AUTOMATIC PUNCTUATION CORRECTION SYSTEM WITH TURKISH BERT MODELS

Mehmet Şadi Özcan, Doğukan Kahya
Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi, 34220 Istanbul, Türkiye
{sadi.ozcan, dogukan.kahya}@yildiz.edu.tr

***Özetçe* —Türkçe'de noktalama işaretlerinin doğru kullanımı, metnin anlamını doğru bir şekilde ifade etmek ve okuyucunun metni kolayca anlamasını sağlamak için büyük bir öneme sahiptir. Bu nedenle metin düzenleme işlemlerinde metnin tam olarak anlaşılabilmesi ve anlam bütünlüğünün sağlanabilmesi için noktalama işaretlerinin doğru bir şekilde düzeltilmesi ihtiyacı doğmaktadır. Projemizde pre-trained Türkçe BERT modellerinin fine-tune işlemleri yapılarak bu problem çözülmeye çalışılmıştır. Projenin sonucunda, kullanıcıların etkileşime geçebileceği bir web uygulaması da tasarlanmıştır.**

Bu işlemlerin yapılabilmesi için öncelikle kullanıma uygun bir veri seti seçilmiş, bu veri seti çeşitli ön işleme aşamalarından geçirilmiş ve modellerin eğitimi için hazır hale getirilmiştir. Sonrasında modelleri eğitme işlemlerine başlanmıştır. Modellerin eğitimi, elde edilen metrikler doğrultusunda sürekli olarak yeniden şekillendirilmiştir. Fine-tuning işlemleri sonucunda base modelde 0.42'lik bir recall ve 0.43'lük bir F1 skoru ile karşılaşılmıştır. Bu, genel olarak olumlu bir sonuç olarak değerlendirilebilse de modeller hala geliştirilebilmeye açık görünmektedir. Bu iyileştirme işlemleri, veri setinin genişletilmesi veyahut modellerin eğitimi sırasında farklı parametreler kullanılması yoluyla yapılabilir.

Anahtar Kelimeler—Fine-tuning, pre-trained model, BERT, noktalama işaretleri, F1 skoru, recall skoru

***Abstract*—The correct use of punctuation marks in Turkish is of great importance to express the meaning of the text correctly and to ensure that the reader can easily understand the text. For this reason, it is necessary to correct the punctuation marks correctly in text editing processes in order to fully understand the text and to ensure the integrity of meaning. In our project, we tried to solve this problem by fine-tuning the pre-trained Turkish BERT models. As a result of the project, a web application was also designed that users can interact with.**

In order to perform these operations, a suitable dataset was first selected, this dataset was subjected to various preprocessing stages and made ready for training the models. Afterwards, the process of training the models started. The training of the models was continuously reshaped in line with the metrics obtained. As a result of the fine-tuning process, the base model achieved a recall of 0.42 and an F1 score of 0.43. Although this can be considered as a positive result in general, the models still seem to be open for improvement. This could be done by expanding the dataset or by using different parameters during the training of the models.

Keywords—Fine-tuning, pre-trained model, BERT, punctuation marks, F1 score, recall score

I. Introduction

In this project, an Automatic Punctuation Correction System will be implemented with Turkish Bert Models.

A. Objectives and Goals

The main goal of this project is to develop a model that can correct punctuation errors in Turkish texts using Turkish BERT models of different dimensions. Accordingly, the primary objectives of the project are as follows:

- Evaluate the performance of BERT models of different sizes.
- Determine the optimal model size.
- Creating an interactive web interface for users.
- To evaluate the accuracy and speed of the developed model with comparative tests.

B. Methods

The project will be implemented in the following steps:

- Fine-tuning of Turkish BERT models of different sizes at HuggingFace ytu-ce-cosmos will be performed.
- Training and test datasets will be carefully prepared and data pre-processing steps will be defined.
- Extensive tests will be performed on real-world data to measure the performance of the developed model.
- The results obtained will be analyzed and the performance of BERT models of different sizes will be evaluated comparatively.
- Once the optimal model size is determined, an interactive web interface will be developed for users.
- The findings obtained in the project will be compiled into a detailed report to be shared on platforms such as arxiv.

C. Outputs

The outputs of the project will include a comparative analysis of the test results of Turkish BERT models of different sizes, an interactive web interface for users, and a detailed report evaluating the performance and usability

of the model to be shared on arxiv. In the web interface, the user will provide the model with an unpunctuated text in Turkish, and after it passes through the model, the user will be able to view the corrected text.

D. Project Motivation

The main motivation for this project is the need to develop a model that can correct punctuation errors in Turkish texts. Although Turkish is a widely used language around the world, the existing spell checking tools are not sufficient for Turkish texts, which is a problem for many users. Therefore, a more advanced spell checking solution is needed to improve the quality of Turkish texts and enhance communication.

The model to be developed is expected to be an artificial intelligence model that can better understand Turkish language structure and correct errors in Turkish texts more effectively. The usability of this model will help make Turkish texts more understandable and improve the quality of communication. For example, using this model on automatically generated unpunctuated Turkish texts on platforms such as YouTube will make these texts more understandable. Furthermore, the development of such a model will support research and technological advances in the field of natural language processing in Turkish and increase the body of knowledge in this field.

The motivation of this project is to improve the quality of Turkish texts and support technological advances in the field of Turkish natural language processing by developing a more efficient model that can correct spelling errors in Turkish texts.

E. Literature Review

This report presents a preliminary review of a project to develop a model that can correct punctuation and capitalization errors in Turkish texts.

In one of the prominent studies in this field in foreign languages, data was extracted from TEDX talks for English and a model was created to correct punctuation errors. The F1 score of the model was measured as 79.8. In the same study, a model was created for Hungarian language and this model was found to have an F1 score of 82.2.[1]

Another study was conducted in 2022 for the Swedish language, which also showed an F1 score of 78.9.[2]

With these studies in the field, it has been seen that punctuation correction can be done successfully in BERT models.

Various machine learning and natural language processing methods have been used to correct punctuation errors in Turkish texts. However, the existing models generally focus on English texts and there is a limited number of models customized for Turkish texts. In particular, there is a lack of work on developing models that can better understand Turkish language structure and correct punctuation errors in Turkish texts more effectively.

The main aim of this project is to develop a model that can correct punctuation errors in Turkish texts. Writing Turkish texts correctly and clearly is an important factor in communication, and punctuation errors can make it difficult for the reader to understand the text. Existing spell checking tools are often limited and not effective for Turkish texts. The rich but complex morphology of Turkish, the fact that it is a suffixal agglutinative language and the difficulties arising from grammar are the main problems in solving multiple classification problems, which is one of the text classification methods, but with the BERT deep learning technique, this problem has become easier to solve.[3]

In this context, the objectives of the project include evaluating Turkish BERT models of different sizes, determining the most appropriate model size and measuring the performance of the developed model. In line with these objectives, finetuning of BERT models of different sizes will be performed, training and test datasets will be prepared, and extensive tests will be conducted on real-world data.

II. Feasibility

The feasibility study is analyzed under this heading in terms of Technical, Labor and Time, Legal and Economic aspects.

A. Technical Feasibility

1) Software Feasibility: Software Feasibility is examined under 3 separate headings: the operating system we use, the programming language and the IDE.

- **Operating System:** We preferred to use the Windows 10 operating system already installed on our computers where we realized the project.
- **Programming Language:** Python was used as the programming language in the project. The most important reasons for this are the wide range of libraries that Python provides for areas such as natural language processing. Libraries such as HuggingFace's transformers library, numpy, torch and pandas play an important role in our project. HTML, JavaScript and CSS codes were written for the interface of the program. Python's Flask library was used to connect the backend part of the web interface with the frontend part.
- **IDE:** Since large data sets and models will be processed in the project, notebooks on Google Colab were used instead of personal computers in order to perform operations that require large processing capacity such as training faster and more effectively.

2) Hardware Feasibility: For the training of the models, the L4 GPU within the scope of Google Colab Pro was used. For other uses, this system has been used.

Table 1 System that's been used in the project

Hardware	Feature
İşlemci	Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 2592 Mhz
Ekran Kartı	Nvidia GTX1650 Ti 4GB GDDR6
Bellek	16 GB (2x8) GB DDR4
Depolama	512 GB SSD

B. Labor and Time Feasibility

1) Project Developers:

- Developer 1: Mehmet Şadi Özcan (Software Developer)
- Developer 2: Doğukan Kahya (Software Developer)

2) Planning: This is the Gannt Schematic for the project timeline.

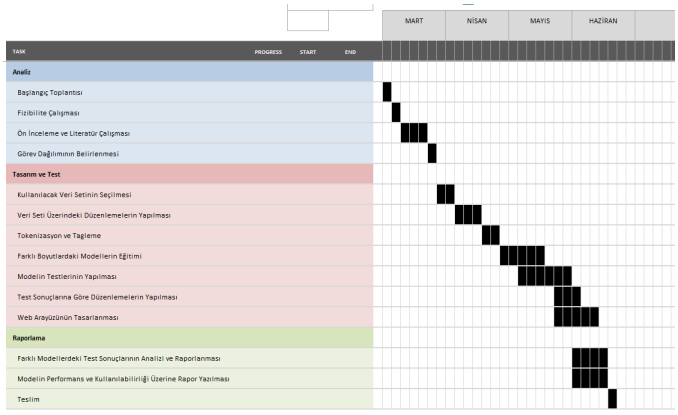

Figure 1 Gannt Schematic

C. Legal Feasibility

In this section, the compliance of the project with the required laws and the necessary license permissions of the programs used in its production will be examined. All rights of this project belong to Yıldız Technical University, Department of Computer Engineering. The software development environment and programming language used in this project are open source.

D. Economic Feasibility

This section specifies the software and hardware costs used in the system. Pricing is based on June 2024. Since all of the software development environments we used in the project are open source, there are no software costs. In terms of hardware, there are the costs of the two computers on which the project was realized and the electricity and internet costs spent during the implementation period of the project. The license of the operating systems on the computers where the project was carried out is also included in the computer costs. Apart from these, an additional expenditure was made for Google Colab Pro membership. All of these costs are included in the table below.

Table 2 Cost table

Expenses	**Price**
Computer	40000 TL
Electricity	1100 TL
Internet	4200 TL
Google Colab Pro	350 TL
Total	**45650 TL**

III. System Analysis

System analysis is usually done early in a project and is used to understand the objectives and requirements of the project. In this section, the requirements of the project are analyzed to form the basis for the system design.

A. Determination of Project Objectives

The main goal of this project is to develop a model that can correct punctuation errors in Turkish texts using Turkish BERT models of different sizes. At the end of the project, the user will be able to interact with the model through a web interface and see the output for the text they want to correct.

B. Definition of Project Requirements

In this section, the main requirements of the project are identified. The following main requirements are aimed to be met for the successful completion of the project:

- **Data Sets**: The datasets to be used in the project must have sufficient amount of data to fulfill the natural language processing tasks. In addition, the data sets should be in a certain format that can be processed by the BERT model. The data set should be suitable for daily Turkish usage. Punctuation marks must be used correctly. The content of the data set should not be focused on a specific topic and should be generalized. Very short sentences in the dataset may restrict the presence of punctuation marks such as commas, so such datasets will not be suitable for the project.
- **Models**: The 5 different models that will be finetuned with the selected dataset were determined at the beginning of the project and are available on the YTU-CE-COSMOS page on HuggingFace.
- **Tools**: The programming language and libraries to be used to perform the specified natural language processing tasks must be chosen correctly. In addition, the languages used to implement the user interface must be chosen carefully and correctly.

C. Performance Requirements

The performance of this entire process needs to be monitored, reported and optimized if necessary. The system needs to work correctly with different data sets from various sources. Since the data set can be changed to improve performance when necessary, the system should be designed to work with different data sets.

IV. System Design

A system design is planned in accordance with the requirements described in the previous section. The details of this design will be analyzed in the following section.

A. Software Design

In general terms, the following software design will be used in this project. As a first step, a dataset that meets the necessary requirements will be selected, in case of errors and deficiencies in the dataset, these deficiencies will be corrected and cleaning operations will be performed on the dataset. After tokenization and tagging, finetuning of Turkish BERT models of different sizes will be performed. This will make the models suitable for the punctuation correction task. In the next stage, a web interface will be developed for users to use the project interactively. From this interface, users will be able to enter text and see the corrected text. During the implementation of these stages, the speed of the model will be increased and the model will run more efficiently through continuous testing. In this way, the model will be able to process the data faster and produce the processed texts faster.

Initially, data sets were created and organized. These operations were implemented on Python. Then training operations were performed. Training was done with transformers library. One of the biggest problems in data sets in such projects is the class imbalance in the data set due to the fact that the number of words that are not followed by punctuation marks in a sentence, regardless of the language, is always higher than the others.[4] Since this problem would affect the training of the model, we searched for the function that would train the model most accurately among different loss functions and decided on the Focal Loss function as a result of the experiments. Apart from this, weights were initially assigned for each class and the weight of the NONE class was reduced to prevent it from affecting the model.

The Focal Loss function used in the training part is a function used when working with unbalanced data sets.[5] It differs from the widely used Cross-Entropy function in that it gives more weight to classes that are represented as minorities in the dataset. For this reason, it was thought to be more appropriate for this project, and since it was found experimentally to lead to small improvements, it was decided to use this function. The formula for this function can be given as follows:

$$\text{Focal Loss} = -\alpha_t (1 - p_t)^{\gamma} \log(p_t)$$

- α_t: Separately assigned weighting factor for each class used to compensate for class imbalance.
- p_t: The predicted probability of the correct class.
- γ: Focusing parameter. It allows the model to take into account more instances that the model has difficulty predicting. As γ increases, the model gives more weight to the difficult examples.

After the training and testing of the models were completed, the user interface was designed in the last stage. The interface was developed by writing HTML, CSS and JavaScript codes. The interface was integrated into a web application using the Flask framework in Python.

B. Database Design

Database design involves the process of identifying and organizing the data to train the model. In order to create the data set, data sets were first extracted from HuggingFace. In the later stages of the project, it was determined that the data set extracted from HuggingFace was not suitable for the project. The main reason for this is that the dataset was found to have a number of punctuation errors that would affect the performance of the model. Another reason was that this dataset was insufficient to train the models. After this stage, the datasets were reconstructed and Turkish novels were used. After the dataset was created, it was preprocessed and converted into a format to be used in the next stages. The dataset was designed to be scalable. That is, it can be optimized for future growth or downsizing needs. The final dataset contains 56309 rows, each with an average of 2 sentences.

After pre-processing, the frequency of the tags in the dataset is as shown in the figure below.

	Frequency
NONE	1246412
UPPER	163571
NOKTA	107094
VIRGUL	97333
SORU	11694
IKINOKTA	9424
UPPER_VIRGUL	9379
UPPER_NOKTA	1813
UPPER_IKINOKTA	692
UPPER_SORU	425

Figure 2 The number of tags in the dataset.

C. Input-Output Design

The input-output design of this project involves the model processing the text input by the user through an interface and returning the corrected text back to the user. The input-output design is shaped in accordance with the main goal of the project.

- **Input**: The user will enter a text in Turkish via the web interface. The text to be entered as input should not contain capital letters. In addition, the user should make sure that there are no punctuation marks or special characters in the sentence. Otherwise, the model will take this data as a token and this may cause the output to be incorrect.
- **Output**: The model will process the Turkish text given as input by the user, correcting any punctuation errors or capitalization. The model will then return the corrected text back to the user on the web interface. The web interface will display both the corrected sentence and the label of each token to reflect the result.

This input-output design aims to improve the user experience by enabling the user to easily input text and quickly see the text corrected by the model, and is designed in a simple way to do so.

V. Application

In the web application, a simple screen welcomes the user. There is a dropdown menu that allows us to select one of the 10 models, 5 pretrained and 5 fine-tuned, and make predictions on it. Below this is a text box where we can enter a sentence. After selecting the model and entering the input, the user can press the run button and see the result outputs on the screen below. Below are screenshots showing the operation and appearance of the application:

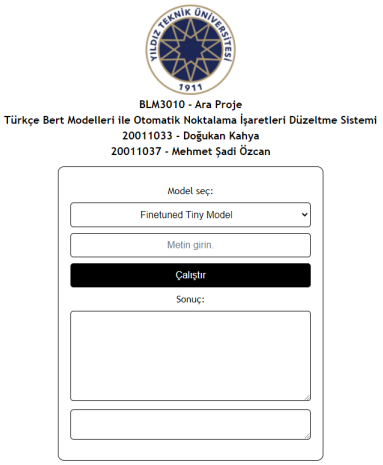

Figure 3 The image the user sees on the home screen.

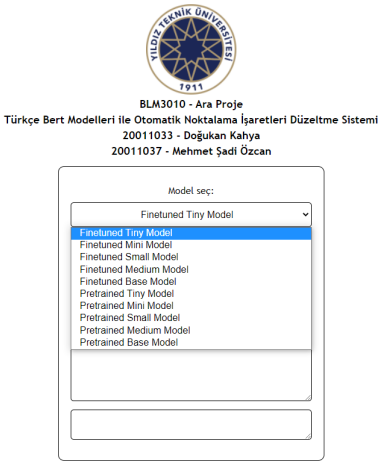

Figure 4 Dropdown menu for model selection.

In the application, predictions can be made with 10 different models. These models are the 5 pretrained Turkish BERT models determined at the beginning of the project and their finetuned versions with the data generated within the scope of the project. Above is the visualization of the dropdown menu where model selection can be made.

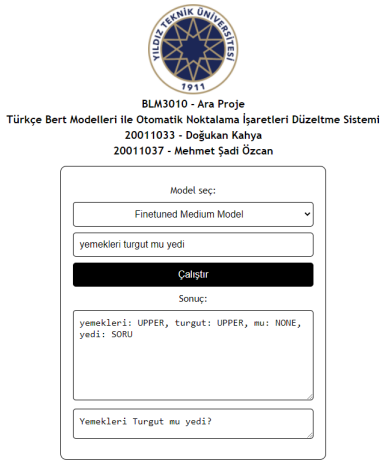

Figure 5 The resulting image when running the application with the finetuned model.

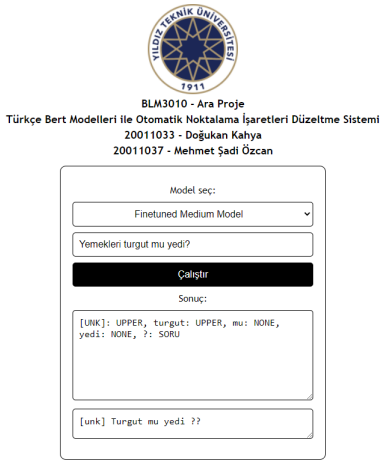

Figure 6 Image when the wrong input is given to the application.

This is the result of the user entering an incorrect input. The use of capital letters in the word causes the model to fail to tokenize the word during tokenization and identify it as “unknown”. In addition, the use of punctuation marks or special characters in the input sentence causes the model to consider this character as a token and may prevent it from giving correct results.

VI. Experimental Results

In this part of the report, we will explain how the application works with 5 different sizes of Turkish BERT models available at ytu-ce-cosmos in HuggingFace with experimental results. While making these explanations, the metrics before and after fine-tuning will be examined and compared and analyzed in detail. This analysis mainly focuses on the F1 and recall metrics.

The pre-fine-tuning metrics provide important information to evaluate the performance of the model before it is retrained with our dataset. These values indicate the success of the model in the initialization

phase and are important to understand how well the model performs after training by comparing with the post-fine-tuning values.

Below we discuss the metrics we use to measure the accuracy of the model. Before that, there are a few definitions that need to be made.

- **True Positives**: If the model predicts a token as positive (a label other than NONE) and this prediction is true, it will be a true positive.
- **False Positives**: If the model predicts a token as positive (a label other than NONE) and this prediction is false, this will be a false positive.
- **True Negatives**: If the model predicts a token as negative (NONE) and this prediction is true, it will be a true negative.
- **False Negatives**: If the model predicts a token as negative (NONE) and this prediction is false, it will be a false negative.

Below are a few metrics that evaluate the performance of the models using them.

- **Precision**: This is a metric that measures how many of the samples predicted as positive are actually positive. The formula for this is given below. In models trained with datasets with class imbalance, this metric was not included in the calculations in the project, as it would cause this metric to give a misleadingly high result if the precision calculation predicts the dominant label frequently.[6]

$$\text{Precision} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Positives (FP)}} \quad (1)$$

- **Recall**: Recall is a metric that measures the rate at which a model detects correctly classified positive samples. That is, it gives the percentage of all true positives that the model is able to detect. A high recall rate indicates that the model is good at correctly detecting positive classes. Recall is calculated as follows:

$$\text{Recall} = \frac{\text{True Positives (TP)}}{\text{True Positives (TP)} + \text{False Negatives (FN)}} \quad (2)$$

- **F1 Score**: The F1 score is a metric calculated as the harmonic mean of precision and recall. It measures the overall accuracy of the model by considering both precision and recall. The F1 score is particularly useful in cases of imbalance between classes because it provides a balanced evaluation of precision and recall.[7] The F1 score is calculated as follows:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

The next part of this section presents the performance outputs of 5 different models before and after fine-tuning. The tests whose results are given below were performed on a test set consisting of approximately 5600 rows obtained by removing 0.1 part of our data set consisting of 56309 rows. In the tables below, the performance values of the models are given and generally interpreted. At the end of the section, there are general performance data for the final version of the model.

A. Tiny Bert Model Results

Table 3 Detailed Metrics

Class	Pretrained Recall	Pretrained F1	Fine-Tuned Recall	Fine-Tuned F1
NONE	0.8732	0.8079	0.8507	0.8261
UPPER	0.1206	0.1089	0.4542	0.5068
VIRGUL	0.0	0.0	0.0416	0.0733
NOKTA	0.0	0.0	0.2549	0.1945
IKINOKTA	0.0	0.0	0.0	0.0
SORU	0.0	0.0	0.1348	0.1577
UPPER_VIRGUL	0.0	0.0	0.1022	0.1682
UPPER_NOKTA	0.0	0.0	0.0	0.0
UPPER_İKINOKTA	0.0	0.0	0.0	0.0
UPPER_SORU	0.0	0.0	0.0	0.0
Genel	0.0994	0.0917	0.1838	0.1927

The table above shows the results of the Tiny BERT model. As can be seen, there is a slight decrease in the recall of the NONE class. The reason for this is that during the fine-tuning process, the model focused on the classes in which it performs well. There is an improvement in the UPPER class. There is an increase in the pretrained recall results of the NOKTA, SORU, UPPER-VİRGÜL classes, which were 0. For the other classes, the scores remained constant at 0. An increase of nearly 2-fold is observed in the overall scores.

B. Mini Bert Model Results

Table 4 Detailed Metrics

Class	Pretrained Recall	Pretrained F1	Fine-Tuned Recall	Fine-Tuned F1
NONE	0.4005	0.5159	0.7720	0.8033
UPPER	0.5823	0.1706	0.5487	0.5223
VIRGUL	0.0	0.0	0.2011	0.2426
NOKTA	0.0	0.0	0.3930	0.2612
IKINOKTA	0.0	0.0	0.1253	0.1445
SORU	0.0	0.0	0.3796	0.2399
UPPER_VIRGUL	0.0	0.0	0.1902	0.2357
UPPER_NOKTA	0.0	0.0	0.0	0.0
UPPER_İKINOKTA	0.0	0.0	0.0	0.0
UPPER_SORU	0.0	0.0	0.0	0.0
Genel	0.0983	0.0687	0.2610	0.2449

As can be seen from the table, there is an improvement in the NONE class. UPPER class showed a slight decrease compared to its pretrained state. For the VİRGÜL, NOKTA, İKİNOKTA, SORU, UPPER-VİRGÜL classes, the score of 0 has increased. Scores remained constant at 0 for the other classes. There was a significant increase in the overall score.

C. Small Bert Model Results

Table 5 Detailed Metrics

Class	Pretrained Recall	Pretrained F1	Fine-Tuned Recall	Fine-Tuned F1
NONE	0.2508	0.3814	0.7979	0.8261
UPPER	0.7712	0.1781	0.5985	0.5843
VIRGUL	0.0	0.0	0.2589	0.3112
NOKTA	0.0	0.0	0.4805	0.3179
IKINOKTA	0.0	0.0	0.2075	0.2423
SORU	0.0	0.0	0.4782	0.3143
UPPER_VIRGUL	0.0	0.0	0.2010	0.2654
UPPER_NOKTA	0.0	0.0	0.0	0.0
UPPER_İKINOKTA	0.0	0.0	0.0	0.0
UPPER_SORU	0.0	0.0	0.0	0.0
Genel	0.1022	0.0559	0.3023	0.2861

In this model, similar results to the above model were obtained. There is an improvement in the NONE class and a slight decrease in the UPPER class. There is an increase from 0 in the VİRGÜL, NOKTA, İKİNOKTA, and SORU classes. The other classes remained constant at 0. There is a 3-fold increase in the overall recall score.

D. Medium Bert Model Results

Table 6 Detailed Metrics

Class	Pretrained Recall	Pretrained F1	Fine-Tuned Recall	Fine-Tuned F1
NONE	0.6993	0.7188	0.7324	0.7817
UPPER	0.2131	0.1107	0.5035	0.4984
VIRGUL	0.0	0.0	0.2767	0.2755
NOKTA	0.0	0.0	0.4326	0.2578
IKINOKTA	0.0	0.0	0.2095	0.1961
SORU	0.0	0.0	0.3933	0.2567
UPPER_VIRGUL	0.0	0.0	0.2109	0.2672
UPPER_NOKTA	0.0	0.0	0.0854	0.1459
UPPER_IKINOKTA	0.0	0.0	0.1231	0.1758
UPPER_SORU	0.0	0.0	0.0	0.0
Genel	0.0912	0.0830	0.2967	0.2855

In this model, there was a significant increase in the UPPER class and a small increase in the NONE class. The UPPER-SORU score remained constant at zero. The scores of the other classes, which were 0, showed a significant improvement. The overall score is close to the above model.

E. Base Bert Model Results

Table 7 Detailed Metrics

Class	Pretrained Recall	Pretrained F1	Fine-Tuned Recall	Fine-Tuned F1
NONE	0.4634	0.5828	0.7155	0.7741
UPPER	0.6202	0.1897	0.5535	0.5010
VIRGUL	0.0	0.0	0.2475	0.2633
NOKTA	0.0	0.0	0.4567	0.2631
IKINOKTA	0.0	0.0	0.1987	0.1918
SORU	0.0	0.0	0.4061	0.2741
UPPER_VIRGUL	0.0	0.0	0.2457	0.2950
UPPER_NOKTA	0.0	0.0	0.1709	0.1884
UPPER_IKINOKTA	0.0	0.0	0.1692	0.1982
UPPER_SORU	0.0	0.0	0.0444	0.0741
Genel	0.1084	0.0772	0.3208	0.3023

The table above shows an improvement in the NONE class. There is a slight decrease in the UPPER class. The scores of all other classes have increased from 0. The increase in the UPPER-SORU class was very small. Compared to other models, this model showed the highest overall score increase.

F. Comparative Results

The best performing model in terms of overall recall and F1 score is the base model. This model has the highest overall recall of 0.3208 and F1 score of 0.3023 after fine-tuning. The best performing model in the NONE class is the small model. The best performing model in the UPPER class is the base model. In general, base model, tiny model and small model perform well in NONE and UPPER classes. In the VİRGÜL and NOKTA classes, the base model stands out especially in terms of recall and F1 scores. Overall, and considering the class-by-class performance, it can be said that the base model is the most suitable model for use.

VII. Performance Analysis

In this section, the performance of the trained models will be analyzed and the results obtained will be interpreted.

For the training of the models, the Focal Loss function and the parameters shown in the figure below were used for training.

```
#Training argümanları
training_args = TrainingArguments(
    output_dir='./results',  #output dizini
    num_train_epochs=8,  #epoch sayısı
    per_device_train_batch_size=64,  #training için batch size
    per_device_eval_batch_size=64,  #evaluation için batch size
    warmup_steps=500,  #learning rate zamanlayıcısı için öğrenme adımlarının sayısını belirler
    #// learning rate başlangıçta default olarak = 5e-5
    weight_decay=0.01,  #overfittingi önlemek için kullanılmıştır. her bir adımda
    #ağırlıkların 0.01 oranında azaltılacağı anlamına gelir.
    logging_dir='./logs',  #logları kaydetmek için dizin
    logging_steps=900,
    evaluation_strategy="steps",
    save_strategy="epoch",  #her epoch'ta model kaydedilecektir.
)
```

Figure 7 Models' training parameters.

As can be seen in the figure, all models were trained with 8 epochs. Batch sizes were set to 64. The change in the recall value of the models during training is given in the graphs below.

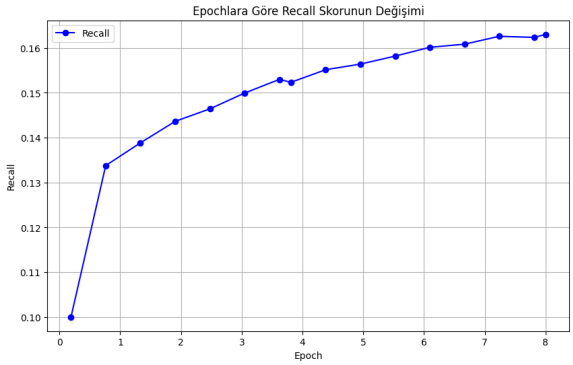

Figure 8 The recall value in the Tiny model.

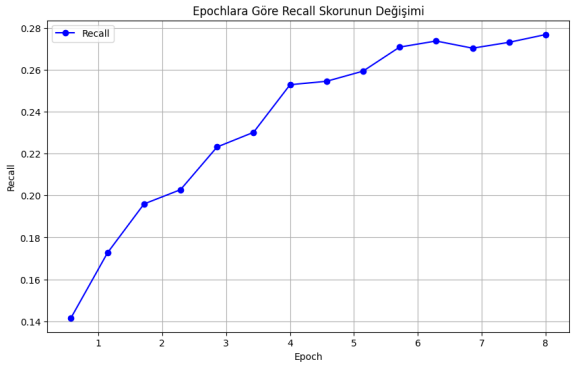

Figure 9 The recall value in the Mini model.

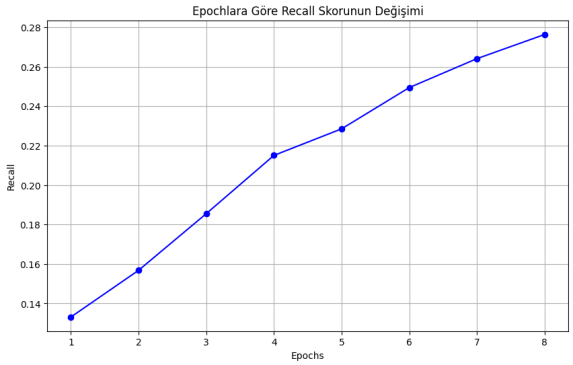

Figure 10 The recall value in the Small model.

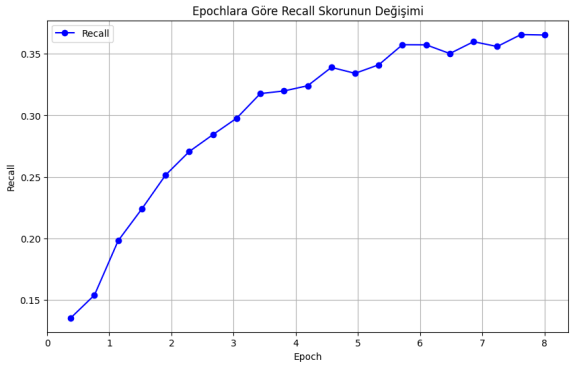

Figure 11 The recall value in the Medium model.

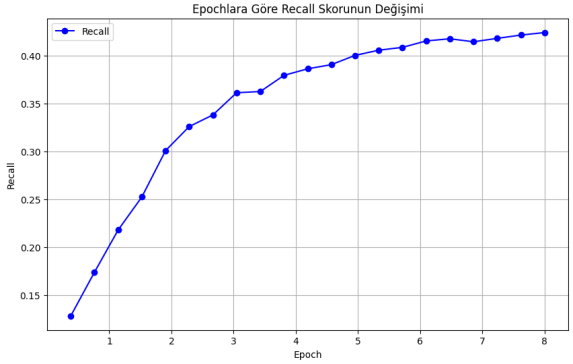

Figure 12 The recall value in the Base model.

Table 8 Model Performance Metrics

Model	Recall	F1 Score
Tiny Model	0.16299	0.17890
Mini Model	0.27653	0.28540
Small Model	0.27934	0.28525
Medium Model	0.36493	0.37448
Base Model	0.42425	0.43965

As can be seen in the figures and the table, the models show a rapid improvement in the beginning and then the improvement slows down. Based on the recall values resulting from the training, the performance of the base model can be interpreted as quite good compared to the other models. Since the GPUs used during the training of the models were insufficient, the training took too long, especially for the medium and base models, and not enough experiments could be performed on the parameters. Increasing the batch sizes, decreasing the learning rate, increasing the number of epochs, etc. may give more beneficial results for the development of the models.[8]

VIII. Conclusion

The main goal of this project is to develop a model that can correct punctuation errors in Turkish texts by using different sizes of Turkish BERT models on the HuggingFace page of YTU Computer Engineering Cosmos Research Group. While doing this development, the models on the ytu-ce-cosmos page need to be fine-tuned. After these operations, a web application will be designed that the user can interact with.

At the beginning of the project, the data of Turkish novels on the internet were extracted, then these sentences were subjected to certain preprocessing processes and a dataset of 56309 lines was created with an average of 2 sentences per line. In the final version of this dataset to be inserted into the model, there is a column containing the tokens of the preprocessed sentences tokenized by the tokenizers of the models to be used, and a column of labels consisting of integer values indicating which punctuation marks are found after these tokens. After preparing the dataset, the models were trained using the Focal Loss function with the parameters specified in the report. A 0.1 part of the dataset was set as a test set and then these fine-tuned models were put to certain tests on this test set. Recall and F1 scores were analyzed in the tests and these scores are explained and interpreted in detail in the report. In addition, an interactive web application was designed where the user can correct punctuation by entering a random sentence.

As a result of this study, a significant improvement was observed between the results before and after fine-tuning. The results show that the model is successful in some classes but needs further improvement in others. This can be explained by the fact that the unsuccessful classes are underrepresented in the dataset or the model has difficulty in learning these classes. In future studies, it may be suggested to diversify the dataset, use larger datasets, or use training parameters that may be more appropriate to improve the success of the model.

References

[1] A. Nagy, B. Bial, and J. Ács, "Automatic punctuation restoration with BERT models," *CoRR*, vol. abs/2101.07343, 2021. [Online]. Available: https://arxiv.org/abs/2101.07343

[2] J. B. Nilsson, "Punctuation restoration in swedish through fine-tuned kb-bert," 2022.

[3] M. Özkan and G. Kar, "TÜrkÇe dİlİnde yazilan bİlİmsel metİnlerİn derİn ÖĞrenme teknİĞİ uygulanarak Çoklu siniflandirilmasi," *Mühendislik Bilimleri ve Tasarım Dergisi*, vol. 10, no. 2, p. 504–519, 2022.

[4] Y. Wu, K. Fang, Y. Zhao, H. Zhang, L. Shi, and M. Zhang, "Ff2: A feature fusion two-stream framework for punctuation restoration," 2022.

[5] L. N. Smith, "Cyclical focal loss," 2022.

[6] M. Buda, A. Maki, and M. A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *CoRR*, vol. abs/1710.05381, p. 9, 2017. [Online]. Available: http://arxiv.org/abs/1710.05381

[7] S. Raschka, "How can the f1-score help with dealing with class imbalance?" 2018. [Online]. Available: https://sebastianraschka.com/faq/docs/computing-the-f1-score.html

[8] S. L. Smith, P. Kindermans, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *CoRR*, vol. abs/1711.00489, p. 9, 2017. [Online]. Available: http://arxiv.org/abs/1711.00489