Convolutional Neural Networks for Clickbait Detection

The Emperor Clickbait Detector at the Clickbait Challenge 2017

Erdan Genc, Tim Gollub, Martin Potthast
Bauhaus University Weimar
erdan.genc@uni-weimar.de

ABSTRACT

The usage of teasing headlines that lure readers into clicking on accompanying content, known as clickbait, has become a tool for many media providers to increase user engagement. This behaviour creates two potential risks. First of all, the risk of information pollution, that relevant information becomes more and more difficult to retrieve among noise such as spam, fake news, clickbait etc. Second of all, the media providers put their reliability at stake, risking to alienate their readers with misleading promises. To counter steer this movement it becomes important to design algorithmic models that can classify clickbait. We report on an experiment using a convolutional neural network (CNN) on top of the pre-trained word2vec GoogleNews word vectors to classify clickbait tweets. The model described in this paper scored second in the Clickbait Challenge 2017.

1. INTRODUCTION

Searching for a common definition for *clickbait* we stumble upon several explanations:

"Clickbait refers to social media posts that are, at the expense of being informative and objective, designed to entice its readers into clicking an accompanying link."[1]

"Clickbait is a pejorative term for web content whose main goal is to get users to click on a link to go to a certain webpage. Clickbait headlines typically aim to exploit the "curiosity gap", providing just enough information to make readers curious, but not enough to satisfy their curiosity without clicking through to the linked content."[2]

"The term clickbait refers to a form of web content that employs writing formulas and linguistic techniques in headlines to trick readers into clicking links but does not deliver on promises."[7]

According to these depictions, *clickbait* can be determined by two characteristics:

1. A teasing text that lures users into clicking on an accompanying link.
2. Content behind the link that doesn't satisfy the promises of the teasing text.

As a conclusion, a model that aims to correctly classify clickbait needs to be capable of understanding the teasing text and matching it against the content behind the link to disclose whether the promise has been met or not. However, since the second step is unequally more ambitious than the first one, most of the current work on this topic, does not consider the linked content but focuses on identifying the linguistic features that distinguish the teasing text of clickbait from ordinary content.

2. RELATED WORK

The research field of clickbait detection is still growing but already features a variety of related work.

In 2014, Potthast et al. [4] first crawled an initial training dataset and set a clickbait detection baseline using a random forest approach. In the follow up work [6] the task got shifted from a classification to a regression problem, the collection of annotated tweets was extended and made available to public to be used to train and evaluate models in the Clickbait Challenge 2017 [5]. The challenge was hosted on the TIRA platform [4].

Since the advent of word embeddings and the recent success of deep learning methods in general, it has become a popular pattern in NLP tasks to first embed terms into a dense vector space representation, which encodes the semantic relationships between the terms. The embedded words than need to be accumulated to not only represent the single terms but the whole input text. This representation usually is passed to a final softmax layer which maps the output to a range between [0,1]. When following this pattern one has to make two general decisions:
Firstly, to either train an own embedding or use a pre-trained one. In the case of working on news articles the GoogleNews embedding[3] is a standard choice for a pre-trained embedding.
Secondly, how to accumulate the word vectors to obtain a representation of the whole text sequence. Anand et al. [1] sequentially feed the word vectors to a bidirectional LSTM recurrent neural network which is a valid solution for lengthy texts of arbitrary length. Rony et al. [7] conducted a series of experiments that show excellent results by averaging the vectors and shows that training your own embedding can increase the performance.

Our approach is inspired by the work of Kim [3], who shows that a convolution layer followed by max pooling to accumulate the text representation out of the words vectors can be an effective technique when working on input data with restricted lengths such as sentiment analysis on a sentence level. Kim further discuss variations of the model, such as keeping the embedding static or dynamic during

[1] http://www.clickbait-challenge.org, 30.09.17
[2] https://en.wikipedia.org/wiki/Clickbait, 30.09.17
[3] https://code.google.com/archive/p/word2vec/, 30.09.17

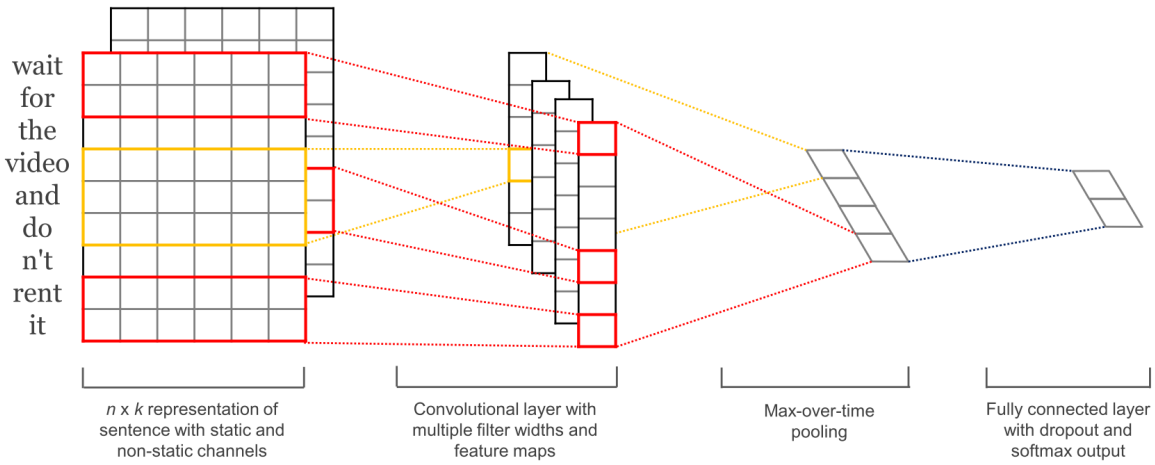

Figure 1: Two channel model architecture example. Taken from Kim 3

Batch Size	100
Initial Learning Rate	0.05
Dropout Rate	50%
l_2-Regularization Degree	0.05

Table 1: Hyperparameters

training or the usage of multiple embeddings (channels). Figure 2 illustrates the two channel model.

3. APPROACH

Our approach uses the GoogleNews embedding as a single channel model. The longest tweet in the dataset consisted of 27 terms which determined the maximum number of words to be considered. Tweets with a smaller amount of terms were padded using the "unk" token. Terms that didn't appear in the GoogleNews vocab were replaced by the "unk" token as well. Filter sizes in the range of [1,15] where used, with 64 random initialized filters for each filter size, summing up to $14 \cdot 64 = 896$ values in the last hidden layer. The weights between the hidden and output layer were initialized using the xavier initializer. Furthermore, dropout and l_2-regularization were applied to the last layer to prevent overfitting. To train the model, the AdaGrad [2] optimizer were applied.

A step by step tutorial that shows how to build a TextCNN using Tensorflow can be found here. [4]

4. EVALUATION RESULTS

After optimizing the hyperparameters on a 10% test split, I experienced a best fit for the values shown in Table 1.

5. CONCLUSION

Using convolutional neural nets for clickbait regression is a straight forward approach that yielded competitive results. Yet the model still offers a variety of options to improve such as:

1. Training an own embedding on twitter data that includes tweet specific tokens such as hashtags or smileys.
2. Switching from a single channel to a multichannel model.
3. Integrating specific tokens for unknown and padding tokens.

Furthermore, the model solely relies on the tweet text and therefore can be be integrated into any kind of service, e.g. clickbait detection browser-plugin, efficiently.

[4]http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/, 30.09.17

References

[1] A. Anand, T. Chakraborty, and N. Park. We used neural networks to detect clickbaits: You won't believe what happened next! *CoRR*, abs/1612.01340, 2016. URL http://arxiv.org/abs/1612.01340.

[2] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. Technical Report UCB/EECS-2010-24, EECS Department, University of California, Berkeley, Mar 2010. URL http://www2.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-24.html.

[3] Y. Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL http://arxiv.org/abs/1408.5882.

[4] M. Potthast, T. Gollub, F. Rangel, P. Rosso, E. Stamatatos, and B. Stein. Improving the Reproducibility of PAN's Shared Tasks: Plagiarism Detection, Author Identification, and Author Profiling. In *CLEF*, pages 268–299. Springer, 2014.

[5] M. Potthast, T. Gollub, M. Hagen, and B. Stein. The Clickbait Challenge 2017: Towards a Regression Model for Clickbait Strength. In *Proceddings of the Clickbait Challenge*, 2017.

[6] M. Potthast, T. Gollub, K. Komlossy, S. Schuster, M. Wiegmann, E. Garces, M. Hagen, and B. Stein. Crowdsourcing a Large Corpus of Clickbait on Twitter. In *(to appear)*, 2017.

[7] M. M. U. Rony, N. Hassan, and M. Yousuf. Diving deep into clickbaits: Who use them to what extents in which topics with what effects? *CoRR*, abs/1703.09400, 2017. URL http://arxiv.org/abs/1703.09400.