

Efficient Text Classification with Deep Learning on Imbalanced Data Improved with Better Distribution

Beytullah YILDIZ^{1*}

¹ Department of Software Engineering, School of Engineering, Atilim University, Ankara, Turkey

*1 beytullah.yildiz@atilim.edu.tr

(Geliş/Received: 06/02/2022;

Kabul/Accepted: 22/02/2022)

Abstract: Technological developments and the widespread use of the internet cause the data produced on a daily basis to increase exponentially. An important part of this deluge of data is text data from applications such as social media, communication tools, customer service. The processing of this large amount of text data needs automation. Significant successes have been achieved in text processing recently. Especially with deep learning applications, text classification performance has become quite satisfactory. In this study, we proposed an innovative data distribution algorithm that reduces the data imbalance problem to further increase the text classification success. Experiment results show that there is an improvement of approximately 3.5% in classification accuracy and over 3 in F1 score with the algorithm that optimizes the data distribution.

Key words: Text classification, Data Imbalance, Data Distribution, Deep learning, Word Embedding.

Daha İyi Dağıtımla İyileştirilmiş Dengesiz Veriler Üzerinde Derin Öğrenme ile Verimli Metin Sınıflandırması

Öz: Teknolojik gelişmeler ve internetin yaygınlaşması, günlük olarak üretilen verilerin katlanarak artmasına neden olmaktadır. Bu veri tufanının önemli bir kısmı sosyal medya, iletişim araçları, müşteri hizmetleri gibi uygulamalardan gelen metin verilerinden kaynaklanmaktadır. Bu büyük miktarda metin verisinin işlenmesi otomasyona ihtiyaç duymaktadır. Son zamanlarda metin işlemede önemli başarılar elde edilmiştir. Özellikle derin öğrenme uygulamaları ile metin sınıflandırma performansı oldukça tatmin edici hale gelmiştir. Bu çalışmada, metin sınıflandırma başarısını daha da artırmak için veri dengesizliği sorununu azaltan yenilikçi bir veri dağıtım algoritması önerdik. Deney sonuçları, veri dağılımını optimize eden algoritma ile sınıflandırma doğruluğunda yaklaşık %3,5 ve F1 puanında 3'ün üzerinde bir iyileşme olduğunu göstermektedir.

Anahtar kelimeler: Metin sınıflandırma, Veri Dengesizliği, Veri Dağıtım, Derin öğrenme, Kelime Gömme.

1. Introduction

With the widespread use of the Internet, data production speed, volume, and variety of data have increased significantly. Text data has a significant share in the produced data. Many applications such as social media, customer service, and communication tools are increasingly generating huge text data. The processing and analysis of these data become very important. Institutions and organizations hire employees and experts who will carry out the tasks of reading, classifying, evaluating, and responding to texts. The correct and fast processing and the response of the text are extremely important in terms of quality and satisfaction. However, the current business model has some drawbacks that need improvement. When the workload increases, either the number of employees or the working hours of the employees must be increased in order to provide the necessary service. This results in compromising the quality of the work. In addition, increasing the number of employees or working hours is a costly solution. Therefore, Automation has become necessary for the processing of increasing text data.

Classification is one of the main tasks for text processing. We see its application in many natural language processing (NLP) tasks. Recently, very successful classification results have been obtained with the prominent deep learning models [1-3]. Well-known deep learning networks such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and recently networks with Attention layer, especially Transformer, have been used text classification [4, 5]. In order to better classify the text with these networks, data preparation is important; it is vital that the data is appropriately represented and well distributed among the training batches. In addition to simple word representation methods such as one-hot encoding, bag-of-words, and term frequency-inverse document frequency (TF-IDF), advanced methods that take into account semantic and syntactic information such as Word2Vec [6], FastText [7], Glove [8], BERT [9] can be used. To improve performance and success, it is

* Corresponding author: beytullah.yildiz@atilim.edu.tr. ORCID Number of authors: ¹ 0000-0001-7664-5145

advantageous to use features that best describe the text in terms of classification, rather than using the entire text. The correct representation of the text and the selection of the correct features are processes that require expertise depending on the data and method to be used. With the use of deep learning and word representation methods, there have been important conveniences and gains in this regard.

Deep learning models are trained with batches of data. The data distribution in these batches should ideally be homogeneous so that the models can learn the patterns efficiently. However, it is not always possible to have ideal data and it is difficult to eliminate this deficiency, which we can define as imbalanced data [10]. One way to do this is to use shuffle to randomly distribute data. However, this may not eliminate the imbalanced data problem. Another way is to use data augmentation. Text data augmentation has some problems. It is difficult to create new text data that contains the original information and properties of the data. Therefore, in this study, we introduce an algorithm to create better distribution to mitigate imbalanced data problems. We classified 263168 documents containing 15 classes using deep learning models. Word2Vec word vectors were created using 2803125 documents of approximately 203 million words. Long short-term memory (LSTM) based deep learning models were created for classification tasks. We proposed an innovative algorithm for better distribution of the training dataset to increase classification success. Experimental results show an increase of about 3.5% in classification accuracy and an improvement of over 3 in F1 score with the proposed data distribution.

In the second section of this article, related works are presented. In the third section, we describe the methodology of our research. Results and evaluations are given in the fourth section. In the last section, we provide the conclusion of our research.

2. Related Work

A brief overview of text classification algorithms was provided by Kowsari et al. in [11]. In the article, existing algorithms and techniques, text feature extraction methods, and dimensionality reduction methods were discussed. The authors also explored the limitations of each technique in real-world problems. An effective text classification requires good word representation and data distribution, in addition to other requirements. The importance of word representation was discussed in [12, 13]. The authors examined the effect of better word representation on classification success. On the other hand, the problem of data imbalance, which hinders classification success, has also attracted the attention of many studies.

Sun et al. studied imbalanced data where the number of text data in some classes was relatively small [14]. The authors provided some conclusions as a result of the experiments. They stated that when the number of texts in the classes is the same, the difference in the number of words in the texts is a factor affecting the success of classification. They also claimed that if the number of texts in the classes is different, increasing the number of texts in the class with a small number of texts does not affect the success much. However, we think that the reason for this claim is due to the difficulty of producing texts representing the class. In [15], the authors presented an approach to measure and reduce unwanted bias in machine learning models. In this context, it was shown how models with imbalances in the training dataset can lead to undesirable bias and thus potentially unfair practices. To provide a solution, a reduction method, which is an unsupervised approach based on balancing the training dataset, was proposed. The approach was claimed to reduce unwanted bias without sacrificing overall model quality. [16] presented a KNN-based method for unevenly distributed large sets of documents. Experimental results showed that the approach provides better text classification.

Imbalance in classes is often come across in real applications of text classifications, especially one-vs-all methods. Therefore, it is quite important to address the issue for reasonable performance. To mitigate the problem, Ogura et al. focused their attention on a feature selection scheme and explored various criteria for feature selection [17]. They examined three different types of metrics and showed that feature selections using the appropriate metrics in the unbalanced dataset yield satisfactory classification success. The problem of underrepresentation of categories with fewer examples was attempted to be solved by Liu et al. using a simple probability-based term weighting scheme [18]. This scheme directly used two critical information ratios, namely, relevance indicators. Using Support Vector Machines and Naïve Bayes classifiers on two benchmark datasets, including Reuters-21578, the proposed work was compared with other classical weighting schemes and showed significant improvement for categories with fewer examples. [10] presented an experimental analysis using various text data representations and data balancing schemes to obtain a classification model with the highest success. The authors' proposed schemes to deal with data imbalance and to analyze it with a numerical optimization problem in which the costs are derived by a Differential Evolution algorithm. In the book chapter, Liu et al. explained the approaches adopted to resolve data imbalance in text classification and group them according to their primary focus [19]. The authors

showed the effects of class imbalance on classification models in [20]. They conduct extensive experiments to highlight the nature of the relationship between the degree of class imbalance and classifier performance.

3. Methodology

In many real-world applications, some classes in training datasets have less representation than others. This imbalanced data structure causes problems in Machine Learning classification and results in poor classification success as there is not enough data to learn. Therefore, we presented an algorithm that optimizes the data distribution as a solution to the data imbalance. We studied this algorithm in an LSTM-based deep learning model. We made text classification on 263168 Turkish documents labeled in 15 categories. The same text data and the same word representations were used when evaluating the model. Since deep learning methods use numbers instead of text, we drew attention to the vectorization of texts using a word representation. Python programming language was used in the development of the application. Word vectors were created using Gensim [21] library and the TensorFlow library was used for deep learning model development.

We performed the steps in Figure 1 to train and test the classification model. We trained and tested our model on the normal dataset containing Turkish documents and the dataset with the removed data imbalance in which we applied our algorithm. The algorithm was applied while generating batches for training. The deep learning model was trained with batches of text documents because the data size was huge. In the remainder of this section, we'll explain our solution to the data imbalance. We will provide detailed information about the classification model, the dataset we used, and the Word2vec model we used to create the word representations.

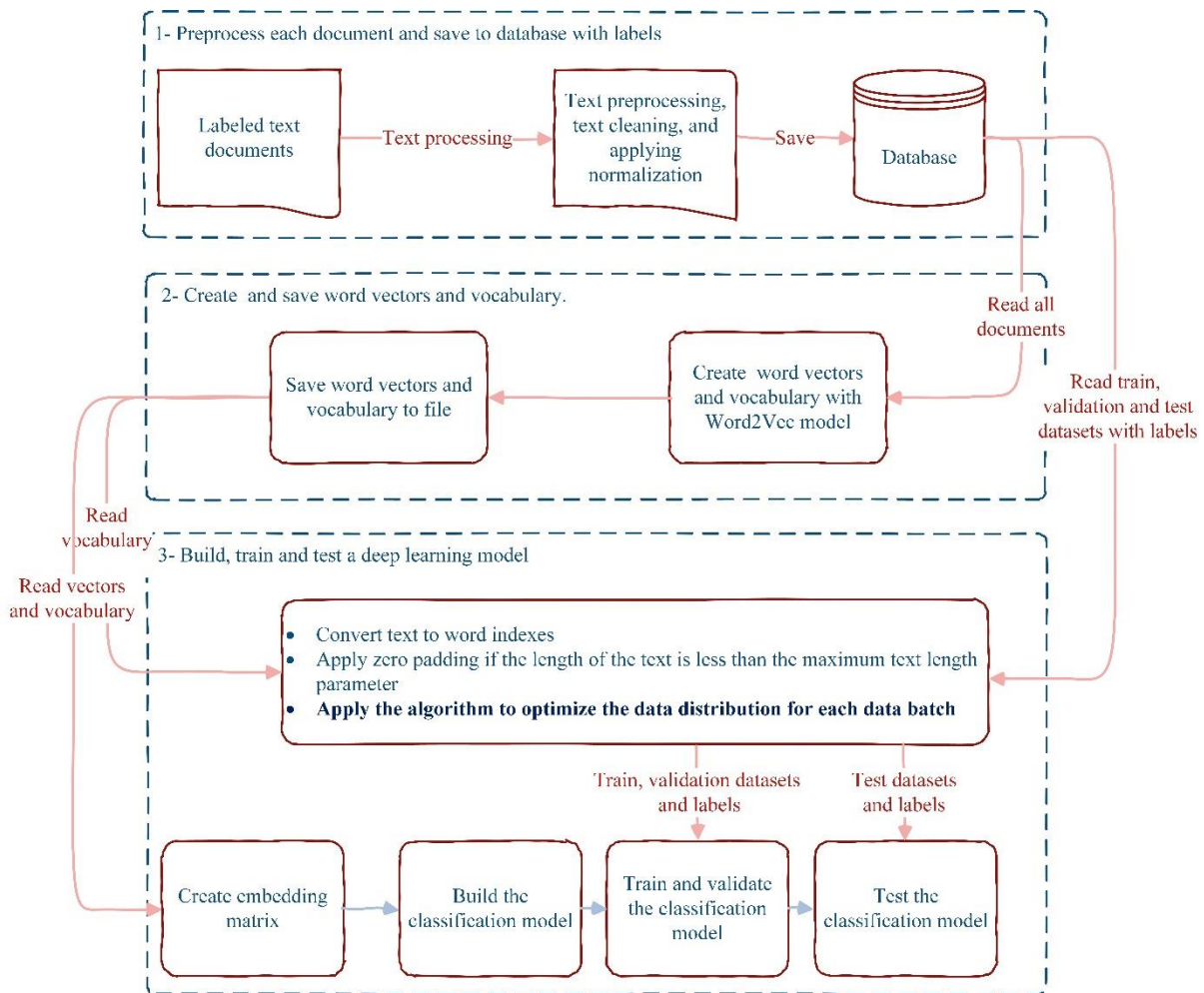


Figure 1. Training and testing the classification model and applying algorithms that resolve data imbalances.

3.1. Solving Data Imbalance

In many real-life application datasets, some classes are less represented than others. In other words, the data is skewed. When the number of texts in some classes is less than in others, the training process is negatively affected. This problem is called imbalanced data. There are solutions to mitigate this problem, such as data augmentation and deleting data from classes with more data. Data deletion is generally not a good choice as it can remove some necessary information. On the other hand, data augmentation is difficult for text data as opposed to image data. For this reason, we prefer to distribute the data optimally into the training batches, considering the number of documents in the classes. Hence, we created the following algorithm that optimally performs the data distribution in the datasets. Algorithm 1 creates ideal batches, while Algorithm 2 performs optimal data distribution in each training batch.

Algorithm 1: Distribute Data

```

1: procedure GetBatches()
2:   Number of classes is m
3:   for i ← 1, m do
4:     classes(i) ← getClasses()
5:     batches ← CreateWellDistributedBatches (classes)
6:   end for
7: end procedure

```

Algorithm 2: Create Well-distributed Batches

```

1: procedure CreateWellDistributedBatches (classes)
2:   normIndex is a two dimensional array
3:   for i ← 1, m do
4:     class ← classes(i)
5:     max ← len(class), min ← 1, row ← 1
6:     while class is not empty do
7:       index (row) ← an incremental value starting from 1 to max for each row
8:       normIndex (i ,row) ← (index(row) – min)/ (max – min)
9:       row ← row+ 1
10:    end while
11:  end for
12:  classesWithNormIndex ← join classes with normalized indexes
13:  range ← 1/numberOfBatch
14:  r ← 0
15:  for i ← 1, numberOfBatch do
16:    while data in classes do
17:      if r < normIndex for the data < r+range then
18:        selectToBatch(classesWithNormIndex)
19:      end if
20:    end while
21:    r ← r + range
22:  end for
23: end procedure

```

We will give an example shown in Table 1 to explain how the algorithm works. Let's say we have a dataset with categories A, B, and C. Category A consists of 10 texts, Category B consists of 5 texts, and Category C consists of 3 texts. There are 18 texts in total in the dataset. It is aimed to divide the dataset into 3 optimal batches of text. When the data is divided into three equal parts without any mixing or distribution as shown in Table 1 (Left-Original), it is seen that the classes are stacked in certain parts. The first part consists of A class texts, the second part consists of A and B class texts. The third part consists of B and C class texts. Generally, classes are randomly distributed using shuffle. However, this method does not guarantee that the categories are optimally distributed in the batches for classification training. The texts in Table 1 (right) were distributed according to the text frequency in the classes using the algorithm we suggested. When the data is divided into batches, it is seen

that the texts in each class are distributed as evenly as possible. Data that is not evenly distributed across 3 batches is placed in an appropriate batch. As the number of data increases, this small error will decrease and the data distribution will approach the ideal.

Table 1. A sample distribution of data.

Original					Part	Distributed				
Indices	Class	Min	Max	Normalized Indices		Indices	Class	Min	Max	Normalized Indices
1	A	1	10	0.00	1	1	A	1	10	0.00
2	A	1	10	0.11	2	1	B	1	5	0.00
3	A	1	10	0.22	3	1	C	1	3	0.00
4	A	1	10	0.33	4	2	A	1	10	0.11
5	A	1	10	0.44	5	3	A	1	10	0.22
6	A	1	10	0.56	6	2	B	1	5	0.25
7	A	1	10	0.67	1	4	A	1	10	0.33
8	A	1	10	0.78	2	5	A	1	10	0.44
9	A	1	10	0.89	3	3	B	1	5	0.50
10	A	1	10	1.00	4	2	C	1	3	0.50
1	B	1	5	0.00	5	6	A	1	10	0.56
2	B	1	5	0.25	6	7	A	1	10	0.67
3	B	1	5	0.50	1	4	B	1	5	0.75
4	B	1	5	0.75	2	8	A	1	10	0.78
5	B	1	5	1.00	3	9	A	1	10	0.89
1	C	1	3	0.00	4	10	A	1	10	1.00
2	C	1	3	0.50	5	5	B	1	5	1.00
3	C	1	3	1.00	6	3	C	1	3	1.00

3.2. Word Representation

We used the Word2Vec model, an unsupervised method proposed by Mikolov et al. [6, 22], for representing words in texts. Word2Vec takes a corpus and creates word vectors of several hundred dimensions. It creates numerical vectors to represent words by trying to position similar syntactic and semantic words close to each other in vector space. There are two different methods for creating word vectors. The first of these is CBOW (Continuous Bag of Words) and the second is SG (Skip-Gram). While the CBOW method tries to guess the word from the words in a particular window to the right and left of the word, the SG method tries to guess the words to the right and left of the word from the word itself. Besides word vectors, we also come across researches that create vectors for documents, patterns, users, and classes [23, 24].

We created the word vectors using the SG model. The SG model has an input layer, a hidden layer, and an output layer. The input vector $x = \{x_1, \dots, x_v\}$ is one-hot encoded. The weights between the input layer and the hidden layer can be represented by the $V \times N$ matrix W shown in formula 1, where V is the vocabulary size and N is the unit size in the hidden layer. Between the output layer and the hidden layer, there is another $V \times N$ weight matrix W' shown in formula 2.

$$W_{V \times N} = \begin{bmatrix} v_{11} & \dots & v_{1N} \\ \vdots & \vdots & \vdots \\ v_{V1} & \dots & v_{VN} \end{bmatrix} \quad (1)$$

$$W'_{V \times N} = \begin{bmatrix} v'_{11} & \dots & v'_{1N} \\ \vdots & \vdots & \vdots \\ v'_{V1} & \dots & v'_{VN} \end{bmatrix} \quad (2)$$

Given a single word x_k , and assuming $x_k = 1$ and $x_{k'} = 0$ for $k \neq k'$, we obtain the formula 3 for the hidden-layer outputs, where v_{w_i} denotes the input vector of the associated word w of the input layer.

$$h = x^T W = W_{(k,)} := v_{w_i} \quad (3)$$

The output utilizing the Softmax function can be calculated by using $v'_w \cdot h$. Applying Formula 3, we get the following formula:

$$p(w_o|w_I) = \frac{\exp(v'_{w_o} \cdot v_{w_I})}{\sum_{j'=1}^V \exp(v'_{w_{j'}} \cdot v_{w_I})} \quad (4)$$

Thus, the loss function is formula 5 where c is the number of context words.

$$E = -\log p(w_{o,1}, w_{o,2}, \dots, w_{o,c}|w_I) \quad (5)$$

After providing the mathematical background information, we explain how we generate the word vectors. A collection of 2.8 million Turkish texts was used to create the Word2Vec model. The corpus has a total of 243 million words and 2.6 million unique words, including misspellings words. All uppercase letters have been converted to lowercase to avoid creating a new token for the same word. While Word2vec training, windows size, negative sample, minimum word count, and vector size were set to 20, 5, 5 250, respectively, in the SG method. When the minimum word count parameter was set to 5, the number of unique words decreased from 2.6 million to 603 thousand words. This is because there are many misspelled words in the corpus. Therefore, the minimum word count parameter was chosen as 5.

3.3. Deep Learning Model

For the experiments, we used an LSTM-based deep learning model, which is known to classify well on text data. With the algorithm we proposed, the dataset that solved the problem of data imbalance and the regular dataset, in which this algorithm was not applied, were trained using the same model. In other words, two datasets were tested on the same deep learning model. In the deep learning model, 1 Embedding layer, 2 bidirectional LSTM layers of 128 units, 2 Dense layers of 500 and 300 units, and 1 output layer of 15 units were used. A 160 x 250 matrix was created for each text, with the embedding layer set to a maximum text length of 160 and using Word2Vec vectors of size 250. Zero padding was applied to texts less than 160 words. 0.2 Dropout was utilized in layers containing LSTM and Dense. Softmax activation function was applied to the output layer. "Adam" was preferred as the optimization function and "sparse_categorical_crossentropy" was used as the loss function.

3.4. Dataset

Statistics of the text dataset used for classification are given in Table 2. The dataset contains a total of 263168 labeled documents with 15 classes. Of these documents, approximately 70% corresponding to 185344 documents were used for training, approximately 15% corresponding to 38912 documents were used for verification, and approximately 15% corresponding to 38912 documents were used for testing.

Table 2. Dataset used for classification.

Class Name	Total	Train	Validation	Test
1.Class	12055	8490	1782	1783
2.Class	18150	12782	2684	2684
3.Class	14405	10145	2130	2130
4.Class	26086	18372	3857	3857
5.Class	9550	6726	1412	1412
6.Class	20945	14751	3097	3097
7.Class	37180	26186	5497	5497
8.Class	11280	7944	1668	1668
9.Class	15505	10919	2293	2293
10.Class	15435	10871	2282	2282
11.Class	10768	7584	1592	1592
12.Class	30312	21348	4482	4482
13.Class	11795	8307	1744	1744
14.Class	12109	8528	1791	1790
15.Class	17593	12391	2601	2601
Total	263168	185344	38912	38912
Ratio (~%)	100	70	15	15

4. Benchmark and Evaluation

We used the documents whose statistics are given in Table 2 in the experiments. We first trained the dataset as it provided. We collected the experimental metrics. We then optimally redistribute documents from datasets to batches to find the improvement our algorithm provides to mitigate the data imbalance issue. Although the train, validation, and test datasets contain the same number of documents, we rearrange the documents in the batches by class types so that the distribution is optimal. Then, we collected experimental results. We did not make use of the "EarlyStopping" callback to see the experimental behaviors and used 20 epochs for model training. However, we save the best models for both experiments. A batch size of 1024 was used for each experiment. Precision, recall, F1-score metrics, and model accuracies were used to evaluate approaches.

4.1. Classification Model with Regular Dataset

The loss and accuracy graphs of the train and validation datasets are given in Figure 2 and Figure 3. The model starts overfitting from Epoch 14. After this point, the validation loss and accuracy are not improving even though training loss and accuracy values are getting better. Hence, we save the model at this point to measure precision, recall, and F1 scores and accuracy on test datasets.

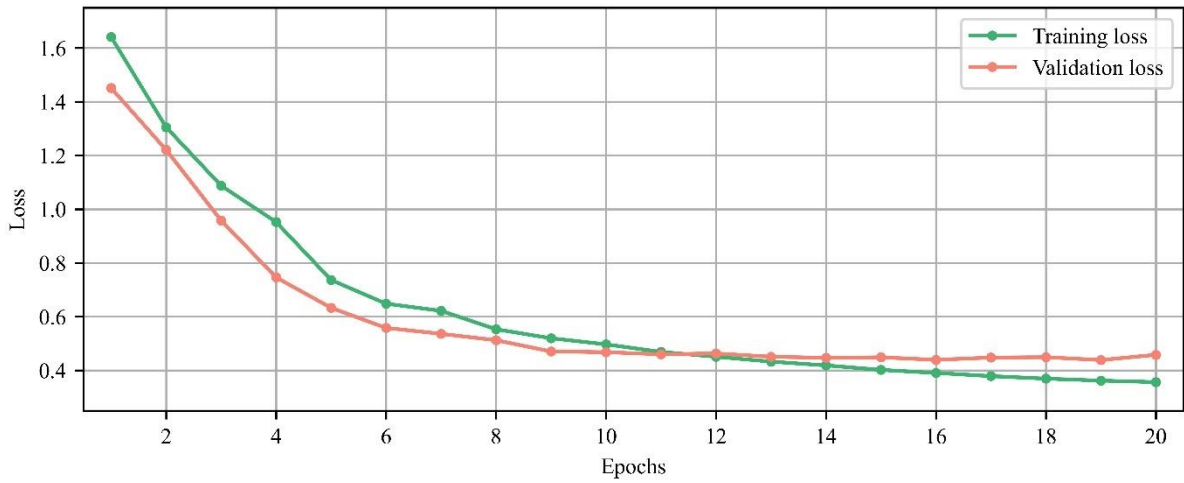


Figure 2. The loss results obtained during the training with the dataset in which the proposed distribution algorithm is not applied.

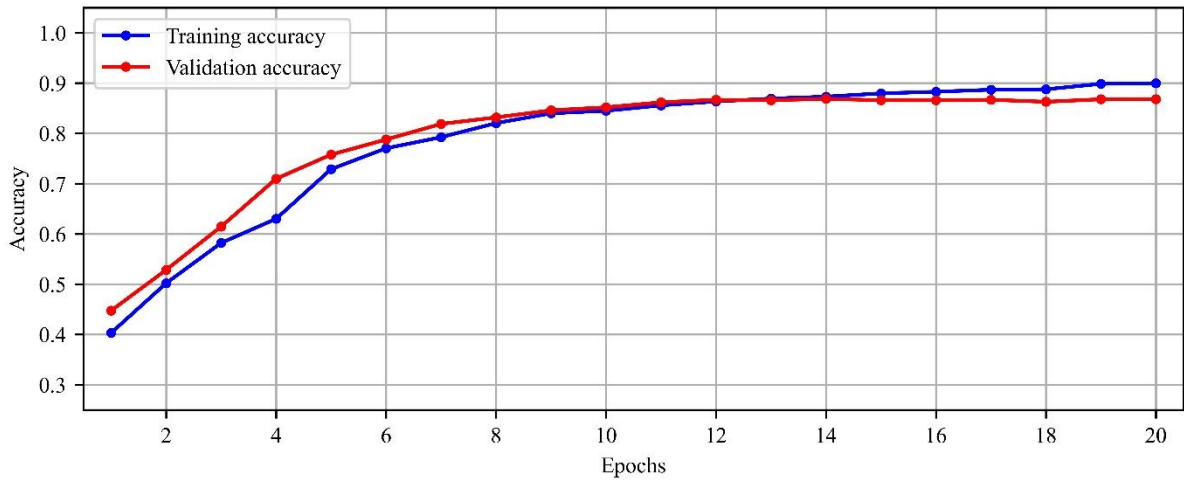


Figure 3. The accuracy results obtained during the training with the dataset in which the proposed distribution algorithm is not applied.

The saved model was used to collect benchmarks. The accuracy of the test dataset is 87.96%. The precision, recall, and F1-score values for each class are given in Table 3. The model's F1 score is 87.84. The other metrics are also reasonable. Class transitivity seems to have caused bad results in some classes. Since the dataset we use was obtained from a real-life application, some classes such as economics and education have similar content with other classes. Therefore, metrics are worse in these classes.

Table 1. Precision, recall, and F1-score metrics obtained with the test dataset in the model trained with the dataset to which the distribution algorithm is not applied.

Class Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	All
Support	1783	2684	2130	3857	1412	3097	5497	1668	2293	2282	1592	4482	1744	1790	2533	38912
Precision	90.63	83.15	83.33	89.77	91.18	90.65	89.53	83.84	85.1	89.65	83.43	89.27	91.94	87.28	86.38	87.99
Recall	92.5	90.29	64.1	88.74	92.76	88.41	92.51	94.28	73.67	85.03	87.51	96.15	83.26	86.44	90.16	88
F1-Score	91.56	86.57	72.46	89.25	91.96	89.52	91	88.75	78.97	87.28	85.42	92.58	87.38	86.86	88.23	87.84

4.2. Classification Model with Well-distributed Dataset

For the optimally distributed dataset to the batches, loss and accuracy plots of the train and validation datasets are given in Figure 4 and Figure 5. The model started to encounter the overfitting problem after Epoch 11. Therefore, we save the model at this point to measure precision, recall, F1 score, and the accuracy value on the test dataset. Compared to the model described in the previous section, the model is trained faster with the optimally distributed dataset using our proposed algorithm. In addition, it is observed that the training and validation accuracy values are much higher.

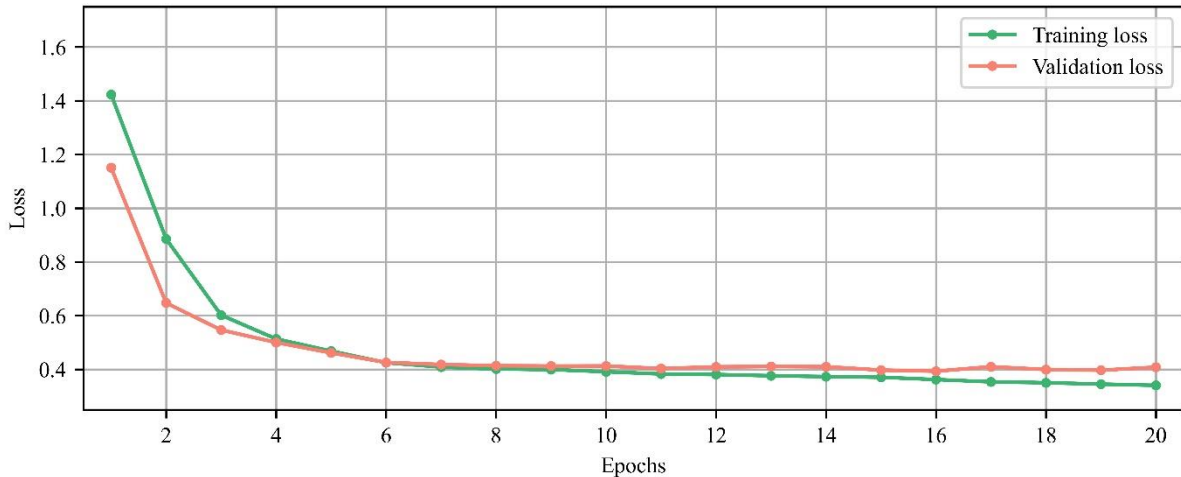


Figure 4. The loss results obtained in the model trained with the dataset created with the proposed distribution algorithm.

Table 2. Precision, recall, and F1-score metrics obtained with the test dataset using the model trained with the dataset created with the distribution algorithm.

Class Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	All
Support	1783	2684	2130	3857	1412	3097	5497	1668	2293	2282	1592	4482	1744	1790	2533	38912
Precision	94	86.25	86.23	93.67	95.51	92.78	92.72	88.16	90.91	92.63	87.85	91.79	92.31	89.76	91.11	91.3
Recall	95.79	93.22	70.28	91.73	94.83	91.22	94.78	96.88	76.8	88.65	90.83	98.55	92.2	91.12	92.62	91.31
F1-Score	94.89	89.6	77.44	92.69	95.17	91.99	93.74	92.31	83.26	90.6	89.32	95.05	92.25	90.43	91.86	91.19

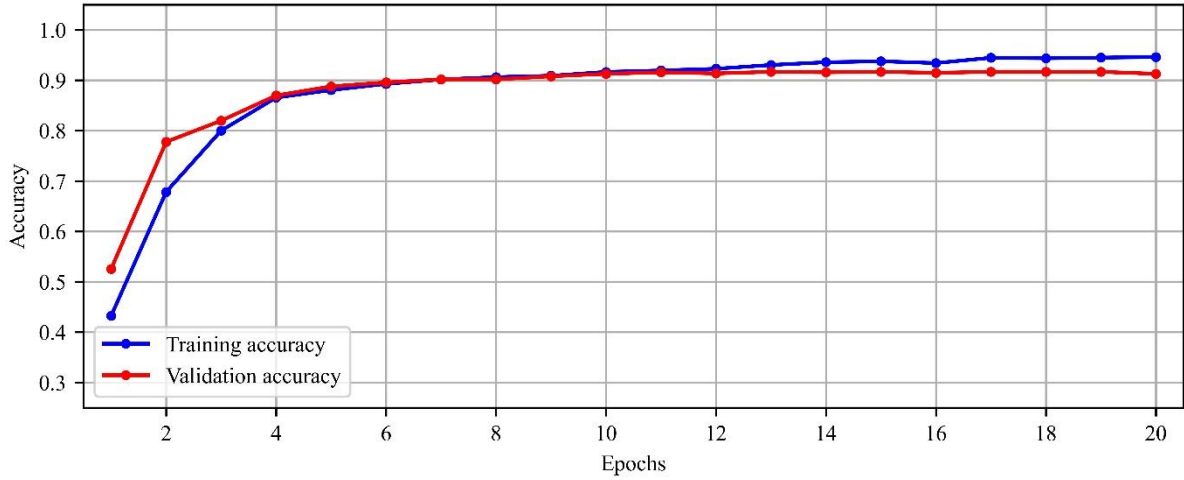


Figure 5. The accuracy results obtained in the model trained with the dataset created with the proposed distribution algorithm.

When we look at the metrics collected for the test dataset. The accuracy of the test dataset was measured as 91.45%. Precision, recall, and F1 score values are given in Table 4. For the entire dataset, they are 91.3, 91.31, and 91.19, respectively. It is clearly seen that these metrics are also better than the results we obtained in the previous experiment.

4.3 Comparison

In terms of accuracy, we observe that our algorithm to distribute data into batches optimally provides 3.46% better output. In addition, the average F1 score improved by 3.35. We see similar advances in recall and precision measures. When we look at each of the classes, we witness improvement in almost every metric. In particular, it appears that improvement is higher in classes with the worst metrics of previous experiments. This shows that our proposed algorithm contributes better to solve the problem of data imbalance in underrepresented or transitive classes. In addition, training takes less time as the model trained with the data set with the improved distribution is optimized faster.

5. Conclusion

Nowadays, huge amounts of text data are produced from many sources. This excessive amount of text data creates more workload. Dealing with this workload with new employees or more working hours is an expensive method. Therefore, document classification, which is an important task for text data, requires atomization. Deep learning, which has achieved significant success recently, offers an ideal solution. However, real-life application datasets often do not have balanced datasets. Data imbalance is ubiquitous in real-life applications. This problem leads to poor classification success.

In order to reduce the data imbalance problem, we proposed an algorithm that can better distribute the data to the batches. We analyzed our proposed method using a very large Turkish dataset containing 263168 documents with 15 classes. We conducted our experiments using an LSTM-based deep learning model. First, we trained the deep learning model and collected the experimental metrics without using the proposed method on the data set we collected. Then, we trained the same model using the dataset improved with our proposed distribution algorithm. We compared the metrics we gathered from both experiments. Our proposed solution yielded approximately 3.5% better accuracy than the experiment using a regular approach. It also shows an increase of more than 3 in the F1 score. Similar improvements are seen in other metrics. These results clearly demonstrate the importance of better data distribution to training batches in text classification. Our proposed algorithm, which mitigates the data imbalance problem, offers an important solution in this regard.

References

- [1] Lai S, Xu L, Liu K, Zhao J. Recurrent convolutional neural networks for text classification. In: 29th AAAI conference on artificial intelligence, Austin, Texas USA, January 25–30, 2015 2015.
- [2] Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J. Deep Learning-based Text Classification: A Comprehensive Review. *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1-40, 2021.
- [3] Tufek A, Aktas M S. On the provenance extraction techniques from large scale log files: a case study for the numerical weather prediction models. In: *European Conference on Parallel Processing*, 2020: Springer, pp. 249-260.
- [4] Tezgider M, Yildiz B, Aydin G. Text classification using improved bidirectional transformer. *Concurrency and Computation: Practice and Experience*, p. e6486.
- [5] Soyalp G, Alar A, Ozkanli K, Yildiz B. Improving Text Classification with Transformer. In: *2021 6th International Conference on Computer Science and Engineering (UBMK)*, 2021; Ankara, Turkey, IEEE pp. 707-712.
- [6] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: *26th International Conference on Neural Information Processing Systems*, 2013, Lake Tahoe, Nevada, pp. 3111-3119.
- [7] Joulin A, Grave E, Bojanowski P, Mikolov T. Bag of Tricks for Efficient Text Classification. In: *15th Conference of the European Chapter of the Association for Computational Linguistics*, April 2017, Valencia, Spain: Association for Computational Linguistics, in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427-431.
- [8] Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation. In: *The Conference on Empirical Methods in Natural Language Processing (EMNLP)*. October 2014 Doha, Qatar: Association for Computational Linguistics, pp. 1532-1543.
- [9] Devlin J, Chang M W, Lee K, Toutanova K. BERT: pre-training of deep bidirectional transformers for language understanding. In: *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019, Minneapolis, MN, USA.
- [10] Padurariu C, Breaban M E. Dealing with data imbalance in text classification. *Procedia Computer Science*, 2019, vol. 159, pp. 736-745.
- [11] Kowsari K, Jafari Meimandi K, Heidarysafa M, Mendu S, Barnes L, Brown D. Text Classification Algorithms: A Survey. *Information*, 2019, vol. 10, no. 4, p. 150.
- [12] Yildiz B, Tezgider M. Improving word embedding quality with innovative automated approaches to hyperparameters. *Concurrency and Computation: Practice and Experience*, 2021 p. e6091.
- [13] Yildiz B, Tezgider M. Learning Quality Improved Word Embedding with Assessment of Hyperparameters. In *European Conference on Parallel Processing*, 2019: Springer, pp. 506-518.
- [14] Li Y, Sun G, Zhu Y. Data imbalance problem in text classification. In: *2010 Third International Symposium on Information Processing*, 2010: IEEE, pp. 301-305.
- [15] Dixon L, Li J, Sorensen J, Thain N, Vasserman L. Measuring and mitigating unintended bias in text classification. In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, 2018, pp. 67-73.
- [16] Shi K, Li L, Liu H, He J, Zhang N, Song W. An improved KNN text classification algorithm based on density. In: *2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, 2011: IEEE, pp. 113-117.
- [17] Ogura H, Amano H, Kondo M. Comparison of metrics for feature selection in imbalanced text classification. *Expert Systems with Applications*, 2011, vol. 38, no. 5, pp. 4978-4989.
- [18] Liu Y, Loh H T, Sun A. Imbalanced text classification: A term weighting approach. *Expert systems with Applications*, 2009, vol. 36, no. 1, pp. 690-701.
- [19] Liu Y, Loh H T, Kamal Y T, Tor and S B. Handling of imbalanced data in text classification: Category-based term weights. In: *Natural language processing and text mining*: Springer, 2007, pp. 171-192.
- [20] Thabtah F, Hammoud S, Kamalov F, Gonsalves A. Data imbalance in classification: Experimental evaluation. *Information Sciences*, 2020, vol. 513, pp. 429-441.
- [21] Rehurek R, Sojka P. Software framework for topic modelling with large corpora. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 2010: Citeseer.
- [22] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. *arXiv preprint*, 2013, arXiv:1301.3781.
- [23] Olmezogullari E, Aktas M S. Pattern2Vec: Representation of clickstream data sequences for learning user navigational behavior. *Concurrency and Computation: Practice and Experience*, 2021, p. e6546.
- [24] Hallac I R, Makinist S, Ay B, and Aydin G. user2vec: Social media user representation based on distributed document embeddings. In: *2019 International Artificial Intelligence and Data Processing Symposium (IDAP)*, 2019: IEEE, pp. 1-5.