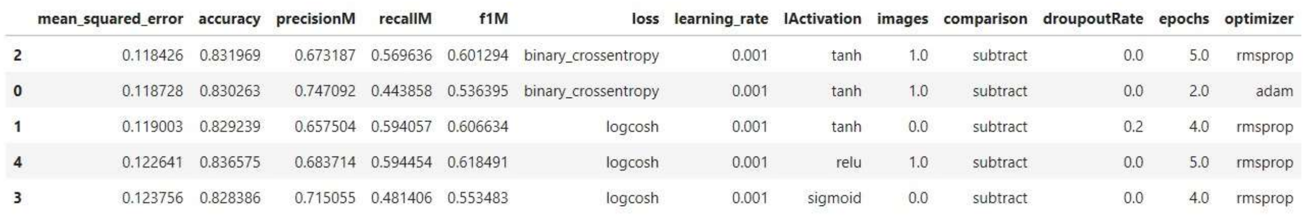




EVALUATION MEASURES

Considering the top 5 models with the lower value of RSME, the hyperparameters presented were all optimized, however the learning rate of 0.01 and the subtraction comparison function were the same for all top 5 models. The activation function – tanh, the optimizer – rmsprop, the dropout rate – 0.0 and the consideration of the images were almost the same for all the models too. The loss function in the top 2 models were the binary_crossentropy however the logcosh function showed good performance too in the other 3 models.

The best model presents the RSME of 0.11 (differences between predicted values and observed values), accuracy of 83% (fraction of predictions our model got right), precision of 67% (proportion of positive identifications was actually correct), recall of 60% (proportion of actual positives was identified correctly) and the F1 score of 60% (balance between Precision and Recall).



	mean_squared_error	accuracy	precisionM	recallM	f1M	loss	learning_rate	lActivation	images	comparison	droupoutRate	epochs	optimizer
2	0.118426	0.831969	0.673187	0.569636	0.601294	binary_crossentropy	0.001	tanh	1.0	subtract	0.0	5.0	rmsprop
0	0.118728	0.830263	0.747092	0.443858	0.536395	binary_crossentropy	0.001	tanh	1.0	subtract	0.0	2.0	adam
1	0.119003	0.829239	0.657504	0.594057	0.606634	logcosh	0.001	tanh	0.0	subtract	0.2	4.0	rmsprop
4	0.122641	0.836575	0.683714	0.594454	0.618491	logcosh	0.001	relu	1.0	subtract	0.0	5.0	rmsprop
3	0.123756	0.828386	0.715055	0.481406	0.553483	logcosh	0.001	sigmoid	0.0	subtract	0.0	4.0	rmsprop

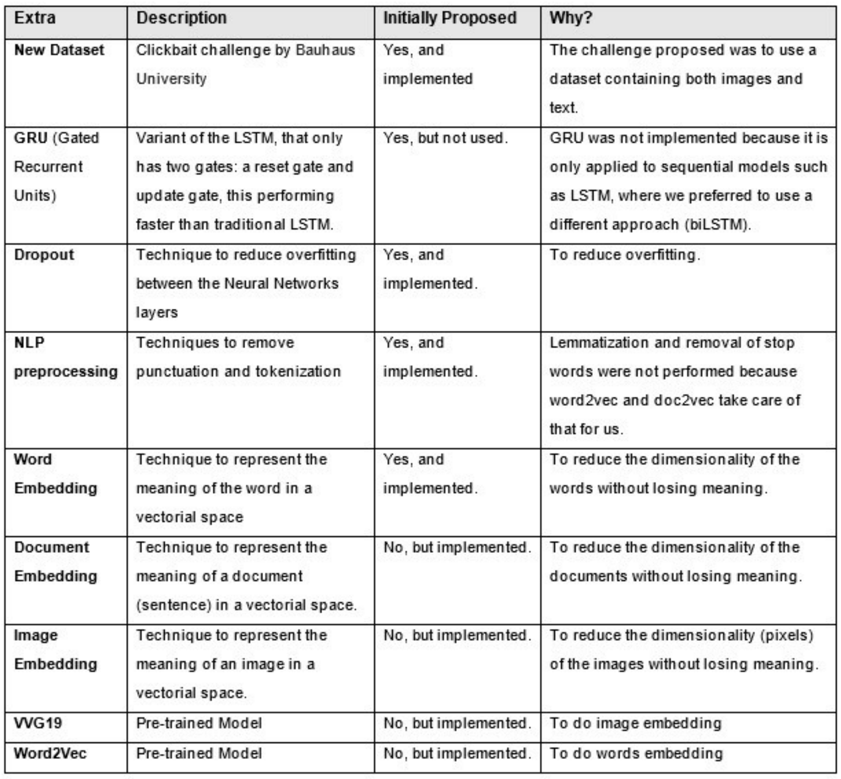
CONCLUSION

This project aimed to detect clickbait posts using a deep learning neural network architecture, using both linguistic information and image information found in tweet posts. The architecture employed not only a single words component, but also a single image component and a text comparison component. When compared with other models with the same objective but similar or very different approaches, the model proposed in this article performed as well or even better.

Nonetheless, there is a list of improvements that could have been implemented to further increase the performance of the proposed model:

- Optimization of the model structure – the presented structured was inspired in other papers but more layers could have been introduced;
- The model presented very early (few epochs) overfitting. Besides using the dropout, other overfitting mitigation techniques could have been explored and implemented;

- The image embedding was predicted from a model (VGG19) trained on images that may not be relevant.



Extra	Description	Initially Proposed	Why?
New Dataset	Clickbait challenge by Bauhaus University	Yes, and implemented	The challenge proposed was to use a dataset containing both images and text.
GRU (Gated Recurrent Units)	Variant of the LSTM, that only has two gates: a reset gate and update gate, this performing faster than traditional LSTM.	Yes, but not used.	GRU was not implemented because it is only applied to sequential models such as LSTM, where we preferred to use a different approach (biLSTM).
Dropout	Technique to reduce overfitting between the Neural Networks layers	Yes, and implemented.	To reduce overfitting.
NLP preprocessing	Techniques to remove punctuation and tokenization	Yes, and implemented.	Lemmatization and removal of stop words were not performed because word2vec and doc2vec take care of that for us.
Word Embedding	Technique to represent the meaning of the word in a vectorial space	Yes, and implemented.	To reduce the dimensionality of the words without losing meaning.
Document Embedding	Technique to represent the meaning of a document (sentence) in a vectorial space.	No, but implemented.	To reduce the dimensionality of the documents without losing meaning.
Image Embedding	Technique to represent the meaning of an image in a vectorial space.	No, but implemented.	To reduce the dimensionality (pixels) of the images without losing meaning.
VVG19	Pre-trained Model	No, but implemented.	To do image embedding
Word2Vec	Pre-trained Model	No, but implemented.	To do words embedding

REFERENCES

[1] https://www.valenciaplus.es/2019/06/05/w0002-what-is-clickbait/
[2] Vaibhav Kumar, Dhruv Khattar, Siddhartha Gairola, Yash Kumar Lal, and Vasudeva Varma. 2018. Identifying Clickbait: A Multi-Strategy Approach Using Neural Networks. In SIGIR '18: The 41st International ACM SIGIR Conference on Research Development in Information Retrieval, July 8–12, 2018, Ann Arbor, MI, USA. ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3209978.3210144
[3] https://www.clickbait-challenge.org/ "We invite you to participate in our ongoing challenge on the detection of clickbait posts in social media."
[4] Radečić, Dario 2019. Calculating String Similarity in Python (https://towardsdatascience.com/calculating-string-similarity-in-python-276e18a7d33a)
[5] https://medium.com/@mishra.thedeepak/doc2vec-simple-implementation-example-df2afbbfbad5
[6] Chansung, Park 2018. Transfer Learning in Tensorflow (VGG19 on CIFAR-10): Part 1 (https://towardsdatascience.com/transfer-learning-in-tensorflow-9e4f7eae3bb4)
[7] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural computation 9, 8 (1997), 1735–1780.
[8] https://www.quora.com/When-should-one-use-bidirectional-LSTM-as-opposed-to-normal-LSTM
[9] https://pathmind.com/wiki/attention-mechanism-memory-network
[10] https://www.quora.com/What-are-the-advantages-of-a-convolutional-neural-network-CNN-compared-to-a-simple-neural-network-from-the-theoretical-and-practical-perspective
[11] https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a
[12] https://towardsdatascience.com/understanding-learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10
[13] https://medium.com/deep-learning-demystified/loss-functions-explained-3098e8ff2b27