Heliyon 9 (2023) e15670

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon

Research article

Comparison of pre-trained language models in terms of carbon emissions, time and accuracy in multi-label text classification using AutoML

Pinar Savci [a], Bihter Das [b,*]

[a] Arçelik A.Ş. Karaağaç Caddesi 2-6, Sütlüce Beyoğlu 34445 Istanbul, Turkey
[b] Department of Software Engineering, Technology Faculty, Fırat University, 23119, Elazig, Turkey

ARTICLE INFO

Keywords:
Natural language processing
Autotrain
Multi-text classification
Pre-trained language models
Artificial intelligence
AutoNLP

ABSTRACT

Since Turkish is an agglutinative language and contains reduplication, idiom, and metaphor words, Turkish texts are sources of information with extremely rich meanings. For this reason, the processing and classification of Turkish texts according to their characteristics is both time-consuming and difficult. In this study, the performances of pre-trained language models for multi-text classification using Autotrain were compared in a 250 K Turkish dataset that we created. The results showed that the BERTurk (uncased, 128 k) language model on the dataset showed higher accuracy performance with a training time of 66 min compared to the other models and the CO2 emission was quite low. The ConvBERTurk mC4 (uncased) model is also the best-performing second language model. As a result of this study, we have provided a deeper understanding of the capabilities of pre-trained language models for Turkish on machine learning.

1. Introduction

With the advancement of technology and the development of the internet, the power of information and the number of documents produced in the computer environment is increasing day by day. Along with the numerous benefits brought by a large amount of information, many phenomena in the internet world have begun to be referred to as information garbage [1]. Finding and using what one is looking for in this heap of information led to the need to organize information according to certain rules. The solution to this problem can only be solved by classifying documents [2]. One of the problems that arise in this context is the classification of texts in electronic media. The purpose of text classification is to determine which of the predetermined classes it belongs to, by looking at the features of the current text. Text classification, also known as 'document classification' or 'text categorization', has many uses such as information retrieval, information extraction, document filtering, and automatic metadata acquisition [3]. Text classification is one of the application areas of Natural Language Processing, an area of artificial intelligence. Text classification has study topics such as sentiment analysis, emotion detection, spam analysis, and document indexing.

Since they can learn rich grammar from large-scale corpus, the development of pre-trained language models and deep learning architectures has brought solutions to natural language processing problems in many subjects such as text classification [4,5].

Natural language processing applications have been carried out in many languages, especially in English. The fact that the Turkish language is an agglutinative language, has free word order, has idioms, figurative expressions and allusions, and the sentences with

* Corresponding author.
E-mail address: bihterdas@firat.edu.tr (B. Das).

https://doi.org/10.1016/j.heliyon.2023.e15670
Received 20 December 2022; Received in revised form 15 April 2023; Accepted 18 April 2023
Available online 1 May 2023

different word orders create diversity in terms of subject make it difficult to process Turkish automatically.

In this study, we present a comparative analysis of the performance of pre-trained language models using Autotrain in multi-text classification in Turkish dataset. The performance of the language models is evaluated in the 250 K dataset written in Turkish. The results show that the BERTurk (uncased, 128 k) and ConvBERTurk mC4 (uncased) language models are more effective than other models in terms of accuracy and CO2 emissions in Turkish multi-text classification.

The remainder of this paper is organized as follows. Section 2 gives information about the studies on text classification in the literature. Section 3 contains details of the data and methodology. Section 4 presents experimental results and discussion. Section 5 presents the conclusion.

The main contributions of this paper are outlined below.

- Comparative deep analysis of pre-trained language models for Turkish text classification was performed for the first time.
- The effect of Autotrain on pre-trained language models was evaluated.
- In the Turkish dataset, the performances of the pre-trained language models were compared in terms of time and CO2 emissions.

2. Background

In this section, a literature review on studies on text classification using pre-trained language models, detailed description of language models, information on transformer, Autotrain and text classification is presented.

2.1. Related works

Related works were examined in two groups as text classification and studies on pre-trained language models. Studies on text classification:

Yildirim et al. performed a Turkish text classification analysis by comparing traditional bag-of-words and machine learning approaches. As a result of the study, traditional methods were performed as well as word embedding approaches [6]. Velioglu et al. made positive, negative, and neutral sentiment analyses on Turkish tweets using two different approaches based on bag-of words and fastText. Machine learning algorithms such as Naive Bayes and Logistic Regression were applied to the bag-of-words applied to text. Experimental results have shown that the two approaches achieve similar results in sentiment analysis [7]. Kılıç et al. performed a text classification process by applying five classifiers and two feature selection algorithms on a data set they created from news sites and articles written in Turkish. As a result of the classification, it was seen that the Random Forest algorithm reached the highest performance with 91.03% accuracy [8].

Aydogan et al. created the largest unlabeled Turkish dataset and word vectors. The performances of Convolutional Neural Network (CNN), Gated Recurrent Unit (GRU), Recurrent Neural Network (RNN), and Long Short-Term Memory (LSTM) methods were compared on the existing data set. It has been shown that LSTM and GRU methods are more successful than other methods in text classification [5]. Cheng et al. proposed a new 2-component model for text classification written in English. The first component uses the attention mechanism in hierarchical feature extraction, and the second component uses capsular networks to extract the semantic relationship between the part-whole of the text. With the proposed model, a 5-class text classification was actualized [9]. Ren et al. proposed a coding method based on Capsule networks to generate an 8-class text classification and word embeddings. They showed that the proposed method achieved as good performance as the existing classification methods with fewer parameters [10]. Studies on pre-trained language models: Gertner et al. actualized a text classification of hate speech detection texts on Twitter, written in Spanish. The multilingual BERT (mBERT) model, one of the pre-trained language models, was used in the experiment [11]. Plaza-del-Arco et al. did a comparative analysis of Bert, XML, and BETO pre-trained language models based on the transformer mechanism to detect hate speech from tweets written in Spanish [12]. Dai et al. actualized a sentiment analysis on unlabeled data using the pre-trained language model. In the study, a mechanism that predicts the next word and automatic encoder approaches are used [13]. Singh et al. developed a model that includes the HL-MTRGU network, using a pre-trained language model for text classification of different lengths. They compared the performance of the proposed model with existing text classification methods. Experimental results show that the proposed model performs better in short texts [14]. Palenzuela et al. explored how pre-trained language models such as BERT and DistilBERT and transfer learning are used in Second Language Acquisition modeling. They stated that the DistilBERT language model achieved 85.20% AUC performance in learning [15]. Zhang et al. performed a text classification by applying a new language model to a large-scale Chinese dataset. In the study, they trained a language model with 2.6 billion parameters and developed a pre-trained model called the Chinese Pre-trained Language Model [16].

2.2. Pre-trained language models

Pre-trained language models were first proposed in 2015 [17]. These language models have recently been found to be quite useful for many unsupervised learning tasks. Language model embeddings can be used as attributes in a target model, or a language model can be adjusted based on target task data [18,19]. It has been seen that pre-trained language models make it possible to learn with fewer data because language models require only unlabeled data, they are particularly useful for situations where labeled data is scarce. With pre-trained language models, it is possible to predict the next word in a given text. In this way, one can learn how languages work by taking advantage of the huge amount of text data available. In addition, language models calculate the probability of a set of words appearing in a given sentence. In this study, pre-trained language models BERTurk, ElecTRa, DistilBERTurk, and

ConvBERTurk were used.

BERT: The Bidirectional Encoder Representations from Transformers (BERT) algorithm, like many other algorithm updates from Google, is a language model developed to better understand queries and provide more accurate results to its users [20]. The BERT model can be explained as a natural language processing technique that uses artificial intelligence and machine learning technologies together. It is trained by Google on BooksCorpus (800 M words) and English Wikipedia (2.5 B words). This language model evaluates the words in front and behind together with similar and synonymous words instead of evaluating the words in the queries one by one. Thus, it aims to better understand long-tail or complex user queries. In addition, with the Google BERT algorithm update, it understands much better what the conjunctions and prepositions used in the queries add to the query. The BERT model, based on fine-tuning, uses the same architectures in both pre-training and fine-tuning. BERT pre-trains deep bidirectional representations from unlabeled text through co-conditioning in both left and right contexts across all layers. However, it was not considered suitable for text production. Because there are absolute position placements on BERT, it is generally recommended to fill the entries on the right rather than the left. BERT was trained using the next sentence prediction (NSP) target using the [CLS] token as a sequence prediction, as well as masked language modeling (MLM). The user can use the first token in a string created with custom tokens instead of a token prediction. For masked language modeling (MLM), 15% of the text is randomly selected to be masked. On these tokens, 80% is exchanged with [MASK], 10% is exchanged with a random token from the alphabet, and the remaining 10% remains the same. Because the purpose of the pre-trained model is to recover the original texts from this masked version. For next sentence prediction, BERT's pre-trained model takes the sentence as 2 inputs designated as A and B, B's job is to actually predict whether there is a sentence after A in the corpus. In this task, it aims to strengthen BERT's ability to reason between sentences. This purpose is also helpful for tasks such as question answering and natural language extraction. There is a difference between the cased and uncased models used in the Bert pre-trained model, in terms of text status and presence of diacritical marks in the WordPiece annotation step. For example, the input "OpenSource" for Bert-cased remained the same, while on Bert-uncased the input was changed from "OpenSource" to "opensource". While the accent marks on the letters used in the Latin language remain the same for Bert-cased, all accent marks are removed in Bert-uncased. In addition to the benefits of the BERT model, there are also some limitations. For example, the BERT language model learns from only 15% of the input tokens and the subsequent sentence prediction is very poor compared to the masked language modeling. Also, the training data size is small and the masking of the training data is done once, it is reused for all epochs.

BERTurk: Stefan Schweter, who made the first publication of the BERT pre-trained model for Turkish, presented the BERTurk-cased and BERTurk-uncased pre-trained models. Later, it also published the community-oriented DistilBERTurk, ElecTRa and ConvBERTurk models for Turkish. BERTurk consists of a basic transformer, an encoder to read the text input, and a decoder to generate an estimate for the task. In the BERTurk model, the input to the encoder is a set of tokens that are first converted to vectors and then processed in the neural network. These sequences are marker embeddings, segment embeddings, and positional embeddings. In fact, the transformer creates a layer stack that maps arrays to arrays, so the output is also a vector array that corresponds one-to-one in the same index between the input and output tokens. In general, BERTurk can be used for a variety of natural language understanding tasks with just a change in the output layer. Fig. 1 Shows the fine-tuning BERTurk architecture from different missions.

The current version of the BERTurk pre-trained model is trained on a filtered and sentence segmented version of the Turkish OSCAR

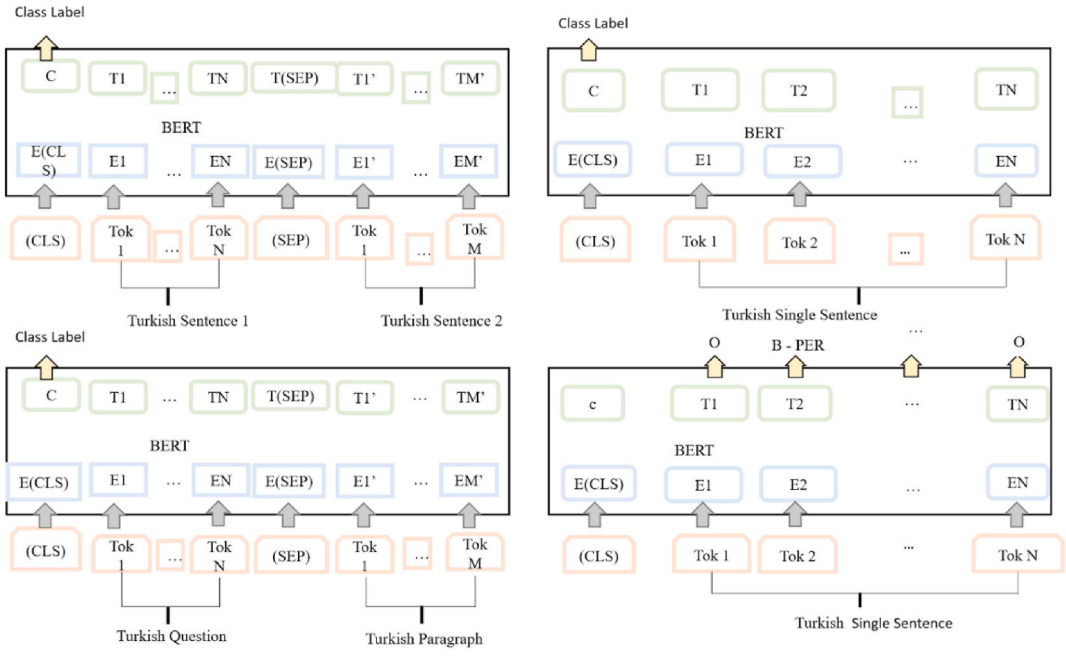

Fig. 1. The architecture of the BERTurk.

corpus, a new Wikipedia transcript and special corpus provided by Kemal Oflazer [21,22]. The size of the final education corpus is 35 GB and has 4,404, 976, 662 coins. Table 1 shows the accuracy values of BERTurk language models in the BOUN and IMST dataset for PoS tagging and in the XTREME Turkish dataset for NER.

ELECTRA: This language model is a self-monitoring pre-trained model. It is also a new pre-training approach that trains two transformer models such as the generator and the discriminator [23]. The task of the model is to train it to distinguish between "real" and "fake" input data. The model is first randomly masked by the masked language modeling task and is used to emulate some tokens in the input. Tokens changed in the generator are forwarded to the discriminator. Discriminator, a hidden small-sized BERT model, distinguishes for each token whether it is original or modified by the generator. After pre-training, the generator is dropped and the discriminator fine-tuned for use in subsequent tasks. All models use transformer neural network architecture. Contextual representations learned by ELECTRA significantly outperform those learned by BERT given the same model size, data, and calculation. While the Electra pre-trained model is being trained, the embedding size and the hidden size are changed, unlike the BERT model. Embedding size is usually smaller and hidden size is larger. An additional projection layer (linear) is used to project embeddings from the embedding dimensions to the hidden dimension. If the embedding size is the same as the hidden size, the projection layer is not used. While the Electra performs well even in a small workout, the Electra-Small produces very good results in terms of performance considering its size.

ElecTRa: ElecTRa small and base trained on the same data as BERTurk is a pre-trained language model published in Turkish as a pre-trained model. IMST dataset and BOUN datasets were used while training the self-monitoring ElecTRA. Fig. 2 shows the architecture of the ElecTRa model.

In the model, the data is trained for 1 M steps for both base and small models. Pre-training checkpoints are written every 100 thousand steps. Averaged Accuracy (PoS tagging) or averaged F1-Score (NER) is reported over 5 runs for each control point. Table 2 shows the Post tagging and NER values of ElecTRa in training. In the ElecTRa-small model, the best results were obtained for the average accuracy rate and the average F1-Score at 1 M steps. In the ElecTRa-base model, the control point selection was tested as 1 M and 800 thousand steps. The values recorded as average accuracy rate and average F1-Score yielded the best performance result for the average F1-Score in 1 M steps, while the average accuracy in 800 K steps performed best. It has been observed that slightly better performance is achieved in 800 K steps for the final control point selection.

DistilBERT: The DistillBERT pre-trained language model emerged as a smaller, faster, cheaper, lighter, distilled version of BERT. The term distillation used here can be defined as a compression technique in which a small model is trained to reproduce the behavior of a larger model. It has 40% fewer parameters than the BERT-base-uncased model and works 60% faster than the other model. Compared to Bert, it offers shorter training and transfer learning time. As a success, they experience a loss of about 2% in natural language comprehension tasks. DistilBERT model, which is pre-trained with Masked Language Modeling tasks, has shown that variations in the tensor's latent size dimension have a smaller effect on computational efficiency for a fixed parameter budget. DistilBERT model, which is pre-trained with Masked Language Modeling tasks, has shown that variations in the tensor's latent size dimension have a smaller effect on computational efficiency for a fixed parameter budget [24]. While other models aim to optimize the performance of BERT, DistilBERT aims to reduce the size of the BERT language model and increase its speed. BERT-base and BERT-large have parameters 110 M and 340 M, respectively. DistillBERT reduces the BERT-base size by 40% and preserves its abilities by 97%. In addition, DistilBERT increases the speed by 60%.

DistilBERTurk: DistilBERTurk is the distilled version of the community-focused cased BERT for Turkish. Distillation is used when moving a model to smaller equipment because a distilled model runs faster and takes up less space. BERTurk is mainly based on a series of layers of attention stacked on top of each other. Therefore, the confidential information that BERTurk learns is located in these layers. This model has the same general architecture as BERT [25]. The number of tiers is reduced by 2 times, and in this model, token embeds and pooling are also removed. Many of the operations used in transformer architecture, such as linear layer and layer normalization, are optimized in modern linear algebra frameworks. DistilBERTurk also outperforms the 24-layer XLM-Roberta model for PoS tagging. Fig. 3 shows the architecture of the DistilBERTurk pre-trained language model.

DistilBERTurk has been trained for 5 days on 4 RTX 2080 TIs with 7 GB of original training data. Table 2 shows the performance values of BOUN and IMST dataset for average accuracy (PoS tagging) and XTREME Turkish dataset accuracy for average F1-Score

Table 1
Accuracy values of BERTurk models.

BERTurk pre-trained model	Development Accuracy	Test Accuracy
BERTurk (cased, 128 k) (IMST-Pos Tagging)	96.614 ± 0.58	96.846 ± 0.42
BERTurk (cased, 32 k) (IMST-Pos Tagging)	97.138 ± 0.18	97.096 ± 0.07
BERTurk (uncased, 128 k) (IMST-Pos Tagging)	96.964 ± 0.11	97.060 ± 0.07
BERTurk (uncased, 32 k) (IMST-Pos Tagging)	97.080 ± 0.05	97.088 ± 0.05
BERTurk (cased, 128 k) (BOUN-Pos Tagging)	90.828 ± 0.71	91.016 ± 0.60
BERTurk (cased, 32 k) (BOUN-Pos Tagging)	91.460 ± 0.10	91.490 ± 0.10
BERTurk (uncased, 128 k) (BOUN-Pos Tagging)	91.010 ± 0.15	91.286 ± 0.09
BERTurk (uncased, 32 k) (BOUN-Pos Tagging)	91.322 ± 0.19	91.544 ± 0.09
BERTurk (cased, 128 k) (XTREME Benchmark-NER)	93.796 ± 0.07	93.8960 ± 0.16
BERTurk (cased, 32 k) (XTREME Benchmark-NER)	93.470 ± 0.11	93.4706 ± 0.09
BERTurk (uncased, 128 k) (XTREME Benchmark-NER)	93.604 ± 0.12	93.4686 ± 0.08
BERTurk (uncased, 32 k) (XTREME Benchmark-NER)	92.962 ± 0.08	92.9086 ± 0.14

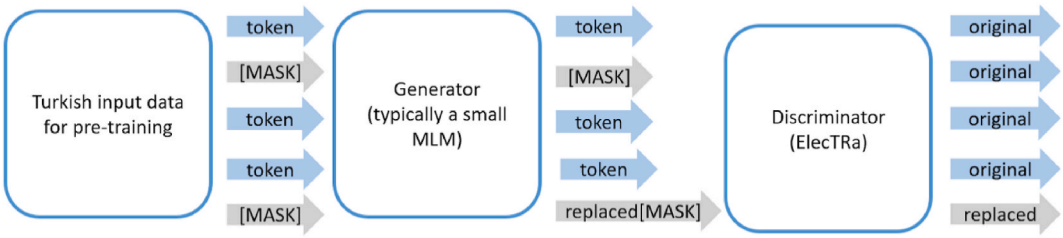

Fig. 2. The architecture of ElecTRa.

Table 2
Accuracy values of DistilBERTurk models.

DistilBERTurk pre-trained model	Development Accuracy	Test Accuracy
DistilBERTurk (IMST-Pos Tagging)	96.362 ± 0.05	96.560 ± 0.05
DistilBERTurk (BOUN-Pos Tagging)	91.166 ± 0.10	91.044 ± 0.09
DistilBERTurk (XTREME Benchmark-NER)	92.012 ± 0.09	91.5966 ± 0.06

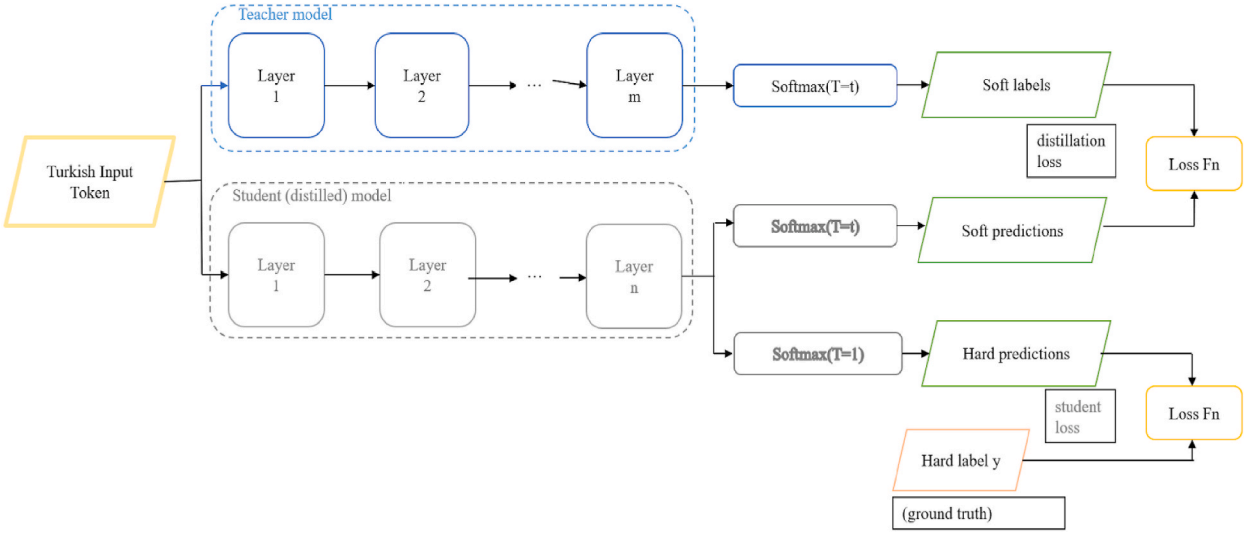

Fig. 3. The architecture of the DistilBERTurk.

(NER).

ConvBERT: CovnBERT is a pre-trained model based on the new propagation-based dynamic convolution module that will replace this self-attention to directly model local dependencies specific to the BERT [25,26]. It can be seen as a slightly more advanced version of ELECTRA. It preserves ELECTRA's modified token detection task and offers a reduced parameter size and a much faster pre-training process to optimize the attention mechanism. Also, the modified icon detection task is pre-trained from ELECTRA using 32 GB of text data. It also learns global dependencies and dynamic convolution local dependencies, but the same word is generated at the same convolution kernel level, so it doesn't work very well. Therefore, refinement has been made here to create a convolution kernel so that range dependencies are well exploited while also guaranteeing different meanings of words. Hybrid attention combines the self-attention module and the intermittent dynamic convolution module. Here, the number of attentions is also reduced, the computational cost of the attention mechanism is reduced, and the self-attention mechanism is forced to learn more useful information. Instead of a fully connected layer, the whole connection grouping method is used to reduce parameters, making it more efficient. Fine-tuning an unsupervised, pre-trained transformer model has become a task in a variety of Natural language processing applications. However, the pre-training phase is slow and costly. ConvBERT reduces training costs by four times while achieving better performance. In short, ConvBERT can outperform ELECTRA with less than 1/4 training cost.

ConvBERTurk: BERTurk has a disadvantage in terms of memory and cost due to its structure based on self-attention block. Because of this situation, a pre-trained language model, CovnBERTurk, is proposed to directly model local dependencies. ConvBERTurk is a cased ConvBERT model for Turkish. The model is pre-trained at 512 sequence lengths for 1 M steps. Convolution layers were used as a complement to self-attention in the pre-training phase, as the extraction of local features increased the convolution success. It reduces

the number of attention heads by projecting input directly onto a smaller docking area for self-attention and diffusion-based dynamic convulsions. ConvBERTurk, a Turkish language version, is trained on a 32 K word capitalized version for Turkish. Fig. 4 shows the architecture of the ConvBERTurk pre-trained language model. Fig. 4 shows the architecture of the ConvBERTurk.

The Turkish part of the model multilingual dataset has also been studied as ConvBERTurk mC4 (cased) and ConvBERTurk mC4 (uncased). Table 3 shows the BOUN and IMST dataset for the average accuracy (PoS tagging) and XTREME Turkish dataset for the average F1-Score (NER) of the models.

Table 4 shows the hyperparameters of all pre-trained language models used during the training process.

2.3. Transformers

Transformers, just like recurrent neural networks (RNNs), are designed to process sequential data such as translation and text summarization, and natural language. RNNs require sequential processing of sequential data, whereas transformers do not require sequential processing of sequential data [27,28]. In the Transformer structure, the Transformer does not need to process the beginning before the end when the input data is a natural language sentence. Because of this feature, Transformer allows much more parallelization than RNNs and therefore reduces training times. Moreover, the transformers are designed around the concept of attentional mechanism designed to help memorize long source sentences in neural machine translation. Fig. 5 shows the architecture of the transformer [29]. The transformer architecture is based on an encoder-decoder. Encoders consist of a series of encoding layers that iteratively process the input one after the other, while decoders consist of a series of decoding layers that do the same to the encoder's output. The transformer is embedded when a sentence is transmitted to a converter and transmitted to an encoder stack. The output from the last encoder is then forwarded to each decoder block in the decoder stack. The decoder stack then produces the output. While all the encoder blocks in the transformer are the same, so are all the decoder blocks in the transformer.

2.4. Autotrain

Automated Machine Learning (AutoML) is a term used to automate data cleaning, model, and hyperparameter selection. Since model selection and hyperparameter optimization is a difficult and time-consuming process, fully automating this with Autotrain provides great convenience in NLP applications involving different languages. Autotrain supports binary and multiclass text classification, token classification, question-answering, text summarization, and text scoring. Autotrain supports English, German, French, Spanish, Swedish, Hindi, and many other languages. Autotrain automatically carries out all the steps of machine learning algorithms from the preprocessing step to the processing of the model to achieve optimum performance on a data set [30,31]. Hyperparameter optimization is also an important problem in machine learning methods that affects the learning process [32]. Autotrain also helps to reduce the cost of the process by eliminating the hyperparameter optimization problem. While the Autotrain concept continues to be developed, it is seen that it produces impressive results when the examples in the literature are examined [33–35]. Fig. 5 shows the pipeline of a classic Autotrain process.

In a classic Autotrain pipeline, there is a data preprocessing stage where basic operations such as missing data, extreme values, and scaling are performed, and the data is separated from the noise. In the second stage, the preprocessing process of the features is carried out. In machine learning methods, which are designed to best represent the real world, the increase in the number of attributes creates a more complex model. As the feature size increases, feature extraction is needed. This point is also included in Autotrain, and the features on which the model should be built are revealed in the feature preprocessing step. The last step is to classify the features by creating a model that learns from the data in line with the relevant features.

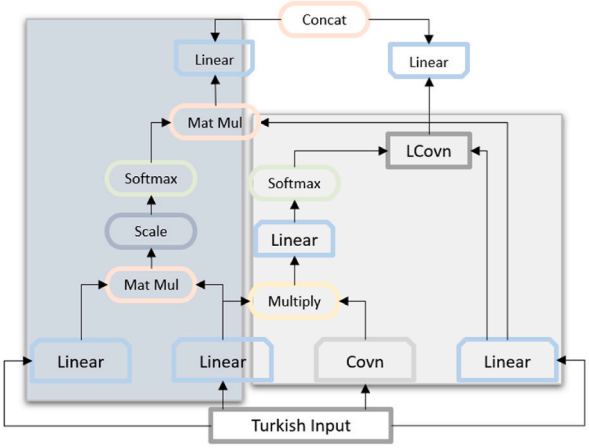

Fig. 4. The architecture of the ConvBERTurk.

Table 3
Accuracy values of ConvBERTurk models.

CovnBERTurk pre-trained model	Development Accuracy	Test Accuracy
ConvBERTurk (IMST-Pos Tagging)	97.208 ± 0.10	97.346 ± 0.07
ConvBERTurk (BOUN-Pos Tagging)	91.250 ± 0.14	91.524 ± 0.07
ConvBERTurk mC4 (cased) (IMST-Pos Tagging)	97.148 ± 0.07	97.426 ± 0.03
ConvBERTurk mC4 (uncased) (IMST-Pos Tagging)	97.308 ± 0.09	97.338 ± 0.08
ConvBERTurk mC4 (cased) (BOUN-Pos Tagging)	91.552 ± 0.10	91.724 ± 0.07
ConvBERTurk mC4 (uncased) (BOUN-Pos Tagging)	91.202 ± 0.16	91.484 ± 0.12
ConvBERTurk (XTREME Benchmark-NER)	93.822 ± 0.14	93.9286 ± 0.07
ConvBERTurk mC4 (cased) (XTREME Benchmark-NER)	93.778 ± 0.15	93.6426 ± 0.15
ConvBERTurk mC4 (uncased) (XTREME Benchmark-NER)	93.586 ± 0.07	93.6206 ± 0.13

Table 4
The hyper parameters of the pre-trained language models.

	Hidden Layer Activation	Hidden Size	Intermediate size	Max position embeddings	Hidden layers	Attention heads	Learning Rate	Vocab_size
ElecTRa Small (cased)	gelu	64	256	512	12	1	1e-12	32,000
ElecTRa Base (cased)	gelu	256	1024	512	12	4	1e-12	32,000
ElecTRa Base mC4 (cased)	gelu	256	1024	512	12	4	1e-12	32,000
ElecTRa Base mC4 (uncased)	gelu	256	1024	512	12	4	1e-12	32,000
BERTurk (cased, 32 k)	gelu	768	3072	512	12	12	1e-12	32,000
BERTurk (uncased, 32 k)	gelu	768	3072	512	12	12	1e-12	32,000
BERTurk (cased, 128 k)	gelu	768	3072	512	12	12	1e-12	128,000
BERTurk (uncased, 128 k)	gelu	768	3072	512	12	12	1e-12	128,000
DistilBERTurk (cased)	gelu	768	None	512	6	12	None	32,000
ConvBERTurk (cased)	gelu	768		512	12	12	1e-12	32,000
ConvBERTurk mC4 (cased)	gelu	768	3072	512	12	12	1e-12	32,000
ConvBERTurk mC4 (uncased)	gelu	768	3072	512	12	12	1e-12	32,000

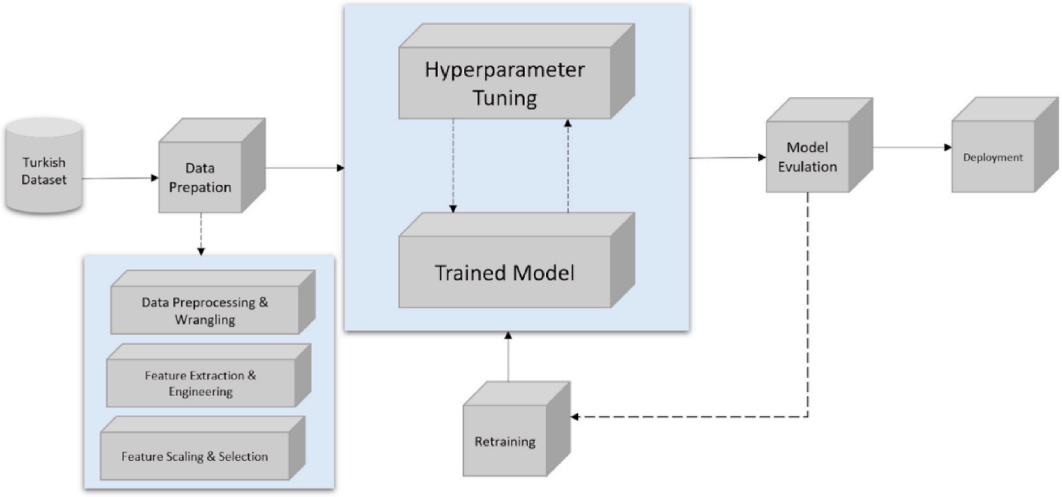

Fig. 5. Linear pipeline of the classic Autotrain.

2.5. Next sentence prediction

The Next Sentence Prediction (NSP) function is defined in natural language processing applications for understanding the relationships between sentences in a corpus and for a binary classification function. For example, given two sentences X and Y, the NSP

determines whether sentence Y is an immediate succession of sentence X in the original document. The NSP function is placed in the [CLS] token with the output IsNext (positive input) or NotNext (negative output). Negative and positive data sets are trained at equal 50% rates. In short, natural language processing is performed with masked input. The output mask is an estimate of the tokens. Adding bi-directionality to the process results in more accurate results. Also, with NSP, the machine understands natural input sentences better. With MLM, the model can learn the statistical properties of word sequences. The model is used in situations where it has to guess one or more words using other words presented in a sentence [36].

2.6. Masked language modeling

Masked language modeling (MLM) is one of the applications of natural language processing that performs the prediction of hidden words. With MLM, a certain percentage of words in a sentence are masked. The model is then expected to predict these masked words based on other words in that sentence. As in NSP, there is bi-directionality learning in MLM because the representation of masked words is learned based on both the right and left words. This process can be thought of as the problem of placing the appropriate word in the blank in fill-in-the-blank questions. With MLM, the model can learn the statistical properties of word sequences. The model is used in situations where it has to guess one or more words using other words presented in a sentence [37]. Fig. 6 shows a structure of the MLM.

3. Methodology and implementation

At this section, the details of the stages of the experiment which contain the architecture of the pre-trained language models, the used dataset, preprocessing, training and text classification are given. Fig. 7 shows the flowchart of the implementation.

3.1. Data description and preprocessing

Data description and preprocessing is the process of preparing raw data and fitting it into a machine learning model. When creating a machine learning system, it is always necessary to work with clean and formatted data.

Data collection: Our dataset was created from customer product reviews on different e-commerce sites in Turkish. The size of the created data set consists of 250 K data. The dataset is shared on Github: https://github.com/BihterDass/Turkish-Dataset.

Data cleaning: At this stage, records with corrupt or invalid values and records with missing columns are removed from the raw data. Turkish letters, punctuation marks, emojis, similar lines have been cleaned. Stop words are not cleared because labeling will be performed for the dataset.

Data formatting: After the data set splitting process, 2 columns as text and multi-label were created for train/test/validation sets in. csv format.

Data labeling: Since the cleaned dataset is used for multi-text classification, which is one of the natural language processing tasks, the labeling process has been carried out. The dataset is divided into 3 classes, namely negative (0), normal (1) and positive (2). Three dictionaries were created for the labeling process. Clustering classes belonging to positive words, normal words, and negative words were created in the dictionary, and labeling was carried out through these dictionaries. Table 5 is given as an example of part of the dictionary. The data were created in 3 classes and made ready to be used in pre-trained models. Table 6 shows detail the corresponding description and class.

Data splitting: After labeling, the data set is divided into training, evaluation, and test sets. Clustering is set to 80% (train)/10% (test)/10% (validation).

4. Experimental results and discussion

In this study, the performance of pre-trained language models in multi-class text classification from the Turkish dataset using Autotrain was compared. In order to accurately evaluate the effectiveness of language models in Autotrain, 15 times of training were

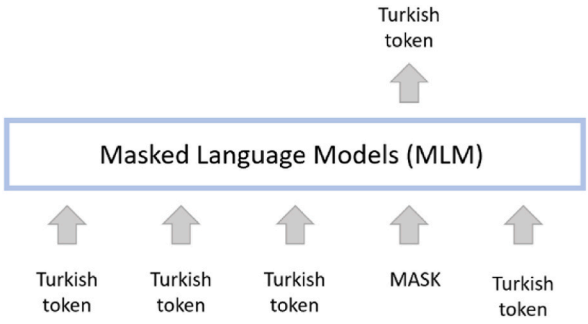

Fig. 6. The structure of MLM.

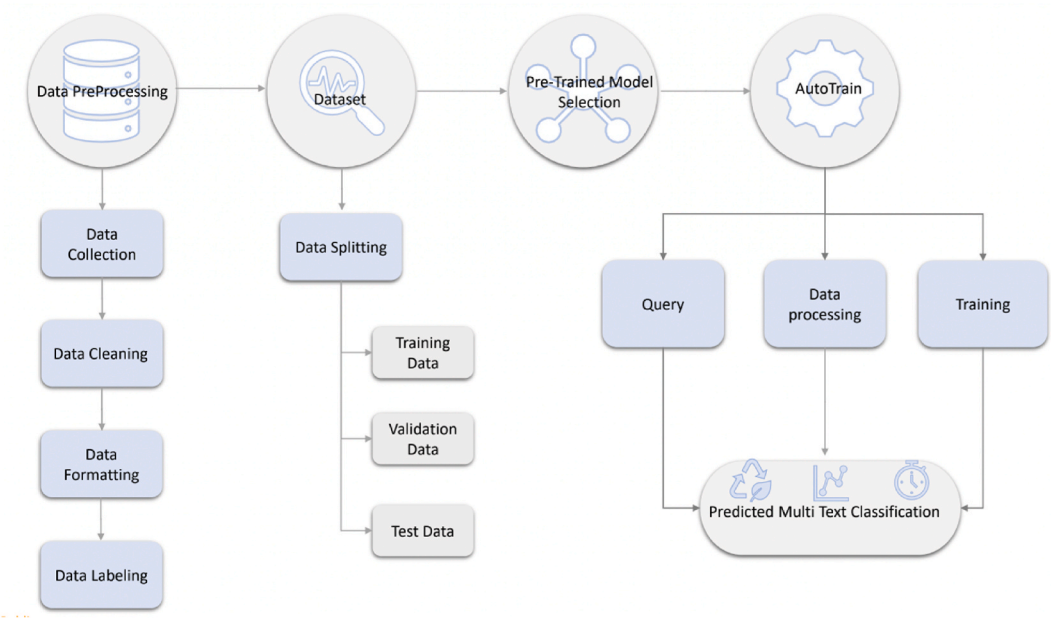

Fig. 7. The flowchart of the implementation.

Table 5
An example of part of the dictionary.

Negative	Neutral	Positive
"kötü/dad"	"fakat/but"	"bayıldım/loved it"
"asla/never"	"iyi gibi/like good"	"mükemmel/excellent"
"korkunç/terrible"	"idare eder/hands down"	"teşekkürler/thanks"
"iyi değil/not good"	"hızlı fakat/quick but"	"çok begendim/like it"
"Memnun değilim/not satisfied"	"kötü değil/not bad"	"tavsiye ederim/recommend"

Table 6
The structure of Dataset.

Text (string)	Class label
kaliteli bekledigimden iyi geldi guzel	2 (pos)
bu magaza kadar kotu bir magaza gormedim	0 (neg)
cok iyi bir urun ama kirik geldi	1 (neu)
urun gayet guzel gonul rahatligiyla alabilirsiniz	2 (pos)
ofis icin aldik oldukca kullanisli	2 (pos)
Urun guzel ama kargosu bekledigimiz zamanda gelmedi erkan kucuk	1 (neu)
ilk defa elektronik bir urun aldim cok iyi bir ekrani var	2 (pos)
urun hemen kargoya verildi ancak hafta icinde teslim edilmedi henuz kurulum yapilmadi	1 (neu)

conducted. Sample observation sections known as validation sets were created from the training data set, and the performance of the language models was measured against each new sample observation section. Table 7 lists the auto-train process results, time, CO2 emission, and accuracy values of pre-trained language models using 15 times different hyperparameters for training (learning rate, maximum length, batch size, etc.). CO2 emission is how much CO2 the model produces, taking into account computer hardware, location, usage, and training time. First, it is based on the carbon intensity of the power grid used for training. This is the amount of CO2 produced by the KwH electricity used. The carbon density depends on the location of the equipment and the total energy used at that location. The more renewable energy is used for training, the less carbon-intensive. If power consumption and carbon intensity are multiplied by the training time of the model, an estimate is made for CO2 emissions. The estimate is not an exact number because there are other factors that will increase carbon emissions, such as the energy used for data center heating and cooling.

The reason for doing 15 different training processes in the Autotrain process is to interpret the results obtained from the training at an optimum level. Because when the language models are evaluated in terms of CO2 emissions, training results would be insufficient if 5 training processes were performed. If more than 15 trains were performed, it would drag the system to overtraining and prevent an accurate evaluation. Data cleaning, model selection, and hyperparameter optimization steps are all fully automated in the Autotrain

Table 7
Results of pre-trained language models for 15 different training.

Pre-trained Models		1.Tr	2.Tr	3.Tr	4.Tr	5.Tr	6.Tr	7.Tr	8.Tr	9.Tr	10.Tr	11.Tr	12.Tr	13.Tr	14.Tr	15.Tr
M-1	Min.	26	31	39	20	20	32	33	22	22	29	30	25	19	25	25
	CO_2	40.5	50.7	0.28	0.13	0.13	0.21	52.0	36.8	37.0	0.20	45.6	0.16	0.11	34.2	0.15
	ACC	.87	.87	.86	.85	.85	.85	.85	.85	.85	.84	.84	.82	.82	.82	.81
M-2	Min.	89	102	57	45	45	45	45	89	78	33	70	34	81	72	81
	CO_2	0.54	119	120	87.5	91.8	0.40	0.41	92.8	0.41	62.1	0.29	0.30	0.30	0.22	72.8
	ACC	.87	.87	.87	.86	.86	.86	.86	.86	.86	.82	.82	.82	.39	.39	.39
M-3	Min.	57	69	69	46	46	45	44	45	46	33	35	34	37	27	37
	CO_2	120	0.63	115	85.3	94.9	89.3	0.42	0.42	89.8	64.2	60.2	64.8	0.32	49.5	67.9
	ACC	.88	.88	.87	.86	.86	.86	.86	.86	.86	.82	.82	.82	.39	.39	.39
M-4	Min.	69	59	57	45	45	45	45	63	63	33	35	41	28	37	37
	CO_2	0.64	119	115.	91.5	93.3	0.40	0.42	128	128	61.0	61.8	81.1	0.22	70.9	73.2
	ACC	.88	.88	.87	.86	.86	.86	.86	.86	.86	.82	.82	.82	.39	.38	.38
M-5	Min.	70	58	57	44	45	45	58	45	46	33	33	34	37	38	37
	CO_2	93.5	121	118.9	0.41	88.6	93.5	50.8	0.40	0.40	61.6	0.29	0.30	70.7	47.7	72.5
	ACC	.87	.87	.87	.86	.86	.86	.86	.86	.86	.82	.82	.82	.39	.39	.39
M-6	Min.	47	69	59	36	36	37	36	37	38	33	28	34	38	28	38
	CO_2	97.6	83.9	0.55	74.4	0.33	74.7	73.6	0.30	73.5	0.29	0.25	65.1	71.9	49.4	74.4
	ACC	.88	.87	.87	.86	.86	.86	.86	.86	.86	.82	.82	.82	.39	.39	.39
M-7	Min.	66	66	67	52	53	41	45	53	52	41	41	41	63	58	63
	CO_2	136	135	0.63	108	110	84.3	83.1	0.49	0.48	0.35	82.2	80.7	0.55	0.50	0.56
	ACC	.88	.88	.88	.87	.87	.87	.87	.87	.87	.84	.84	.84	.38	.38	.38
M-8	Min.	66	79	66	64	64	54	54	54	54	41	42	43	64	64	50
	CO_2	0.61	0.72	0.61	0.58	0.60	0.49	0.49	113	107	77.8	0.37	0.36	125	125	94.4
	ACC	.88	.88	.88	.87	.87	.87	.87	.87	.87	.85	.85	.84	.39	.39	.38
M-9	Min.	37	37	44	24	24	30	30	39	40	22	23	30	27	21	21
	CO_2	69.9	0.32	0.37	47.5	46.3	0.24	55.4	0.31	0.31	39.7	37.6	0.23	0.21	0.16	0.15
	ACC	.87	.87	.87	.86	.86	.86	.86	.86	.85	.82	.82	.81	.80	.55	.55
M-10	Min.	91	76	77	61	60	63	60	62	60	46	47	48	49	39	49
	CO_2	194	166	169	127	0.58	124	124	0.57	123	0.43	91.1	96.2	0.42	72.2	97.7
	ACC	.88	.87	.87	.86	.86	.86	.86	.86	.86	.82	.82	.82	.53	.53	.53
M-11	Min.	77	76	91	61	61	60	72	73	61	47	47	47	52	39	49
	CO_2	168	0.72	196	0.57	125	127	149	152	0.55	0.42	93.5	93.2	101	75.3	95.7
	ACC	.88	.88	.88	.86	.86	.86	.86	.86	.86	.82	.82	.82	.53	.53	.39
M-12	Min.	77	76	76	66	66	66	66	73	62	47	48	49	39	50	49
	CO_2	0.75	161	167	122	0.57	129	0.56	152	125.	0.42	95.3	93.9	70.3	101.	97.0
	ACC	.88	.88	.88	.86	.86	.86	.86	.86	.86	.83	.83	.83	.53	.53	.39

Tr: Train, 1.Tr:First train, 2.Tr:Second train etc. M-1:Model 1, Min:Minutes, CO_2: CO_2 Emission, ACC: Accuracy.

process. The Autotrain process uses a principle known as transfer learning, which makes the process much simpler. Using the pre-trained language models for multi-class text classification on the Turkish dataset, it has been transformed into a fine-tuned model ready for distribution.

Table 8 shows the best performance results for multi-text classification of pre-trained models in Turkish dataset. In the 250 K data set, the train values were first queued, and after the pre-processing step, the training process was carried out. As can be seen from Table 8, the BERTurk (uncased, 128 k) pre-trained model performed better than the other models as a result of auto-training. In terms of training time, BERTurk (uncased, 32 k) is the model that produces the fastest results among BERTurk pre-trained models. Almost similar performance values were obtained for the results obtained in ElecTRa models trained with both generator and discriminator structures. However, if a fast training is desired, ElecTRa Small (cased) can definitely be preferred. The table shows that we got the best Accuracy value in the ElecTRa Base mC4 (cased) model, one of the ElecTRa pre-trained models, but it had a bad result in the ML models in terms of CO2 Emission. The DistilBERTurk (cased) model performed as well as the BERTurk and other pre-trained models. ConvBERTurk mC4 (uncased), one of the span-based dynamic convolution models, has a higher accuracy than other ConvBERTurk models and has achieved very good performance in terms of CO2 emission.

The difference between the cased and uncased models is the use of capitalization and the presence of diacritics in the Turkish text. Therefore, the training of models with and without a case is also different. In the uncased models, emphasis marks in the text are removed, for case models, emphasis marks are preserved. Emphasis marks are the marks on the letters, which are usually used in the Latin language. Since these models have different training, their CO2 emissions are also different. In addition, full-word masking is used in uncased models, while the original text is used in cased models.

Since machine learning models are generally energy intensive in education, these models can produce a significant carbon footprint. Therefore, it is important to monitor the CO2 emissions of the models in order to have an idea about the environmental impact of these processes. While these models improve accuracy in many NLP tasks, they also draw attention to large computational resources that require significant energy consumption. Training and developing these models is costly, both financially due to hardware, electrical, and cloud computing time, and environmentally due to the carbon footprint required to fuel modern tensor processing hardware. A100 PCIe 40/80 GB Hardware, Google Cloud Platform Provider, and Asia-east1 Region of Compute were used to calculate the carbon footprint of the models in the study. The carbon emission formula generated according to the local power grid is shown in

Table 8
The performance results of pre-trained language models for multi-text classification.

Pre-trained Models	Loss	Accuracy	Precision	Recall	F1 score	Time	CO_2 Emission
ElecTRa Small (cased)	.30	.8630	.87	.87	.87	26 *m*	40.50
ElecTRa Base (cased)	.27	.8749	.87	.89	.88	89 *m*	0.54
ElecTRa Base mC4 (cased)	.27	.8765	.88	.88	.88	57 *m*	119.94
ElecTRa Base mC4 (uncased)	.26	.8713	.88	.89	.88	69 *m*	0.64
BERTurk (cased, 32 k)	.26	.8748	.87	.88	.88	69 *m*	93.50
BERTurk (uncased, 32 k)	.26	.8759	.88	.89	.88	47 *m*	97.58
BERTurk (cased, 128 k)	.27	.8807	.88	.89	.88	66 *m*	135.60
BERTurk (uncased, 128 k)	**.25**	**.8809**	**.88**	**.89**	**.88**	**66 *m***	**0.61**
DistilBERTurk (cased)	.27	.8742	.88	.88	.88	37 *m*	69.89
ConvBERTurk (cased)	.27	.8755	.89	.88	.88	91 *m*	193.80
ConvBERTurk mC4 (cased)	.26	.8763	.88	.89	.88	77 *m*	168.43
ConvBERTurk mC4 (uncased)	**.26**	**.8777**	**.88**	**.88**	**.88**	**77 *m***	**0.75**

Equation (1).

Power consumption x Time x Carbon (1)

In Table 8, the carbon emissions obtained from the models are related to the environment, provider, hardware type and times in which these models are trained. The reason why the models produce different CO2 emissions is that the training times of the models are different according to different hyper parameters.

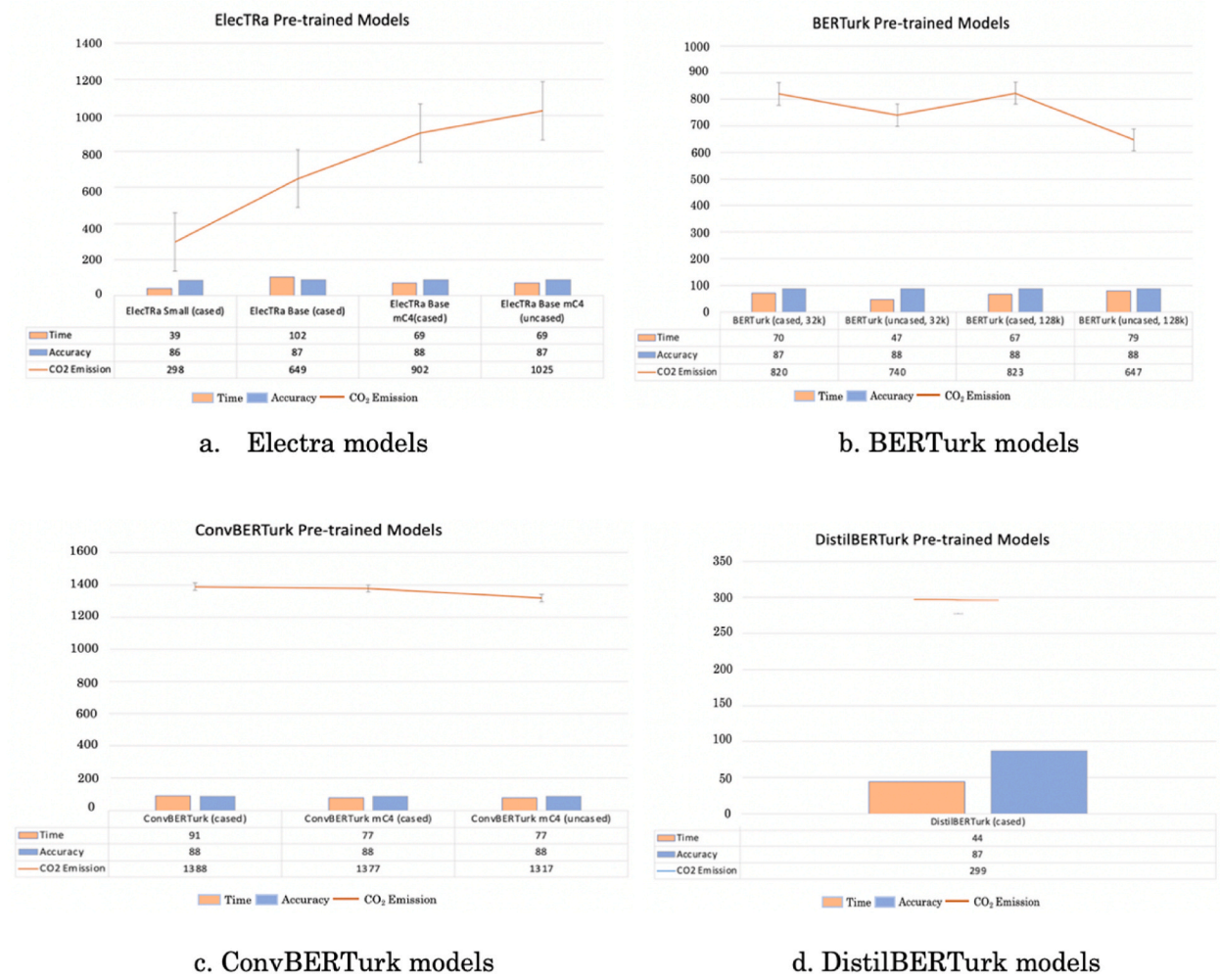

Fig. 8. Accuracy, time and CO2 emission values of pre-trained language model families for multi text classification.

Fig. 8(a–d) shows the accuracy values, training time, and CO2 Emission values of ElecTRa, BERTurk, ConvBERTurk, and DistilBERTurk models for multi-text classification, respectively. The CO2 Emission value is equal to the total value obtained on 15 training processes. For the ElecTRa language family, the ElecTRa Small (cased) model showed the best performance in terms of CO2 Emission during the 15-time training process on different hyperparameters. The ElecTRa Base mC4 (uncased) model has the worst result for CO2 Emission. For the BERTurk language family, the BERTurk (uncased, 128 k) model performed well for CO2 Emission over all training periods. In addition, the BERTurk (uncased, 32 k) model produced a good result for training time and CO2 emissions. The ConvBERTurk family has achieved a better result than other language models for CO2 emission during the train process. The ConvBERTurk mC4 (cased) is the best performing model in this language family. For the DistilBERTurk language family, it has achieved better performance than other models in terms of CO2 emission, accuracy and time during the train process.

5. Conclusion and future works

This study aims to find the best NLP model for the triple Turkish text classification task and to show the configuration of the best model. For this purpose, we evaluate the performance of the models in multi-text classification on the 250 K Turkish dataset we created. The performances of all language models trained for the Turkish language, such as BERTurk, CovBERTurk, ElecTRA, and DistilBERTurk, were compared in terms of training time, accuracy, CO2 emissions, and cased and uncased in three-class text classification using Autotrain. We also tried to show the factors affecting CO2 emission and accuracy in this study. We have seen that the hyperparameters used in the pre-trained language models change both the training time and accuracy as well as the CO2 emission even if a data set of the same size is used. Experimental results have shown that the BERTurk (uncased, 128 k) pre-trained model performs best as a result of auto-training, with high accuracy and low CO2 emissions compared to other models. This is because BERT uses a transformer, an attention mechanism that learns the contextual relationships between words or subwords in a text. BERTurk (cased) is better than a BERTurk (uncased) in most applications, except for applications where text state information is important. Although AutoML models perform training in a shorter time with the technical features they offer, we cannot see the batch size, number of epoch and which epoch gives the highest accuracy value. We also don't know how the performance of the models and CO2 emissions may vary on larger datasets and in different languages. The lack of clarity of these can be shown as the limitations of the study. In future studies, named entity recognition (NER), sentiment analysis and multi-class text classification applications of these models will be carried out in order to see the performance of these models in different NLP applications and how they will facilitate training times.

Author contribution statement

Pinar Savci: Conceived and designed the experiments; Performed the experiments; Wrote the paper.
Bihter Das: Performed the experiments; Analyzed and interpreted the data; Contributed reagents, materials, analysis tools or data.

Data availability statement

Additional information for this paper can be found at: https://github.com/BihterDass/Turkish-Dataset.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper

References

[1] L. Li, J. Geissinger, W.A. Ingram, E.A. Fox, Teaching Natural Language Processing through big data text summarization with problem-based learning, Data and Inform. Manag. 4 (1) (Mar. 2020) 18–43, https://doi.org/10.2478/dim-2020-0003.
[2] P. Fortuna, S. Nunes, A survey on automatic detection of hate speech in text, ACM Comput. Surv. 51 (2018) 85:1–85:30.
[3] D.S. Moirangthem, M. Lee, Hierarchical and lateral multiple timescales gated recurrent units with pre-trained encoder for long text classification, Expert Syst. Appl. 165 (Mar. 2021) 113898, https://doi.org/10.1016/j.eswa.2020.113898.
[4] E. Çelik, T. Dalyan, Unified benchmark for zero-shot Turkish text classification, Inf. Process. Manag. 60 (3) (2023), https://doi.org/10.1016/j.ipm.2023.103298.
[5] M. Aydoğan, A. Karci, Improving the accuracy using pre-trained word embeddings on deep neural networks for Turkish text classification, Phys. Stat. Mech. Appl. 541 (Mar. 2020) 123288, https://doi.org/10.1016/j.physa.2019.123288.
[6] S. Yıldırım, T. Yıldız, A comparative analysis of text classification for Turkish language, Pamukkale Univ. J. Engin. Sci. 24 (5) (2018) 879–886, https://doi.org/10.5505/pajes.2018.15931.
[7] R. Velioğlu, T. Yıldız, S. Yıldırım, Sentiment analysis using learning approaches over emojis for Turkish tweets, in: 2018 3rd International Conference on Computer Science and Engineering (UBMK), Sep. 2018, pp. 303–307, https://doi.org/10.1109/UBMK.2018.8566260.
[8] D. Kılınç, A. Özçift, F. Bozyigit, P. Yıldırım, F. Yücalar, E. Borandag, TTC-3600: a new benchmark dataset for Turkish text categorization, J. Inf. Sci. 43 (2) (2015) 174–185.
[9] Y. Cheng, et al., HSAN-capsule: a novel text classification model, Neurocomputing 489 (Jun. 2022) 521–533, https://doi.org/10.1016/j.neucom.2021.12.064.
[10] H. Ren, H. Lu, Compositional coding capsule network with k-means routing for text classification, Pattern Recogn. Lett. 160 (Aug. 2022) 1–8, https://doi.org/10.1016/j.patrec.2022.05.028.
[11] A.S. Gertner, J. Henderson, E. Merkhofer, A. Marsh, B. Wellner, G. Zarrella, MITRE at SemEval-2019 task 5: transfer learning for multilingual hate speech detection, in: Proceedings of the 13th International Workshop on Semantic Evaluation, 2019, pp. 453–459.
[12] F.M. Plaza-del-Arco, M.D. Molina-González, L.A. Ureña-López, M.T. Martín-Valdivia, Comparing pre-trained language models for Spanish hate speech detection, Expert Syst. Appl. 166 (Mar. 2021) 114120, https://doi.org/10.1016/j.eswa.2020.114120.

[13] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018 arXiv preprint arXiv: 1810.04805.
[14] D.S. Moirangthem, M. Lee, Hierarchical and lateral multiple timescales gated recurrent units with pre-trained encoder for long text classification, Expert Syst. Appl. 165 (Mar. 2021) 113898, https://doi.org/10.1016/j.eswa.2020.113898.
[15] Á.J.J. Palenzuela, F. Frasincar, M.M. Truşcă, Modeling Second Language Acquisition with pre-trained neural language models, Expert Syst. Appl. 207 (Nov. 2022) 117871, https://doi.org/10.1016/j.eswa.2022.117871.
[16] C.P.M. Zhang, et al., A large-scale generative Chinese Pre-trained language model, AI Open 2 (2021) 93–99, https://doi.org/10.1016/j.aiopen.2021.07.001.
[17] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, et al., Aligning Books and Movies: towards Story-like Visual Explanations by Watching Movies and Reading Books IEEE International Conference on Computer Vision (ICCV 2015), IEEE, 2015, pp. 19–27, https://doi.org/10.1109/ICCV.2015.11.
[18] M. Peters, S. Ruder, N.A. Smith, To Tune or Not to Tune? Adapting Pretrained Representations to Diverse Tasks 4th Workshop on Representation Learning for NLP (RepL4NLP 2019), ACL, 2019, pp. 7–14, https://doi.org/10.18653/v1/W19-4302.
[19] B. Settles, Data for the 2018 Duolingo Shared Task on Second Language Acquisition Modeling (SLAM), 2018, https://doi.org/10.7910/DVN/8SWHNO.
[20] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, arXiv, May 24, 2019. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/1810.04805.
[21] Kemal Oflazer, Two-level description of Turkish morphology, Lit. Ling. Comput. 9 (2) (1994) 137–148.
[22] F.B. Fikri, K. Oflazer, B. Yanıkoğlu, Turkish dataset for semantic textual similarity, in: 2021 29th Signal Processing and Communications Applications Conference (SIU), 2021, pp. 1–4, https://doi.org/10.1109/SIU53274.2021.9477982.
[23] K. Clark, M.-T. Luong, Q.V. Le, C.D. Manning, ELECTRA: Pre-training Text Encoders as Discriminators rather than Generators, arXiv, Mar. 23, 2020. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/2003.10555.
[24] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, arXiv, Feb. 29, http://arxiv.org/abs/1910.01108, 2020.
[25] Z. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, S. Yan, ConvBERT: Improving BERT with Span-Based Dynamic Convolution, arXiv, Feb. 02, 2021. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/2008.02496.
[26] A. Conneau, et al., Unsupervised Cross-Lingual Representation Learning at Scale, 2020, https://doi.org/10.48550/arXiv.1911.02116 arXiv, Apr. 07.
[27] Łukasz Kaiser, Ilya Sutskever, Neural GPUs learn algorithms, in: International Conference on Learning Representations (ICLR), 2016.
[28] Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, Koray Kavukcuoglu, Neural Machine Translation in Linear Time, 2017 arXiv preprint arXiv:1610.10099vol. 2.
[29] A. Vaswani, et al., Attention Is All You Need, arXiv, Dec. 05, 2017. Accessed: Aug. 03, 2022. [Online]. Available: http://arxiv.org/abs/1706.03762.
[30] A. Romblay, Automated Machine Learning" Datahack Summit, Anlytics Vidhya, 2017.
[31] S. Ozdemir, S. Orslu, New perspective on machine learning process: AutoML, Journal of Information Systems and Management Researchp 7 (2019).
[32] Y. Bengio, Gradient-based optimization of hyperparameters, Neural Comput. 12 (8) (2000) 1889–1900.
[33] B. Komer, J. Bergstra, C. Eliasmith, Hyperoptsklearn: automatic hyperparameter configuration for scikit-learn, in: ICML Workshop on AutoML, 2014.
[34] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, Efficient and robust automated machine learning, Adv. Neural Inf. Process. Syst. 28 (2015) 2962–2970.
[35] M. Aydogan, V. Kocaman, TRSAv1: a new benchmark dataset for classifying user reviews on Turkish e-commerce websites, J. Inf. Sci. (Feb. 2022), https://doi.org/10.1177/01655515221074328.
[36] W. Shi, V. Demberg, Next sentence prediction helps implicit discourse relation classification within and across domains, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP, Hong Kong, China, Nov, 2019, pp. 5790–5796, https://doi.org/10.18653/v1/D19-1586.
[37] T. Arici, et al., MLIM: Vision-And-Language Model Pre-training with Masked Language and Image Modeling, arXiv, Sep. 24, 2021. Accessed: Aug. 04, 2022. [Online]. Available: http://arxiv.org/abs/2109.12178.