symmetry

MDPI

Article

Clickbait Convolutional Neural Network

Hai-Tao Zheng [1,*], Jin-Yuan Chen [1], Xin Yao [1], Arun Kumar Sangaiah [2] and Yong Jiang [1] and Cong-Zhi Zhao [3]

[1] Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, Guangdong, China; jy-chen13@mails.tsinghua.edu.cn (J.-Y.C.); yaox14@mails.tsinghua.edu.cn (X.Y.); jiangy@sz.tsinghua.edu.cn (Y.J.)

[2] School of Computing Science and Engineering, VIT University, Vellore 632014, India; sarunkumar@vit.ac.in

[3] Giiso Information Technology Co., Ltd., Shenzhen 518055, Guangdong, China; zhaocz@giiso.com

* Correspondence: zheng.haitao@sz.tsinghua.edu.cn; Tel.: +86-180-3815-3089

Received: 31 March 2018; Accepted: 27 April 2018; Published: 1 May 2018

Abstract: With the development of online advertisements, clickbait spread wider and wider. Clickbait dissatisfies users because the article content does not match their expectation. Thus, clickbait detection has attracted more and more attention recently. Traditional clickbait-detection methods rely on heavy feature engineering and fail to distinguish clickbait from normal headlines precisely because of the limited information in headlines. A convolutional neural network is useful for clickbait detection, since it utilizes pretrained Word2Vec to understand the headlines semantically, and employs different kernels to find various characteristics of the headlines. However, different types of articles tend to use different ways to draw users' attention, and a pretrained Word2Vec model cannot distinguish these different ways. To address this issue, we propose a clickbait convolutional neural network (CBCNN) to consider not only the overall characteristics but also specific characteristics from different article types. Our experimental results show that our method outperforms traditional clickbait-detection algorithms and the TextCNN model in terms of precision, recall and accuracy.

Keywords: clickbait detection; convolutional neural network; deep learning

1. Introduction

With the development of web advertisements these years, publishers want more clicks on their web pages to increase revenue from advertisements. Under this circumstance, clickbait appears on the net, and tries to attract users' attention and encourage them to click the link. Generally, clickbait utilizes erotic words, misleading contents, unverified news and exaggerated tones to achieve their goals. While increasing the click-through rates (CTR) of the articles, clickbait decreases users' satisfaction substantially because users feel a gap between what they want to know (the headline) and what they really read (the content). Besides, clickbait also causes fake news to spread on the Internet since many users forward them without further reading the contents. Therefore, it is important to develop technologies to detect clickbait.

Research aiming at detecting clickbait is still in its infancy. Current algorithms about detecting clickbait can be categorized into lexical similarity-based algorithms [1] and machine learning-based algorithms [2–5]. Lexical similarity-based algorithms detect clickbait by the semantic similarities between headlines and corresponding contents. However, information in headlines is limited, so it is hard to calculate the similarities between headlines and contents. Moreover, some headlines of clickbait are relevant to the contents literally. Consequently, the performances of such algorithms are poor. Machine learning-based algorithms solve the information-lack problem by taking additional features into consideration [6–9]. The performance of these algorithms is better than lexical algorithms. However, these algorithms suffer from two limitations. Firstly, many features used in these algorithms

are domain specific. The performance of such methods are doubtful when they are applied to normal headlines. Secondly, heavy feature-engineering tasks are required for such methods, which implies significant time resources and effort to define new features in new domains. The performance in this situation needs to be examined accordingly.

Convolutional neural networks (CNNs) [10–12] have been applied to various domains and have improved machine-learning performance significantly in the last six years. In the nature language processing (NLP) area, Word2Vec [13,14] is the most popular model to produce word embedding, which can discover semantic information of words. Based on that, CNN utilizes kernels with different sizes to find various features from the input text. These features are automatically detected and are used to train a neural network for the corresponding task. Thus, CNN understands the input text from different perspectives, and it can be applied to various nature language processing tasks. For example, CNNs are applied to part-of-speech recognition [15], named entity recognition [15], relation extraction [16], event extraction [17] and sentence similarity calculation [18]. Agrawal [19] shows the performance of using CNN to detect clickbait. However, there is no optimized CNN model for clickbait detection until now.

Unlike traditional text-analyzing problems, headlines of different article types tend to use different ways to attract users' attention, that is, the characteristics of clickbait vary between article types. For example, headlines of blogs and forums are more colloquial than those of ordinary news. A news headline using the word "amazing" is more likely to be clickbait than the headlines containing the same word but appearing in forums or blogs. Traditional CNN models utilize the pretrained Word2Vec model for all the articles, and the different characteristics in different types of articles are neglected.

To address these issues, we propose a clickbait convolutional neural network (CBCNN) model in this work. The CBCNN model utilizes a new word-embedding structure, which not only considers the overall word meaning but also the type-related word meaning. Experimental results show the optimized CNN model is the best one among the baseline methods. Our main contributions are listed below:

1. We proposed a clickbait convolutional neural network (CBCNN) model for the clickbait-detection problem. To the best of our knowledge, this is the first attempt to optimize a CNN model in clickbait detection.
2. We designed a new word-embedding structure in this work. The new word-embedding layer takes both overall and type-related word meanings into consideration.
3. We proposed a new loss function to regulate the influence of type-related word meaning.
4. We conducted extensive experiments, and the results show that the CBCNN model outperforms all the five baseline methods in terms of accuracy, precision and recall.

This paper is organized as follows: In the next section, we review the related work about clickbait detection. Section 3 elaborates the CBCNN model. Section 4 shows the evaluation of the proposed method and discusses the experimental results. The last section presents the conclusions and the future work.

2. Related Work

Clickbait appeared in recent years, and research about detecting clickbait is still in an early stage. We categorize current clickbait-detection algorithms into two types: lexical similarity algorithms and machine learning algorithms.

2.1. Lexical Similarity Algorithms

Lexical similarity algorithms try to figure out clickbait based on the similarity between headlines and content texts. Headlines are supposed to tell readers about the topic of corresponding article. Therefore, the semantic meaning of the headline and the content should be highly related. Otherwise,

the headline probably contains some attractive information to draw readers' attention. Consequently, such uncorrelated relations are the characteristic of clickbait articles.

Wang et al. [1] represented the similarity between a headline and the content by the similarity between the headline and the core sentence of the article. They also proposed a method extracting the core sentence from the article. Some other methods calculating semantic similarity between sentences can also be adapted into clickbait-detection tasks. For example, Gabrilovich and Markovitch [20] proposed a method computing semantic similarities based on Wikipedia concepts. Kiros et al. [21] proposed a method encoding sentences into vectors. The similarity between a headline and the content can be calculated based on these vectors.

However, some clickbait utilizes exaggerated words to attract users' attention, and the headline and the content are relevant semantically. Thus, these clickbait articles are misjudged by these algorithms. Besides, if the information in headlines is limited, it is hard to calculate the similarities between headlines and contents.

2.2. Machine Learning Algorithms

The main idea of machine learning algorithms is to find out the features of clickbait. For example, most clickbait headlines contain exaggerated words or interrogatives. Machine learning algorithms [22–26] learn these characteristics based on human-designed features. The main differences between machine learning algorithms is the choice of features. When features are determined, the algorithm extracts features from documents, trains the learning model and predicts new documents by the trained model.

Potthast et al. [4] proposed a model detecting clickbait on Twitter. Three different machine learning algorithms are examined in their work. The algorithms include logistic regression [27], naive Bayesian [28] and random forests [29]. Three types of features are used in their research, including text features, page features and tweetter features. There are 203 text features in their work, including the tweetter tags provided by authors, the mentioned persons in the tweet, emotional polarity, stop-words number, simple words number and punctuation number. Page features are the features of the web page this tweet links to, including the page's readability, the content length, n-grams and other content-related features. Twitter features are the features about the tweet itself, such as the author's name, the attachment type of the tweet, retransmission or not, and the frequency of the author publishing new tweets.

Chakraborty et al. [3] compared the differences between clickbait and normal articles in various views. Four types of features are used in their research: sentence structure features, n-grams, special words and POS features. Consequently, Biyani et al. [2] also proposed series types of features to detect clickbait.

All these models require a number of features to detect clickbait. Obtaining these features suffers from various problems, such as domain limitation, language limitation, being labor intensive and so on.

Chen et al. [5] suggested to use nonlexical features, image features and user-behavior features to detect clickbait.However, they have not yet found a appropriate way to use these features. After that, Zheng et al. [30] proposed a model that takes into account the user-behavior features to detect clickbait. Their model takes the outputs of traditional models as inputs, and recalculates the clickbait probability according to user behaviors. In real situations, clickbait should be detected as soon as it appears, while only when the articles are displayed to users can the user behaviors be collected. Therefore, these models cannot be deployed independently in real situations. In this work, we focus on detecting clickbait based on semantic meanings of headlines.

Agrawal [19] examined the performance of TextCNN [12] in a clickbait-detection task. He proved that CNN is useful in clickbait detection. Anand et al. [31] utilized recurrent neural networks [32,33] and character-level word embedding [34] to detect clickbait. However, their models are general

classification models, which are not designed for clickbait detection. In addition, the character-level word embedding cannot be applied to languages such as Chinese and Japanese.

As a conclusion, lexical similarity-based methods suffer from the information-limitation problem. Traditional machine learning methods need heavy feature engineering to guarantee the performance. User behavior-based methods cannot be deployed independently because only when the articles are displayed to users can the user behaviors be collected. Deep-learning methods solve these problems by word embedding and convolutional networks, but are not adapted to the new situation of clickbait detection, that is, the semantic meanings of words change according to the article types. In this work, we focus on optimizing a CNN model for clickbait-detection tasks. The proposed CBCNN model takes the article types into consideration, and the meanings of words in the model are combined by two factors: the overall meaning and the type-related meaning. Therefore, CBCNN is easy to train and can overcome all the disadvantages mentioned above. Our experimental results show the significant improvement of CBCNN.

3. Methodology

The main difference between clickbait and normal headlines is the linguistic character of the headline, such as questioning, exaggerating, wondering. Therefore, only the headlines are taken into consideration in the CBCNN model.

Figure 1 shows the main steps of CBCNN. When training CBCNN, the headlines are preprocessed, including segmentation, stop-words filtering, and part-of-speech filtering. After that, these headlines are used to train the CBCNN model. The CBCNN model consists of $1 + |T|$ Word2Vec [13,14] models and a CNN model, where T is the collection containing all article types, and $|T|$ is the size of T. One of the Word2Vec models is the overall Word2Vec model and the other $|T|$ Word2Vec models are type-related models. These Word2Vec models are used to embed words into the CNN model. On the predicting side, the input headline is preprocessed as the same as that of training. After that, the Word2Vec models are utilized to convert the headline to a weight matrix. Thus, the weight matrix contains both general characteristics and type-related characteristics. At last, the CNN model is used to predict the clickbait.

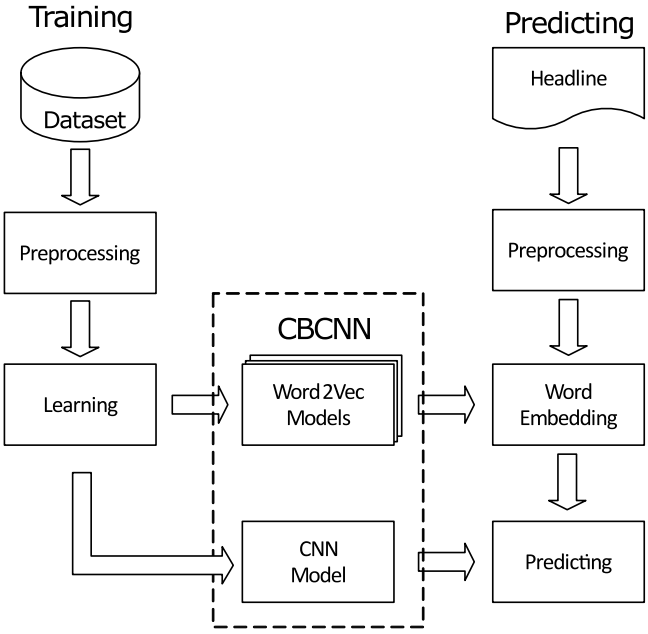


Figure 1. The main steps of CBCNN while training and predicting.

3.1. Word Embedding

As described in the first section, article type is a necessary feature in detecting clickbait. Different types of articles tend to write headlines in different ways, and detecting clickbait should vary accordingly. We also realize that clickbait articles have their own characteristics which should also be taken into consideration. To address this issue, we designed a new word-embedding structure for the CBCNN model.

The word-embedding model consists of $1 + |T|$ Word2Vec models. The first one is the overall Word2Vec model, which learns from all the headlines in the dataset. The other $|T|$ models are type-related Word2Vec models. The type-related Word2Vec models learn word vectors from the headlines of the corresponding article type. The overall Word2Vec model is used to learn general clickbait characteristics, and the type-related Word2Vec models learn type-related characteristics from the headlines. After that, a headline of article type t with n words is represented by $\mathbf{V}_{1:n}$.

$$\mathbf{V}_{1:n} = v^{'}(w_1) \oplus v^{'}(w_2) \oplus ... \oplus v^{'}(w_n), \tag{1}$$

$$v^{'}(w) = v(w) + v_t(w), \tag{2}$$

where $\mathbf{V}_{1:n}$ is a $k \times n$ matrix, k is the length of the word vector, $\oplus$ is the concatenation operator, $w_1, w_2 ... w_n$ and w are words, t is the article type, $v^{'}(w)$ is w's final word vector, $v(w)$ is w's word vector based on the whole dataset and $v_t(w)$ is w's word vector based on articles with type t.

Therefore, the word-embedding layer of the CBCNN model contains both overall characteristics and type-related characteristics, and the convolutional layer can learn more diversified features from this embedding layer.

Obviously, overall characteristics are much more important than type-related characteristics. However, they are treated equivalently in function (2). Thus in the CBCNN model, we train Word2Vec and CNN together, and add a regulation to the loss function of CBCNN. The regulation function makes sure that type-related characteristics are weighted lower than overall characteristics. The new loss function is as follows:

$$L(X) = -\frac{1}{n} \sum_{x \in X} ln[o(x)] + \frac{\alpha}{|T| * k * |V|} \sum_{t \in T} \sum_{w_j \in V} \|v_t(w_j)\|^2, \tag{3}$$

where X is the training dataset, $o(x)$ is the probability of the CBCNN model's correct prediction, k is the length of the word vector, V is the vocabulary, T is the article type, α is the weight of regularization and $|.|$ is the size of corresponding collection. In our model, each headline x is represented by the words $(w_1, w_2, ..., w_n)$ appearing in it. The first part of the loss function is that of the traditional CNN model. This loss function is used to make sure the prediction results of CBCNN are the same as the training data. Janocha and Czarnecki [35] proposed that log loss is the best choice in classification tasks. Therefore, we use log loss in CBCNN. The second part of the loss function is the regulation part, which limits the 2-norm of the type-related word vectors. $\|v_t(w_j)\|$ is the 2-norm of word w_j. In CBCNN, we have $|T|$ types, where each type contains $|V|$ words and each word consists of k dimensions, and the final regulation is the average value of all dimensions.

3.2. Clickbait Convolutional Neural Network

We have designed the clickbait convolutional neural network model as depicted in Figure 2. The inputs of the CBCNN model are the type of articles and the headlines. The word-embedding layer learns the word matrix as described in the previous subsection. In the CBCNN model, we set the length of each word vector to 50. The rows n of the word matrix are set to 12 because the lengths of most headlines are shorter than 12 words. Matrices of headlines with less than 12 words are padded with zeros, otherwise the first 12 words of the headline are used to form the matrix. Therefore, the size of the word-embedding layer is 12×50.

The next layer is the convolutional layer. The convolutional layer learns different features from the embedded word matrix with different kernels. Each kernel consists of a filter $\mathbf{W} \in \mathbb{R}^{hk}$ and a bias term $b \in \mathbb{R}$. The input of the kernel is a window of words with height h. The output of the kernel is shown in Equation (4).

$$c_i = f(\mathbf{W} \cdot \mathbf{V}_{i:i+h-1} + b), \tag{4}$$

where c_i is a feature learned from the ith row to the $i + h - 1$th row. The kernel is applied to each possible window of the headline, and learns $n + 1 - h$ features.

$$\mathbf{c} = [c_1, c_2, ..., c_{n+1-h}] \tag{5}$$

The CBCNN model contains 60 kernels in the convolutional layer. One third of the kernels have the size 2×50, one third 3×50, and one third 4×50.

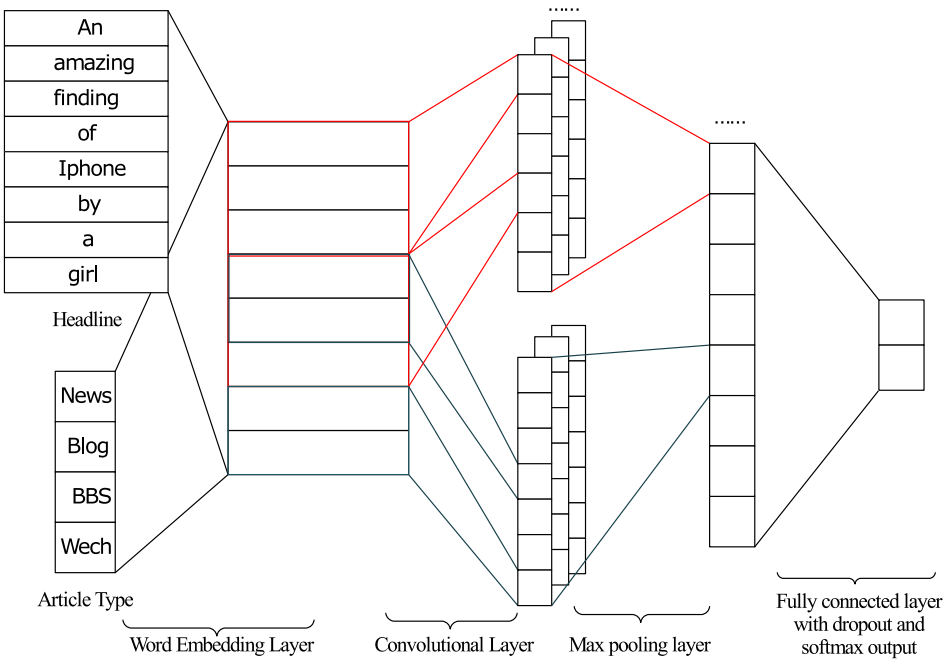


Figure 2. The structure of the clickbait convolutional neural network.

The outputs of each kernel are passed to the max-pooling layer [15]. The max-pooling layer finds the maximum value—the most important feature—for the corresponding kernel.

The last layer is a fully connected layer, which takes all the outputs of the max-pooling layer as input and utilizes a dropout [36] mechanism to prevent from over-fitting. The softmax function is utilized to limit the output of the fully connected layer into $[0, 1]$, which proved useful in TextCNN [12]. The softmax result indicates the possibility that the corresponding headline is clickbait.

We defined a new loss function in the CBCNN model in Equation (3). When training the CBCNN model, we need to calculate the new error functions of each layer in backpropagation. Since the regularization part only depends on the type-related characteristics, the error functions of the word-embedding layer, convolutional layer, pooling layer and fully connected layer in the CBCNN model are the same as that of TextCNN [12].

Bouvrie [37] proposed a detailed derivation progress of CNN models. Following their derivation, we can obtain the error $\delta_v(w)$ of word w in the word-embedding layer. After that, we need to calculate the errors of corresponding Word2Vec models. In each iteration, only the specialized type-related Word2Vec model is involved. Therefore, the word-embedding error is caused by the overall Word2Vec

model, and the specialized type-related Word2Vec model. According to Equation (2), the two Word2Vec models are weighted the same. Therefore, the overall Word2Vec model is updated as follows:

$$v^{itr+1}(w) \Leftarrow v^{itr}(w) + \frac{\beta}{2}\delta_v(w), \tag{6}$$

where $v^{itr}(w)$ is the word vector of w in the itrth step, $\delta_v(w)$ is the error of the word-embedding layer, and β is the learning rate.

The type-related Word2Vec model is relevant to the regulation part of the loss function. It is updated as follows:

$$\begin{aligned} v_t^{itr+1}(w) &\Leftarrow v_t^{itr}(w) + \frac{\beta}{2}(\delta_v(w) + \frac{\partial L(X)}{\partial w}) \\ &\Leftarrow v_t^{itr}(w) + \frac{\beta}{2}(\delta_v(w) + \frac{2\alpha}{|T| * k * |V|}\sum_{t \in T} v_t^{itr}(w)). \end{aligned} \tag{7}$$

4. Experiments and Discussions

In this section, we conduct experiments and discuss the results. Firstly, we introduce the dataset setup, the experimental environment, the baseline methods and the effectiveness matrices used in our experiments. After that, we present the experimental results of our method, including the parameter setup and the comparative results between our method and the baseline methods. At last, we discuss the experimental results and the usage of clickbait-detection methods.

4.1. Experiment Setup

As the article-type information is missing in the existing datasets, we had to construct our own dataset for conducting the experiments. Our dataset contains 14,922 headlines, where half of them are clickbait. These headlines are taken from four famous Chinese news websites (Tencent, 163, Sohu, Sina), well-known blogs, popular BBSs and Wechat official accounts. Therefore, we have four article types: news, blogs, BBSs and Wechats. In order to learn knowledge as much as possible, the training dataset should be as large as possible. Therefore, 90% of the data is used for training and the remaining 10% is used for evaluating. Since our dataset is large enough, 10% of the data is enough to evaluate the corresponding methods. The experiment is conducted in a computer with 8 GB memory and 2.5 GHz, 8-kernel CPU. Our method is implemented by TensorFlow V1.4. Our experimental data and source code are uploaded to GitHub (https://github.com/chenjinyuan87/cbcnn).

Three evaluation metrics are used in this work: precision, recall and accuracy. Their definitions are listed as follows:

$$\text{precision} = \frac{tp}{tp + fp}, \tag{8}$$

$$\text{recall} = \frac{tp}{tp + fn}, \tag{9}$$

$$\text{accuracy} = \frac{tp + tn}{N}, \tag{10}$$

where tp is the number of clickbait articles that have been predicted correctly by the algorithm, fp is the number of non-clickbait articles that have been predicted as clickbait, fn is the number of clickbait articles that have been predicted as non-clickbait, tn is the number of non-clickbait articles that have been predicted correctly, and N is the number of the predicted articles.

Five baseline methods are chosen to verify the effectiveness of the CBCNN model. The first baseline is the lexical similarity-based method, the second to the fourth methods are machine learning-based methods and the last one is TextCNN.

- CIBTSS. Wang et al. [1] proposed a method detecting clickbait based on the lexical similarity between headline and content. Their method is named as CIBTSS.

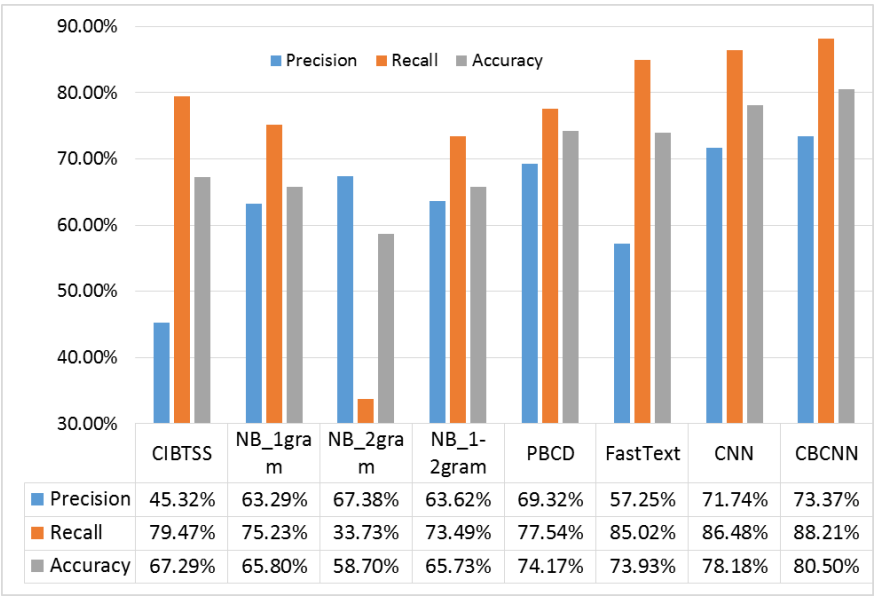
- NB. The classic text classification method naive Bayes. We utilized unigrams and bigrams as the features of Bayes. The method that uses only unigrams is marked as NB_{1gram}, the method that uses only bigrams is named as NB_{2gram}, and the method that utilizes both unigrams and bigrams is marked as $NB_{1-2gram}$.
- PBCD. Biyani et al. [2] proposed series types of features for detecting clickbait, which is the latest machine learning-based method. Their features include unigram, bigram and a series of other newly defined features, such as the number of words, exclamatory marks and question marks. We named it as PBCD in this study.
- FastText. FastText [38] is a text-classification method similar to Word2Vec. Like Word2Vec, the sequence of words is considered in FastText. The learning algorithm of FastText is similar to the continuous bag-of-word (CBOW) model [39], which is a model learning distributed representations of words based on ordered words.
- TextCNN. TextCNN is evaluated by Agrawal [19]. As shown by Agrawal, the performance of TextCNN is the best among the five baselines.

4.2. Experimental Results

The first experiment is set up to show the influence of the regulation weight α.

As shown in the Table 1, the first five columns are the performance of CBCNN with different α, and the last column is the performance of TextCNN. When $\alpha = 1$, the performance of CBCNN reaches the best. The precision, recall and accuracy are 73.37%, 88.21% and 80.5%, respectively. When α is smaller than 1, the type-related characteristics are over-weighted. Thus, the CBCNN's performance rises when $\alpha < 1$. On the other hand, fewer weighting type-related characteristics also cause the performance of CBCNN to go down. An important finding is when α is bigger than 4, the performance of CBCNN is the same as TextCNN. The reason is that when α is big enough, all the type-related characteristics are weighted as zero, and the CBCNN model degrades into TextCNN.

Figure 3 shows the comparison's results between the CBCNN model and the baseline methods. Our method CBCNN performs the best among all the methods. The precision of our method is 73.37%, recall is 88.21% and accuracy is 80.50%. All the three metrics are the best using CBCNN. Comparing with TextCNN, our model is 1.63%, 1.73% and 1.68% better in terms of precision, recall and accuracy, respectively. Comparing with FastText, our model is 16.22%, 3.09% and 6.57% better, respectively. Comparing with PBCD, our model is 4.05%, 10.67% and 6.33% better, respectively.



	CIBTSS	NB_1gram	NB_2gram	NB_1-2gram	PBCD	FastText	CNN	CBCNN
Precision	45.32%	63.29%	67.38%	63.62%	69.32%	57.25%	71.74%	73.37%
Recall	79.47%	75.23%	33.73%	73.49%	77.54%	85.02%	86.48%	88.21%
Accuracy	67.29%	65.80%	58.70%	65.73%	74.17%	73.93%	78.18%	80.50%

Figure 3. Comparing the results obtained by the CBCNN model and the five baselines.

Table 1. The influence of the regulation weight α.

	$\alpha = 0.25$	$\alpha = 0.5$	$\alpha = 1$	$\alpha = 2$	$\alpha >= 4$	**TextCNN**
Precision	69.21%	71.32%	73.37%	72.59%	71.74%	71.74%
Recall	81.34%	84.31%	88.21%	87.35%	86.48%	86.48%
Accuracy	74.16%	77.54%	80.5%	79.84%	78.18%	78.18%

4.3. Discussion

Since the headlines do not contain enough information, calculating similarity between headlines and contents is difficult. In practice, all the similarity values are small because of the word-impedance problem (only few words in the headline appear in the content). Thus, the lexical similarity-based method CIBTSS tends to treat most headlines as clickbait. It finds out 79.47% clickbait, but only 45.32% are really clickbait. Overall, the lexical similarity-based method fails to distinguish normal headlines from clickbait.

The method NB_{1gram} utilizes unigrams as features. Only word features are considered in this method. NB_{2gram} utilizes bigrams as features. It is supposed to perform better than the unigram method since bigrams contain some word-sequence information. However, in our experiment, the headlines are too short for bigrams, and most bigrams are unique in the dataset. These unique bigrams are useless for machine learning. Therefore the performance of NB_{2gram} does not meet the expectation. $NB_{1-2gram}$ tries combine the two methods, but the same problem exists as seen in NB_{2gram}. The PBCD method considers word-level features the same as many other expert-designed features. Its performance is much better than the Bayesian methods. Both Bayesian methods and PBCD fail to consider sequence information, which is important for understanding nature languages. Therefore, their performance is worse than that of TextCNN and CBCNN.

The method FastText utilizes word-sequence information to predict clickbait. The performance is similar to PBCD even if no expert-designed features are used. This result shows the importance of word-sequence information for detecting clickbait.

TextCNN not only considers sequence information but also utilizes a convolutional neural network to detect various level features. Its performance is much better than FastText and PBCD. Comparing with CBCNN, TextCNN neglects the influence of writing habit differences between various article types. Therefore, its performance is worse than that of CBCNN.

CBCNN utilizes the Word2Vec model to learn word-sequence information, combines overall word vectors and type-related word vectors together to adjust the writing habits of different article types, employs a convolutional neural network to find various features, and regulates the influence of type-related characteristics by a loss function. Its performance is the best among all the methods.

To conclude, we have the following five findings:

1. A number of clickbait articles tend to use similar words to attract users' attention. Therefore, unigram-based machine learning algorithms figure out clickbait to a certain extent.
2. Feature engineering is useful for clickbait detection, discarding the robustness problem.
3. Word-sequence information helps machine learning algorithms to understand clickbait semantically.
4. Various features are necessary for detecting clickbait, no matter if they had been extracted by feature engineering or by convolutional neural network means.
5. The type-related features are important but undesirable when overvalued.

Another important finding is that all the methods run relatively quickly. We can train CBCNN in less than one minute, and it only takes dozens of milliseconds to identify a clickbait article. Therefore, we did not compare the efficiency of those methods.

One limitation of CBCNN is that the maximum length of the headline is limited, which may cause information loss for long headlines. Besides, CBCNN does not contain a pretrained Word2Vec model, which may help it understand the overall word meaning more precisely.

In real environments, clickbait-detection methods benefit news searching and recommendation systems (such as Google news (https://news.google.com), Smartnews (https://www.smartnews.com), etc.) significantly. These systems attract more users by providing information with high quality, and consequently growing their profits. The articles of these systems are downloaded by crawlers, many of them are clickbait and it is difficult to distinguish them from normal articles. For these systems, clickbait-detection methods such as CBCNN help them detect clickbait and differentiate between these articles.

In order to implement CBCNN in the real world, we need to tag data for training CBCNN. Since our model is time insensitive, the training data is not required to be the latest. At the same time, we do not need to collect a huge amount of data for training,because clickbait-generation patterns are limited. Based on our experience, ten-thousand headlines, where half of them are clickbait, are enough. A computer installing Python 3 and TensorFlow V1.4 or higher is required also. The hardware is not important, since CBCNN runs relatively quickly. After the CBCNN model is trained, it can be used to identify clickbait articles in real-time. We suggest to retrain the CBCNN model every three months when new words or new clickbait-generation patterns appear, otherwise retraining is not necessary. To retrain the model, previous training data should be stored in the server and additional training data should be tagged. In case of considering user personalization, the training data may vary among users. Users are required to select the clickbait in a survey. They can also provide feedback to the server when the network fails to identify a clickbait. The feedback is used to update users' profiles in the future.

5. Conclusions and Future Work

In this paper, we proposed a CBCNN model for clickbait detection. Considering the word-sequence information and learning word meanings from the whole dataset, CNN models perform much better than traditional machine learning algorithms in clickbait-detection tasks. The CBCNN model utilizes article types as an additional input for the CNN model. It takes advantage of the different writing habits between different article types. The experimental results show that CBCNN outperforms the five baseline methods in terms of precision, recall and accuracy. Besides, CBCNN runs relatively quickly, the training process takes less than 60 seconds, and it only takes dozens of milliseconds to identify a clickbait article. Therefore, it can be deployed in real systems.

In the future, we will study how to interpret long headlines in CBCNN to address the information-loss problem. We will also study how to improve the CBCNN model by incorporating user-behavior analysis.

Author Contributions: Hai-Tao Zheng and Jin-Yuan Chen conceived and designed the experiments; Jin-Yuan Chen and Xin Yao performed the experiments; Hai-Tao Zheng and Jin-Yuan Chen analyzed the data; Cong-Zhi Zhao contributed reagents/materials/analysis tools; Hai-Tao Zheng and Jin-Yuan Chen wrote the paper; Arun Kumar Sangaiah and Yong Jiang proofread the paper.

Acknowledgments: This research is supported by National Natural Science Foundation of China (Grant No. 61773229), Shenzhen Science and Technology Project (Grant No. CYZZ20150408152315667), Basic Scientific Research Program of Shenzhen City (Grant No. JCYJ20160331184440545), and Cross fund of Graduate School at Shenzhen, Tsinghua University (Grant No. JC20140001).

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Wang, Z.-C.; Weng, N.; Wang, Y. Research of Title Party News Identification Technology Based on Topic Sentence Similarity. *New Technol. Lib. Inf. Serv.* **2011**, 11, 48–53. [CrossRef]

2. Biyani, P.; Tsioutsiouliklis, K.; Blackmer, J. "8 Amazing secrets for getting more clicks": Detecting clickbait in news streams using article informality. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
3. Chakraborty, A.; Paranjape, B.; Kakarla, S.; Ganguly, N. Stop clickbait: Detecting and preventing clickbait in online news media. In Proceedings of the IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, San Francisco, CA, USA, 18–21 August 2016; pp. 9–16.
4. Potthast, M.; Köpsel, S.; Stein, B.; Hagen, M. Clickbait Detection. In Proceeding of the 38 European Conference on Information Retrieval, Padua, Italy, 20–23 March 2016.
5. Chen, Y.; Conroy, N.J.; Rubin, V.L. Misleading online content: Recognizing clickbait as "False News". In Proceedings of the ACM Workshop on Multimodal Deception Detection, Seattle, WA, USA, 9–13 November 2015; pp. 15–19.
6. Abbasi, A.; Zhang, Z.; Zimbra, D.; Chen, H.; Nunamaker, J.F. Detecting fake websites: The contribution of statistical learning theory. *Mis Q.* **2010**, *34*, 435–461. [CrossRef]
7. Abbasi, A.; Chen, H. A comparison of fraud cues and classification methods for fake escrow website detection. *Inf. Technol. Manag.* **2009**, *10*, 83–101. [CrossRef]
8. Ntoulas, A.; Najork, M.; Manasse, M.; Fetterly, D. Detecting spam web pages through content analysis. In Proceedings of the World Wide Web Conference, Edinburgh, Scotland, 23–26 May 2006; pp. 83–92.
9. Lahiri, S.; Mitra, P.; Lu, X. Informality judgment at sentence level and experiments with formality score. In *CICLing 2011: Computational Linguistics and Intelligent Text Processing*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 446–457.
10. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.
11. Haykin, S.; Kosko, B. *GradientBased Learning Applied to Document Recognition*; IEEE: New York, NY, USA, 2009; pp. 306–351.
12. Kim, Y. Convolutional Neural Networks for Sentence Classification. *Eprint Arxiv* **2014**, arXiv:1408.5882.
13. Mikolov, T.; Le, Q.V.; Sutskever, I. Exploiting Similarities among Languages for Machine Translation. *Comput. Sci.* **2013**. [CrossRef]
14. Le, Q.V.; Mikolov, T. Distributed Representations of Sentences and Documents. *Comput. Sci.* **2014**, *4*, 1188–1196.
15. Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; Kuksa, P. Natural language processing (almost) from scratch. *J. Mach. Learn. Res.* **2011**, *12*, 2493–2537.
16. Zeng, D.; Liu, K.; Chen, Y.; Zhao, J. Distant Supervision for Relation Extraction via Piecewise Convolutional Neural Networks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Edinburgh, UK, 27–31 July 2015; pp. 1753–1762.
17. Chen, Y.; Xu, L.; Liu, K.; Zeng, D.; Zhao, J. Event Extraction via Dynamic Multi-Pooling Convolutional Neural Networks. In Proceedings of the Association for Computational Linguistics, Beijing, China, 26–31 July 2015.
18. He, H.; Gimpel, K.; Lin, J. Multi-Perspective Sentence Similarity Modeling with Convolutional Neural Networks. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, Lisbon, Portuga, 17–21 September 2015; pp. 1576–1586.
19. Agrawal, A. Clickbait detection using deep learning. In Proceedings of the IEEE 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 14–16 October 2016; pp. 268–272.
20. Gabrilovich, E.; Markovitch, S. Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, 6–12 January 2007; pp. 1606–1611.
21. Kiros, R.; Zhu, Y.; Salakhutdinov, R.; Zemel, R.S.; Torralba, A.; Urtasun, R.; Fidler, S. Skip-Thought Vectors. *Comput. Sci.* **2015**, arXiv:1506.06726.
22. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2012**, *12*, 2825–2830.
23. Friedman, J.H. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]

24. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
25. John, G.H.; Langley, P. Estimating Continuous Distributions in Bayesian Classifiers. In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Montréal, QC, Canada, 18–20 August 2013; pp. 338–345.
26. Molek-Kozakowska, K. Coercive Metaphors in News Headlines a Cognitive-Pragmatic Approach. *Brno Stud. Engl.* **2014**, *40*, 149–173. [CrossRef]
27. Cessie, S.L.; Houwelingen, J.C.V. Ridge Estimators in Logistic Regression. *J. R. Stat. Soc.* **1992**, *41*, 191–201. [CrossRef]
28. Lewis, D.D. *Naive (Bayes) at Forty: The Independence Assumption in Information Retrieval*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 1204–1206.
29. Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
30. Zheng, H.T.; Yao, X.; Jiang, Y.; Xia, S.T.; Xiao, X. Boost clickbait detection based on user behavior analysis. In *APWeb-WAIM 2017: Web and Big Data*; Springer: Cham, Switzerland, 2017; pp. 73–80.
31. Anand, A.; Chakraborty, T.; Park, N. We used Neural Networks to detect clickbait: You won't believe what happened Next! In *ECIR 2017: Advances in Information Retrieval*; Springer: Cham, Switzerland, 2017; pp. 541–547.
32. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
33. Cho, K.; Van Merriënboer, B.; Bahdanau, D.; Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv* **2014**, arXiv:1409.1259.
34. Santos, C.D.; Zadrozny, B. Learning character-level representations for part-of-speech tagging. In Proceedings of the 31st International Conference on Machine Learning (ICML-14), Beijing, China, 21–26 June 2014; pp. 1818–1826.
35. Janocha, K.; Czarnecki, W.M. On Loss Functions for Deep Neural Networks in Classification. *Schedae Inform.* **2016**, 25, 49–59. [CrossRef]
36. Hinton, G.E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *Comput. Sci.* **2012**, *3*, 212–223.
37. Bouvrie, J. Massachusetts Institute of Technology, Cambridge, MA , USA. Notes on Convolutional Neural Networks. Unpublished work, 2006.
38. Joulin, A.; Grave, E.; Bojanowski, P.; Mikolov, T. Bag of Tricks for Efficient Text Classification. *Comput. Sci.* **2016**, arXiv:1607.01759.
39. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient Estimation of Word Representations in Vector Space. *Comput. Sci.* **2013**, arXiv:1301.3781.