Received 1 May 2024, accepted 28 May 2024, date of publication 31 May 2024, date of current version 7 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3407983

RESEARCH ARTICLE

BERT2D: Two Dimensional Positional Embeddings for Efficient Turkish NLP

YİĞİT BEKİR KAYA AND A. CÜNEYD TANTUĞ
Computer Engineering Department, Istanbul Technical University, Sarıyer, 34468 İstanbul, Turkey
Corresponding author: Yiğit Bekir Kaya (kayayig@itu.edu.tr)

ABSTRACT This study addresses the challenge of improving the downstream performance of pretrained language models for morphologically rich languages, with a focus on Turkish. Traditional BERT models use one-dimensional absolute positional embeddings, which, while effective, have limitations when dealing with complex languages. We propose BERT2D, which is a novel BERT-based model that contributes to positional embedding systems. This approach introduces a dual embedding system that targets all the words and their subwords. Remarkably, this modification, coupled with whole word masking, resulted in a significant increase in performance despite a negligible increase in the parameters. Our experiments showed that BERT2D consistently outperformed the leading Turkish-focused BERT model, BERTurk, in terms of various performance metrics in text classification, token classification, and question-answering downstream tasks. For a fair comparison, we pretrained our BERT2D language model on the same dataset as that of BERTurk. The results demonstrate that two-dimensional positional embeddings can significantly improve the performance of encoder-only models in Turkish and other morphologically rich languages, suggesting a promising direction for future research in natural language processing.

INDEX TERMS Transformer models, BERT, BERT2D, sentiment analysis, named entity recognition, question answering, Turkish, NLP, positional embeddings, positional encoding.

I. INTRODUCTION

Transformer-based [1] large language models, such as BERT [2], BART [3] and GPT [4], and the models derived from them have been used as de facto tools in almost all natural language processing (NLP) tasks for many years. Their outstanding ability to represent the meaning of words in a sentence as vectors has led to easy learning of the latent semantics of words by training on massive unlabeled text corpora, which are widely available for almost any language. These models require words (more specifically, tokens generated by the tokenizer) and learn the internal interactions of words through self-attention layers. Because the basic self-attention mechanism does not consider word positions, these models resort to augment the input word vectors with position vectors that indicate the absolute position of the word in the sentence or text. Transformer-based models have achieved most of the state-of-the-art results using learned absolute position embeddings for many downstream NLP tasks, including named entity recognition (NER), part-of-speech (POS) tagging, sentiment analysis, and question answering, to name a few.

The order of the words in a sentence is crucial for English, since the positions of the words together with the prepositions determine their syntactic functions. However, some languages, such as Turkish, have almost free word order, or more precisely, free constituent order, which makes the word position less important. Consider the following example, which presents several Turkish sentences that convey the same meaning (*my little brother came home from school* in English) with slight differences in emphasis:

```
Küçük kardeşim okuldan eve geldi.
Küçük kardeşim eve okuldan geldi.
```

The associate editor coordinating the review of this manuscript and approving it for publication was Jolanta Mizera-Pietraszko.

`Okuldan eve küçük kardeşim geldi.`
`Okuldan eve geldi küçük kardeşim.`

The syntactic functionalities of the constituents in these sentences are determined by the affixes attached to the words (usually the last word of the constituent). While the word order within the constituent is important, the order of the constituents in a sentence is not important for Turkish. For example, while *küçük kardeşim* translates into English as *my little brother*, *kardeşim küçük* translates into English as *my brother is small*, which rebders the arbitrary word order within the constituent invalid. This fact can be interpreted as input word position embeddings still being beneficial for the transformer architecture for Turkish, even though it is a free constituent order language.

On the other hand, Turkish and many other languages, such as Finnish and Hungarian, have complex agglutinative morphological structures, where several suffixes can be added to words. These suffixes can cause the word to undergo inflectional or derivational changes, which eventually makes the vocabulary size large, even theoretically unlimited. Because transformer-based architectures use fixed-size vocabularies, they incorporate various tokenization methods to mitigate out-of-vocabulary (OOV) words when the actual vocabulary is too large. Common tokenization methods such as WordPiece [5] and BPE [6], break words into subwords.

For example, the English word *unbelievable* can be tokenized as `un #believ #able`, with three subwords. The granularity of tokenization (i.e., the number of subwords generated after tokenization) is much higher for agglutinative languages [7] because a wordform can consist of a root word and several suffixes. An example Turkish wordform *göremediklerimizden* (English: *(of) what we cannot see*) is split into `göre #me #dik #lerimiz #den`.

In the basic approach of the Transformers tokenization procedure, all tokens (mostly subwords in the Turkish case) are treated as separate words in a sentence, and their positions are used in the input embedding computation by completely ignoring their relationship with the wordform, root, and other subwords of the wordform, although the last subword is very important in determining the part of speech of a word. The positional encoding used in the original Transformers paper and later in BERT relies on a linear, one-dimensional positional embedding that is absolute and does not distinguish between whole words and tokenized subwords. Treating whole words and subwords in the same way has little impact in languages such as English, where tokenization granularity is low. However, for agglutinative languages, this tokenization granularity is higher for Turkish, which means that the treatment of subwords and whole words is more critical. Therefore, we hypothesized that if we replace these 1D positional embeddings with two positional embeddings, one for whole word positions and one for subword positions relative to the whole word to which it belongs, we will better capture the effect of word position. To test our hypothesis, we introduced a new BERT variant called BERT2D.

Masked language models replace some of the tokens with mask tokens, and their training task is to predict masked tokens. This approach makes BERT a successful language model that captures the representation of tokens in a self-supervised manner. However, this approach does not distinguish between the subwords and whole words during masking. For example, the word *misconception* can be tokenized as `mis` + `##concept` + `##ion`, and if the first token `mis` is masked, it becomes trivial to predict the token. Because the information gain is reduced by not distinguishing between whole words and subwords, the whole word masking approach is introduced in the BERT repository as a masking strategy that improves the results even for the English language. This improvement is expected to be much more significant for languages with compound or agglutinated words, such as Turkish, German, and Chinese. In Turkish, sentences with agglutinated words are ubiquitous like miniature sentences, so that a single conjugated word can consist of an entire English sentence. An extreme example is *Muvaffakiyetsizleştiriveremeyeleceklerimizdenmişsinizcesine*, which translates into English as: *as if you were one of those we would not be able to turn into a maker of.* Because of this property of Turkish, masking part of the agglutinated word makes the mask prediction (cloze) task trivial. This reduces the information obtained to a greater extent than in many other non-agglutinative languages.

In this study, we empirically validated the effectiveness of BERT2D in various NLP tasks by pretraining, fine-tuning, and evaluation with grid search experiments to compare the performance of novel language models with benchmark BERT models (i.e., BERTurk and multilingual BERT). In these experiments, we used a grid search to determine the best configurations for 2D positional encoding and whole word masking. The experiments show that BERT2D, especially when combined with whole word masking, achieves better performance than traditional BERT models in most cases, setting a new state-of-the-art for Turkish NLP applications.

Our contributions are as follows.

(1) We propose a novel positional encoding, which we call 2D positional encoding, that includes word and subword relative positional encodings. We use these positional encodings to construct a BERT model, which we call BERT2D, in which word position embedding and subword relative position embedding replace the 1D absolute positional embedding in BERT.
(2) We trained BERT and BERT2D models with and without whole word masking and BERT2D with different hyperparameters and evaluated the models on various text classification, token classification, and question-answering tasks.

II. RELATED WORK

A. POSITIONAL EMBEDDINGS

Positional embeddings have been extended in many ways to accommodate various modalities and encoding paradigms.

For example, [8] introduced Continuously Augmented Positional Augmentations (CAPEs), which exploit continuous data augmentation to enhance a model's generalization capabilities in various domains, including speech and images. Similarly, [9] extended traditional sinusoidal positional encodings by incorporating complex-valued embeddings and adding nuanced encoding capabilities. Reference [10] explored the limitations of traditional sinusoidal encodings and proposed alternative encodings based on a functional analysis.

Research has also been conducted on relative position embeddings. Reference [11] used an additional attention mechanism for scalar biases to improve the standard benchmarks of GLUE and SuperGLUE. Another paper, called KERPLE [12], introduced an expressive kernel-based framework for the nature of positional embeddings with speed and perplexity higher than any other contemporary counterpart. Reference [13] found that by multiplicatively combining positional information, they could improve the standard tasks of benchmarks, including SQuAD 1.1 and SQuAD 2.0 with 'Method 4' and M4M. Reference [14] proposed TUPE to solve the drawbacks observed in the positional encodings of BERT. In this design, different projection matrices were used for words and positions to improve the expressiveness of the model. Reference [15] proposed ALiBi, which mitigates bias by linearly adding bias to the attention score and does not require additional parameters from classical positional embeddings.

Reference [16] proposed Rotary Positional Embeddings (RoPE), which use a rotation matrix to encode positions. This method provides a stable coding process and has been shown to outperform traditional methods in text classification tasks.

Reference [17] extended the self-attention mechanism by incorporating dependency trees as structural positional representations. This incorporation significantly improved the translation tasks such as NIST Chinese to English and WMT14 English to German.

In contrast, [18] questioned the need for explicit positional encodings. The NoPos models show that competitive performance can be achieved across a range of benchmarks and sequence lengths even without learned positional embeddings.

B. WHOLE WORD MASKING

Whole word masking (WWM) has attracted attention because of its effectiveness in various languages and domains. Reference [19] explored its utility in the morphologically complex Turkish language, particularly in the banking sector, and found that the overall increase in the accuracy of WWM is particularly beneficial when traditional token-based approaches fail. Similarly, [20] and [21] focused on the Chinese language, highlighting the advantages of WWM in mitigating the limitations of partial token masking and improving NER by reducing the negative impact of random masking.

In the German context, [22] found that WWM significantly improves BERT model performance in tasks by enhancing the training signal. Reference [23] also emphasized the role of WWM in Chinese BERT models for grammatical error correction, although its effectiveness levels off in downstream sentence-level tasks. Reference [24] uniquely applied WWM in the specialized domain of energy marketing for NER, emphasizing its ability to effectively handle terms with multiple meanings. Finally, [25] applied WWM as an advanced strategy in Masked Language Models (MLM), increasing the complexity of the task and improving the robustness of the model, a feature that proved beneficial to various Chinese NLP datasets. Despite its widespread use, the effectiveness of WWM varies depending on the language and the specific NLP task at hand, warranting further investigation into its limitations and potential for optimization.

III. PRELIMINARIES

In the original Transformers paper [1], the output of the attention layer is computed as z_i^l for each layer l and position i, as shown in (1). x_i^l is the contextual representation of the attention layer input for each layer. The attention scores α_{ij} were computed for each of these layers, as shown in (2).

$$\mathbf{z}_i^l = \sum_{j=1}^{n} \frac{\exp(\alpha_{ij})}{\sum_{j'=1}^{n} \exp(\alpha_{ij'})} (\mathbf{x}_j^l \mathbf{W}_l^V), \tag{1}$$

$$\text{where} \quad \alpha_{ij} = \frac{1}{\sqrt{d}} (\mathbf{x}_i^l \mathbf{W}_l^Q)(\mathbf{x}_j^l \mathbf{W}_l^K)^T. \tag{2}$$

A. ABSOLUTE POSITIONAL EMBEDDINGS

Note that these results do not depend on the positional embeddings. The original Transformers paper suggested using positional embeddings only for the first layer, that is, the input layer, replacing $x_i^1 = w_i + p_i$. The formula is given by (3).

$$\alpha_{ij}^{Abs} = \frac{1}{\sqrt{d}} \left((\mathbf{w}_i + \mathbf{p}_i) \mathbf{W}^{Q,1} \right) \left((\mathbf{w}_j + \mathbf{p}_j) \mathbf{W}^{K,1} \right)^T \tag{3}$$

This formula represents the absolute attention scores α_{ij}^{Abs} in a self-attention mechanism, incorporating both the word embeddings $\mathbf{w}$ and position embeddings $\mathbf{p}$, with the query and key transformations $\mathbf{W}^{Q,1}$ and $\mathbf{W}^{K,1}$, respectively. Dimension d is typically the size of the query and key vectors, and the square root serves as a scaling factor to control the variance of the dot products.

For sinusoidal position encoding, the p_i becomes a periodic function, depending on the word position and dimension as shown in (4). To provide a mathematical explanation of sinusoidal positional encoding, we consider the formula used to calculate the positional encoding for each position i and dimension t.

$$\begin{aligned} \mathbf{p}_{i,2t} &= \sin\left(\frac{i}{10000^{\left(\frac{2t}{d_{\text{model}}}\right)}} \right) \\ \mathbf{p}_{i,2t+1} &= \cos\left(\frac{i}{10000^{\left(\frac{2t}{d_{\text{model}}}\right)}} \right) \end{aligned} \tag{4}$$

Using these sinusoidal functions, positional encodings for different positions and dimensions can be calculated, providing a unique value for each position in the input sequence. However, the limitations of this method in capturing the complexities of Turkish have been acknowledged, leading to research on alternative positional encoding mechanisms that better accommodate the linguistic characteristics of Turkish. For this purpose, BERT uses the learned absolute positions instead of periodic functional values.

B. 2D POSITIONAL EMBEDDINGS

When we replace p_i in absolute positional embedding further with $\mathbf{p}_{2D_i}$, that is, two-dimensional positional embedding, we obtain (5) for the self-attention score calculation.

$$\alpha_{ij}^{2D} = \frac{1}{\sqrt{d}} \left((\mathbf{w}_i + \mathbf{p}_{2D_i})\mathbf{W}^{Q,1} \right) \left((\mathbf{w}_j + \mathbf{p}_{2D_j})\mathbf{W}^{K,1} \right)^T \quad (5)$$

The $\mathbf{p}_{2D_i}$ encoding is a linear combination of two other embeddings, which are the whole word position embedding $\mathbf{p}_{e_i}$ and the relative subword position embedding $\mathbf{p}_{s_i}$ denoted in (6).

$$\mathbf{p}_{2D_i} = \mathbf{p}_{e_i} + \mathbf{p}_{s_i} \quad (6)$$

IV. 2D POSITIONAL ENCODING

A word can be split into subwords depending on the granularity of the tokenization. If a word is not tokenized, the number of tokens per word was only one. If a word was tokenized into two subwords, the number of tokens per word is two, with a start token and an end token. If a word is tokenized into more than two subwords, we call the subwords between the start and end tokens intermediate subwords and denote their number by m. A word tokenized into fewer than three tokens has $m = 0$.

In Fig. 1, we compare 1D absolute positional encoding with 2D positional encoding consisting of whole word positional encoding denoted by E and subword positional encoding denoted by S. Note that relative subword position encoding changes according to the maximum number of intermediate subword positions to encode, denoted by M. If M is one, then all intermediate subwords are encoded as two, regardless of the number of intermediate subwords. If M is greater than one, the intermediate subword encodings are evenly distributed, as shown in (9).

Note that the whole word positional encoding e_i is the same for every subword of the same word, as depicted in (8), and the subword positional encoding of the beginning subword of each word is always equal to 0. If there is more than one subword per word, the encoding of the last subword of each word is always equal to one. In this way, we ensure that we embed the last subword of each word with the same vector so that the model learns what the first and last subwords mean for a word. These two encodings will create two learned embeddings that will make a two-dimensional separation between words, where, for each word, a word positional embedding vector is learned. In addition, a relative subword

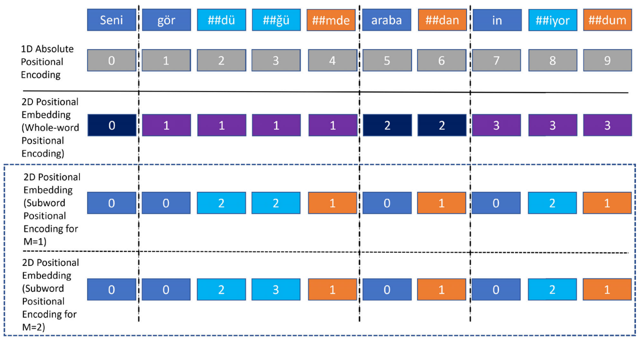

Seni
gör
##dü
##gü
##mde
araba
##dan
in
##iyor
##dum
1D Absolute Positional Encoding
0 1 2 3 4 5 6 7 8 9
2D Positional Embedding (Whole-word Positional Encoding)
0 1 1 1 1 2 2 3 3 3
2D Positional Embedding (Subword Positional Encoding for M=1)
0 0 2 2 1 0 1 0 2 1
2D Positional Embedding (Subword Positional Encoding for M=2)
0 0 2 3 1 0 1 0 2 1

FIGURE 1. The encoding of a Turkish sentence using two different methods - 1D Absolute Positional Encoding (the original BERT encoding [2]) and 2D positional encoding proposes in this work. The 2D positional encoding consists of one whole word positional encoding and one subword positional encoding, which is shown for two different values of *M*: *M* = 1 and *M* = 2. (English: I was getting out of the car when I saw you.)

position embedded relative to the word vector was learned. The model learns a two-dimensional representation of the word position feature as shown in Fig. 2.

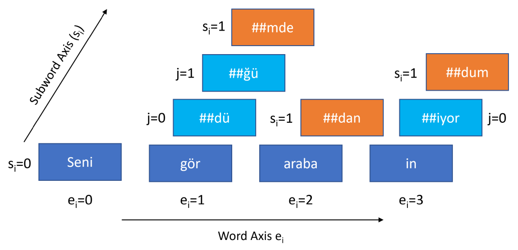


FIGURE 2. How two-dimensional information is represented using the Word Axis (e_i) and Subword Axis (s_i). The dark blue boxes represent whole words or the first subword of a word. Light blue boxes represent intermediate subwords, whereas orange boxes represent the last subword in a word. The token index is denoted by *i* and is independent of dimension, whereas *j* is the index of intermediate subwords used in (9) to calculate subword encoding. (English: I was getting out of the car when I saw you.)

In Fig. 3, we show that changing vocabulary size leads to different tokenizations. Note that for each sentence, the 1D absolute positional encodings change almost randomly, but the word position embeddings remain the same for each tokenization, so the word position encoding is tokenization-agnostic. The subword positional encoding changes with different tokenization methods but again remains relative to the word. These properties of the 2D encoding make it more robust to hyperparameter changes that influence the tokenization of a word, making positional encoding more versatile than 1D positional encoding.

$$\begin{aligned} T &= \{t_1, t_2, \ldots, t_n\} \\ E &= \{e_1, e_2, \ldots, e_n\}, \quad e_i \in \mathbb{N} \\ S &= \{s_1, s_2, \ldots, s_n\}, \quad s_i \in \mathbb{N} \end{aligned} \quad (7)$$

$$e_i = \begin{cases} 0 & \text{if } t_i \text{ belongs to the first word} \\ e_{i-1} + 1 & \text{if } t_i \text{ belongs to a new word} \\ e_{i-1} & \text{otherwise} \end{cases} \quad (8)$$

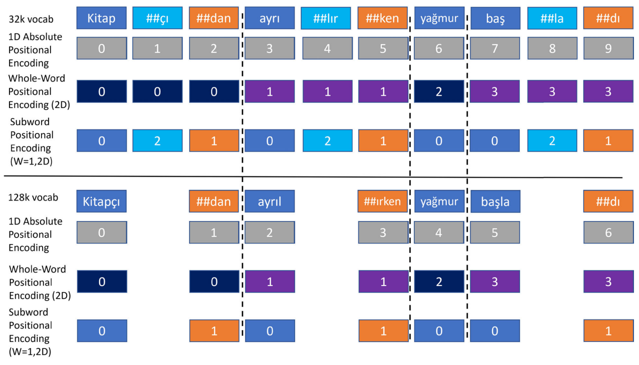


FIGURE 3. **How vocabulary size affects the encodings. Above, we show how a sentence tokenized with 32k vocabulary is encoded, comparing original 1D absolute positional encoding with 2D positional encoding. When we change the tokenization vocabulary size to 128k, the whole word positional embedding does not change for words. However, the subword positional embedding changes while staying relative to its position in a word. The last subwords are always encoded with 1 (orange), and the first subword or the whole word is always encoded with 0 (dark blue) for subword positional encoding. The intermediate subwords (light blue) disappear when the vocabulary size goes from 32k to 128k, due to decreased tokenization granularity with increased vocabulary size. (English: It started raining as I was leaving the bookstore)**

$$s_i = \begin{cases} 0 & \text{if } t_i \text{ is the first subword} \\ 1 & \text{if } t_i \text{ is the last subword} \\ 2 + \left\lfloor \dfrac{j \times M}{m} \right\rfloor & \text{if } m > M \\ j & \text{otherwise} \end{cases} \tag{9}$$

In (7), t_i represents the i-th token in an input text with n tokens. In (8), e_i is the word position IDs, and in (9), s_i is the subword position IDs, m is the number of intermediate subwords, and M is the maximum number of intermediate subword positions encoded per word. The i-th index is the token index in the sentence, and the j-th index is the intermediate subword index in a word (see Fig. 2).

For example, if there is only one token per word, the subword position ID for that word is 0. If only two tokens exist, the subword positional encoding for that word is always [0, 1]. However, if there are more than two tokens per word, subword positional encoding depends on M. If M is one and the word consists of three subwords, there is only one intermediate subword. Thus, the subword encoding of this word is [0, 2, 1]. Note that the last token is always encoded as one because encoding the last subword of a word with one, regardless of M or the number of subwords, increases attention to the last token, which is more appropriate for Turkish.

Positional encoding IDs function as indices for a vector lookup table. Because the whole word positional encoding e is the same for all subwords of a word, they contribute to the same embedding. Thus, instead of using $\mathbf{p}_i$ as the embedding vector as in absolute positional embedding, we use a linear combination of two embeddings $\mathbf{p}_{e_i}$ and $\mathbf{p}_{s_i}$, which we denote as $\mathbf{p}_{2D_i}$, represented in (6).

Note that for the 768 intermediate layer size, we add $(M + 2) * 768$ parameters to the model, which are only $(1+2)*768 = 2304$ for $M = 1$ and $(10+2)*768 = 9216$ for $M = 10$, which are negligible compared with the sheer size of the entire BERT model. In general, the number of parameters in the model does not change for the BERT2D family models; therefore, there is no additional training cost.

V. WHOLE WORD MASKING

The BERT language model learns by optimizing two tasks simultaneously: masked token prediction (or Cloze task) and next sentence prediction. However, modern BERT variations have dropped the next sentence prediction in favor of only the Cloze task, making new BERT models MLMs only.

Initially, the authors of BERT decided to use random word masking (RWM) to mask a portion of tokens with a default masking probability of 15% for self-supervised learning. This approach eliminates the need for human data annotators, because it can automatically generate annotations. The task then becomes to predict the masked tokens. However, the original authors of the BERT model added whole word masking with an update to the model's repository[1] so that the model can mask only whole words instead of partial tokens, improving performance even for the English language.

In Table 1, the first sentence has 14 tokens, and with a word masking probability of 15%, the masking algorithm should mask two tokens. Using a random word masking approach, any two tokens can be masked, whether partial or whole words. Note that in the first sentence, large compound words are broken down into many partial words. Partial masking of compound words provides no additional information during pretraining for masked token prediction; however, when whole words are masked, the words '*yarın*' (*tomorrow* in English) and '*ben*' (*me* in English) are learned. With whole word masking, it is impossible to partially mask long compound words, which makes learning more coherent.

The results were similar for the second sentence because there were no long compound words. Nevertheless, random word masking can mask only a single subword, and compared to whole word masking, it is more challenging for the masked language model to predict if the token is a partial token or a whole word.

VI. EXPERIMENTAL EVALUATION AND RESULTS

A. EXPERIMENTAL SETUP

We pretrained, fine-tuned, and evaluated the language models according to the pipeline shown in Fig. 4.

1) BERTURK CORPUS

We pretrained all our language models using the corpus curated in [26] (which we refer to as *the BERTurk corpus*) to train the BERTurk language model in the same work. The BERTurk corpus combines and processes four datasets: the OSCAR corpus [27], a recent Wikipedia dump, OPUS corpora [28], and an unpublished corpus by Kemal Oflazer. Because the BERTurk corpus is curated from various sources

[1]https://github.com/google-research/bert/commit/0fce551

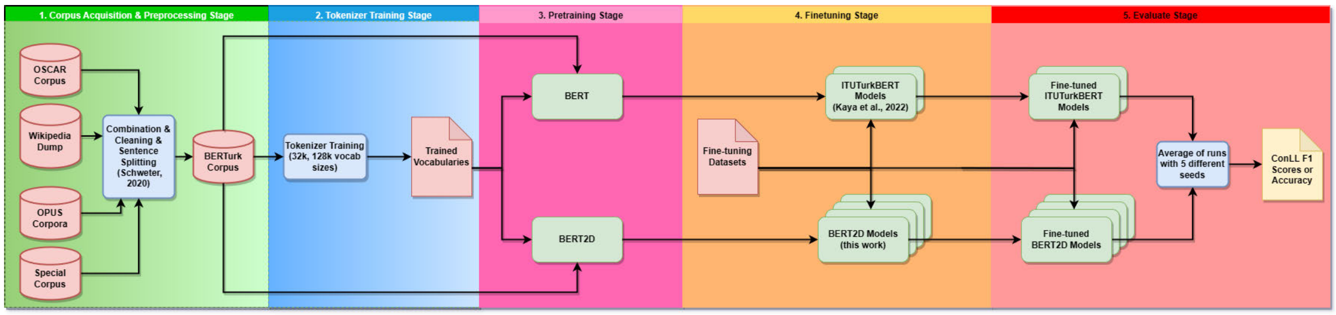

1. Corpus Acquisition & Preprocessing Stage
2. Tokenizer Training Stage
3. Pretraining Stage
4. Finetuning Stage
5. Evaluate Stage
OSCAR Corpus
Wikipedia Dump
OPUS Corpora
Special Corpus
Combination & Cleaning & Sentence Splitting (Schweter, 2020)
BERTurk Corpus
Tokenizer Training (32k, 128k vocab sizes)
Trained Vocabularies
BERT
BERT2D
Fine-tuning Datasets
ITUTurkBERT Models (Kaya et al., 2022)
BERT2D Models (this work)
Fine-tuned ITUTurkBERT Models
Fine-tuned BERT2D Models
Average of runs with 5 different seeds
ConLL F1 Scores or Accuracy

FIGURE 4. **The five stages of experimenting and evaluating the language models. Fist, we start with pretraining corpus acquisition (BERTurk corpus [26]). Second, we train the vocabularies with the corpus, Thirdly, we pretrain BERT and BERT2D family language models. Finally, we fine-tune and evaluate the pretrained language models with respective downstream task datasets and report results.**

TABLE 1. **Word Masking Comparison.**

Masking Strategy	Tokenized Sentence
Original Text 1	Yarın gel ##di ##ğin ##de beni bura ##da bula ##ma ##ya ##bilir ##sin ##iz.*
Random Word Masking	Yarın gel [MASK] ##ğin ##de beni [MASK] ##da bula ##ma ##ya ##bilir ##sin ##iz.
Whole Word Masking	[MASK] gel ##di ##ğin ##de [MASK] bura ##da bula ##ma ##ya ##bilir ##sin ##iz.
Original Text 2	Sen de mi on ##lar gibi ##sin?**
Random Word Masking	Sen de mi on [MASK] gibi ##sin?
Whole Word Masking	Sen [MASK] mi on ##lar gibi ##sin?

* **English:** You may not find me here when (you come tomorrow/he comes tomorrow/she comes tomorrow/tomorrow comes).
** **English:** Are you like them, too?

that are not manually checked for bias and quality, researchers are advised to consider these biases in the sentences when interpreting the results of this study.

The BERTurk corpus has 300M sentences with 4.4B tokens.

2) VOCABULARY LEARNING

We trained WordPiece vocabulary models using the BERTurk corpus with 32k and 128k vocabulary sizes, each yielding different vocabularies and tokenizations.

3) PRETRAINING

To answer our research questions, we pretrained the novel models based on the following parameters:

- Positional embedding type
- Vocabulary size
- Word masking type
- Training corpus
- Maximum number of intermediate subwords to be mapped (M) for BERT2D family models

Fig. 5 shows the parameters leading to the pretrained model. There are two families of models, BERT and BERT2D. The difference lies in the choice of positional encoding used in the first layer of the BERT architecture. The BERT family models used 1D absolute position embeddings, whereas the BERT2D family models used the proposed 2D position embeddings.

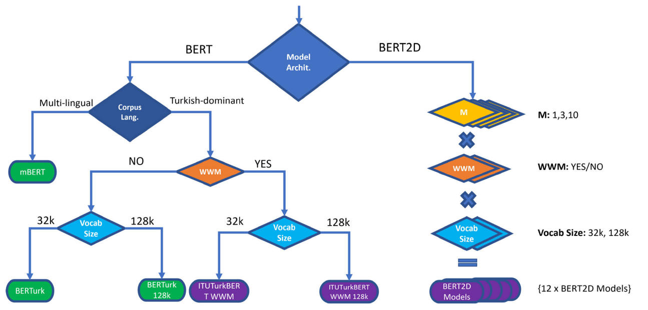


FIGURE 5. **Classification of BERT and BERT2D family models. Green denotes benchmark models, whereas purple denotes the models pretrained for this work. *M* denotes the number of intermediate subwords, and `WWM` denotes whole word masking. Combining three *M* options, two `WWM` options, and two vocab sizes results in 12 BERT2D-family model combinations.**

Additionally, we fine-tuned multilingual BERT (mBERT) as a multilingual baseline, which includes Turkish. Any model other than mBERT was pretrained with the same BERTurk corpus. For our Turkish-dominant BERT family, where random word masking was the previous state-of-the-art baseline, we fine-tuned 32k and 128k BERTurk models.

We used the same dataset as that used for the BERTurk. In addition to the new BERT2D family model that we pretrained, we pretrained two BERT family models with whole word masking to isolate the effect of whole word masking: the 32k and 128k ITUTurkBERT-WWM models (ITUTurkBERT refers to our previous work [7], where we analyzed the effect of tokenization granularity and mainly vocabulary size using the BERTurk corpus).

We pretrained 12 BERT2D-family models, which Were combinations of:

- 32k and 128k vocabulary (2 options)
- Random word masking and Whole Word Masking (2 options)
- Maximum number of intermediate subwords to be mapped (M) for the BERT2D family models (1, 3, and 10; 3 options)

A word can consist of one, two, three, or more subwords. In the first two cases, there were no intermediate subwords. For three or more subwords, we have intermediate subwords, and their count m is calculated as two subtracted by the total subwords for the first and last subwords.

The BERT family models were pretrained using the default configuration of the BERT base architecture. The configuration comprised a Transformer with 12 attention heads, which was used only for decoding. It has 12 hidden layers, each consisting of 768 hidden vector units, and a feedforward intermediate size of 3072. The model had a maximum sequence length of 512 and used GELU activation. Furthermore, the model has a dropout rate of 0.1 for both hidden and attention probabilities.

The hyperparameters used for the BERT2D family of models were the same as those used for the BERT family of models. However, the 1D absolute positional embedding layer was replaced with the 2D positional embedding layer, resulting in a negligible difference in the parameters with respect to M and hidden vector units.

Furthermore, we pretrained some of our language models with whole word masking, and others with random word masking. Furthermore, we pretrained some of our models using a 32k vocabulary and others with a 128k vocabulary.

4) FINE-TUNING

For fine-tuning, we used hyperparameters as close as possible to [29]. The original BERTurk model also used [29] as a reference for hyperparameters. The hyperparameters used to fine-tune our language models are shown in Table 2. We fine-tune the language models using train splits for the downstream tasks of named entity recognition, POS tagging, question answering, and sentiment analysis, and evaluated them using the dev and test splits of the data provided by the respective data repositories.

TABLE 2. Fine-tuning hyperparameters of language models for downstream tasks.

Hyper Parameter	NER	POS	QA	SA
Batch size	16	16	64	64
Epochs	10	10	20	3
Max Sequence Length	512	512	384	256
Doc Stride	N/A	N/A	128	N/A
Max Answer Length	N/A	N/A	64	N/A
Learning Rate	5e-5	5e-5	3e-5	3e-5

B. TEXT CLASSIFICATION

For the text classification task, we fine-tuned all language models, including the novel BERT2D models using a combination of two-dimensional embeddings and whole word masking using the same corpus as BERTurk, the ITUTurkBERT models using whole word masking with the original BERT architecture and using the same corpus as BERTurk, the BERTurk models using the Turkish corpus with the original BERT architecture, and the multilingual BERT which uses a multilingual corpus including Turkish with the original BERT architecture for sentiment analysis on an unbalanced sentiment dataset from HuggingFace.

1) DATASET

For sentiment analysis, we used the "turkish-sentiment-analysis-dataset"[2] provided in the HuggingFace `datasets` repository. The dataset consists of 441k training examples and 49k test examples. Of these 441k train examples, 53.5% were positive, and 11.6% were labeled as negative, collected from various repositories, including the HUMIR dataset prepared for the [30] paper. The remainder of the dataset consists of neutral examples. For binary sentiment analysis (i.e., classification as `Positive` or `Negative`), neutral examples were removed. After up-sampling the negative sentences, there were 235k positive and 50k negative unique sentences in the training split and 26k positive and 5.6k negative unique sentences in the test split. If we remove positive examples to balance negative and positive sentences, we would have lost a considerable amount of valuable data. Instead, we randomly upsampled the negative sentences so that there were an equal number of positive and negative sentences in the training and test sets.

2) HOW WE EVALUATE

In this study, we addressed the imbalance in our Turkish sentiment analysis dataset by increasing the number of negative sentences in the training set, instead of creating synthetic sentences using methods such as SMOTE [31]. This decision was driven primarily by the unique linguistic characteristics of Turkish. Turkish is an agglutinative and morphologically rich language, that poses significant challenges for the generation of syntactically and semantically accurate synthetic sentences. The Turkish language's extensive inflection and derivation make it challenging to generate synthetic sentences that accurately reflect the language's nuances.

Furthermore, techniques such as SMOTE would have introduced an additional layer of dependency on the quality of synthetic sentence generation. This could have led to a misrepresentation of the actual task, which was to compare the performance of language models in processing natural Turkish language data. In this context, using a simpler, more direct approach of upsampling negative sentences ensures

[2]https://huggingface.co/datasets/winvoker/turkish-sentiment-analysis-dataset

that the training data remain grounded in actual language use, thereby avoiding the pitfalls of potentially inaccurate or misleading synthetic data generation.

Additionally, [32] found empirically that balancing is useful in moving the focus of the classifiers towards the minority class, resulting in better prediction of the minority samples and worse prediction of the majority samples. This was found to improve the prediction performance for weak classifiers, but not for strong ones. Given that BERT is a strong classifier, it is reasonable to avoid using the SMOTE.

The upsampling method aligns with established machine learning practices for addressing class imbalances in the datasets. By augmenting the representation of the underrepresented class (in this case, negative sentences), the goal was to train a model that was more sensitive to the intricacies of both positive and negative sentiments in Turkish. The decision to train on this artificially balanced dataset and test on an unaltered, real-world class distribution ensures that the model's performance is evaluated under realistic conditions, providing a more accurate measure of its effectiveness in processing Turkish sentiment analysis. This methodological choice prioritizes linguistic authenticity and practical feasibility, reflecting a nuanced understanding of the constraints and requirements of Turkish language processing.

Sentiment analysis is typically evaluated based on accuracy. However, owing the highly skewed dataset, we also provided the F1 metric, which is a straightforward method for sentence classification (refer to (10)). To calculate F1, the precision and recall must be calculated. Precision is defined as the ratio of correctly predicted positive sentences (True Positive, `TP`) to all sentences predicted to be positive as shown in (11) (i.e., True Positive, `TP` + False Positive, `FP`). Recall is defined as the ratio of true positives to all real positive sentences in the test dataset (True Positive, `TP` + False Negatives, `FN`), as shown in (12). F1 provides a balanced measure of precision and recall for unbalanced datasets.

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{10}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

3) RESULTS

Table 3 shows that while all BERT2D models significantly outperform the mBERT and BERTUrk models, the ITUTurkBERT model, which combines BERT with whole word masking, performs the best. It can be argued that this is because sentiment analysis is a sentence classification task that focuses only on a single special sentence token `[CLS]`; thus, the use of the BERT2D architecture does not contribute. However, whole word masking clearly improved accuracy and F1 values for this task.

TABLE 3. Sentiment Analysis (Unique) Accuracy/F1.

Model	Vocabulary Size			
	32k		128k	
	RWM	WWM	RWM	WWM
mBERT	N/A	N/A	94.60/96.72	N/A
BERTurk	96.04/97.61	N/A	96.02/97.60	N/A
ITUTurkBERT	N/A	96.14/97.66	N/A	**96.19/97.70**
BERT2D ($M = 1$)	**96.16**/97.67	**96.16/97.68**	96.15/**97.68**	96.11/97.65
BERT2D ($M = 3$)	96.11/97.64	96.14/97.66	96.08/97.63	96.15/97.67
BERT2D ($M = 10$)	96.14/97.66	96.15/97.67	96.14/97.67	**96.16/97.68**

C. TOKEN CLASSIFICATION

We fine-tuned all language models for named entity recognition and POS tagging downstream tasks for token classification evaluation.

1) DATASET

The Turkish-WikiNER[3] [33] dataset was utilized for this study due to its high data quality and the inclusion of more than three named entity classes. This dataset contains 18,000 training, 1000 test, and 1000 development sentences, and includes 19 label classes such as `DATE`, `TIME`, `ORG`, and `PERSON`. It is available in the CoNLL format.

The most recent version of the IMST[4] dataset was used for the POS tagging. The dataset is a semi-automatic conversion of the IMST Treebank [34] (and [35]), which in turn is a re-annotated version of the METU-Sabancı Turkish Treebank [36] (and [37]). All three treebanks share identical raw data, consisting of a collection of 5635 sentences obtained from daily news reports and novels.

We also used the BOUN dataset[5] [38], which includes 9,761 manually annotated sentences from various topics, including biographical texts, national newspapers, instructional texts, articles on popular culture and essays for the POS tagging task.

2) HOW WE EVALUATE

The performance of the named entity recognition task was evaluated using the CoNLL F1 score [39]. This score quantifies the accuracy of named entity recognition by balancing precision and recall. Precision is the proportion of correctly identified entities, whereas recall is the proportion of entities that are actually recognized. The CoNLL F1 score is calculated using the harmonic mean of the precision and recall, ensuring that both metrics are given equal weights. This method achieves a balance between the accurate identification and comprehensive detection of entities, making the F1 score a robust measure for evaluating system performance in named entity recognition tasks.

[3]https://github.com/turkish-nlp-suite/Turkish-Wiki-NER-Dataset
[4]https://github.com/UniversalDependencies/UD_Turkish-IMST
[5]https://github.com/UniversalDependencies/UD_Turkish-BOUN

For the POS tagging task, we simply evaluated the accuracy of the dev and test splits and reported the results as dev/test results.

3) RESULTS

Table 4 presents the NER results for the dev/test sets. The `ITUTurkBERT+WWM@128k` model achieves the best dev performance, closely followed by the `BERT2D(M=1)@32k` model. The same BERT2D model achieved the best results for the test split, with significant improvement over the baseline. These results demonstrate that BERT2D and whole word masking consistently outperformed the baseline for different dataset splits.

TABLE 4. NER dev/test F1.

	Vocabulary Size			
	32k		128k	
Model	RWM	WWM	RWM	WWM
mBERT	N/A	N/A	76.53/76.13	N/A
BERTurk	77.03/77.46	N/A	77.06/77.38	N/A
ITUTurkBERT	N/A	77.38/77.54	N/A	**77.40/77.93**
BERT2D ($M=1$)	77.17/77.68	77.30/77.90	**77.39/78.22**	76.54/77.62
BERT2D ($M=3$)	77.24/77.24	77.12/77.46	77.18/77.44	77.31/78.20
BERT2D ($M=10$)	76.94/77.39	77.06/77.41	76.87/77.90	77.35/77.74

Table 5 shows that BERT2D models with whole word masking outperformed the baseline in the POS-tagging tasks. Furthermore, in Table 6, BERT2D significantly outperformed the baseline in dev and matched the accuracy of whole word masking ITUTurkBERT in the test. These results demonstrate that the combination of BERT2D and whole word masking is the most effective model for this problem.

TABLE 5. POS IMST dev/test accuracy.

	Vocabulary Size			
	32k		128k	
Model	RWM	WWM	RWM	WWM
mBERT	N/A/N/A	N/A/N/A	94.47/94.70	N/A/N/A
BERTurk	**96.01**/96.17	N/A/N/A	95.70/**96.25**	N/A/N/A
ITUTurkBERT	N/A/N/A	95.92/96.20	N/A/N/A	95.66/96.00
BERT2D ($M=1$)	95.93/96.28	95.89/96.22	95.88/96.23	95.76/96.20
BERT2D ($M=3$)	95.94/96.21	**96.06/96.36**	95.67/96.19	95.77/96.13
BERT2D ($M=10$)	95.96/96.31	95.95/96.26	95.76/96.09	95.96/96.28

TABLE 6. POS BOUN dev/test accuracy.

	Vocabulary Size			
	32k		128k	
Model	RWM	WWM	RWM	WWM
mBERT	N/A/N/A	N/A/N/A	91.93/92.51	N/A/N/A
BERTurk	92.58/92.85	N/A/N/A	90.90/90.99	N/A/N/A
ITUTurkBERT	N/A/N/A	**92.75/93.05**	N/A/N/A	92.57/92.64
BERT2D ($M=1$)	92.81/**93.05**	92.81/92.97	92.61/92.82	92.72/93.01
BERT2D ($M=3$)	92.80/92.92	92.75/93.02	92.79/92.90	92.81/92.91
BERT2D ($M=10$)	92.69/92.94	92.74/92.96	92.74/92.92	**92.86**/92.96

D. QUESTION ANSWERING

We fine-tuned all language models for question-answering downstream tasks, such as other downstream tasks.

1) DATASET

The THQuad[6] [40] dataset was used to evaluate the performance of the language models in a downstream task. The dataset was used to assess the performance of questions and answers. This Turkish QA dataset focuses on Turkish and Islamic science history, and was curated as part of the Teknofest 2018 AI competition. It includes 753 titles, 2,507 paragraphs, and 9,300 question-answer pairs. We used the train/dev split provided by the repository as shown in Table 7.

TABLE 7. THQuad Train/Dev Split.

	Title	Paragraph	Q&A
Train	681	2232	8308
Dev	72	275	892

2) HOW WE EVALUATE

Fig. 6 shows how to calculate the precision, recall, and F1 scores for the question-answer task. The exact match score was the proportion of correctly answered questions without any errors.

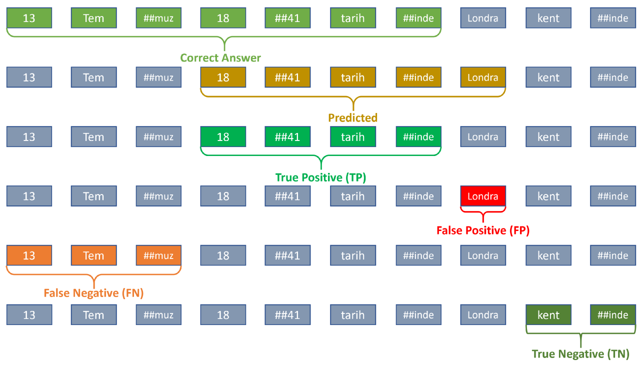


FIGURE 6. The true positive (TP), false positive (FP), true negative (TN), and false negative (FN) for the question-answering task. (English: On July 13, 1841, in the city of London)

3) RESULTS

Table 8 lists the EM and F1 scores for the THQuad dataset. BERT2D outperformed the baseline in both EM and F1, especially when combined with whole word masking.

TABLE 8. QA EM/F1.

	Vocabulary Size			
	32k		128k	
Model	RWM	WWM	RWM	WWM
mBERT	N/A/N/A	N/A/N/A	58.54/77.29	N/A
BERTurk	60.07/77.60	N/A/N/A	61.19/78.56	N/A/N/A
ITUTurkBERT	N/A/N/A	61.35/79.00	N/A/N/A	**61.64/79.07**
BERT2D ($M=1$)	60.11/78.03	60.61/77.85	**62.00**/78.85	61.30/78.31
BERT2D ($M=3$)	60.99/78.95	61.86/**79.69**	60.58/77.86	61.61/78.30
BERT2D ($M=10$)	60.56/77.95	61.35/78.57	61.70/79.29	60.78/78.43

[6]https://github.com/TQuad/turkish-nlp-qa-dataset

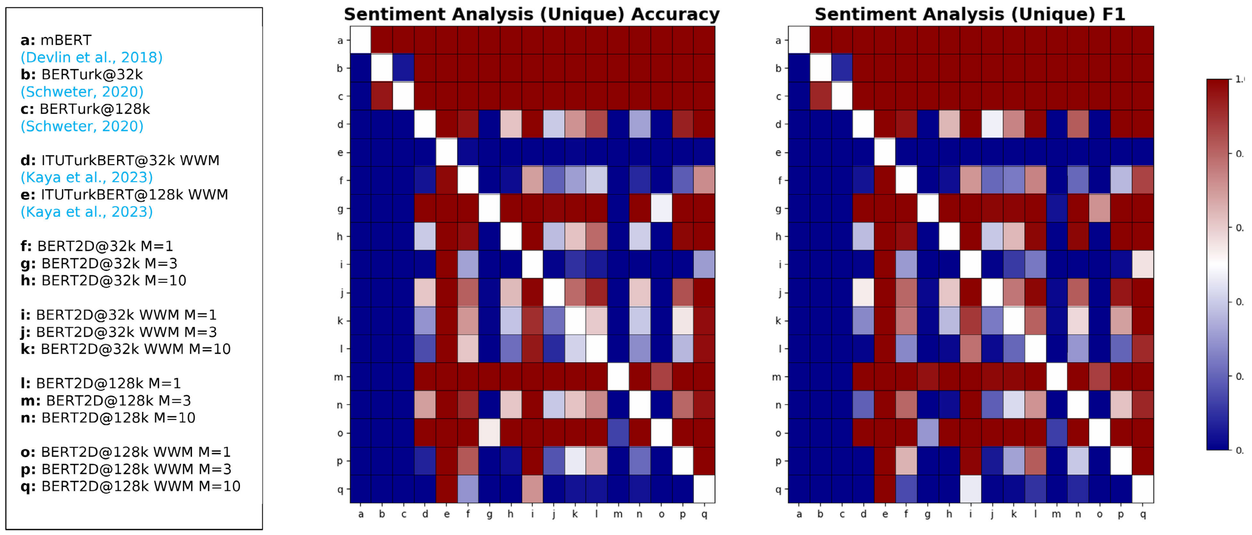


FIGURE 7. Statistical significance for Sentiment Analysis Accuracy/F1: Minimal distance ϵ for Almost Stochastic Order at level $\alpha = 5\%$. Blue cells mean that the left model is significantly better than the bottom model. Red cells mean the opposite.

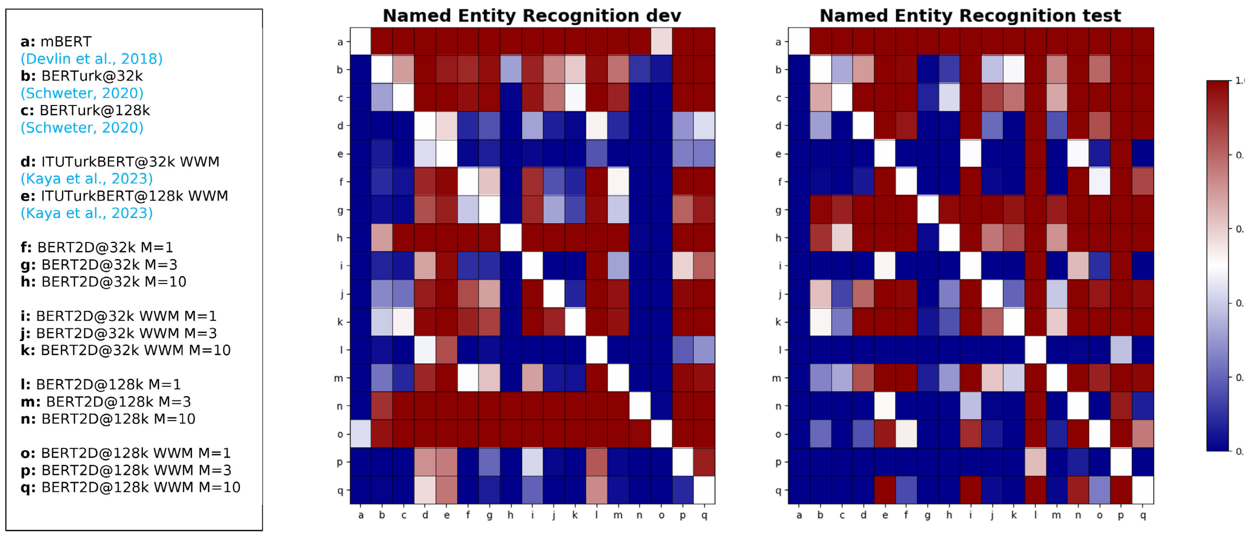


FIGURE 8. Statistical significance for Named Entity Recognition CoNLL F1 score for dev/test splits: Minimal distance ϵ for Almost Stochastic Order at level $\alpha = 5\%$. Blue cells mean that the left model is significantly better than the bottom model. Red cells mean the opposite.

VII. DISCUSSION AND CONCLUSION

Our studies show that BERT2D models consistently outperform or match the performance of BERT-based models in token classification, text classification, and question answering, with only a negligible increase in the number of parameters. The improvement in performance in these tasks was found to be statistically significant, which underscores the robustness of our findings.

To ensure the rigor of our conclusions, we employed extensive statistical methods to analyze performance differences. These methods are detailed in the Appendix (see Appendix), which explains the statistical tests used, such as the Almost Stochastic Dominance (ASD) and Almost Stochastic Order (ASO) tests. These tests are particularly suited to the complex nature of Deep Neural Network (DNN) performance evaluation, where traditional statistical methods may fall short.

We demonstrated that whole word masking is a significant hyperparameter of BERT, that improves performance without altering the architecture. In addition, we combined whole word masking and BERT2D to consistently achieve the best results. We concluded that two-dimensional embeddings benefit token-focused downstream tasks, such as token classification and question-answering, whereas whole word masking benefits text classification tasks.

A. FUTURE WORK

The encoding has two components consisting of relative subword positions and absolute whole-word positions. For the sake of brevity, we focus only on subword relativity, but in

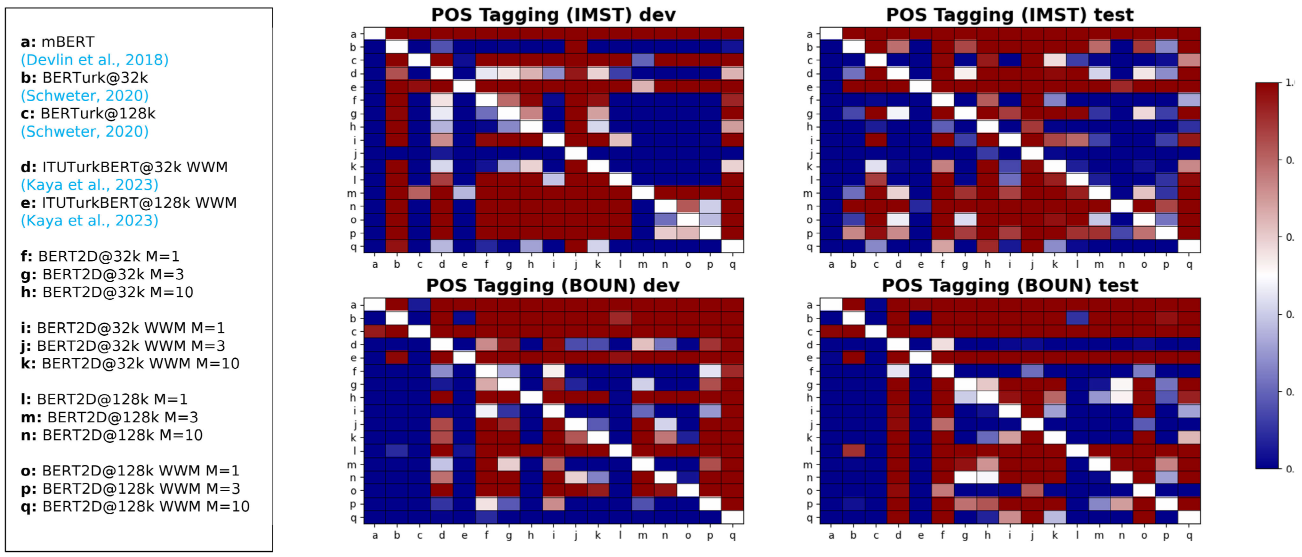


FIGURE 9. Statistical significance for POS-tagging with IMST/BOUN datasets for dev/test splits: Minimal distance ϵ for Almost Stochastic Order at level $\alpha = 5\%$. Blue cells mean that the left model is significantly better than the bottom model. Red cells mean the opposite.

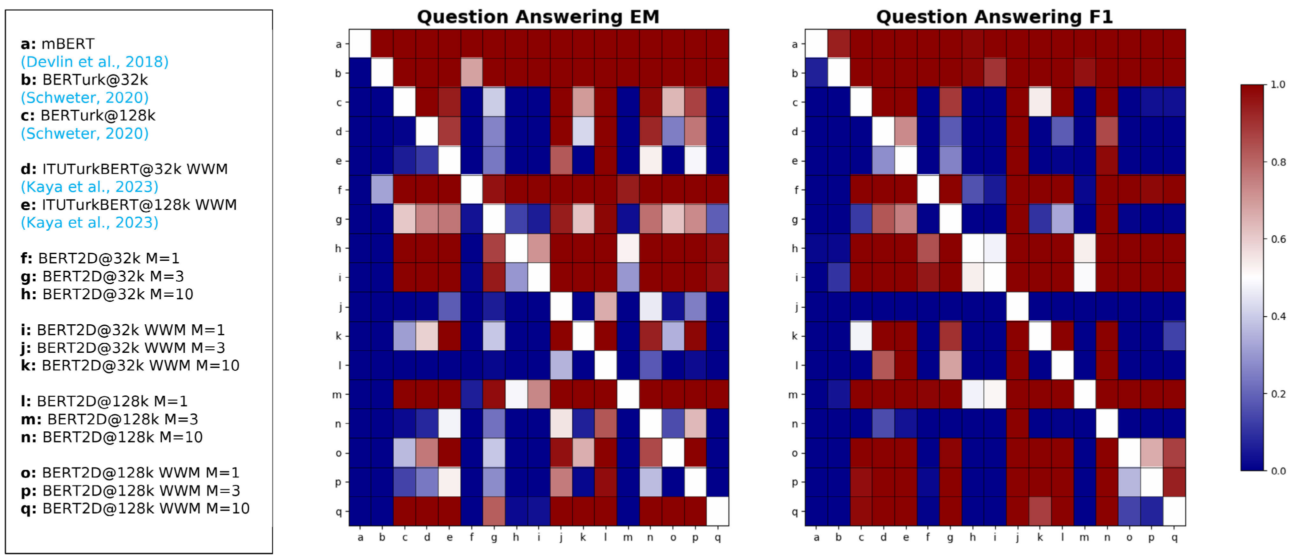


FIGURE 10. Statistical significance for Question answering EM/F1: Minimal distance ϵ for Almost Stochastic Order at level $\alpha = 5\%$. Blue cells mean that the left model is significantly better than the bottom model. Red cells mean the opposite.

future work we are considering adding relative word positions as well. In addition, we plan to pretrain BERT2D models in other agglutinative and non-agglutinative languages to analyze their impact. We also plan to add 2D positional encoding to newer BERT-based models to evaluate the impact of 2D positional encoding in general.

APPENDIX

STATISTICAL SIGNIFICANCE STUDY

In NLP, evaluating DNN models for tasks such as NER, POS tagging, sentiment analysis, and question answering is a complex challenge due to their stochastic nature. Traditional statistical significance tests often fail to provide a comprehensive comparison because they focus on individual scores. This approach does not sufficiently consider the variability inherent in DNNs because of the numerous hyperparameters and non-convex nature of their optimization landscapes. Consequently, the performance of these models can vary significantly, depending on the random initialization and stochastic elements encountered during training. To address this issue, we opted to use ASD [41] as a more robust and informative method to compare DNN models. By running the models multiple times with different seed values (typically five in our analysis), we generated empirical score distributions for each model on unseen data. The Almost Stochastic Dominance test enables a nuanced comparison between these distributions, providing insight into the models' performance across a range of possible outcomes, rather than relying on a single-point estimate.

In our comparative analysis of BERT2D and BERT under various operational conditions, ASO test, conducted at an α level of 0.05 formed the backbone of our methodological approach. This statistical framework is adept at identifying the minimum ϵ value required to detect a significant performance difference between models. This ϵ value is particularly important when evaluating the impact of specific configurations, such as the implementation of the whole word masking hyperparameter. This feature significantly influences how models interpret contextual subtleties during language processing. The robustness of the ASO test ensures that our analysis accurately captures the influence of these hyperparameters on the model performance.

Furthermore, the variation in the number of intermediate subword embeddings is central to our study. This parameter is essential for understanding how the models navigate and process complex linguistic structures. By manipulating this variable, we gain insight into the models' ability to deal with linguistic complexity.

In presenting our results, we use color codes linked to epsilon values as in [42] in Fig. 7, 8, 9, and 10: blue indicates that the model on the left is significantly better than the model on the bottom when the ϵ value is less than 0.5, and red indicates the opposite when the epsilon value exceeds 0.5. This color coding provides an intuitive visual representation of model performance in relation to epsilon values. Adherence to the level of statistical significance 0.05 α ensures a high degree of confidence in our results, effectively ruling out the influence of random variance and solidifying the validity of our conclusions under different test conditions.

ACKNOWLEDGMENT

The authors would like to thank for Google's TPU Research Cloud (TRC) support in providing them with cloud TPUs and the ITU AI Cloud Center for providing them with GPU clusters. They also thank Stefan Schweter for generously sharing the BERTurk dataset and enabling fair comparison.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017, *arXiv:1706.03762*.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019, pp. 4171–4186. [Online]. Available: https://aclanthology.org/N19-1423

[3] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, "BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," 2019, *arXiv:1910.13461*.

[4] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," OpenAI, San Francisco, CA, USA, Tech. Rep., 2018.

[5] Y. Wu et al., "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.

[6] Y. Shibata, T. Kida, S. Fukamachi, M. Takeda, A. Shinohara, T. Shinohara, and S. Arikawa, "Byte pair encoding: A text compression scheme that accelerates pattern matching," Dept. Inform., Kyushu Univ., Fukuoka, Japan, Tech. Rep. DOI-TR-161, 1999.

[7] Y. B. Kaya and A. C. Tantuğ, "Effect of tokenization granularity for Turkish large language models," *Intell. Syst. Appl.*, vol. 21, Mar. 2024, Art. no. 200335. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2667305324000115

[8] T. Likhomanenko, Q. Xu, G. Synnaeve, R. Collobert, and A. Rogozhnikov, "CAPE: Encoding relative positions with continuous augmented positional embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds. Red Hook, NY, USA: Curran Associates, Inc., 2021, pp. 16079–16092. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/865bf46435bd84fa5d89f64cf3ba7347-Paper.pdf

[9] B. Wang, D. Zhao, C. Lioma, Q. Li, P. Zhang, and J. Grue Simonsen, "Encoding word order in complex embeddings," 2019, *arXiv:1912.12333*.

[10] D. Xu, C. Ruan, S. Kumar, E. Korpeoglu, and K. Achan, "Self-attention with functional time representation learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates Inc., Dec. 2019, no. 1426, pp. 15915–15925.

[11] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019, *arXiv:1910.10683*.

[12] T.-C. Chi, T.-H. Fan, P. J. Ramadge, and A. I. Rudnicky, "KERPLE: Kernelized relative positional embedding for length extrapolation," 2022, *arXiv:2205.09921*.

[13] Z. Huang, D. Liang, P. Xu, and B. Xiang, "Multiplicative position-aware transformer models for language understanding," 2021, *arXiv:2109.12788*.

[14] G. Ke, D. He, and T.-Y. Liu, "Rethinking positional encoding in language pre-training," 2020, *arXiv:2006.15595*.

[15] O. Press, N. A. Smith, and M. Lewis, "Train short, test long: Attention with linear biases enables input length extrapolation," 2021, *arXiv:2108.12409*.

[16] J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen, and Y. Liu, "RoFormer: Enhanced transformer with rotary position embedding," 2021, *arXiv:2104.09864*.

[17] X. Wang, Z. Tu, L. Wang, and S. Shi, "Self-attention with structural position representations," 2019, *arXiv:1909.00383*.

[18] A. Haviv, O. Ram, O. Press, P. Izsak, and O. Levy, "Transformer language models without positional encodings still learn positional information," 2022, *arXiv:2203.16634*.

[19] C. B. Gemirter and D. Goularas, "A Turkish question answering system based on deep learning neural networks," *J. Intell. Syst., Theory Appl.*, vol. 4, no. 2, pp. 65–75, Sep. 2021.

[20] C. Zeng and S. Li, "Analyzing the effect of masking length distribution of MLM: An evaluation framework and case study on Chinese MRC datasets," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–17, Nov. 2021. [Online]. Available: https://www.hindawi.com/journals/wcmc/2021/5375334/

[21] C. Liu, C. Zhu, and W. Zhu, "Chinese named entity recognition based on BERT with whole word masking," in *Proc. 6th Int. Conf. Comput. Artif. Intell.* New York, NY, USA: Association for Computing Machinery, Aug. 2020, pp. 311–316, doi: 10.1145/3404555.3404563.

[22] B. Chan, S. Schweter, and T. Müller, "German's next language model," in *Proc. 28th Int. Conf. Comput. Linguistics*, Barcelona, Spain, Dec. 2020, pp. 6788–6796. [Online]. Available: https://aclanthology.org/2020.coling-main.598

[23] Y. Dai, L. Li, C. Zhou, Z. Feng, E. Zhao, X. Qiu, P. Li, and D. Tang, "'Is whole word masking always better for Chinese BERT?': Probing on Chinese grammatical error correction," 2022, *arXiv:2203.00286*.

[24] Y. Chen, Z. Liang, Z. Tan, and D. Lin, "Named entity recognition in power marketing domain based on whole word masking and dual feature extraction," *Appl. Sci.*, vol. 13, no. 16, p. 9338, Aug. 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/16/9338

[25] Y. Cui, W. Che, T. Liu, B. Qin, and Z. Yang, "Pre-training with whole word masking for Chinese BERT," *IEEE/ACM Trans. Audio, Speech, Language Process.*, vol. 29, pp. 3504–3514, 2021.

[26] S. Schweter, "Berturk—BERT models for Turkish," Bayerische Staatsbibliothek MDZ Digital Library Team, Munich, Germany, Tech. Rep., 2020.

[27] P. J. O. Suárez, B. Sagot, and L. Romary, "Asynchronous pipeline for processing huge corpora on medium to low resource infrastructures," in *Proc. 7th Workshop Challenges Manage. Large Corpora*, 2019, pp. 9–16.

[28] J. Tiedemann and L. Nygaard, "The OPUS corpus—Parallel and free," in *Proc. LREC*, 2004, pp. 1183–1186.

[29] P. Rust, J. Pfeiffer, I. Vulić, S. Ruder, and I. Gurevych, "How good is your tokenizer? On the monolingual performance of multilingual language models," 2020, *arXiv:2012.15613*.

[30] A. Ucan, B. Naderalvojoud, E. A. Sezer, and H. Sever, "SentiWordNet for new language: Automatic translation approach," in *Proc. 12th Int. Conf. Signal-Image Technol. Internet-Based Syst. (SITIS)*, Nov. 2016, pp. 308–315.

[31] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[32] Y. Elor and H. Averbuch-Elor, "To SMOTE, or not to SMOTE?" 2022, *arXiv:2201.08528*.

[33] D. Altinok, "A diverse set of freely available linguistic resources for Turkish," in *Proc. 61st Annu. Meeting Assoc. Comput. Linguistics*. Toronto, ON, Canada: Association for Computational Linguistics, 2023, pp. 13739–13750. [Online]. Available: https://aclanthology.org/2023.acl-long.768

[34] U. Sulubacak and G. Eryiğit, "Implementing universal dependency, morphology, and multiword expression annotation standards for Turkish language processing," *Turkish J. Electr. Eng. Comput. Sci.*, vol. 26, no. 3, pp. 1662–1672, 2018.

[35] U. Sulubacak, G. Eryiğit, and T. Pamay, "Imst: A revisited Turkish dependency treebank," in *Proc. 1st Int. Conf. Turkic Comput. Linguistics*, 2016, pp. 1–6.

[36] K. Oflazer, B. Say, D. Z. Hakkani-Tür, and G. Tür, "Building a Turkish treebank," in *Treebanks: Building and Using Parsed Corpora*. Dordrecht, The Netherlands: Springer, 2003, pp. 261–277.

[37] N. B. Atalay, K. Oflazer, and B. Say, "The annotation process in the Turkish treebank," in *Proc. 4th Int. Workshop Linguistically Interpreted Corpora*, 2003, pp. 33–38.

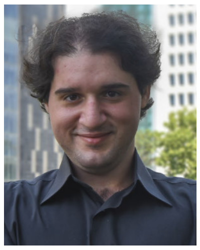
[38] B. Marşan, S. F. Akkurt, M. Şen, M. Gürbüz, O. Güngör, Ş. B. Özateş, S. Üsküdarlı, A. Özgür, T. Güngör, and B. Öztürk, "Enhancements to the BOUN treebank reflecting the agglutinative nature of Turkish," 2022, *arXiv:2207.11782*.

[39] E. F. T. K. Sang and F. De Meulder, "Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition," in *Proc. 7th Conf. Natural Lang. Learn. HLT-NAACL*, 2003, pp. 142–147.

[40] F. Soygazi, O. Çiftçi, U. Kök, and S. Cengiz, "THQuAD: Turkish historic question answering dataset for reading comprehension," in *Proc. 6th Int. Conf. Comput. Sci. Eng. (UBMK)*, Sep. 2021, pp. 215–220.

[41] R. Dror, S. Shlomov, and R. Reichart, "Deep dominance—How to properly compare deep neural models," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 2773–2785.

[42] H. El Boukkouri, O. Ferret, T. Lavergne, H. Noji, P. Zweigenbaum, and J. Tsujii, "CharacterBERT: Reconciling ELMo and BERT for word-level open-vocabulary representations from characters," 2020, *arXiv:2010.10392*.



YİĞİT BEKİR KAYA was born in Şişli, İstanbul, Turkey, in 1990. He received the double B.S. degree in computer engineering and aeronautical engineering and the M.S. degree in aeronautical engineering from Istanbul Technical University, in 2015, where he is currently pursuing the Ph.D. degree in computer engineering. From 2013 to 2015, he was a Data Science Researcher with ITU Control and Avionics Laboratory, Department of Aeronautical and Astronautical Engineering, Istanbul Technical University, Ayazağa Campus. He is the author of one journal article and five conference papers. He was a recipient of the Boeing Graduate Grant, in 2014.

A. CÜNEYD TANTUĞ was born in Adana, Turkey, in 1978. He received the B.S. degree in control and computer engineering and the M.Sc. and Ph.D. degrees in computer engineering from Istanbul Technical University, in 2002 and 2007, respectively. He is currently a full-time Faculty Member with the Faculty of Computer and Informatics, Istanbul Technical University. His research interests include natural language processing and financial artificial intelligence.