International Journal of Assessment Tools in Education
2023, Vol. 10, Special Issue, 148–162
https://doi.org/10.21449/ijate.1394194
Published at https://ijate.net/ https://dergipark.org.tr/en/pub/ijate *Research Article*

Language models in automated essay scoring: Insights for the Turkish language

Tahereh Firoozi[1*], **Okan Bulut**[1], **Mark J. Gierl**[1]

[1]University of Alberta, Edmonton, Alberta, Canada

ARTICLE HISTORY

Received: Nov. 22, 2023
Accepted: Dec. 17, 2023

Keywords:
Automated essay scoring,
Word embedding,
Transformers,
BERT,
Turkish AES.

Abstract: The proliferation of large language models represents a paradigm shift in the landscape of automated essay scoring (AES) systems, fundamentally elevating their accuracy and efficacy. This study presents an extensive examination of large language models, with a particular emphasis on the transformative influence of transformer-based models, such as BERT, mBERT, LaBSE, and GPT, in augmenting the accuracy of multilingual AES systems. The exploration of these advancements within the context of the Turkish language serves as a compelling illustration of the potential for harnessing large language models to elevate AES performance in in low-resource linguistic environments. Our study provides valuable insights for the ongoing discourse on the intersection of artificial intelligence and educational assessment.

1. LANGUAGE MODELS IN AUTOMATED ESSAY SCORING

Automated essay scoring (AES) is a sub-task of text classification that uses computer algorithms to score essays written by humans automatically. Machine and deep learning algorithms are often utilized to build a scoring engine that can model the scoring performance of human raters. The model is then employed to classify essays into different score classes (i.e., score categories). AES systems typically work by analyzing the text of an essay and applying a set of linguistic features to assess its quality. These features may include grammar, vocabulary, sentence structure, coherence, and the presence of relevant arguments or evidence. The AES system builds the scoring model using techniques and procedures from the fields of natural language processing (NLP) and computational linguistics where linguistic features are extracted from the instances of human-scored essays (i.e., labeled data) and turned into numerical representations that a machine or deep learning model can process. The most common NLP techniques for feature extraction include text length features, bag of words, and pre-trained large language models such as Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2018).

Text length features are simple and effective in general text analysis and AES (Fleckenstein et al., 2020; Hussein et al., 2019), as they have been widely used to evaluate essays based on their

*CONTACT: Tahereh FIROOZI ✉ tahereh.firoozi@ualberta.ca University of Alberta, Edmonton, Alberta, Canada

e-ISSN: 2148-7456 /© **IJATE** 2023

length and structure. These features include, for example, the average number of words per paragraph, or the average number of characters per word. Using text length features, AES systems compare the length of each essay to the length of the essay prompt or an ideal essay length with a high score in the training corpus. Text length features are typically used in conjunction with other syntactic properties, such as part of speech (POS) and discourse characteristics of a text, including cohesion and coherence. Statistical language models are used to analyze the syntactic properties in a text (e.g., Rodriguez et al., 2019). N-gram is an example of the statistical model that captures the likelihood of a sequence of n words occurring in a given text based on the frequency of those word sequences in a training corpus. Practically, the syntactical property of texts is assessed using the existing natural language toolkit libraries in programming languages, such as the NLTK library in Python (Bird, 2006). In addition, the linguistic and discourse characteristics of written texts in English can be assessed using text analysis tools, such as Coh-Metrix (Graesser et al., 2004), which were developed and used for the English language.

Word embeddings have become fundamental in various NLP applications, including AES. Word embedding models, such as Word2vec and GloVe embeddings, are a class of NLP techniques to represent words as dense vectors in a continuous vector space (Firoozi et al., 2022). These vectors capture hidden information about a language, like word analogies or semantics. The information can be used to examine the proximity of the semantic relationship between the word and the context. For example, in a well-trained word embedding model, the vectors for "king" and "queen" would be closer together than the vectors for "king" and "car." Calculating the proximity in the vector space allows the model to capture semantic relationships, such as analogies ("king" is to "queen" as "man" is to "woman"). This knowledge is learned in pre-trained word embedding models through unsupervised learning using large amounts of text corpus. Depending on the corpus and the learning techniques, the word embedding models capture different information in the vectors. The most popular word embedding language models are Word2Vec (Mikolov et al., 2013), GloVe (Pennington et al., 2014), and FastText (Bojanowski et al., 2017) developed by Google Inc., Stanford University, and Facebook AI Research, respectively. The models were pre-trained using different corpora and learning techniques.

Word2Vec pre-trained vectors were trained on a part of the Google News dataset with about 100 billion words. Mikolov et al. (2013) proposed two model architectures, including a continuous bag of words (CBOW) and Skip-gram, for learning distributed representations of words. CBOW predicts a word in a sequence of words given the average distributed representation of all the surrounding words in the sequence. A word is predicted based on the largest semantic similarity between the word vector and the average distributed representation vector of the surrounding words in context. The Skip-gram model is similar to CBOW, but instead of predicting a word based on the surrounding sequence of words, it tries to predict the surrounding words of a given word in a sentence.

FastText (Bojanowski et al., 2017) has a similar training process (i.e., CBOW and Skip-gram) as Word2Vec. FastText differs from Word2Vec regarding the corpora used for training and word representation technique. FastText is trained on Wikipedia data in nine different languages: Arabic, Czech, German, English, Spanish, French, Italian, Romanian, Russian, and Turkish (Kuyumcu et al., 2019). FastText represents words as bags of character n-grams (subword units). In this technique, words are decomposed into character-level n-grams (e.g., "apple" -> {ap, ppl, ple}), including both prefixes and suffixes. Word representations are then generated by summing or averaging the vectors of these n-grams. The character level representation in FastText enables the model to capture morphological and semantic information even for out-of-vocabulary words.

The training process of the Glove model (Pennington et al., 2014) is different from Word2Vec. The GloVe model combines the matrix factorization methods (Cai et al., 2009) and the window-based methods to consider both the statistical and contextual information of words in calculating word vectors. Hence, GloVe learns the embeddings based on a co-occurrence matrix showing the count of the overall statistics of how often words appear together in a text based on their semantic similarity. The vector spaces of the word embedding techniques can be trained on AES datasets with different sizes to fine-tune the pre-trained parameters.

The word embedding models use Recurrent Neural Network (RNN) models for training (Liu et al., 2015). RNN models are neural network models containing a hidden layer that auto-regressively updates the conditional probability of the output vector (e.g., a word or the context of a word) given the hidden state in the next step. The RNN model updates the prediction weights based on the errors it receives in the following steps. While RNN-based models revolutionized the Google translation engines in 2016, they have two main problems. First, these models suffer from the vanishing gradient problem, making it very difficult to capture long-range dependencies within the text (Hochreiter et al., 2001). For instance, if a system is developed to predict the next word in a sentence, the network must have a better knowledge of the preceding words in the text for more accurate predictions. In RNN, the hidden weights are updated recurrently to decrease the error function. In long texts with more hidden weights at different time steps, the initial weights are multiplied by the updated weight. However, because the initial weights are small, this multiplication quickly decreases the gradient value, leading to the early termination of model training before the model can learn the whole text. This problem with sequential training was solved using a parallel structure in encoding the input sequence of different lengths (Vaswani et al., 2017).

2. TRANSFORMER_BASED LANGAUGE MODELS

Research studies on AES skyrocketed in 2018 when the transformer models (Vaswani et al., 2017) were introduced (Ramesh & Sanampudi, 2022). Transformer-based models have revolutionized the field of NLP by offering powerful tools for training language models that significantly increase the accuracy of the pre-trained models in text classification tasks, including AES (Devlin et al., 2018). Transformers are encoder-decoder-based neural networks that solve sequence processing problems by finding a mapping function from an input sequence of vectors (e.g., word or sentence) to the output sequence of vectors (e.g., essay labels). The architecture consists of an encoder and a decoder comprising multiple layers of multi-head attention-based blocks. The encoder takes the input sequence and processes it by repeatedly applying the multi-head attention block to the input sequence of tokens. The attention mechanism in transformers can capture all of the contextual information within a text to calculate the weighted sum of values for each token (e.g., words) in a sequence of input (e.g., sentence). For example, in the sequence of input = “I want to buy a car,” the representation of the fourth word “buy” depends not only on the adjacent words, including “I,” “want,” “to,” “a,” and “car” in the sequence, but also on all other words in the text. This feature allows for the modeling of global dependencies in all sequential inputs without regard to their distance in the input or output sequences. Hence, in encoding or decoding the representation of an input sequence, the attention mechanism allows transformers to learn the context of the input by parallelizing all the surrounding inputs within training examples (Wolf et al., 2020). The decoder takes the output sequence generated by the encoder and processes it by attending to the encoder's output and the previous tokens in the decoder's input sequence. This allows the decoder to generate each output token by selectively attending to different parts of the input sequence. During training, the model learns to assign appropriate weights to the tokens in each layer of the encoder and decoder in order to minimize a given loss function (e.g., cross-entropy

loss) between the predicted output sequence and the ground truth output sequence (Han et al., 2022).

Using this training approach, transformer-based language models can accurately capture the linguistic patterns in a language by learning the long-range dependencies and semantic relationships between tokens (e.g., words/subwords or sentences) in texts (Bouschery et al., 2023). Hence, by transferring the knowledge in these pre-trained language models to the targeted AES task, the accuracy of the AES systems improves significantly without using a large number of labeled essays for training. Unlike the text length and word embedding models that were language-specific and mainly developed and used for the English language, transformer-based models were also trained on languages other than English. Hence, the low-resource languages can benefit from transformer-based language models, including BERT, LaBSE, and GPT, for transfer learning in AES (Firoozi et al., 2023). Low-resource languages are less studied, less digitized, less privileged, less commonly taught, and less accessible compared to English (Cieri et al., 2016; Magueresse et al., 2019).

BERT (Devlin et al., 2018) is a transformer-based encoder model for language representation that uses a multi-head attention mechanism and a bidirectional approach to learn the contextual relations between words and sentences in a text for an accurate representation of the entire text. Multi-head attention is a mechanism for training transformers that compares each input vector with all other text vectors to consider the context in word representations. The bidirectional in BERT refers to the training process where the transformers can generate contextual embeddings based on previous and next tokens of the text (Kenton & Toutanova, 2019). BERT can be trained in different languages. BERT is trained using two approaches. The first approach is masked language modeling (MLM). MLM predicts a missing word in a sentence by randomly masking 15% of the words and running all the masked sentences through the model to predict the masked words. The second approach is next sentence prediction (NSP). NSP is predicting if one sentence naturally follows another. In NSP, the model learns to understand longer-term dependencies across sentences. Using the NSP technique allows the model to predict each two-sentence sequence that follows one another in a text. BERT learns this knowledge by receiving masked sentence embeddings concatenated in pairs as inputs during pre-training. Half of the embeddings are random, and the other half are actual sentence pairs from the pool of training data. For example, the model receives sentence A and sentence B to predict whether sentence B is the next sentence or whether it is not the next sentence. This process continues, and the model learns from the error rates in each prediction until it fully predicts the accurate sequence of sentences in a text (Devlin et al., 2018).

The version of the BERT model using multilingual text is called mBERT (Devlin et al., 2018). mBERT contains the lexical, linguistic, and grammatical knowledge for 104 different languages. The languages in this model were selected because they contain the largest number of Wikipedia entries. mBERT was trained using MLM and NSP, and it implements a monolingual text stream process in which each language's pre-training process is conducted separately. As a result, the feature space for each language is not shared with any other language in the model. mBERT can be used to overcome the problem of data sparsity. For example, Firoozi and Gierl (in press) used mBERT to score essays written in Persian. Persian is a low-resource language that has proven challenging for traditional automated text analysis methods (Roshanfekr et al., 2017). Firoozi and Gierl (In press) compared the result of the mBERT language model with a word-embedding language model. The mBERT model (Quadratic Weighted Kappa =0.84) significantly outperformed the word embedding model (Quadratic Weighted Kappa = 0.75). The Quadratic Weighted Kappa (QWK) is a statistic that measures the agreement between two sets of ratings or classifications. It is commonly used in the field of inter-rater reliability to assess the agreement between human raters or between a human rater

and an automated system. QWK considers quadratic weights for misclassifications based on how close the ratings are to the correct class using an ordinal scale. In the current study, QWK is the main evaluation metric in AES systems because it can easily be used to compare the performance of our model with the performance of models used in similar studies. QWK varies from 0 (random agreement between raters) to 1 (complete agreement between raters). Typically, values between 0.60 and 0.80 QWK are used as a lower bound estimate for an acceptable reliability outcome using human raters in a high-stakes testing situation (Williamson et al., 2012).

Another BERT model that uses multilingual text is called language-agnostic BERT sentence embedding or LaBSE (Feng et al., 2020). LaBSE is an extension of mBERT where, instead of considering monolingual text streams, bilingual text streams are created by using parallel data—a collection of texts in two or more languages aligned at a sentence or phrase level—in the learning process of the model. This sentence embedding method is called translation language modeling (TLM) (Lample & Conneau, 2019). LaBSE uses TLM and MLM for training by randomly masking words in both the source- and target-language sentences. For example, when Italian and German serve as the source and target languages, respectively, LaBSE can predict a word masked in a sentence written in Italian either by attending to the surrounding words written in Italian or by attending to the parallel surrounding words written in German thereby allowing the model to align the Italian and German text representations.

A critical benefit of using LaBSE is that the model can leverage information from the multilingual context to improve its ability to learn the text (Chi et al., 2020). For example, the model can use the German language context to infer the masked Italian word if the Italian language context is not sufficient to infer the masked Italian words. By training LaBSE on parallel sentences using TLM and MLM, the model learns the lexical, linguistic, and grammatical knowledge for each language and connects the knowledge between the two languages, thereby providing a shared embedding space for both languages in the model. Because LaBSE is pre-trained on 109 languages, many different LaBSE-based language models can be fine-tuned using different numbers and types of languages on downstream tasks. The knowledge transfer between languages is essential when considering low-resource languages. Training a language model like LaBSE on several languages with adequate supervised resources allows building AES systems on low-resource languages using their limited data. LaBSE serves as another example of how language models can be used to facilitate transfer learning (Ranathunga et al., 2023).

GPT (Generative Pre-trained Transformer) is another recent groundbreaking transformer-based language model (Radford et al., 2019) developed by OpenAI. Like the BERT models, the core idea behind training GPT models is the attention mechanism introduced by transformers. GPT models differ from BERT-based models in terms of training methods and the dataset used for training. Unlike BERT, a bidirectional transformer-based architecture, GPT is a unidirectional transformer-based architecture trained on texts from start to end. In addition, GPT models use a different training method than mask language modeling used in BERT. GPT models are autoregressive language models that generate text by predicting the next word in a sequence given the previous words (Black et al., 2022; Brown et al., 2020). This type of training enables GPT models not only to understand but also to generate texts.

GPT models are trained unsupervised on a vast amount of textual data available on the internet. GPT was trained on a much larger corpus than the one used for BERT. For example, GPT-3 (Generative Pre-trained Transformer 3), the third version of the GPT series of language models introduced by OpenAI in 2020, contains 175 billion parameters that enable it to generate coherent and fluent text outputs such as text generation, language translation, and question-answering in a human-like manner. While BERT-based models can mainly be utilized for

transfer learning in scoring students' written tasks, the GPT models' capabilities to generate text make them useful for transferring their knowledge to generate detailed feedback to students (Mayer et al., 2023). GPT-3's remarkable capabilities come with computational resource requirements and limitations. The model size and complexity make it computationally intensive, requiring significant computational power to train and deploy effectively. Additionally, GPT-3 text generation can sometimes exhibit biases in the training data, and it may generate plausible but incorrect or misleading information (Mizumoto & Eguchi, 2023).

GPT is just a member of a larger category of models called Large Language Models (LLMs) (MacNeil et al., 2022). As the name suggests, LLMs are large models trained to contain the structure and knowledge of natural languages. Similar to variations of GPT, these models contain billions of parameters and are trained on massive corpora using self-supervised methods. As a result, these models acquire a deep and rich understanding of their target languages. However, as these models are trained on text with a wide range of topics and structures, they have gained a unique generalizability and multitasking ability (Bubeck et al., 2023). These models can be prompted and interactively trained to perform entirely new tasks. One example of such a task could be AES. Like humans, the model will gradually acquire the ability to do AES by prompting a language model to score an essay and providing constructive feedback. LLMs can utilize their comprehensive knowledge of language, common sense, and communication skills to acquire AES skills without needing to be explicitly trained on AES in a supervised fashion (Mizumoto & Eguchi, 2023). Hence, LLMs can be trained as an AES chatbot that scores essays and provides detailed and personalized feedback for each essay. The chatbot can also be prompted for further feedback and automated improvements through a natural language conversation.

3. AUTOMATED ESSAY SCORING IN TURKISH LANAGUGE

Turkish is a language in the Turkic family of Altaic languages, which over 80 million people speak in Turkey, the Middle East, and Western European countries. Despite being the native language of more than 80 million people, like other low-resource languages, Turkish is also relatively less studied and benefited from the developed NLP tools and resources (Oflazer & Saraçlar, 2018). The Turkish language has certain morphological features, such as multiple derivations of a given word via prefixes and suffixes, making language processing more challenging (Koskenniemi, 1983). For example, the single word "ruhsatlandırılmamak" includes five suffixes. Despite these language challenges, NLP research and tools in the Turkish language are growing thanks to the unsupervised learning algorithms that overcome the problem of data sparsity in NLP tasks, such as speech recognition (e.g., Arslan & Barışçı, 2020) and sentiment analysis (e.g., Gezici & Yanıkoğlu, 2018).

Research on Turkish AES has been the focus of very few studies (Cetin & Ismailova, 2019; Dikli, 2006; Uysal & Doğan, 2021). Cetin and Ismailova (2019) attempted to develop a language tool to automatically evaluate students' essays in Turkish. They used the existing NLP tools for the Turkish language, including Zemberek (Akın & Akın, 2007), to extract mechanical features of the language, such as word count, spelling error, and number of sentences for evaluation of written essays. In another study, Uysal and Doğan (2021) compared different machine learning (ML) algorithms, including support vector machines, logistic regression, multinominal Naive Bayes, long-short term memory (LSTM), and bidirectional long-short term memory (BILSTM) to score open-ended response items in the Turkish language. They also used the existing NLP tools in the Turkish language for text representations. Uysal and Doğan (2021) concluded that the BLSTM model outperformed (QWK=0.77) the other models, including Logistic regression (QWK=0.70), Naïve Bayes (QWK=0.64), support vector machine (QWK=0.69), and LSTM (QWK=0.58) in terms of scoring accuracy.

Despite the popularity of AES models, they have not been studied widely in the Turkish language. One reason is that the NLP tools for feature extraction in low-resource languages such as Turkish are limited (Cetin & Ismailova, 2019). In addition, there are very few, if any, labeled essays available for public research. For example, in the Turkish language, the available few labeled data (e.g., Benchmark Data[†]) are developed for text analysis tasks, such as sentiment analysis (Kavi, 2020), and there are no labeled essays that can be used for AES tasks. Given that the recent large language models such as mBERT and LaBSE are trained in hundreds of languages, including Turkish, the rich knowledge in these LLMs can be transferred to downstream tasks such as AES using even a few training data (Firoozi et al., 2022). The existing challenges in the Turkish AES research, including the limited NLP tools for feature extraction and the insufficient labeled essays available for public research, can be solved by using the large language models, such as mBERT and LaBSE, which were reviewed in the current study. Using transformers, like mBERT, can help decrease the gap in the AES literature between English and low-resource languages. The following steps summarize the process of applying the mBERT model to the Turkish Language using Python.

3.1. Installing Transformers Library

Google Colab (https://colab.research.google.com/) gives free access to writing and executing arbitrary Python code through the browser. It also provides easy-to-use hardware acceleration for deep learning models. First, on the Google Colab page, we install transformers with pip package manager and import the installed packages (Figure 1). The Tensorflow package is a computational graph processor–a fundamental tool for implementing and using deep learning models in Python. The Transformers package by Hugging Face also enables us to employ the latest transformer models and their pre-trained weights.

Figure 1. *Installing libraries.*

```
!pip install tensorflow
!pip install transformers

import tensorflow as tf
from transformers import AutoTokenizer
from transformers import TFAutoModelForSequenceClassification
```

3.2. Loding Turkish Dataset

The code snippet in Figure 2 shows how to write a Python function to read the Turkish AES dataset and return two lists: one containing the texts and one containing the labels. The Turkish AES dataset is a collection of texts in the Turkish language and their corresponding AES scores.

Figure 2. *Loading datasets.*

```
def read_texts(path=ADDR_TURKISH):
    with open(path, "rb") as file:
        dataset_turkish = pickle.load(file)
    texts = [item[0] for item in dataset_turkish])
    labels = [round(float(item[1]) for item in dataset_turkish])

    return texts, labels
```

[†] https://www.kaggle.com/datasets/savasy/ttc4900

3.3. Data Preprocessing

For tokenization, we can use either mBERT Tokenizer or one of the existing Turkish-specific language models, such as BERTurk (https://huggingface.co/dbmdz/bert-base-turkish-cased), using the codes in Figure 3.

Figure 3. *Data preprocessing*.

```
 tokenizer = AutoTokenizer.from_pretrained(
    " bert-base-multilingual-cased"
)
Or
tokenizer = AutoTokenizer.from_pretrained(
    " dbmdz/bert-base-turkish-cased"
)
```

Text tokenization is the process of splitting a text into smaller units, such as words or subwords, that can be mapped to numerical representations. Different methods of text tokenization for transformers are:

Byte-Pair Encoding (BPE): This method uses a statistical algorithm to learn a fixed-size vocabulary of subword units from a large corpus of text. It starts with a set of characters as the initial vocabulary and then iteratively merges the most frequent pair of symbols until the vocabulary reaches the desired size. BPE can handle rare or unknown words by breaking them into smaller subwords. BPE is used by models such as GPT-2 and RoBERTa1.

WordPiece: This method is similar to BPE, but instead of merging the most frequent pair of symbols, it merges the pair that maximizes the likelihood of the data. WordPiece also uses a special symbol to mark the beginning of a word so that it can distinguish between different occurrences of the same subword in different words. WordPiece can also handle rare or unknown words by breaking them into smaller subwords. WordPiece is used by models such as BERT and DistilBERT1.

SentencePiece: This method is a generalization of BPE and WordPiece, which can operate on raw texts without pre-tokenization or pre-segmentation. SentencePiece can learn a vocabulary of subword units from any language and encode texts into sequences of subwords or characters. SentencePiece can also handle rare or unknown words by breaking them into smaller subwords or characters. SentencePiece is used by models such as ALBERT and XLNet1. An example of how a word piece tokenizer—which is used in our sample—might work on a Turkish language sentence is as follows. In the sentence “Türkiye'nin en büyük şehri İstanbul'dur,” means “The largest city of Turkey is Istanbul,” the word piece tokenizer would first split the sentence into words by whitespace as: [Türkiye'nin, en, büyük, şehri, İstanbul'dur.]

3.4. Model Training

BERTurk[‡] is a cased BERT model for the Turkish language that can be used for tokenization. The model is trained on various free-access Turkish corpora, including a filtered and sentence-segmented version of Turkish open parallel corpus (OPUS), OSCAR corpus, and a special local corpus. The final training corpus has a size of 35GB and 44,04,976,662 tokens§. The following codes can be used for tokenization using BERTurk.

To load the BERTurk model using the Hugging Face’s transformers package, we need to use the AutoModel and AutoTokenizer classes from the transformers module. Given the model’s

[‡] https://github.com/stefan-it/turkish-bert
[§] https://huggingface.co/dbmdz/bert-base-turkish-cased

name or path, these classes can automatically load any model from the huggingface model hub. The BERTurk model is available on the model hub under "dbmdz/bert-base-turkish-cased." We can load this model using the codes in Figure 4.

Figure 4. *Importing a pretarined model.*

```
 from transformers import AutoModel, AutoTokenizer


tokenizer = AutoTokenizer.from_pretrained("dbmdz/bert-base-turkish-cased")
model = AutoModel.from_pretrained("dbmdz/bert-base-turkish-cased")
```

The codes in Figure 5 is an example of how to train the BERTurk model using PyTorch. The code defines the loss function, optimizer, and scheduler for the training process. The loss function is a cross-entropy loss, which measures how well the model predicts the correct class for each text. The optimizer is an AdamW optimizer–a stochastic gradient descent method that updates the model parameters based on the gradients of a loss function. The scheduler is a linear schedule with a warmup, which adjusts the learning rate during the training process. The code also defines a training loop, which iterates over the batches of data and labels, feeds them to the model, computes the loss, and updates the model parameters using the optimizer and the scheduler.

Cross-entropy loss is a standard loss function used in machine learning, especially for classification tasks. It measures how well a model predicts the correct class for a given input by comparing the probability distribution output of the model with the true distribution of the classes. The lower the cross-entropy loss, the better the model predicts the correct class. In the context of AES, cross-entropy loss can be used to train a model that assigns scores to essays as an alternative to human grading. AES is a challenging task that requires a model to understand the content, structure, and style of an essay, and to compare it with a predefined rubric or criteria. One way to approach this task is to formulate it as a classification problem, where each possible score is treated as a class. For example, if the scoring scale is from one to fix, then there are six classes to predict.

Figure 5. *Model training.*

```
# Define data and labels
texts = ["Bu bir örnek cümledir.", "Bu başka bir örnek cümledir.", ...] # Your texts here
labels = [0, 1, ...] # Your labels here


# Encode data and labels
inputs = tokenizer(texts, padding=True, truncation=True, return_tensors="pt")
labels = torch.tensor(labels)


# Create data loader
batch_size = 32
data_loader = DataLoader(list(zip(inputs["input_ids"], inputs["attention_mask"], labels)), batch_size=batch_size)


# Define loss function, optimizer, and scheduler
loss_fn = CrossEntropyLoss()
```

```
optimizer = AdamW(model.parameters(), lr=2e-5)
total_steps = len(data_loader) * epochs
scheduler = get_linear_schedule_with_warmup(optimizer,
num_warmup_steps=0, num_training_steps=total_steps)

# Define training loop
epochs = 4
for epoch in range(epochs):
  # Train model on batches of data
  for batch in data_loader:
    # Get batch data and labels
    input_ids, attention_mask, labels = batch

    # Forward pass
    outputs = model(input_ids=input_ids, attention_mask=attention_mask)
    logits = outputs[0]

    # Compute loss
    loss = loss_fn(logits, labels)

    # Backward pass and update parameters
    loss.backward()
    optimizer.step()
    scheduler.step()

    # Reset gradients
    optimizer.zero_grad()
```

3.5. Model Evaluation

The codes in Figure 6 can be implemented to evaluate the BERTurk model on the AES dataset. The code defines the evaluation metrics and writes a function to compute them for a given data loader and model. The code uses accuracy, F1-score, QWK, or Kappa as the evaluation metrics, which measure how well the model predicts the scores of the essays. The code then writes a function that loops over the batches in the data loader, computes the logits (output scores) of the model, gets the predicted labels by taking the argmax of the logits, and calculates the metrics for the predictions and the true labels. The code then runs this function on the test and validation sets and prints the results.

The evaluation metrics used in the context of automated essay scoring are measures of how well the automated system can mimic human raters in grading essays. Each metric captures a different aspect of writing quality and can be used to compare the performance of different models or systems. Here is a brief explanation of why each metric was used:

Figure 6. *Model Evaluation.*

```
# Import the libraries
from sklearn.metrics import accuracy_score, f1_score, cohen_kappa_score

# Define the evaluation metrics
metrics = {"accuracy": accuracy_score, "f1": f1_score, "kappa": cohen_kappa_score}

# Write a function to compute predictions for a given data loader and model
def evaluate(dataloader, model):
  # Set the model to evaluation mode
  model.eval()
  # Initialize empty lists to store the predictions and the true labels
  preds = []
  truths = []
  # Loop over the batches in the data loader
  for batch in dataloader:
    # Get the inputs and labels from the batch
    input_ids = batch["input_ids"]
    attention_mask = batch["attention_mask"]
    labels = batch["labels"]
    # Move them to the device (cpu or gpu)
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    input_ids = input_ids.to(device)
    attention_mask = attention_mask.to(device)
    labels = labels.to(device)
    # Compute the logits (output scores) with no gradient calculation
    with torch.no_grad():
      outputs = model(input_ids, attention_mask)
      logits = outputs.logits
    # Get the predicted labels by taking the argmax of the logits
    pred_labels = torch.argmax(logits, dim=1)
    # Append the predictions and the true labels to the lists
    preds.extend(pred_labels.tolist())
    truths.extend(labels.tolist())
  # Compute the metrics for the predictions and the true labels
  results = {}
  for name, metric in metrics.items():
    results[name] = metric(truths, preds)
  # Return a dictionary of results
  return results

# Run the evaluation function on the test and validation sets and print the results
test_results = evaluate(test_loader, model)
valid_results = evaluate(valid_loader, model)
print("Test results:")
print(test_results)
print("Validation results:")
print(valid_results)
```

Accuracy: Accuracy is the simplest and most intuitive metric. It measures how often the automated system assigns the same score as the human rater. Accuracy is easy to calculate and interpret, but it does not account for the variability or agreement among human raters, nor does it reflect the severity of errors made by the system.

F1-score: The F1-score is the harmonic mean of precision and recall. Precision measures how many of the essays scored by the system are correct, while recall measures how many of the correct essays are scored by the system. F1-score balances both aspects and gives a higher score to precise and recall-oriented systems. F1-score is useful for evaluating systems that assign binary or categorical scores, such as pass/fail or low/medium/high.

QWK or Kappa: QWK or kappa is a measure of agreement between two raters that accounts for the chance agreement. It compares the observed agreement with the expected agreement under random scoring. QWK or kappa ranges from -1 to 1, where 1 means perfect agreement, 0 means no agreement beyond chance, and negative values mean worse than a chance agreement. QWK or Kappa is useful for evaluating systems that assign ordinal or numerical scores, such as 1 to 6 or 0 to 100. It also takes into account the magnitude of disagreement, such that a small difference in scores is less penalized than a large difference.

4. DISCUSSION and CONCLUSION

Pre-trained language models, encompassing both word embedding techniques and transformer-based architectures, present a robust foundation for harnessing extensive knowledge to enhance the efficacy and efficiency of AES models (Singh & Mahmood, 2021). This is especially pertinent in scenarios where data is scarce or challenging to procure. BERT-based language models, in particular, offer a versatile and potent framework for capitalizing on large-scale pre-training to optimize the performance of AES models, even in resource-constrained environments.

The inherent flexibility of BERT-based models extends beyond mere performance enhancement (Devlin et al., 2018). These models facilitate knowledge transfer across languages by undergoing training on a shared set of parameters. Subsequently, this knowledge can be fine-tuned for specific languages or tasks. Our paper succinctly encapsulates key language models that have transformative implications for AES applications in both English and non-English contexts. Furthermore, we expound upon the practical application of mBERT in the Turkish language, displaying its adaptability across linguistic landscapes. As a forward-looking proposition, this research lays the groundwork for future endeavors to implement the methodologies outlined herein. Researchers can utilize the provided codebase to analyze essays written in Turkish, potentially culminating in the development of the inaugural Turkish AES system employing large language models."

Future studies can explore the applicability of alternative transformer-based models, including LaBSE and GPT, to assess their efficacy within the Turkish language. Furthermore, delving into the ramifications of domain-specific fine-tuning on these models' performance in the realm of Turkish essay scoring holds promise for yielding valuable insights.

The scalability of the proposed methodology across diverse languages, coupled with its adaptability to various educational levels and essay genres, opens compelling avenues for subsequent research. Undertaking comparative studies that scrutinize different language models in terms of computational efficiency, interpretability, and bias mitigation could significantly contribute to honing the selection of models tailored for specific AES applications (Yang et al., 2020).

In conclusion, the substantial language models expounded upon in this study serve as a springboard for future AES research across a spectrum of linguistic and educational contexts. Harnessing the capabilities of large language models can empower researchers to actively contribute to the evolution of sophisticated and flexible AES systems, effectively tackling the distinct challenges posed by diverse languages and educational landscapes.

Declaration of Conflicting Interests and Ethics

The authors declare no conflict of interest. This research study complies with research publishing ethics. The scientific and legal responsibility for manuscripts published in IJATE belongs to the authors.

Contribution of Authors

Tahereh Firoozi: Investigation, Resources, Visualization, Software, Formal Analysis, and Writing-original draft. **Okan Bulut**: Methodology, Supervision, and Writing-original draft. **Mark J. Gierl**: Methodology, Supervision, and Writing-original draft.

Orcid

Tahereh Firoozi https://orcid.org/0000-0002-6947-0516
Okan Bulut https://orcid.org/0000-0001-5853-1267
Mark J. Gierl https://orcid.org/0000-0002-2653-1761

REFERENCES

Akın, A.A., & Akın, M.D. (2007). Zemberek, an open source NLP framework for Turkic languages. *Structure*, *10*(2007), 1-5.

Arslan, R.S., & Barişçi, N. (2020). A detailed survey of Turkish automatic speech recognition. *Turkish Journal of Electrical Engineering and Computer Sciences*, *28*(6), 3253-3269.

Bird, S. (2006, July). NLTK: the natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions* (pp. 69-72).

Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., ... & Weinbach, S. (2022). Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5, 135–146.

Bouschery, S.G., Blazevic, V., & Piller, F.T. (2023). Augmenting human innovation teams with artificial intelligence: Exploring transformer-based language models. *Journal of Product Innovation Management, 40*(2), 139-153.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., ... & Amodei, D. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33, 1877-1901.

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., ... & Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Cai, D., He, X., Wang, X., Bao, H., & Han, J. (2009, June). Locality preserving nonnegative matrix factorization. In *Twenty-first International Joint Conference on Artificial Intelligence*.

Cetin, M.A., & Ismailova, R. (2019). Assisting tool for essay grading for Turkish language instructors. *MANAS Journal of Engineering*, *7*(2), 141-146.

Chi, Z., Dong, L., Wei, F., Yang, N., Singhal, S., Wang, W., ... & Zhou, M. (2020). InfoXLM: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834*.

Conneau, A., & Lample, G. (2019). Cross-lingual language model pretraining. *Advances in Neural Information Processing Systems*, *32*.

Devlin, J., Chang, M.W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dikli, S. (2006). Automated essay scoring. *Turkish Online Journal of Distance Education*, *7*(1), 49-62.

Firoozi, T., Bulut, O., Epp, C.D., Naeimabadi, A., & Barbosa, D. (2022). The effect of fine-tuned word embedding techniques on the accuracy of automated essay scoring systems using Neural networks. *Journal of Applied Testing Technology*, *23*, 21-29.

Firoozi, T., & Gierl, M.J. (in press). Scoring multilingual essays using transformer-based models. Invited chapter to appear in M. Shermis & J. Wilson (Eds.), *The Routledge International Handbook of Automated Essay Evaluation*. New York: Routledge.

Firoozi, T., Mohammadi, H., & Gierl, M.J. (2023). Using Active Learning Methods to Strategically Select Essays for Automated Scoring. *Educational Measurement: Issues and Practice*, *42*(1), 34-43.

Feng, F., Yang, Y., Cer, D., Arivazhagan, N., & Wang, W. (2020). Language-agnostic BERT sentence embedding. *arXiv preprint arXiv:2007.01852*.

Fleckenstein, J., Meyer, J., Jansen, T., Keller, S., & Köller, O. (2020). Is a long essay always a good essay? The effect of text length on writing assessment. *Frontiers in Psychology*, *11*, 562462.

Gezici, G., & Yanıkoğlu, B. (2018). Sentiment analysis in Turkish. In K. Oflazer & M. Saraçlar (Eds.) *Turkish Natural Language Processing. Theory and Applications of Natural Language Processing* (pp. 255-271). Springer, Cham.

Graesser, A.C., McNamara, D.S., Louwerse, M.M., & Cai, Z. (2004). Coh-Metrix: Analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers*, *36*(2), 193-202.

Han, T., & Sari, E. (2022). An investigation on the use of automated feedback in Turkish EFL students' writing classes. *Computer Assisted Language Learning*, 1-24.

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *Neural Computation*, *9*(8):1735–1780.

Hussein, M.A., Hassan, H., & Nassef, M. (2019). Automated language essay scoring systems: A literature review. *PeerJ Computer Science*, *5*, e208.

Kavi, D. (2020). Turkish Text Classification: From Lexicon Analysis to Bidirectional Transformer. *arXiv preprint arXiv:2104.11642*.

Kenton, J.D.M.W.C., & Toutanova, L.K. (2019, June). Bert: Pre-training of deep bidirectional transformers for language understanding. *In Proceedings of naacL-HLT*, *1*(2).

Koskenniemi K (1983) Two-level morphology: A general computational model for word-form recognition and production. PhD dissertation, University of Helsinki, Helsinki.

Kuyumcu, B., Aksakalli, C., & Delil, S. (2019, June). An automated new approach in fast text classification (fastText) A case study for Turkish text classification without pre-processing. In *Proceedings of the 2019 3rd International Conference on Natural Language Processing and Information Retrieval* (pp. 1-4).

Liu, P., Joty, S., & Meng, H. (2015, September). Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods İn Natural Language Processing* (pp. 1433-1443).

MacNeil, S., Tran, A., Mogil, D., Bernstein, S., Ross, E., & Huang, Z. (2022, August). Generating diverse code explanations using the gpt-3 large language model. In *Proceedings of the 2022 ACM Conference on International Computing Education Research-Volume 2* (pp. 37-39).

Mayer, C.W., Ludwig, S., & Brandt, S. (2023). Prompt text classifications with transformer models: An exemplary introduction to prompt-based learning with large language models. *Journal of Research on Technology in Education*, *55*(1), 125-141.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems, 26*.

Mizumoto, A., & Eguchi, M. (2023). Exploring the potential of using an AI language model for automated essay scoring. *Research Methods in Applied Linguistics, 2*(2), 100050.

Oflazer, K., & Saraçlar, M. (Eds.). (2018). *Turkish natural language processing*. Springer International Publishing.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532-1543).

Ramesh, D., & Sanampudi, S.K. (2022). An automated essay scoring systems: a systematic literature review. *Artificial Intelligence Review, 55*(3), 2495-2527.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI Blog, 1*(8), 9.

Ranathunga, S., Lee, E.S.A., Prifti Skenduli, M., Shekhar, R., Alam, M., & Kaur, R. (2023). Neural machine translation for low-resource languages: A survey. *ACM Computing Surveys, 55*(11), 1-37.

Rodriguez, P.U., Jafari, A., & Ormerod, C.M. (2019). Language models and automated essay scoring. *arXiv preprint arXiv:1909.09482*.

Roshanfekr, B., Khadivi, S., & Rahmati, M. (2017). Sentiment analysis using deep learning on Persian texts. *2017 Iranian Conference on Electrical Engineering (ICEE*).

Singh, S., & Mahmood, A. (2021). The NLP cookbook: modern recipes for transformer based deep learning architectures. *IEEE Access, 9*, 68675-68702.

Uysal, I., & Doğan, N. (2021). How Reliable Is It to Automatically Score Open-Ended Items? An Application in the Turkish Language. *Journal of Measurement and Evaluation in Education and Psychology, 12*(1), 28-53.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems, 30*.

Williamson, D.M., Xi, X., & Breyer, F.J. (2012). A framework for evaluation and use of automated scoring. *Educational Measurement: Issues and Practice, 31*(1), 2-13.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A.M. (2020, October). Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations* (pp. 38-45).

Yang, R., Cao, J., Wen, Z., Wu, Y., & He, X. (2020, November). Enhancing automated essay scoring performance via fine-tuning pre-trained language models with combination of regression and ranking. In *Findings of the Association for Computational Linguistics: EMNLP 2020* (pp. 1560-1569).