TEXT ADVERTISEMENTS ANALYSIS USING CONVOLUTIONAL NEURAL NETWORKS

AbdulwahedAlmarimi and Asmaa Salem

Department of Computer Science, Bani Waleed University, Libya

ABSTRACT

In this paper, we describe the developed model of the Convolutional Neural Networks CNN to a classification of advertisements. The developed method has been tested on both texts (Arabic and Slovak texts).The advertisements are chosen on a classified advertisements websites as short texts. We evolved a modified model of the CNN, we have implemented it and developed next modifications. We studied their influence on the performing activity of the proposed network. The result is a functional model of the network and its implementation in Java and Python. And analysis of model results using different parameters for the network and input data. The results on experiments data show that the developed model of CNN is useful in the domains of Arabic and Slovak short texts, mainly for some classification of advertisements.

KEYWORDS

Convolutional neural networks, advertisement text, back-propagation algorithm, classification, encoding of text

1. INTRODUCTION

Advertisement texts form a big set of data that we can individually choose an appropriate category. Advertisements authors can select the category but sometimes they do not recognize the suitable category. Generally, an advertisement consists of sentences, which are not long and can be categorized in a good automatic way. Its good classification can be found in [1], [2] using convolutional neural networks (CNN). Convolutional neural networks are used in many domains of data processing and realize the best results mostly in image processing as it is presented in [3]. There exist models that have searched their use for word processing and get great results [1]. For the processing of sentence and the classification of advertisements, modified CNN are used.

The classification of advertisement texts can be done using method for sentence classification in two ways: (1) To suppose that all advertisement text is one sentence formed by words, or (2) to analyze each sentence and then to estimate results of all sentences. We have used the first method, In the first step of our analysis. Arabic and Slovak languages have different free grammars. This indicates that positions of subjects and predicates in a sentence are not constant, and it looks that the Arabic grammar is more complicated than the Slovak grammar. We used grammar information in text data preparation as input to CNN and it can help to do some better classification. We used some connections between pairs of words as assistance information in input. As we explained in [4],[5] working on Arabic and English texts, the results were less complicated than which we have got using Arabic and Slovak texts.

The structure of the paper as the following: The second section contains a description of data and their preparation as input to the network. In the third section, we describe a developed model of

the convolutional neural network. The results and their comparison for both languages are given in the fourth section, we illustrated the results and their comparison for both languages and we formulate summary of results in the conclusion.

2. DATA PREPROCESSING

The advertisements are very similar in both languages, Arabic and Slovak. But we would like to do a comparison for the results of the advertisement classification using a convolutional neural network for both languages and between both languages.

2.1. Information on used Databases of Advertisements

- **Arabic Advertisements** - advertisements on the web: 3qaratonline.com portal [6]. There are 3 used categories of different advertisements: realty, furniture and electric devices. We supposed that electric devices descriptions have many common features with descriptions of furniture, while the realty category has very little similarity to other categories.
- **Slovak Advertisements** - advertisements on the Bazos.sk portal [7], from which we have selected specific categories that we will examine. We chose 3 categories, namely mobiles, computers and laptops and animals. We supposed that cell phone descriptions have many common features with descriptions of computers and notebooks, while the animal category has very little similarity to the rest of categories, and therefore advertisements in this category could be better evaluate using the neural network.
- **The numbers of used advertisements** in categories are in Table 1.

Table 1. The numbers of used advertisements from three categories in both languages.

Category	**Arabic texts**	**Slovak texts**
Realty	205	-
Furniture	194	-
Electric devices	206	-
Mobiles	-	11470
Computers and laptops	-	4968
Animals	-	3247

2.2. Creating Word Vectors

In order for the neural network to be able to process the text, we need to convert it to a vector of real numbers. In our situation, we want to process the word by word in order to determine the category based on the words used. Therefore, we need to create a uniform length vector for each word that will be used in the input data for CNN. Most often, preprocessed models are used, which are trained on a large number of texts to recognize the correlations between different words based on their use in text. They create a vector model that indicates words into a vector space, where semantically connected words are represented by points situated close to each other. The most widely used of these models is the software package word2vec [8].

The disadvantage of these solutions is that all well-trained prepared models are in English. Given the time-consumption of creating our own model for a similar learning of word vectors and the characteristics of our data, we have chosen not to use this field. We will look to the words as to images of coded letters (one letter represents one pixel) and we create vectors for words so that

each word codes the letter after the letter. For each letter a-z, we randomly generated the code v $\in$ (0, 1), taking into account that the code for each letter is unique.It means, different words have different codes.

Thereafter, we have removed the diacritics for each word in our data and created an encoding vector

$$\bar{v} = (v_1, v_2, ..., v_d, 0, ...0), v_i \in (0,1), \leq i \leq d, \quad (1)$$

where $v_1, v_2, ..., v_d$ are codes for the individual letters in the word with the length d. The vector is completed with zeros for the uniform length of vectors l. Thus, we created a table which has a word and its encoding vector. After text processing, we received a matrix of real numbers for each advertisement with a fixed number of columns equal to the selected vector length and variable number of rows depending on the number of words in the advertisement. For neural networks, we used these matrices as input. The preprocessing was very similar for both languages.

3. DEVELOPED MODEL OF CNN

We developed a similar network structure as suggested by [1], which was used for the processing of sentences in some texts, and we tried to find parameters such that the network would well evaluate our data using the knowledge found by [9].

3.1. Input Layer

As mentioned above, a network input is a matrix of real numbers, where each row is a vector representation of the word in the advertisement after its processing. The number of rows differs for each output, while the number of columns is equal to the selected vector length *l* for the word. Let $x_i \in R^l$ be a vector of the length *l* representing the i-th word in the processed advertisement. If the advertisement has n words, the resulting representation is

$$\bar{x} = \bar{x}_1 \oplus \bar{x}_2 \oplus ... \oplus \bar{x}_n \quad (2)$$

where $\oplus$ is a symbol of a concatenation operation. We will get the vector $\bar{x}$ with the length *n*l* and it should be represented as a matrix of the type *n* l*.

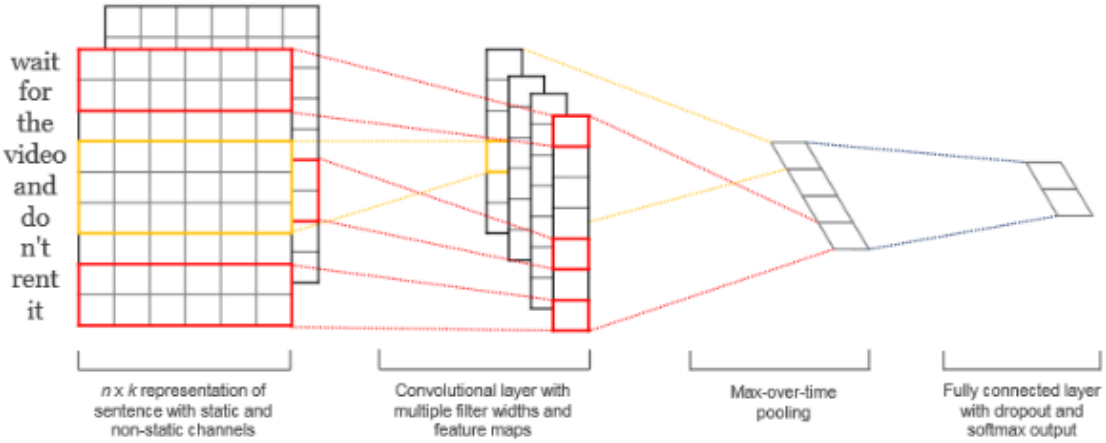

Figure 1. The basic model of the prepared convolutional network [1].

3.2. Convolutional Layer

The main idea of a convolutional layer is to work in an invariant way across a text. The convolutional layer consists of filters $\phi_j \in R^{h*l}, 1 \le j \le n_\phi$, where n_ϕ isthe number of filters, h is the number of words in the filter which is applied to obtain a new feature. For example, the feature c_j^i is calculated after applying the filter ϕ_j to the word x_i to $x_i + h - 1$

$$c_j^i = f(\phi_j . \bar{x}_{i:i+h-1} + b), \tag{3}$$

where$b \in R$ is the threshold for the filter and f is an activation function such as hyperbolic tangent or identity. In literature, the convolution operation is often performed by rotating the filter before it is applied, but it does not affect to the network processing [Godfellow2016]. If filter ϕ_j is applied to every possible part of the input with shift 1 and there is a vector of features

$$c_j = \left[c_j^1, c_j^2, ..., c_j^{n-h+1}\right]. \tag{4}$$

If there is a situation when $h > n$, we need to modify the input so that we can apply the filter to it. This is achieved by adding of zero vectors (it is called zero padding). It means, we can apply any big filter to any input (we do not have to limit the size of the smallest input). The following vector c^i represents the features of the same words for all filters

$$c^i = \left[c_1^i, c_2^i, ..., c_{n\phi}^i\right] \tag{5}$$

3.3. Pooling Layer

The output of the convolutional layer is the so-called vectors of features. For each filter, we get one such vector and each vector has a different length depending on the length of the input and the length of the filter. We remove these differences by applying subsampling to each vector using the max pooling function, giving to each filter its most distinctive feature

$$c_j^{\max} = \max\left\{c_j^i; 1 \le i \le n-h+1\right\},\ j = 1,2,...,n_\phi . \quad (6)$$

The best feature (the value of the best filter) for words $\bar{x}_i ... \bar{x}_i + h - 1$ is

$$c_{\max}^i = \max\left\{c_j^i; 1 \le j \le n_\phi\right\}. \quad (7)$$

We connect these values with all filters in one vector to make a real number vector with a length equal to the number of used filters, regardless of the length of the input that is the pooling layer output.

3.4. Output Layer

The next layer is the output layer. This layer is fully interconnected to the pooling layer, it means each neuron that come in this layer is connected to each neuron in the previous layer. We compute the value of the neuron of the output layer neuron as

$$o_i = \sum_{j=1}^{n_\phi} c_j . w_{ij} , \quad (8)$$

where*c* is the resulting vector after applying n_ϕ filters and then applying the pooling layer, and *w* is the matrix of the output layer weights. The activation function is a *softmax* function, characterized by converting the vector belonging to R^k to a vector belonging to $(0, 1)^k$, for which the sum of values is equal to 1 used by equation (7). The activation function is applied to the calculated vector o

$$y_i = \frac{\exp(o_i)}{\sum_{j=1}^{k} \exp(o_i)} . \quad (9)$$

Thus, the calculated values of the output neurons can be interpreted as the coefficient of how the network determined that the entry belongs to that category. If the output for i-neurons is equal to 1, we would be able to evaluate that the network has entered the i-th category.

However, in using the *softmax* function is a problem. If o_i is large enough (for example, when using identity as an activation function when applying filters), where $exp(o_i)$ is close to infinity, causing errors in the calculations. To avoid this, we have added the constant$D \in R \setminus 0$ to the expression

$$y_i = \frac{\exp(o_i)}{\sum_{j=1}^{k} \exp(o_i)} = y_i = \frac{\exp(o_i + D)}{\sum_{j=1}^{k} \exp(o_i + D)} . \quad (10)$$

By applying the modified *softmax* function, all inputs to the function are shifted to negative values (with the exception of the largest one, which moves to 0), causing a very low negative value element (if the difference between a given element and the maximum is large enough) has a resulting value of 0. We have thus "NaN" values, but for the output vector y, it can also contain values 0 and 1.

3.5. The Learning Algorithm

Back-propagation algorithm is the algorithm in which network errors are scrolled back across the layers so that the respective weights can be appropriately modified, and the network outputs are gradually improving. The difference between the expected and the actual neuron output is the network error. The value to be adjusted for individual weights depends on how the neuron into which the weight enters contributed to the overall Errorand on the learning ratio η.

- **Output Layer:** The output of our network is a vector of length k, where k is the number of categories we are examining. Each value of the output vector $\bar{y}$ is a number $y_i \in \{0.1\}$. However, the expected output is a vector $\bar{a}$, where one output value $a_i$for the index i associated with the input category i is equal to 1 and the other values are equal to 0. The resulting network error in such cases is best calculated as the cost cross entropy in order to avoid the learning retardation of the network.

$$c = -\sum a_i . \log y_i = -\log \bar{y} \tag{11}$$

fora_i= 1, if the output belongs to the category i, otherwise a_i=0. For each neuron of the output layer we calculated its error signal δ, which we later propagated into the previous layers. Neural error signal is the value that the given neuron has contributed to the resulting network error. In our case, we denoted the output vector of the network $\bar{y}$ and the output vector neural potential values of the output layer before applying the activation function. Expected network output is the vector $\bar{a}$. Subsequently we calculated δ for the output layer with respect to the cross entropy function

$$\delta_i^v = \frac{\delta C}{\delta o_i} = y_i - a_i \quad . \tag{12}$$

- **Pooling Layer:** Since the pooling layer only moves the greatest value from the vectors produced by the filters, its activation function is the identity whose derivation equals 1. Therefore, we have calculated for this layer simply

$$\delta_i^p = \sum_{j=1}^{n} w_{ij} \delta_j^v \quad . \tag{13}$$

- **Convolutional Layer:** The pooling layer moves the largest value from the output vector for the filter to its output. This means that the weights are equal to 0 for all neurons except the maximum for which the weight is equal to 1. The error signal therefore propagates only the neuron that contains the maximum value and is the only neuron whose error signal we need to count. For each *Fi* filter we calculated δ based on the activation function f used in the filters, h_j^i of the maximum value of the resulting vector of the particles before applying the activation function, and δ_i^p pooling the layer

$$\delta_i^F = f'(h_j^i)\delta_i^p . \qquad (14)$$

- **Modification of weights:** We adjusted the weights based on the parameter, which indicates the learning speed and set before the training of the network to a small value (usually 0.001 - 0.05). The weights are adjusted by the following equation (14), where w_{ij}^{new} is the new value of the weight, w_{ij}^{old} is it the original value, δ_i is the calculated error signal of the neuron into which the weight enters, and v_j^{prev} is the output value of the neuron of the previous layer associated with the weight.

$$w_{ij}^{new} = w_{ij}^{old} - \eta.\delta_i v_j^{prev}, \qquad (15)$$

We did not modify the weights in the pooling layer because we still want to return the maximal value of the filter element, but we just adjusted the weights for the filter based on those values of the input layer that contributed to the maximum value (for all others, the value of the error signal is 0). If $c_l = \max\{c\}$ and therefore l is the index of the maximum value of the vector, we adjusted the weights for the filter as follows:

$$w_{ij}^{new} = w_{ij}^{old} - \eta.\delta_i^F x_{i+l-1,j} . \qquad (16)$$

$\bar{x}$ is the input matrix after eventual application of zero paddings.

4. RESULTS OF APPLICATION

For word vectors we chose the length 50. The length was chosen with respect to the longest word in [10], and that the grammatical errors in the scanned text. Some functions are used as the following:

1. **accuracy**: Calculates how often predictions match labels;
2. **false negatives**: Computes the total number of false negatives;
3. **false positives**: The number of sentences when actual class of sentences is yes but predicted class is no;
4. **precision**: Computes the precision of the predictions with respect to the labels;

Table 2. Statistics of results

Language	**set: #segments**	**accuracy**	**false negative**	**false positive**	**precision**
Arabic	TR: 484 TE: 121	0.587 0.735	34.00 10.10	13.00 17.00	0.5810 0.8760
Slovak	TR: 484 TE: 121	0.460 0.589	0.420 124.0	36.00 106.00	0.7330 0.7850

The results in the accuracy are better for Arabic texts than for Slovak and in the precision results for both languages are comparable. All results are positive for us and we know that it is a good way in the research.

5. CONCLUSION

In the paper we deal with one type of neural networks for a classification of advertisements. The model was tested on a short text of advertisements were written in Arabic and Slovak language. We have shown a way to get data from the advertisements websites. The modified model is qualified for using in the area for both languages independent on the obtained results. But it needs to analyze more texts. We designed the convolutional neural network model, it was applied in Java and Python programming languages and was examined using different activation functions, learning rate coefficients, filter count and its size. We have shown the results of the network using the proposed model. The results on testing data demonstrate that the neural network model realize good classification in different data sets (Arabic and Slovak advertisements), after training on the correct example of inputs.According to the given results, Arabic advertisements are classified in the suitable category with 87% while 78% for Slovak advertisements of cases within our dataset. Part of the output is used along with a database model that uses the order file to train the selected network model, and shows the percentage of success on the training and test sets during the learning of the network. As the future work we will continue using different encodings and put more information to codes, Our plan to do many statistic results to evaluations of bigger text sets and to analyze them in different languages and compare our previous results to the results which will get it.

ACKNOWLEDGEMENTS

Thanks to Prof. Gabriela Andrejková, CSc. and Mgr. ŠimonHorváth for their help.

REFERENCES

[1] Kim, Y. (2014) Convolutional Neural Networks for Sentence Classification. arXiv:1408.5882. Chicago, web-page: https://arxiv.org/abs/1408.5882

[2] Johnson, R. Zhang, T. (2014) “Effective Use of Word Order for Text Categorization with Convolutional Neural Networks”, arXiv:1412.1058. web-page: https://arxiv.org/abs/1412.1058, Denver, Colorado.

[3] LeCun, Y. (1990) Handwritten Digit Recognition with a Back-Propagation Network. Morgan Kaufmann, San Francisco.

[4] Asmaa, Salem, Abdulwahed, Almarimi& Gabriela Andrejkova “Text Dissimilarities Predictions using Convolutional Neural Networks and Clustering”, 1st World Symposium on Digital Intelligence for Systems and Machines August 23-25,2018, International Conference in Technical University of Košice, Slovakia, ISBN: 978-1- 5386-5101-8, pp 343-348.

[5] Asmaa, Salem & Gabriela, Andrejková “Analysis of text advertisements using convolutional neural networks”, In proceedings of Cognition and artificial life, Brno, 30. 5. - 1. 6. 2018, ISBN 978-80-88123-24- 8, pp 59-61.

[6] 3qaratonline.com - web-page [online]: https://www.3qaratonline.com/

[7] Bazos.sk - web-page [online]: https://www.bazos.sk/

[8] Mikolov, T (2013) “Distributed Representations of Words and Phrases and their Compositionality”, proceedings of the 26th International Conference on Neural Information Processing Systems.

[9] Zhang, Y. Wallace, B. (2015) “ A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence classification”, arXiv:1510.03820. web-page: http://arxiv.org/abs/1510.03820

[10] Tvaroslovník – web page [online], http:// tvaroslovnik.ics.upjs.sk/

AUTHORS

Master Degree from P. J. Šafárik University in Košice, Institute of Computer Science, Faculty of Science 2012, PhD. from P. J. Šafárik University in Košice, Institute of Computer Science, Faculty of Science2016, Faculty member at Bani Waleed University From 2017, Libya.

Master Degree from P. J. Šafárik University in Košice, Institute of Computer Science, Faculty of Science 2014, PhD. from P. J. Šafárik University in Košice, Institute of Computer Science, Faculty of Science 2019, Faculty member at Bani Waleed University From 2020, Libya..