Article

Clickbait's spoiler classification systems

Authors Marta Mejer and Jakub Pludowski

Warsaw University of Technology, Nowowiejska 15/19, 00-665 Warsaw, Poland;

Abstract: Click baits are starting to become one of the modern world problems. People are being attacked from every side by information channels which are trying to force them to be interested in their contents. Scientists are trying to find a way which would filter that information and remove those that are redundant. In this article we are presenting our attempts to implement NLP classifiers that would decide if a clickbait's spoiler were a phrase or passage which would help future systems to get rid of click baits. We propose some classic algorithms and deep neural networks which were trained and achieved nearly 70% F1 metric value on validation test.

Keywords: NLP, Classification, SVM, Decision tree, neural networks

1. Introduction

According to the World Press Trends report from 2016, 1.3 billion people read news on websites in the form of articles. This is still over two times less than people who read paper newspapers, but the trend is still growing. For this reason, the world we live in is entering (or is already in it) an era in which we will get information mainly from internet articles. So publishers stop selling newspapers and try to make money in other ways. One is introducing a fee for access to the article or a temporary subscription giving access to all articles. This is a form that can discourage readers, so some publishers allow other companies to advertise on their websites. The more people see the article, the more the publisher will earn. For this reason, to make money, scientific, entertainment and news magazines must somehow attract the reader to their article.

Some publishers focus on content, but others prefer to cut corners and encourage readers to visit their sites through clickbaits, which are some traps set for our minds. These pitfalls usually take the form of a sentence that suggests that within the article, the reader will find answers to their questions. They can take the form of an unfinished sentence, e.g. "An American scientist discovered a cure for cancer. It is enough to take once a day…" or an adequately asked question, e.g. "Did you know that this one tablet is able to solve your problems?" or a highly hyperbolized statement, e.g. "Discovery of the year! One little thing, and it solves all the problems of this world.

The biggest problem is that you never know whether inside the article you will find valuable information or worthless axioms, or unproven lies. For this reason, even a reader experienced in avoiding this type of content is able to fall for clickbait and waste his time reading an article from which he will not learn anything.

Science and technology, of course, come to the rescue and try to solve somehow the problem of clickbaits flooding the Internet. Scientists are trying to create programs based on NLP that can correctly analyze the article and get the answer to the trap question. This answer is called a spoiler. It can take the form of one word or sentence equivalent (phrase) or several sentences describing a given

problem (passage). Before the classifier decides what a spoiler is, it or another program must determine what the type is. Knowing this allows us to increase the accuracy of the classifiers significantly. The task of this project is to create a model that, based on clickbait and the article's content, will decide whether the spoiler takes the form of a phrase or a paragraph of the article.

2. Data

Class sizes are as follows: phrase – 1367 records, passage – 1274 records, multi – 559 records. As can be seen, while the first two classes contain a similar number of records, there are more than twice as many in the multi class, which can negatively affect the final results.

3. Methodology

Classical methods

For the task, it was decided to use classical methods due to their simplicity relative to neural networks. To classify the spoiler into one of three groups, the following classifiers were used: a decision tree with both the gini index and entropy as the criterion for division, a random forest, svm with two kernel functions - linear and rbf, and a naive Bayesian classifier.

Deep models

The models described above are technically simple algorithms, which often, thanks to their simplicity, cope very well with various types of problems. They can be debugged and analyzed relatively easily. Thanks to this, you can quickly adapt their operation to the intended goals. The human mind, however, cannot understand how all those processes work, and the action imposed by it only sometimes gives the best results. In such situations, it is worth using deep models such as neural networks because, thanks to the computer's computing power, they can analyze extensive data quickly and, looking at the "big picture", give excellent results. Their parameters are enclosed in a "black box", which can be interpreted as a disadvantage, but in this case, it is their advantage.

In this task, it was decided that the fine-tuning process of previously trained models would give the best effect. This process transforms an existing model trained for a specific task into a model adapted for another purpose in a similar field. In NLP tasks, this can be done by leaving the layers of the "body" of the network unchanged, e.g. for embedding, and modifying the tail (head) of the network with layers for classification.

In this task, three different networks were decided to 'fine tune': BERT, deBERTa and roBERTa. These networks were selected from a set of transformers/hugging face library models. The original networks were used to solve text classification tasks. These networks were downloaded and fed with new data, and the network parameters were trained. The model was trained on the training data and validated on the validation dataset at each epoch.

4. Experimental setup

Preparation for training began with data preprocessing. Files in json format were downloaded and entered in data frame format thanks to the pandas library. Data labels were coded accordingly: phrase = 0; passage = 1; multiple = 2. The input data was also

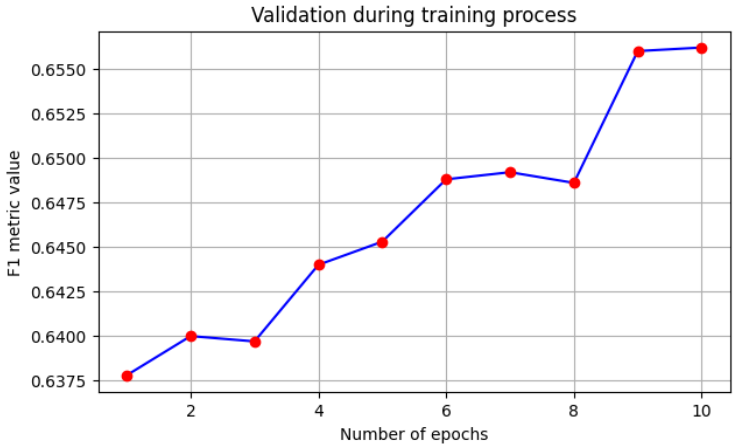
edited by converting all sentences into one string. Further data preprocessing steps varied depending on the type of model.

Classical methods

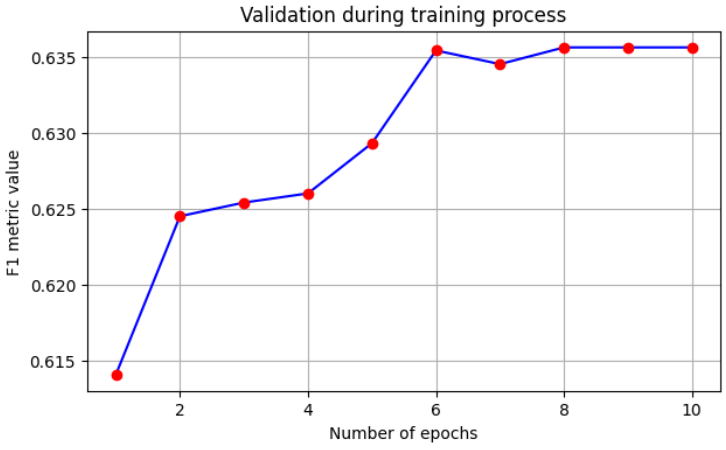
First, the data was cleaned of unnecessary characters and stop words. This was followed by lemmatization. The final feature vectors were determined using the tf-idf method. The class was predicted using two approaches - first using features from both the post and the main content of the linked document, then using only features from the post. In both cases, the steps of the experiment were identical.

Deep models

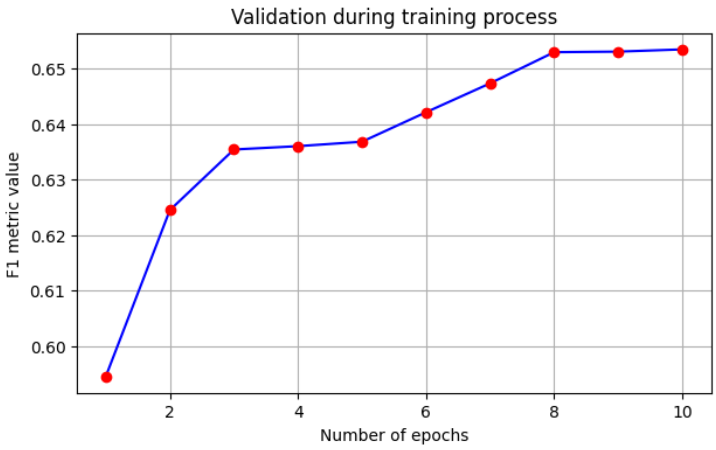
Using a tokenizer from a given model, the input data was converted to tokenized form. A Dropout layer (with parameter 0.1) was added to each of the existing models to prevent overtraining of the network. To this layer, a dense linear layer was added with an output of as many neurons as there were labels. It was set that for every epoch, the model is to be validated on a set of validation data, and the f1 measure is to be calculated for it. In addition, it was set that a checkpoint should be saved every epoch in order to be able to return to the version of the model in which the validation model achieved the best result.



BERT model training progress



DeBERTa model training process



RoBERTa model training process

5. Results

Classical methods

The following tables show the results obtained using classical methods for features extracted from the post concatenated with linked document, and features extracted only from the post. The F1 measure was used to evaluate the models.

	F1 score			
	Phrase	**Passage**	**Multi**	**F1 weighted**
Decision tree, gini	58.36	38.1	15.22	42.49
Decision tree, entropy	59.32	38.94	20.94	44.26
SVM, linear	54.86	50.07	20.99	**46.88**
SVM, rbf	58.06	48.01	12.34	45.88
Multinomial naive Bayes	56.6	45.32	0	41.94
Random forest	52.2	46.5	16.95	43.62

Results for post and linked document features

	F1 score			
	Phrase	**Passage**	**Multi**	**F1 weighted**
Decision tree, gini	60.51	18.62	16.18	35.73

Decision tree, entropy	60.86	17.69	26.37	37.32
SVM, linear	65.3	61.44	30.43	**57.51**
SVM, rbf	64.87	58.88	13.92	53.35
Multinomial Naive Bayes	63.27	56.92	4.11	50.14
Random forest	64.92	61.52	22.99	56.06

Results for post features

The results obtained are not satisfactory. In both cases, the best results were achieved by the SVM classifier with a linear kernel function, but in the first case the result does not exceed 47%, while in the second case it is admittedly 10 percentage points higher, but still not high. However, it should be noted that the results are largely affected by the presence of the third class (*multi*). In order to further investigate the impact of this class on the results, it was decided to conduct an additional experiment with a binary classification that excludes the *multi* class. The table below shows the results expressed by the weighted F1 measure.

	F1 score	
	Post + linked document	**Post**
Decision tree, gini	53.75	46.7
Decision tree, entropy	54.53	46.77
SVM, linear	57.65	69.33
SVM, rbf	**59.35**	68.05
Multinomial Naive Bayes	57.1	66.55
Random forest	55.86	**70.76**

Results for binary classification

In this case, the results obtained are better than for the three classes, managing to exceed 70% in the case of random forest and features derived only from the post.

Deep models

Results:

- BERT network, fine-tuned from 'bert-base-uncased' reached quality of 68% measured with f1 measure.

- deBERTa network, fine-tuned from 'microsoft/deberta-base-mnliuncased' reached quality of 64% measured with f1 measure.
- roBERTa network, fine-tuned from 'textattack/roberta-base-MNLI' reached quality of 66% measured with f1 measure.

6. Conclusions

Classical methods

Classical methods proved to be inadequate in this case and high classification results for the three classes could not be achieved with them. In particular, the multi class proved to be problematic, perhaps because this class was less numerous than phrase and passage, which may have affected the process of training classifiers. In addition, it can be noted that using features only from the post gives better results than including features from the linked document.

Deep models

The deep models gave good but not outstanding results. Poor results are caused by a small training set and limited training power of the computers on which the project was performed. Due to a long model training time, the number of epochs had to be limited to 10 epochs per model. In addition, these deep models have a high tendency to overtrain, and great care has been taken to prevent this from happening. Nevertheless, the results of each of these networks reached nearly 70% of the F1 measure, which is an excellent result for the resources that could be used to complete the project task.

Authors contributed equally to writing the manuscript.

Authors have read and agreed to the published version of the manuscript.

Funding**:** This research received no external funding.

Competing Interest: The authors have no competing interests.

References

1. Tadeusiewicz, R. "Sieci neuronowe w inzynierii˙ biomedycznej", *Acta Bio-Optica et Informatica Medica. Inzynieria˙ Biomedyczna*, vol. 13, no. 3, pp. 184–189, 2007, ISSN: 1234-5563.
2. Pezzotti, N.; Hollt, T.; Van Gemert, J.; Lelieveldt, B.P.; Eisemann, E.; Vilanova, A. "Deepeyes: Progressive visual analytics for designing deep neural networks", *IEEETransactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 98–108,2018. DOI: 10.1109/TVCG.2017.2744358.
3. Chollet, F. *Deep Learning with Python*. Helion, 2019.
4. Hagen, Matthias, Maik Fröbe, Artur Jurk, i Martin Potthast. „Clickbait Spoiling via Question Answering and Passage Retrieval". W Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 7025–36. Dublin, Ireland: Association for Computational Linguistics, 2022. https://doi.org/10.18653/v1/2022.acl-long.484.
5. González-Carvajal, Santiago, i Eduardo C. Garrido-Merchán. „Comparing BERT against traditional machine learning text classification". arXiv, 12 styczeń 2021. https://doi.org/10.48550/arXiv.2005.13012.
6. Sun, Chi, Xipeng Qiu, Yige Xu, i Xuanjing Huang. „How to Fine-Tune BERT for Text Classification?" W Chinese Computational Linguistics, zredagowane przez Maosong Sun, Xuanjing Huang, Heng Ji, Zhiyuan Liu, i Yang Liu, 194–206. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2019. https://doi.org/10.1007/978-3-030-32381-3_16.
7. Yu, Shanshan, Jindian Su, i Da Luo. „Improving BERT-Based Text Classification With Auxiliary Sentence and Domain Knowledge". *IEEE Access* 7 (2019): 176600–612. https://doi.org/10.1109/ACCESS.2019.2953990