# TEXTS CLASSIFICATION WITH THE USAGE OF NEURAL NETWORK BASED ON THE WORD2VEC'S WORDS REPRESENTATION

D. V. Iatsenko

Department of Information and Measuring Technologies, Southern Federal University, Milchakova 10, Rostov-on-Don 344090, Russia

***ABSTRACT***

*Assigning the submitted text to one of the predetermined categories is required when dealing with application-oriented texts. There are many different approaches to solving this problem, including using neural network algorithms. This article explores using neural networks to sort news articles based on their category. Two word vectorization algorithms are being used — The Bag of Words (BOW) and the word2vec distributive semantic model. For this work the BOW model was applied to the FNN, whereas the word2vec model was applied to CNN. We have measured the accuracy of the classification when applying these methods for ad texts datasets. The experimental results have shown that both of the models show us quite the comparable accuracy. However, the word2vec encoding used for CNN showed more relevant results, regarding to the texts semantics. Moreover, the trained CNN, based on the word2vec architecture, has produced a compact feature map on its last convolutional layer, which can then be used in the future text representation. I.e. Using CNN as a text encoder and for learning transfer.*

***KEYWORDS***

*Deep Learning, Text classification, Word2Vec, BOW, CNN*

## 1. INTRODUCTION

Text classification is one of the most common text processing tasks. This task is quite close to the one of relevance of the text based on its query, which is now mostly solved by modern search engines like Yandex, Google, etc. At different times, this type of tasks was solved in various methods. One of the first methods was to use the statistical measure of the word's importance TF-IDF in order to define the text's relevance to the given word. Many other different methods and principles was used afterwards, mainly based on the statistical characteristics of the words and their parts, semantical analysis of the given text, vector representation of those words and on the measurement of the proximity of those words, based on different neural networks, etc [1].

Let's take a look at solving this type of text classification issue, based on different text representation models and different types of neural networks. Let's take commonly known fetch 20newsgroups data set from the sklearn.datasets. pack for analysis.

## 2. DATASET DESCRIPTION

The fetch 20 newsgroups Dataset contains texts of different ads taken from the e-bulletin boards. There are a total of 11314 training texts and 7532 test texts in the dataset. The presented ads are divided into 20 classes:

DOI: 10.5121/ijsc.2023.14201 

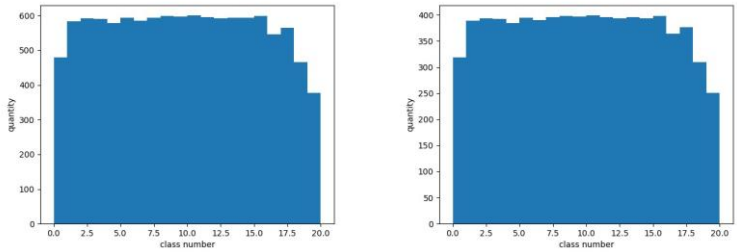
0. alt.atheism,
1. comp.graphics,
2. comp.os.ms-windows.misc,
3. comp.sys.ibm.pc.hardware,
4. comp.sys.mac.hardware,
5. comp.windows.x,
6. misc.forsale,
7. rec.autos,
8. rec.motorcycles,
9. rec.sport.baseball,
10. rec.sport.hockey,
11. sci.crypt,
12. sci.electronics,
13. sci.med,
14. sci.space,
15. soc.religion.christian,
16. talk.politics.guns,
17. talk.politics.mideast,
18. talk.politics.misc,
19. talk.religion.misc

The text contains 200 to 300 words on average. This is an example of the dataset's text, mapped to the rec.autos class:

*"From: lerxst@wam.umd.edu (where's my thing) Subject: WHAT car is this!?*
*Nntp-Posting-Host: rac3.wam.umd.edu*
*Organization: University of Maryland, College Park*
*Lines: 15*

*I was wondering if anyone out there could enlighten me on this car I sawgtfvt the other day. It was a 2-door sports car, looked to be from the late 60s/ early 70s. It was called a Bricklin. The doors were really small. In addition, the front bumper was separate from the rest of the body. This is all I know. If anyone can tell me a model name, engine specs, years of production, where this car is made, history, or whatever info you have on this funky looking car, please e-mail.*

*Thanks,*
*- IL*
*—- brought to you by your neighborhood Lerxst —-"*[2] The text distribution goes quite evenly (l.a Figure 1)




(a) Text classes distribution on (b) Text classes distribution on valtraining dataset. idating dataset.
Figure 1: Text classes distribution.

## 3. TEXT REPRESENTATION MODELS

In order to process the text using neural networks the text must be presented as a vector of real numbers, set in a format, coinciding with the network input. The input data format can vary both in the vector value's dimension and in the coding principle. I.e. Encoding could either be one-hot encoding (positional encoding) or vector representation.

### 3.1. The "Bag of Words" Model

The "Bag of Words" model (BoW) is essentially a simple text vectorization based on a particular word's characteristic (TF-IDF). In order to make this method work, the marked text corpus goes for the following transformations:

- The text corpus gets tokenized.
- A corpus dictionary is being built with a table of frequency for words occurrence, usually sorted in descending order. Additionally, the diagram of the frequency for words occurrence and index in the dictionary corresponds to the Zpif diagram (Figure 2).

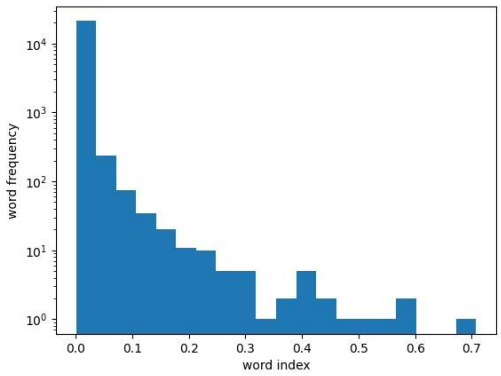



Figure 2: Distribution of the words relative frequencies.

- Words with too high or too low frequency are discarded, because on the former casethey will hardly ever occur in the future (Which means that they will have less of an impact for the overall analysis), and on the latter case they will occur almost everywhere (That way it will not help to classify the text into any category at all). The formula for DF calculation (1) is widely known and, in general, represents the ratio of the number of

texts containing the desired words to the total amount of words in the given corpus. IDF is more like an inverted version of the DF, the value of which is sometimes appears as logarithmic (2) for overall standardization [3]:

$$DF = \frac{\{t \in D\}}{D} \tag{1}$$

$$IDF = log\frac{D}{\{t \in D\}} \tag{2}$$

- Thus, using this method, each text is represented as a vector of real numbers witha size equal to the size of the corpus dictionary, each unit of which contains a number equal to the number of i-th word occurrences in the text by its frequency characteristic $DF_i$(3).

$$val_i = n_i \cdot DF_i \tag{3}$$

- A set of pairs of such vectors and text class numbers are the training data for theneural network that will carry out the classification.

It should be noted, that the final dataset will be represented by a large K*W discharged matrix, where K is the number of overall texts, and W is the dictionary's size.

## 3.2. Word2Vec Model

Let's look at the alternative representation of the vectorization text, using the representation of a word as a vector in space in the word2vec's semantically-distributive language model [4]. Using this model, a word does not encode by the one-hot encoding principle, but goes as a numeric value of a vector in the N-dimensional space of the model's semantic representation . Such models are presented in the form of embedding and shown in public projects by many research groups. The following embedding were used in the presented study:

- word2vec-google-news-300
- glove-wiki-gigaword-50
- glove-wiki-gigaword-100
- glove-wiki-gigaword-200
- glove-wiki-gigaword-300
- glove-twitter-25
- glove-twitter-50
- glove-twitter-100
- glove-twitter-200

Thus, the word is represented by a vector of values with a dimension of 25 to 300 numbers, depending on the selected model. In our case, the text is a set of such vectors.

Using the classic approach with values instead of one-hot encoding is highly inefficient, but in word2vec method it is quite plausible, mostly because the close values in the vector word representation correspond to the semantically close entities in the given model's semantical space. I.e. Any small changes in the input values will not change the behavior of the model significantly [5].

The preparation of the dataset shares the similarity with the stages for the "Bag of Words" model. The corpus of texts is tokenized, each word is replaced by a vector given from a word2vec model, which results in each text getting represented by a two-dimensional N*M matrix, where N is the number of words, and M is the dimension of the selected embedding. The whole dataset

will be presented as a three-dimensional K*N*M matrix, where K is the number of texts in the corpus [6].
This version creates a small issue, based on the fact that the length of the texts is different, which adds complexity within the description of the dimension of the neural network. There are various methods in order to solve this kind of problem:

- Sometimes the N leads to a single predetermined value by adding zero vectors tothe end of the text in cases if the text is shorter, otherwise, if the text is longer, then those vectors get erased.
- You can also use the global pulling operation, as an alternative.
- Pre-trained RNN can also be used for forming fixed vector featuremaps when itgoes through the text.

In our case, we will be using the first method, since the initial data set is assumed to be a collection of small close-sized texts.

# 4. Implementation of the Neural Networks for Working with Various Models of Text Representation

## 4.1. Fully Connected Neural Network and The ”Bag Of Words”

The Fully-connected Neural Network (FNN) may well be one of the output layers by the number of classes. In this case, for two-class training the loss function should be BCELoss (Binary cross Entropy) (4), or Softmax (5) for a multi-class training.

$$L = -p\log_2 p - (1-p)\log_2(1-p) \tag{4}$$

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i log_2(\tilde{y_i}) + (1-y_i) log_2(1-\tilde{y_i}) \tag{5}$$

The network is trained by gradually minimizing the value of the loss function. During this process, adjustment of the model’s parameters take place, i.e. the weights of the network. The current algorithm is used in order to minimize the loss function (6):

$$w_i = w_{i-}1 - \alpha \cdot \nabla(L) \tag{6}$$

It is possible to classify the texts after the network training. In order to do this, the given text gets tokenized, turned into a vector, according to the previous algorithm (while using the same dictionary), after which the resulting network vector is presented [7]. During this, the predicted class output will have the largest value in the range from (0:1), which corresponds to the probability that this test actually belongs to the predicted class (Figure 3).

The presented network has the following kind of architecture: One FNN with the input amount based on the unique works (UNIQUE_WORDS_N) equals 21628 in the following example, and the amount of neurons based on the amount of text classes (UNIQUE_LABELS_N) equals 20 in the given example. Shown below is the network structure made in Keras, taking the tabular model.summary() representation (Table 1).

The source code of the notebook with the network implementation is shown here https://github.com/dyacenko/article_text_processing/blob/main/newo

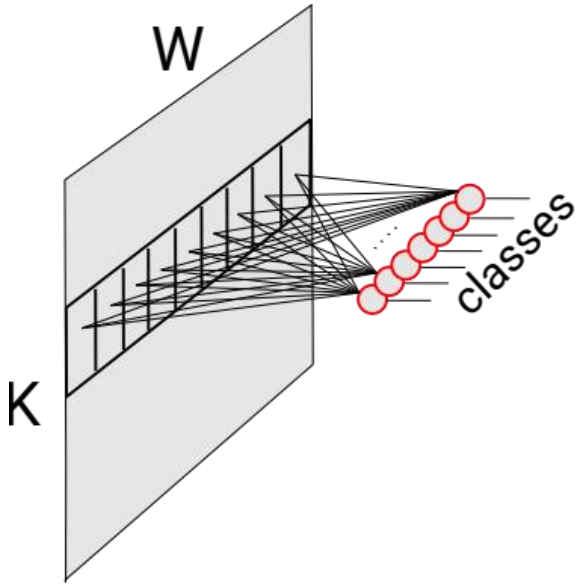



Figure 3: Processing of vectorized text by a Fully-connected Neural Network.

Table 1: Fully-connected NN Architecture.

| Layer (type) | Output Shape | Param# |
|---|---|---|
| Linear-1 | [-1, 20] | 432,580 |
| Total params: 432,580<br>Trainable params: 432,580<br>Non-trainable params:0 | | |
| Input size (MB): 0.32<br>Forward/backward pass size (MB): 0.99<br>Params size (MB): 1.65<br>Estimated Total Size (MB): 2.64 | | |

rk1_1_tfidf.ipynb.

The way network is trained is by presenting it with samples of texts encoded according to the TF-IDF principle. During the process, cross_entropy from the torch.nn.functional package is being used for loss evaluation. Further on, the resulting loss value is minimized by means of applying the gradient descent method with addition of Adam optimizer from the torch.optim. package. The network is represented by a large discharged vector in the given example. Because of this, special data structures were used for more efficient storage of the discharged data for this example.

Shown below are graphs of the loss value functions at each epoch during network training and testing (Figure 4). As show from the graph the network is learning quite steadily and stable, and reaches a plateau at 15th epoch. Relearning process does not occur. As a result, the network shows classification accuracy of 0.766 during the test. Shown below is the confusion matrix of the trained network (Figure 5) when the texts taken from the test set are processed.

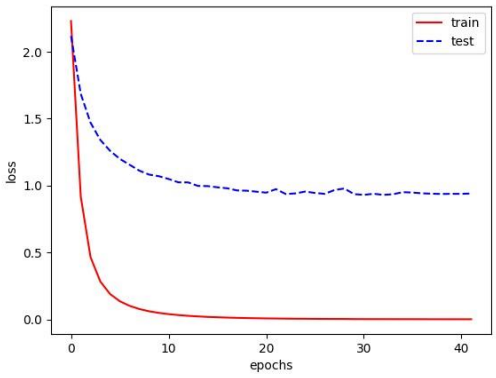



Figure 4: Training of FC network metrics.

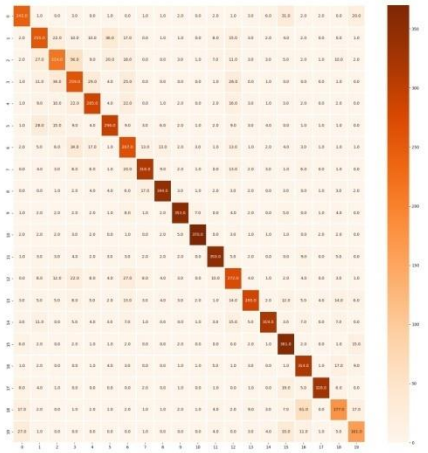


Figure 5: confusion matrix of a FNN.

## 4.2. Convolutional Neural Network and Word2vec Encoding

Now let's use a deep Convolutional neural network to classify the vectorized texts, based on the word2vec distributive semantic model. The convolution process has been in the field of signal processing, as well as image processing and filtering for quite a long period of time. In general, a convolution operation (7) is a sort of operation that works with a pair of matrices $A = (n_x, n_y)$, and $B = (m_x, m_y)$ which results in a matrix $C = A * B$ of size $(n_x m_y + 1)$ . Each element of the given result is same $- m_x + 1) \times (n_y -$ $B$ calculated as a scalar product of thematrix and $A$ submatrix of the size as the $B$. In addition, the submatrix is determined by the position of the element given on the result. I.e:

$$C_{i,j} = \sum_{u=0}^{m_x-1} \sum_{v=0}^{m_y-1} A_{i+u,j+v} B_{u,v} \tag{7}$$

The process of a convolutional neural network described in [8] by Jan Lekun uses quite a similar mathematical apparatus, but in application to neural networks. Which means that the $A$ matrix –

neural network inputs, a $B$ matrix – the convolution core, which essentially acts as a special case of an incompletely connected neural network, and the $C$ matrix is the network output. In this case, the $B$ matrix elements – are the weights of the neural connection, which are adjusted in the process of teaching the network along with the value of the offset that is embedded in the convolution core.

A CNN can be used in a wide variety of case. For the current case the conv1d onedimensional convolution was used. In this case, the convolution core moves in only one direction – through the text, performing the convolution operation on the three adjacent words (Figure 6). In addition, the number of the output channels is take by the number of the input channels. Subsamping operation and activation function are triggering after the convolution layer. Deep convolutional neural networks are often used in a way that makes it so the output of one convolutional layer acts as an input for another one, as it is in modern architecture [9].

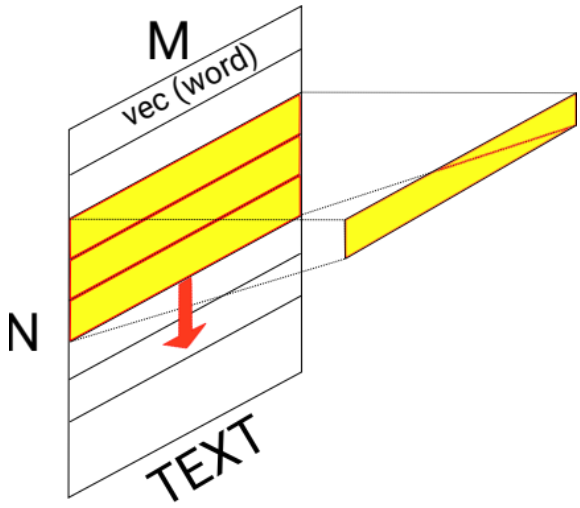



Figure 6: The processing of vectorized text by the means of CNN

As a result of a series of experiments, it was discovered that the optimal network depth for the current task is 9 convolutional layers, with a feature map formed as a single vector with size equal to the size of a word vector given from the output. This example shows 300 values. According to this feature map, the FNN classifies the texts using the number of neurons equal to the number of classes.

That way this network's architecture consists of - 9 convolutional layers, 300 output and input channel with the size of each equals the size of the convolutional core, which is 3, and one FNN with 300 inputs and 20 outputs (Table 2).

It is important to note that with the network this deep, using the gradient descent results in the gradient attenuation [10], i.e. The gradient practically does not reach the first layers of the network; therefore, they are not being trained. In order to minimize the impact of this effect, the ReLU activation function [11] and the residual block mechanism, which were first used by the developers of the deep ResNet network [12], were used in this work.

Table 2: Deep Convolutional Neural Network architecture.

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Conv1d-1 | [-1, 300, 280] | 270,300 |
| RReLU-2 | [-1, 300, 280] | 0 |
| MaxPool1d-3 | [-1, 300, 140] | 0 |
| Conv1d-4 | [-1, 300, 140] | 270,300 |
| RReLU-5 | [-1, 300, 140] | 0 |
| Conv1d-6 | [-1, 300, 140] | 270,300 |
| RReLU-7 | [-1, 300, 140] | 0 |
| Dropout-8 | [-1, 300, 140] | 0 |
| Conv1d-9 | [-1, 300, 140] | 270,300 |
| RReLU-10 | [-1, 300, 140] | 0 |
| MaxPool1d-11 | [-1, 300, 70] | 0 |
| Conv1d-12 | [-1, 300, 70] | 270,300 |
| RReLU-13 | [-1, 300, 70] | 0 |
| Conv1d-14 | [-1, 300, 70] | 270,300 |
| RReLU-15 | [-1, 300, 70] | 0 |
| Dropout-16 | [-1, 300, 70] | 0 |
| Conv1d-17 | [-1, 300, 70] | 270,300 |
| RReLU-18 | [-1, 300, 70] | 0 |
| MaxPool1d-19 | [-1, 300, 35] | 0 |
| Conv1d-20 | [-1, 300, 35] | 270,300 |
| RReLU-21 | [-1, 300, 35] | 0 |
| Conv1d-22 | [-1, 300, 35] | 270,300 |
| RReLU-23 | [-1, 300, 35] | 0 |
| Dropout-24 | [-1, 300, 35] | 0 |
| AdaptiveMaxPool1d-25 | [-1, 300, 1] | 0 |
| Flatten-26 | [-1, 300] | 0 |
| Linear-27 | [-1, 20] | 6,020 |
| Total params: 2,438,720<br>Trainable params: 2,438,720<br>Non-trainable params: 0 | | |
| Input size (MB): 0.32<br>Forward/backward pass size (MB): 5.61<br>Params size (MB): 9.30<br>Estimated Total Size (MB): 15.24 | | |

The source code of the notebook with the network implementation is shown here https://github.com/dyacenko/article_text_processing/blob/main/netwo rk2_vec.ipynb

The network is learning by presenting a network of text samples encoded, according to the principle of encoding words with vectors from the word2vec distributive model. During the process, cross_entropy from the torch.nn.functional package is being used for loss evaluation. Further on, the resulting loss value is minimized by means of applying the gradient descent method with addition of Adam optimizer from the torch.optim. package. Shown below are graphs of the loss value functions at each epoch during network training and testing (Figure 7).

As show from the graph the network is learning quite steadily and stable, and reaches a

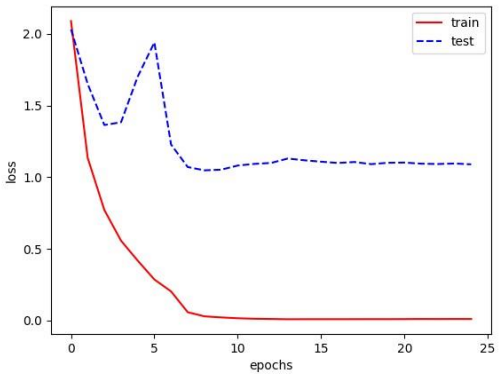



Figure 7: Training of convolutional network metrics.

plateau at 15th epoch. Relearning process does not occur. As a result, the network shows classification accuracy of 0.763 during the test. Shown below is the confusion matrix of the trained network (Figure 8) when the texts taken from the test set are processed.

## 4.3. Adaboost and TF-IDF Encoding

For comparison, let's use the already made adaboost classifying model from the sklearn library. We will use a Decision Tree as a basic algorithm We will also use the same fetch 20newsgroups dataset with TF-IDF encoding. The source code of a notebook with a network implementation is shown here: https://github.com/d-yacenko/artic le_text_processing/blob/main/network1_1_ada.ipynbAs a result, the model demonstrates an accuracy of 0.6523 on the test dataset. Shown below is the confusion matrix of the trained network (Figure9) when the texts taken from the test set are processed.

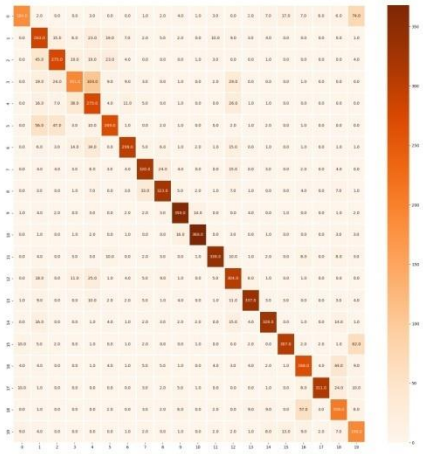


Figure 8: confusion matrix of a CNN.

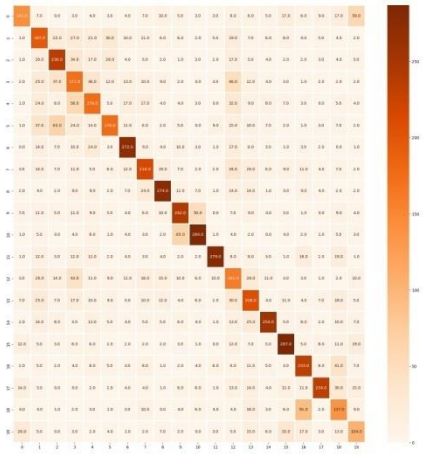


Figure 9: confusion matrix of a Adaboost.

## 5. ANALYSIS RESULT

All of the aforementioned experiments were conducted on a Virtual PC:

- CPU Intel(R) Xeon(R) CPU @ 2.00GHz
- RAM 12Gb
- GPU Tesla P100 with 16GB of RAM
- OS Ubuntu 18.04.5 LTS (Bionic Beaver)

The final characteristics of neural networks of different architectures considered in this article are as follows (Table 3):

Table 3: Pivot charcteristics of different networks.

| | Number of weights | Network training time (s) | Time of pre-diction (ms) | Max accuracy |
|---|---|---|---|---|
| Fully Connected | 432,580 | 96 | 0.4 | 0.766 |
| Convolution | 2,438,720 | 679 | 5.4 | 0.763 |
| Adaboost | – | 1656 | 0.16 | 0.663 |

## 6. DISCUSSIONS AND CONCLUSIONS

The comparison of different encoding text for processing in neural networks from different kinds of architectures has been discussed by many authors[13], [14]. The study also shows the practical results and compares them with different architectures. As can be seen from the presented data, different approaches and methods show similar levels of accuracy. The Convolutional Neural Network has a much more complex architecture and has even more configurable parameters. The time to train for both of the networks is comparable. There are noticeable differences in the distribution of errors by category, which make it possible to interpret the cause in the use of semantic proximity in the work of the Convolutional Neural Network with the word2vec encoding. This is quite noticeable, for example, in the fact that the CNN is much more likely to mistakenly classify texts from the soc.religion.christian (class 15) category as talk.religion.misc (class 19) – 22 vs 14. At the same time, a network which is based on a distributive semantic coding is much less likely to mistakenly classify talk.politics.guns (class 16) with talk.religion. misc (class19) compared to TF-IDF encoding – 5 vs 18. Moreover, it is obvious that a network with the word2vec encoding can classify texts with words not represented in the training samples, if these words are represented in the embedding used to encode the texts.

In addition, a trained CNN can be used to convert texts into a feature map, which can then be used in transfer learning [15] or as an encoder for texts encoding. All in al, despite the similar results show by both of the methods, the CNN with word2vec encoding has an additional way of use.

## REFERENCES


[1] K. Mehta and S. P. Panda, "Sentiment analysis on e-commerce apparels using convolutional neural network," *International Journal of Computing*, vol. 21, pp. 234–241, Jun. 2022.

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[3] S. Robertson, "Understanding inverse document frequency: on theoretical arguments for IDF," *Journal of Documentation*, vol. 60, pp. 503–520, oct 2004.

[4] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013.

[5] Z. Rui and H. Yutai, "Research on short text classification based on word2vec microblog," in *2020 International Conference on Computer Science and Management Technology (ICCSMT)*, pp. 178–182, 2020.

[6] G. Mustafa, M. Usman, L. Yu, M. Afzal, M. Sulaiman, and A. Shahid, "Multi-label classification of research articles using word2vec and identification of similarity threshold," *Scientific Reports*, vol. 11, p. 21900, 11 2021.

[7] B. Das and S. Chakraborty, "An improved text sentiment classification model using TF-IDF and next word negation," *CoRR*, vol. abs/1806.06407, 2018.

[8] Y. Lecun and Y. Bengio, *Convolutional Networks for Images, Speech and Time Series*, pp. 255–258. The MIT Press, 1995.

[9] S. Baker and A. Korhonen, "Initializing neural networks for hierarchical multi-label text classification," in *BioNLP 2017*, (Vancouver, Canada,), pp. 307–315, Association for Computational Linguistics, Aug. 2017.

[10] S. Hochreiter, "Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München," 1991.

[11] A. Petrosyan, A. Dereventsov, and C. G. Webster, "Neural network integral representations with the ReLU activation function," in *Proceedings of The First Mathematical and Scientific Machine Learning Conference* (J. Lu and R. Ward, eds.), vol. 107 of *Proceedings of Machine Learning Research*, pp. 128–143, PMLR, 20– 24 Jul 2020.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[13] B. Jang, I. Kim, and J. W. Kim, "Word2vec convolutional neural networks for classification of news articles and tweets," *PLOS ONE*, vol. 14, pp. 1–20, 08 2019.

[14] R. Kurnia and A. Girsang, "Classification of user comment using word2vec and deep learning," *International Journal of Emerging Technology and Advanced Engineering*, vol. 11, pp. 1–8, 05 2021.

[15] S. Bozinovski, "Reminder of the first paper on transfer learning in neural networks, 1976," *Informatica (Slovenia)*, vol. 44, no. 3, 2020.