Automatika

Journal for Control, Measurement, Electronics, Computing and Communications

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/taut20

Advancing natural language processing (NLP) applications of morphologically rich languages with bidirectional encoder representations from transformers (BERT): an empirical case study for Turkish

Akın Özçift, Kamil Akarsu, Fatma Yumuk & Cevhernur Söylemez

To cite this article: Akın Özçift, Kamil Akarsu, Fatma Yumuk & Cevhernur Söylemez (2021) Advancing natural language processing (NLP) applications of morphologically rich languages with bidirectional encoder representations from transformers (BERT): an empirical case study for Turkish, Automatika, 62:2, 226-238, DOI: 10.1080/00051144.2021.1922150

To link to this article: https://doi.org/10.1080/00051144.2021.1922150

AUTOMATIKA
2021, VOL. 62, NO. 2, 226–238
https://doi.org/10.1080/00051144.2021.1922150

REGULAR PAPER

OPEN ACCESS

Advancing natural language processing (NLP) applications of morphologically rich languages with bidirectional encoder representations from transformers (BERT): an empirical case study for Turkish

Akın Özçift, Kamil Akarsu, Fatma Yumuk and Cevhernur Söylemez

Hasan Ferdi Turgutlu Technology Faculty, Software Engineering Department, Manisa Celal Bayar University, Manisa, Turkey

ABSTRACT
Language model pre-training architectures have demonstrated to be useful to learn language representations. bidirectional encoder representations from transformers (BERT), a recent deep bidirectional self-attention representation from unlabelled text, has achieved remarkable results in many natural language processing (NLP) tasks with fine-tuning. In this paper, we want to demonstrate the efficiency of BERT for a morphologically rich language, Turkish. Traditionally morphologically difficult languages require dense language pre-processing steps in order to model the data to be suitable for machine learning (ML) algorithms. In particular, tokenization, lemmatization or stemming and feature engineering tasks are needed to obtain an efficient data model to overcome data sparsity or high-dimension problems. In this context, we selected five various Turkish NLP research problems as sentiment analysis, cyberbullying identification, text classification, emotion recognition and spam detection from the literature. We then compared the empirical performance of BERT with the baseline ML algorithms. Finally, we found enhanced results compared to base ML algorithms in the selected NLP problems while eliminating heavy pre-processing tasks.

ARTICLE HISTORY
Received 4 May 2020
Accepted 21 April 2021

KEYWORDS
Bidirectional encoder representations transformers; language pre-processing; morphologically rich language; natural language processing; Turkish

1. Introduction

There are many sources and types of information such as social media posts, micro-blogs, news and customer reviews that accumulates data progressively [1,2]. The automated analysis of particularly the large amount of text data is predominantly handled with machine learning (ML) techniques applied to natural language processing (NLP) domain. ML techniques and recently deep learning (DL) models are applied to various language analysis problems such as text categorization, sentiment identification, emotion recognition, fake news identification and spam detection etc.[3–7].

Conventionally, to apply ML methods on NLP problems, the first step is to transform texts into a convenient format such as bag-of-words (BoW) in which text is represented as or Vector Space Model (VSM). More formally, any document in term space can be modelled with $d = (w_1, \ldots, w_n)$, where n denotes the term space or feature size. The effectiveness of the stated model is frequently enhanced with the use of *N-grams* or term weighting schemes such as term frequency-inverse document frequency (*tf* or *tf-idf*) [8,9]. Rather than sparsity or high-dimensional nature of BoW models, another problem is that these representations ignore their syntactic and semantic information [10]. In simpler words, the contextual meaning of words are ignored with BoW models. Therefore, newer text representations, i.e. word embedding models, have been developed to overcome this issue. A word embedding is basically defined as a real-valued vector that is extracted from the context of the corresponding document. The contextual information capturing ability of word embedding models is explained with the "distributional hypothesis". The theory was explained as "words which are similar in meaning occur in similar contexts" [11] and "words with similar meanings will occur with similar neighbours if enough text material is available" [12]. Hence, it can be expected that the words having semantic or syntactic similarity will be closer to each other compared to relatively dissimilar words in vector space. It is evident that the stated *relatedness* or *similarity* is exclusively dependent on the corpus which is used to obtain word embeddings [13]. This concept is first introduced in [14] by Bengio et al. and formally a word embedding scheme is implemented as $\varepsilon : \text{words} \to \mathbb{R}^p$ to transform words into dense vectors. In general, p value in the representation is chosen between 50 and 500. The two sample vectors for words "orange" and "glass" may be represented as $\varepsilon('\text{orange}') = (0.056, -0.170, -0.011, \ldots)$ and $\varepsilon('\text{glass}') = (-0.075, 0.087, -0.315, \ldots)$ [15]. Since the efficiency of the embedding models rely on large vocabulary [12] and they need an excess amount of computational power, the emergence of the practical word embedding models, i.e. *Word2Vec* [16],

CONTACT Akın Özçift akin.ozcift@cbu.edu.tr

GloVe [17] and *fastText* [18], has delayed until 2013. Word2Vec, GloVe or fastText embedding models depend on co-occurrence statistics of corpus used for pre-training. Though these models are more eligible to comprehend semantic or syntactic information, the models are context independent and they generate only one vector embedding for each word. Furthermore, another weakness of these language models is that they only make use of left or right context. However, since language understanding is improved with bi-directional learning [19], new neural language models, i.e. Embeddings from Language Models (ELMo) [20] and Bidirectional Encoder Representations from Transformers (BERT) [19], are developed. Concisely, ELMo, being a contextualized word/character embedding model, uses a bi-directional Recurrent Neural Network (RNN) that takes into account both bi-directional positions of words in a sentence. Similarly, BERT, being another contextualized learning language model, benefits multilayer bi-directional transformer-encoder having parallel attention layers rather than sequential recurrence [13,21,22]. Overall, bi-directional models process words in relation to the other words in a sentence and therefore they are closer to an *intended understanding* with the use of rich contextual information.

Conventional NLP pipelines, in general, consist of basic preprocessing tasks such as tokenization, stopword removal, stemming or lemmatization and possibly a feature engineering task. In particular, as opposed to English which is a widely researched language, the mentioned tasks are not straightforward for morphologically rich languages, i.e. Turkish, Arabic, Czech, Hungarian, Finnish, Hebrew, Korean, etc., due to their nature of complexity [23]. The mentioned preprocessing steps, in particular stemming, lemmatization and feature engineering need complicated analysis tasks because of their agglutinative or inflectional morphologies. For example Turkish, having agglutinative morphology, has a vocabulary size of 474,957 and English has vocabulary size of 97,734. However, if the root forms of Turkish words are obtained, this vocabulary size becomes as 94,235. It is clearly observed that on the average five different words may be derived using the same root [24] and this is an issue from stemming point of view. Moreover, the mentioned language processing tasks affect the performance of ML algorithms [25–27]. In this context, BERT may seem to be a solution to comprehend contextual meaning of morphologically complicated words [28–30] without mentioned language processing tasks. This is one of the first motivations of this study. Furthermore, morphologically rich languages are also probably low-resource languages that have limited research and data sources compared to English [31]. Although Turkish is a widely spoken language, it has limited computational language studies. To the best of our knowledge, this is the first study that uses BERT neural language model for Turkish and this is the second motivation of the current study.

This empirical study compares the traditional BoW (or VSM) based ML algorithms and newly proposed BERT in terms of empirical achievement. In order to evaluate performances of the two approaches, we chose diverse six datasets from literature as cyberbullying identification [32], sentiment analysis of Turkish movie and hotel reviews [33], spam Short Message Service (SMS) detection [34], a text classification dataset of six news categories [35] and emotion recognition dataset with six emotion categories [36]. In particular, the mentioned Turkish datasets are studied in the literature and they have morphological language processing steps with corresponding BoW-ML results as baseline. Therefore, we are able to evaluate those results to make an empirical comparison. The results of this study will probably help the morphologically rich language researchers particularly for Turkish language, in terms of applicability of BERT neural language model to the other real-world NLP tasks.

This study provides the following contributions:

- A novel approach is presented for Turkish language analysis with the use of BERT.
- Detailed experiments were carried out to demonstrate efficiency of BERT over traditional Turkish language analysis pipeline.
- An analysis roadmap is presented for Turkish language.
- An extensive number of experiments on various Turkish datasets were evaluated. This is the largest empirical study for Turkish to validate the efficiency of novel language transformers.

This paper has been organized as follows: Section 2 presents computational modelling challenges of Turkish based on its morphological complexity. While, in Section 3, we explain traditional Turkish language processing pipeline with frequently used ML algorithms, we define the BERT architecture in Section 4. The NLP problems discussed in this study is explained in Section 5. The article presents the detailed experimental results in Section 6 and ends with Conclusion in Section 7.

2. Turkish language modelling challenges based on its morphological complexity

Morphologically rich languages, particularly Turkish, have their own language difficulties while generating a computational NLP model [37,38]. As opposed to English, being an analytical language, Turkish-like morphologically rich languages have *agglutinative* complexity. As the simplest member of the language, i.e. the word, basically consists of a stem and inflectional suffixes generating the context such as time (tense), locality (place) and arity (singular or plural).

English-like analytical languages make use of separate prepositions to generate this context and naturally their analysis is easier compared to morphologically languages such as Turkish. More clearly, even one single Turkish word is generated with morphemes (stems, prefixes and affixes) and that word will have many corresponding inflectional forms [39]. Interestingly, Oflazer has shown this enormous word generation ability of Turkish and "its challenges for language processing" in his detailed study [40]. According to the study, it is possible to derive about 1.5 million different words from a *Noun* [masa (table)] and from a *Verb* [oku (read)] only with the use of derivational morphemes [40]. The morphological structure of Turkish word is shown in Figure 1 [41].

Some samples for morphological productivity of Turkish language are provided in Table 1 [41]. As it is obvious from Table 1, the number of suffixes and their imaginable combinations that can be added to a word generate a serious language analysis problem to obtain actual stem from possible derivations.

From the above discussion, we may state that Turkish NLP studies has to deal with language processing tasks before modelling a solution to the target problem. In general, most-words are composed of many morphemes and they may occur only once on the training data that generates the so called *data-sparsity* and *curse of dimensionality* problems [42,43] from computational modelling point of view. It is important to observe that this complexity constrains implementation of state-of-the-art models and algorithms developed for example for English. In order to overcome data-sparsity in Turkish, dense pre-processing tasks such as stemming or lemmatization (possibly followed with a feature selection step) should be introduced before NLP pipelines [44]. Both stemming and lemmatization has goal to reduce inflectional or derivational forms of words into a common base form. While *stemming* is a heuristic process that chops off the ends of the derived words to obtain a *base form*, *lemmatization* makes use of a vocabulary and morphological analysis to obtain dictionary form, i.e. lemma, of the word [45]. However, the two methods are not interchangeable and it should be carefully examined which one is better for the corresponding language problem. For example, the Turkish words göz (eye), gözlük (eyeglasses), gözlükçü (optician) and gözlem (observation) may all be stemmed from a single word "*göz (eye)*" losing the semantical information [41,46]. Interestingly, it is apparent that the given Turkish words have distinct English equivalents and this may be a concise comparison of two languages in terms of analysis complexity. The aforementioned language processing tasks require specialized software [47] and related lexicons. Other than stemming and lemmatization traditional pre-processing tasks may also include stop-word removal, lower-case conversion and tokenization tasks. Stop-words are the common words that are filtered out in language processing. Prepositions, pronouns and conjunctions are stop-words and bu (that), orda (there) are some example Turkish stop-words. And lexically, tokenization is defined as the process of splitting the input text string into meaningful pieces (tokens) such as words or phrases. Though, we will not make a detailed discussion, we will recap the importance of feature engineering tasks in Turkish [48]. Many Turkish NLP tasks also make use of several feature processing strategies to obtain an enhanced ML model. Practically, traditional ML algorithms developed for morphologically rich languages needs token-based language models that obtained through pre-processing steps. For example, Czech language has five different inflectional variants for English word "king" as *král* (king), *krále* (of king), *králi* (kings), *králem* (king), *králové* (kings) and the corresponding token forms through heavy language pre-processing tasks are required for conventional ML models. Another example from German language is that regen (rain) may generate various forms as *regenschirm* (umbrella), *regenschirmhersteller* (umbrella manufacturer), *regenschirmherstellergewerkschaft* (umbrella manufacturer's trade union). Even if the token-based models are generated after pre-processing steps, the aforementioned data-sparsity problem is another issue. In other words, since morphological processing tasks produces word forms that are very rare or probably non-existent in training corpus, the performance of the ML algorithms will decrease in prediction phase [67].

As a summary, it can be observed that Turkish NLP problems needs several pre-processing steps in order to obtain efficient ML models while recruiting data-sparsity and related problems. Moreover, other morphologically rich languages will also require morphological processing tasks similar to Turkish while generating efficient ML models. The promising solution or answer to the overall discussion is possibly the

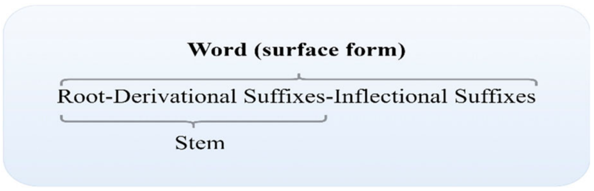

Figure 1. Morphological structure of a Turkish word.

Table 1. Stems and surface form examples derived from teh stem "göz" (eye).

Word (surface form)	Root	Stem	Inflectional suffixes
göz/ler-im (my eyes)	göz	göz	ler-im
göz/ün-de (in the eye)	göz	göz	ün-de
göz/cü (observer)	göz	gözcü	-
göz/lük (eyeglass)	göz	gözlük	-
göz/lük-çü (optician)	göz	gözlükçü	-
göz/lük-çü-y-dü (he/she was optician)	göz	gözlükçü	-y-dü

evaluation of bi-directional neural language models or BERT for our case study.

3. Frequent Turkish language processing pipeline and ML algorithms

Computational modelling of Turkish to solve NLP problems usually require a suitable vector representation accompanied with pre-processing tasks such as tokenization, stop-word removal, stemming/lemmatization and feature engineering. This refined representation of textual information is then feed into ML algorithms to obtain the target identification. This frequent NLP analysis approach is shown in Figure 2.

For pre-processed raw texts, a mathematical representation such as BoW needs to be generated. In particular, BoW reduces a text into a simplified representation based on a criterion such as word frequency. In this model, while raw text is represented as a word bag, the semantic relationships are ignored between words. Since BoW encodes every word in a one-hot-encoding scheme, the model may fast converge to a sparse vector. Term-frequency is the simplest technique to obtain features in BoW. In order to decrease effect of common frequent words in the corpus, a weighting scheme, i.e. tf-idf, is frequently used while obtaining features from text. Mathematically, tf-idf is represented in the following way:

$$W(d,t) = \mathrm{TF}(d,t) * \log\left(\frac{N}{\mathrm{df}(t)}\right) \qquad (1)$$

where N denotes number of documents in the corpus, $\mathrm{df}(t)$ denotes the number of documents containing the term t [49]. Since BoW models are just unordered sequence of words, they are unaware of the semantic relations that may enrich contextual information to be extracted. One solution to this problem is the use of *N-gram* model that preserves the relation between words based on the occurrence of $N-1$ previous words [45]. The whole pre-processing pipeline may have a high-dimension problem that may reduce performances of ML algorithms. In order to deal with this problem, another important step that is known as feature engineering (possibly a filtering) should possibly be applied to the resultant model. Though there are filters, wrappers, and embedded feature selection methodologies, the most versatile feature selection group is *feature filters*. In the NLP literature, the *Information Gain* (IG), *Chi-Square* (CHI) and *Gini Index* (GI) are some of the frequently used feature filtering algorithms [50].

Table 2. Confusion matrix.

	Predicted class	
Actual class	Positive	Negative
Positive	TP	FP
Negative	FN	TN

Once completed the text representation pipeline, then the language problem needs an ML approach in terms of train/validate/test scheme. There are many ML algorithms used in the NLP problems frequently. Explicitly, *Naïve Bayes* (NB), *Support Vector Machines* (SVM), *k-Nearest Neighbours* (k-NN) *Random Forests* (RF) ensemble approaches such as *Boosting* and *Bagging* are to name a few. Recently, DL algorithms, i.e. *Convolutional Neural Networks* and *Recurrent Neural Networks* have also gained great popularity as emerging ML algorithms for NLP domain. An important point to remind here is that the performance of the algorithms may be compared based on some widely used metrics such as Accuracy, Recall, Precision, and F1-measure [51]. The evaluation metrics are derived from the confusion matrix given in Table 2.

Accuracy, Recall, Precision, F1-measure from confusion matrix are defined as follows:

$$\mathrm{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \qquad (2)$$

$$R = \frac{TP}{TP + FN} \qquad (3)$$

$$P = \frac{TP}{TP + FP} \qquad (4)$$

$$F1 - \text{measure} = \frac{2 \times P \times R}{(P + R)} \qquad (5)$$

where *TP*, *FP*, *FN* and *TN* denotes *True Positive*, *False Positive*, *False Negative* and *True Negative*, respectively. Furthermore, we underline two additional significant metrics which are used two evaluate the performance

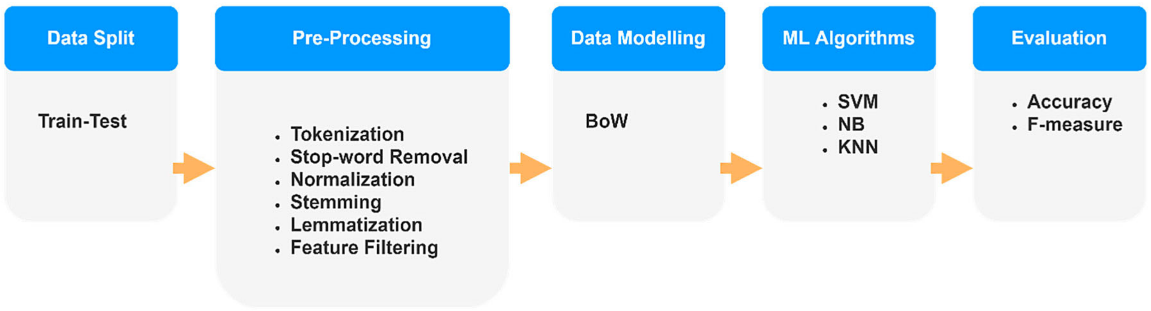

Figure 2. Frequent Turkish language processing pipeline.

and statistical validation of ML algorithms: Matthews Correlation Coefficient (MCC) and Kappa (κ).

MCC is a correlation coefficient between target and prediction whose value varies between 1 and -1 to inform perfect agreement and disagreement, respectively. Since MCC estimates the strength of an ML algorithm with all four results of confusion matrix, it is a comprehensive metric. Therefore, it provides a balanced performance assessment even in imbalanced data applications and it is given as follows:

$$\mathrm{MCC} = \frac{(TP \times TN) - (FP \times FN)}{(TP \times FP) \times (TP \times FN) \times (TN \times FP) \times (TP \times FN)} \quad (6)$$

Based on the inclusiveness, we also made use of MCC metric to evaluate performance of BERT experiments.

Kappa, i.e. κ, is a statistic metric which measures inter-rater reliability for categorical items. Inter-rater reliability is defined as the degree of agreement among the raters. This statistical score measures the consensus degree based on the decisions of predictors. In other words, it measures the agreement between two predictors who each classify N items into exclusive categories and κ is defined as follows:

$$\kappa = \frac{(Po - Pe)}{(1 - Pe)} \quad (7)$$

where Po is given below and it is also identical to accuracy.

$$Po = \frac{TP + TN}{TP + FP + TN + FN}$$

In Equation (7), Pe is defined as the number of n_{ki} times a rater i predicted category k with N observations and it is given as

$$\mathrm{Pe} = \frac{1}{N^2} \sum_k n_{k1} n_{k2}$$

and it is also calculated with the following relation in terms of confusion matrix terms.

$$Pe = \frac{(TP + FP)}{(TP + FP + TN + FN)} \times \frac{(TP + FN)}{(TP + FP + TN + FN)} \times \frac{(FN + TN)}{(TP + FP + TN + FN)} \times \frac{(FP + TN)}{(TP + FP + TN + FN)}$$

The overall score of κ varies between -1 and 1. The obtained score is a statistical measure of how the obtained results far from occurring by chance. More empirically, while values smaller than 0.40 show fair agreement, the values between 0.40 and 0.60 show moderate agreements. Simply, for a confident classification performance evaluation, we should obtain 0.60–0.80 for good agreement and higher than 0.80 for the perfect agreement [51]. Therefore, we calculated κ metric in the experiments to evaluate statistical confidence of obtained results.

4. Problem definition and BERT architecture

In the preceding section, we summarized Turkish language pre-processing steps to obtain a refined vector representation while overcoming data-sparsity and high-dimension problems. We have also made a concise reminding of ML algorithms and frequently used evaluation metrics. The mentioned ML algorithms are used in many Turkish NLP problems and the mentioned algorithms will be used as a baseline while making comparison of the results from literature and the results of the BERT model obtained. First we define the problem and then we explain BERT architecture used in the experiments to realize our tasks.

4.1. Definition of the problem

The basic classification task is the automated assignment of a text into predefined categories. Formally, for a given set of text with set of categories a model $Y = f(T, \theta) + \in$ is constructed. In this model, while T is a text representation scheme, θ denotes the set of unknown parameters of classifier model f that should be estimated during training. The last parameter in the model, i.e. $\in$, is the error of the classification. Since f is simply an approximation of an unknown function h with representation of $Y = h(T)$, the efficiency of the classification task becomes better for smaller values of error term $\in$. The value of Y varies over a set of discrete integers that are denoting corresponding categories. For example, Y may take values of $+1$ and -1 for a binary sentiment identification task or for spam detection problem denoting to be spam or not [52].

4.2. BERT architecture

BERT is a recent attention-based model with a bidirectional Transformer network that was pre-trained on a large corpus. This pre-trained model is then effectively used to solve various language tasks with fine-tuning [53,54]. In brief terms, the task-specific BERT architecture represents input text as sequential tokens. The input representation is generated with the sum of the token embeddings, the segmentation embeddings and the position embeddings [54]. For a classification task, the first word in the sequence is a unique token which is denoted with [CLS]. An encoder layer is followed with a fully-connected layer at the [CLS] position. Finally, a softmax layer is used as the aggregator for classification purposes [53]. If the NLP task has pair of sentences as in question-answer case, the sentence pairs may be separated with another special token [SEP].

BERT has mainly two versions as *base* and *large* with heavy parameter settings of 110M and 340M respectively. While the BERT-base has 12 number of Transformer blocks and 768 hidden layer size, the BERT-large has 24 Transformer blocks and 1024 hidden layer size. As it is obvious BERT-large requires more GPU memory, we decided to use BERT-base-multilingual model for our evaluations.

4.3. BERT unsupervised pre-training tasks

Recently, most of the neural language modelling approaches make use of the concept of borrowing the learned knowledge from the other tasks. In this context, modern NLP systems prefer pre-trained language models such as BERT over embeddings learned from scratch [55]. While using pre-trained language models to down-stream NLP tasks, there are two approaches: *feature-based* and *fine-tuning*. ELMo uses the first approach including pre-trained representations as additional features for the corresponding NLP problem. On the other hand, generative pre-trained transformer is trained for downstream-tasks while fine-tuning all the pre-trained parameters. The drawback of the two approaches is that they make use of a unidirectional language model to obtain language representations. In other words, the unidirectional language modelling may reduce efficiency of the transformers while handling down-stream NLP tasks. The reason is that the directional models proceed with reading text sequentially in left or right direction that limits to grasp the contextual meaning of the words depending on all surroundings. However, the transformer-encoder strategy of BERT digest a complete sequence at once that helps it to acquire the context of a word in relation to all of the neighbours in either left or right (so-called bidirectional). At this point, BERT having bi-directional language representation comes into play enhancing efficiency NLP tasks with fine-tuning scheme. BERT unravelled the unidirectional limitation with masked language model (MLM). This model first masks some percentage of the input tokens randomly and then it continues with the prediction of the masked words depending on its context [54].

BERT model is trained end-to-end under two unsupervised tasks: (i) MLM (also known as *Cloze Task* [56]) and (ii) the Next Sentence Prediction (NSP) [57].

As it was mentioned, BERT operates over vocabulary represented as discrete token sequences and a set of specialized tokens: SEP, CLS and MASK. For an input token, the input embedding is obtained with the summation of *token-specific* learned embeddings and corresponding encodings for *position* and *segment*. While the position defined to be the index of the token in the sequence, the segment is the index of the sentence of the particular token [57]. BERT input representation is shown in Figure 3 [58].

The MLM task is realized by dividing input tokens into disjoint sets: masked and observed. About 15% of tokens are masked (replaced) with the following scheme: (i) Masked tokens are replaced with MASK token 80% of the time, (ii) replaced with a random word with 10% percentage and (iii) and they are retained unchanged for 10% of portion. Following the masking process, the BERT model is trained to reconstruct the masked tokens based on the observed set [57].

The second unsupervised task, i.e. NSP, is particularly important for the tasks that require understanding the relation between two sentences such as question-answering and natural language inference. This binary NSP task is generated from any monolingual corpus as follows: If the sentences A and B are chosen for each pre-training case, 50% of the time B is the genuine *next sentence* labelled with *IsNext*. For the remaining 50% percentage, B is a random sentence from the corpus with label *NotNext*. The two tasks are given in Figures 4 and 5, respectively.

4.4. Fine-tuning BERT in down-stream NLP tasks

As it was stated, pre-trained BERT model can be fine-tuned for various NLP tasks by just additional single output layer without substantial modifications to its main architecture [54]. This approach allows

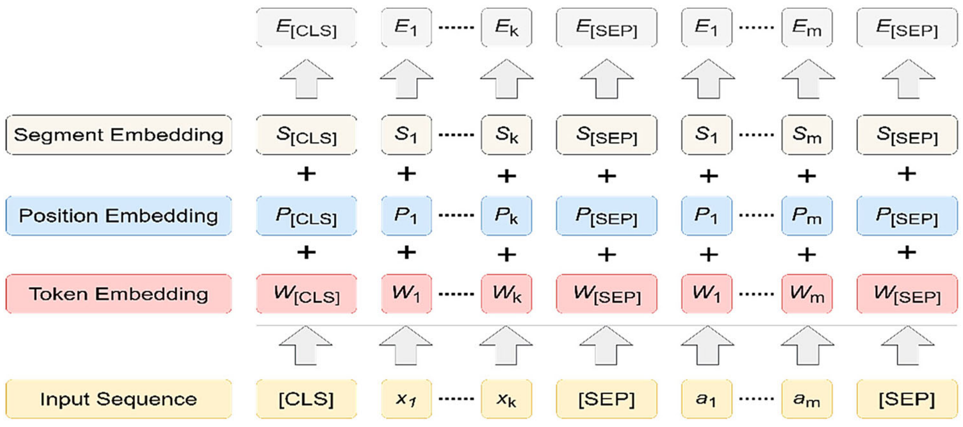
$E_{[CLS]}$ E_1 ...... E_k $E_{[SEP]}$ E_1 ...... E_m $E_{[SEP]}$
Segment Embedding
$S_{[CLS]}$ S_1 ...... S_k $S_{[SEP]}$ S_1 ...... S_m $S_{[SEP]}$
Position Embedding
$P_{[CLS]}$ P_1 ...... P_k $P_{[SEP]}$ P_1 ...... P_m $P_{[SEP]}$
Token Embedding
$W_{[CLS]}$ W_1 ...... W_k $W_{[SEP]}$ W_1 ...... W_m $W_{[SEP]}$
Input Sequence
[CLS] x_1 ...... x_k [SEP] a_1 ...... a_m [SEP]

Figure 3. BERT input representation. Sum of segment, position and token embeddings is the input embedding.

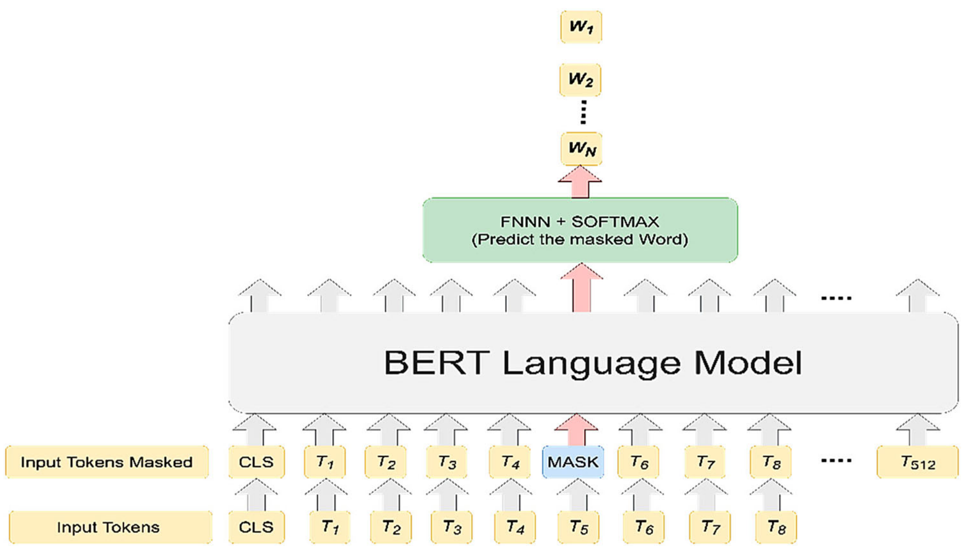

Figure 4. BERT unsupervised MLM task.

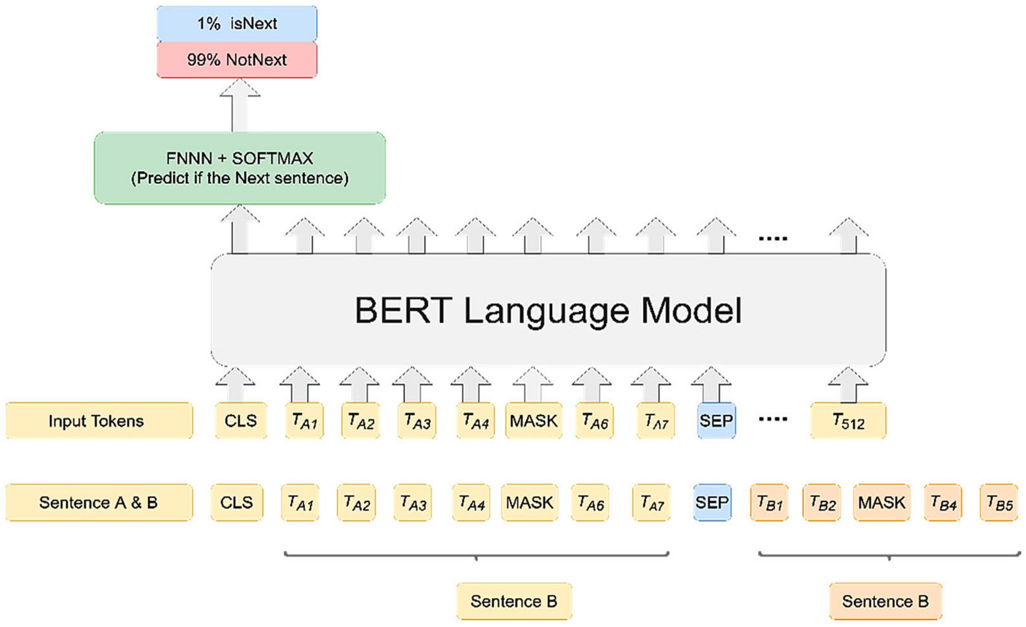

Figure 5. BERT unsupervised next sentence prediction task.

BERT to accomplish state-of-the-art results on various down-stream (supervised learning tasks from pre-trained models [59]) NLP tasks such as Named Entity Recognition (NER), Question Answering (QA), Sentiment Analysis (SA) and Text Classification (TC).

Basically fine-tuning is defined as follows: The default BERT model is a sequence classifier. In this context, final hidden state of the first word ([CLS]) from BERT is introduced to a fully connected layer for softmax evaluation [60]. More formally, lets $\boldsymbol{h}$ denote the final hidden state of the first token [CLS]. Then the prediction of label c [55] is calculated with Equation (8).

$$p(c|\mathbf{h}) = \text{softmax}(W\mathbf{h}) \tag{8}$$

where W is the task-specific weight matrix. BERT parameters are all fine-tuned mutually with W maximizing the log-probability of the correct label [55].

Self-attention mechanism in the Transformer allows BERT to model many NLP tasks by exchanging the proper inputs and outputs. Simply, task-specific input and outputs are plugged into BERT and all parameters are fine-tuned on an end-to-end principle [54]. For a text classification task, a single-layer feed-forward neural network with softmax output is put on top of the fine-tuned BERT model. A sample *binary sentiment classification* scheme is shown in Figure 6.

Having overviewed BERT pre-training and fine-tuning strategies, we first explain the various NLP problems in Turkish. Then the performances of baseline ML algorithms and BERT-base-multilingual architecture are compared in Section 6.

5. NLP problems from Turkish language literature

In order to show efficiency of BERT empirically, we have selected diverse datasets from Turkish NLP domain. In this section, we will present the selected NLP problems

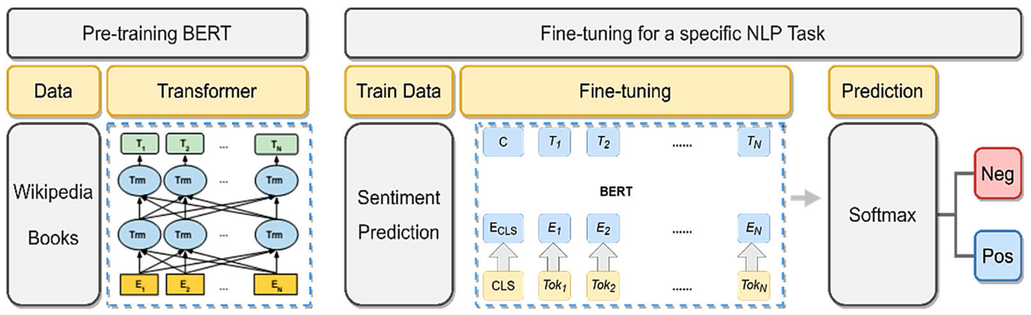

Figure 6. BERT Fine-tuning pipeline for a sample sentiment identification task.

Table 3. Statistical properties of six datasets with maximum sentence lengths in terms of words.

Dataset	Classes	Sample size	Type	Max lengths
Cyberbullying Identification	2	1500 sample in each category	Social Networking	25
News Text Classification	6	600 sample in each category	Topic	891
Emotion Recognition	6	3000–5000 in each category	Emotion	56
Spam SMS	2	normal 188 spam 329	Spam	10
Sentiment Hotel	2	5800 sample in each category	Sentiment	124
Sentiment Movie	2	26,700 sample in each category	Sentiment	166

and their corresponding results belong to ML models. While we handle the corresponding literature for datasets, we mainly focus on Turkish language and we add recent samples from corresponding literature where suitable. We made a search on the Web of Science and Google Scholar using appropriate keywords and to the best of our knowledge, this is the first study applying BERT neural language model to solve various Turkish language problems. We have studied five types of problems with six datasets from Turkish language domain and the statistical properties of datasets are provided in Table 3.

(1) *Cyberbullying Identification*: In the literature, this problem is researched as hate speech detection, harassment detection, cyberbullying identification and detection of offensive language usage. The latest sample research from literature that uses BERT was realized in [61] for Arabic (a morphologically rich language) offensive Tweets. For SVM baseline, they applied data cleaning, normalization, tokenization, varying n-gram feature models and stemming pre-processing steps. On the other hand, their BERT approach was a fine-tuning scheme to obtain the identification of offensive and inoffensive Tweets [61] without mentioned pre-processing tasks. The latest study for Turkish cyberbullying identification was experimented in [32] with the use of tokenization, word n-grams, tf-idf modelling on top of Artificial Neural Network (ANN) and they obtained *91%* in terms of F-measure score as their best value. Since we used the same dataset in our BERT evaluations, therefore we choose their best result as *baseline* in our evaluation.

(2) *Text Classification*: A recent study for Urdu, i.e. a low-resource language, made use of BERT in comparison to SVM for text categorization problem is given in [62]. Traditionally for SVM classification pipeline, they used preprocessing steps such as cleaning, stop-word removal, tokenization and lemmatization together with ten feature filtering algorithms. They obtained SVM and BERT-base text classification results to be similar in terms of F-measure. Text categorization is also a studied topic in Turkish language and we selected the recent TTC-3600 news dataset [35] with economy, culture-arts, health, politics, sports and technology categories. The authors used stop-word filtering, stemming and feature elimination approaches to process data before ML classifiers. They used NB, SVM, J48-tree and RF to evaluate the dataset and they obtained *91.03%* accuracy with RF classifier to be their best performance. We annotate this result later to evaluate with our BERT experiment.

(3) *Sentiment Identification*: Detection of polarity from user reviews is a widely studied research area to improve the quality of services. In a current study [63], a financial sentiment-index for Hong Kong stock exchange market was developed with RNN, LSTM and BERT. The authors compared the performances of three architectures for Chinese sentiment analysis and they obtained comparable results. Sentiment polarity detection also is in general a widely studied research topic for Turkish. As a baseline, we selected a hotel and movie datasets collected for Turkish sentiment research [33]. The collection recently studied with Ersahin et. al. [64] using SVM, NB and J48 tree algorithms on top of substantial preprocessing steps: normalization, tokenization, lemmatization and lexicon-based feature generation and feature filtering. Their best sentiment identification result was obtained with NB *88.68%* in movie reviews

and with SVM *92.26%* in hotel reviews in terms of accuracy. The two results are to be used as baselines while experimenting BERT model.

(4) *Spam SMS Identification*: An unsolicited message from a sender to a user mostly for commercial purposes is known as spam SMS. The importance of developing advanced spam filtering services is obvious. In their empirical study, Houlsby et al. made use of BERT-base to identify spam SMS and they obtained a significant accuracy of 95.1% for English language [65]. In case of Turkish, a recent study evaluating spam SMS detection research was done by Kaya et al. in [34]. The researchers used a sliding-window based motif discovery technique in tandem with feature selection strategies to obtain a data model. Then they used various classifiers RF, SVM, ANN and LR with the best classification result belonging LR to be *93.76%* in terms of accuracy. We made use of the dataset collected with the researchers as our baseline and we will present the corresponding results later in experiments section.

(5) *Emotion Identification*: Detection of emotions from textual data is important particularly for potential applications in marketing, political science, and human–computer interaction etc. In the study by Huang et al., the three emotions (happy, angry, sad) were classified with BERT-large and LSTM ensemble. They found the 77.09% average F-score for SemEval-2019 "EmoContext" dataset [66]. In the recent Turkish emotion detection study, Tocoglu et. al. collected a dataset called TREMO [36] and they experimented the dataset with various stemming preprocessing steps. They obtained four stemmed versions to detect happiness, fear, anger, sadness, disgust and surprise with SVM, J48 and NB. The best average accuracy was obtained with SVM to be *86.29%*. As we will make use of the collected dataset, this result is our baseline for the comparison.

6. Experimental study

In this section, we made use of mentioned datasets and their corresponding best predictive performances from literature to evaluate equivalent results obtained with BERT. We start with explanation of BERT fine-tuning parameters and we give details of the specifications of hardware used for experiments. Then, we will present and interpret the results of the experiments conducted.

6.1. BERT fine-tuning parameter selection

The designers of BERT in their article [54] recommends to keep all parameters constant except the *batch-size*, *learning rate (Adam)* and *epoch-size*. The recommended values for the parameters are as follows:

- Batch-size: 16,32
- Learning Rate: 5e-5, 3e-5, 2e-5
- Epoch-size: 2,3,4

We selected batch-size, learning rate and epoch-size **16**, **2e-5** and **4**, respectively, in all the experiments and we kept them constant. For our tasks, we selected pre-trained BERT-base-multi-language having number of Transformer blocks of 12, the hidden layer size of 768 and self-attention head of 12. BERT has single-language model devoted to English or Chinese and multi-language model that support 100 languages including Turkish. The BERT pipeline for multi-language applications does not require a specific tuning task. In particular, for multi-language version of BERT, no specific parameter tuning strategy was applied for Turkish other than the mentioned fine-tuning values.

While we carrying out experiments, we made use of a Personal Computer (PC) having 128 GB of RAM, Intel Core i9 9900X CPU and GeForce RTX 2080 Ti GPU having 11 GB of memory.

6.2. BERT architecture experiments

We mentioned the datasets from Turkish language studies literature and we summarize the pre-processing steps conducted in the corresponding studies in Table 3. However, being an essential operation of every text processing step, we did not include removal of punctuation, lower-case conversion and removal of ambiguous characters, etc. steps in Table 4. For the sake of convenience, we use symbolization of datasets as follows: cyberbullying (CYB), news text categorization (NEWS), emotion identification (EMOT), spam SMS (SPAM), sentiment hotel and sentiment movie (SHTL and SMOV) respectively.

It is observed from Table 4 that any Turkish language analysis problem makes use of pre-processing steps that can affect the performance of the ML algorithms. The empirical studies show this performance variation depending on the mentioned language processing tasks. It is observed that most of the studies of morphologically rich languages make use of these steps in some way to obtain a sufficient ML model. However, from BERT point of view, *no such preprocessing step is required* and the analyses of most tasks are conducted with state-of-the-art results. We present the

Table 4. Language pre-processing steps conducted in baseline studies.

Pre-processing steps	CYB	NEWS	EMOT	SPAM	SHTL	SMOV
Stop-word removal		+				
Tokenization	+		+	+	+	+
N-gram representation	+					
Stemming or lemmatization		+	+		+	+
Feature engineering		+	+	+	+	+
TF-IDF representation	+	+	+			
Lexicon-based processing	+				+	+

Table 5. Comparison of performances of BERT and ML Algorithms.

Dataset	Baseline ML	Performance Acc or F-1	BERT Acc	Performance comparison
CYB	ANN	91.00	98.27	+7.27
NEWS	RF	91.03	98.12	+7.09
EMOT	SVM	86.29	93.63	+7.34
SPAM	LR	93.76	92.51	−1.25
SHTL	SVM	92.26	99.01	+6.75
SMOV	NB	88.68	97.28	+8.60

Table 6. Evaluation of BERT experiments in terms of various metrics.

DAT	ACC	F1	Precision	Recall	MCC	K
CYB	0.9827	0.9826	0.9819	0.9834	0.9653	0.9650
NEWS	0.9812	0.9829	0.9811	0.9848	0.9620	0.9620
EMOT	0.9363	0.9401	0.9350	0.9453	0.8721	0.8720
SPAM	0.9251	0.9300	0.9384	0.9217	0.8497	0.8500
SHTL	0.9901	0.9901	0.9901	0.9901	0.9802	0.9800
SMOV	0.9728	0.9742	0.9722	0.9761	0.9445	0.9450

performances of the baseline algorithms and the result of BERT experiments in Table 5. Since the datasets are balanced we use Acc or F-1 score interchangeably which one is supported in the corresponding article.

Inspection of Table 5 in comparison with the best baseline performances show that the BERT generates enhanced prediction performances in five out of six datasets. Particularly, overall increase in the performances in average is about 7.5% except spam data. This empirical performance increase in various Turkish language is significant as a possible research direction. We demonstrate the resultant benchmark in Figure 7 visually.

The performance increase in the mentioned Turkish language problems is obvious. We now give more detailed results of BERT experiments in terms of Acc, F1-measure, Precision, Recall, MCC and κ are in Table 6.

From the above table, it is observed that the obtained Acc and F1 are meaningful and they are positively correlated with each other without showing any skewness.

In a ML analysis, Recall-Precision metrics produces important information about the quality of the predictions. The two metrics are needed to be minimized for an efficient classification task. However, there is a trade-off between them and they cannot be minimized both at the same time. In particular, for different unbalanced problem domains (for example health-diagnosis cases) minimizing one metric may be preferred. In our problems, the datasets used are balanced and the generated Recall-Precision pairs are in balance in terms of values. The Recall-Precision pair values are provided in Figure 8.

Inspection of Figure 8 shows that there is a correlation between Recall-Precision values. Therefore, it is reasonable to infer BERT architecture is successful to predict the categories of the tested problems.

MCC and κ are the other important metrics to demonstrate the efficiency of BERT neural model in the prediction tasks of the mentioned problems. As a reminder, we note that a successful ML model has MCC and κ values that tend to 1. In particular MCC for all experiments have value of above 0.85 that is an adequate performance criterion. This numerical observation is shown in Figure 9.

Furthermore from Figure 9, we may also observe that κ values are also reasonable having significantly higher values above baseline of 0.4. The smallest κ is 0.85 and the calculated values prove the statistical validation of BERT predictions.

As the last evaluation of experimental results of BERT model is the elapsed time while carrying out the experiments. The elapsed time on the abovementioned hardware for each of the tasks with 4 epoch are CYB: 2 h 4 min, NEWS: 2 h 34 min, EMOT: 1 h 46 min, SPAM:

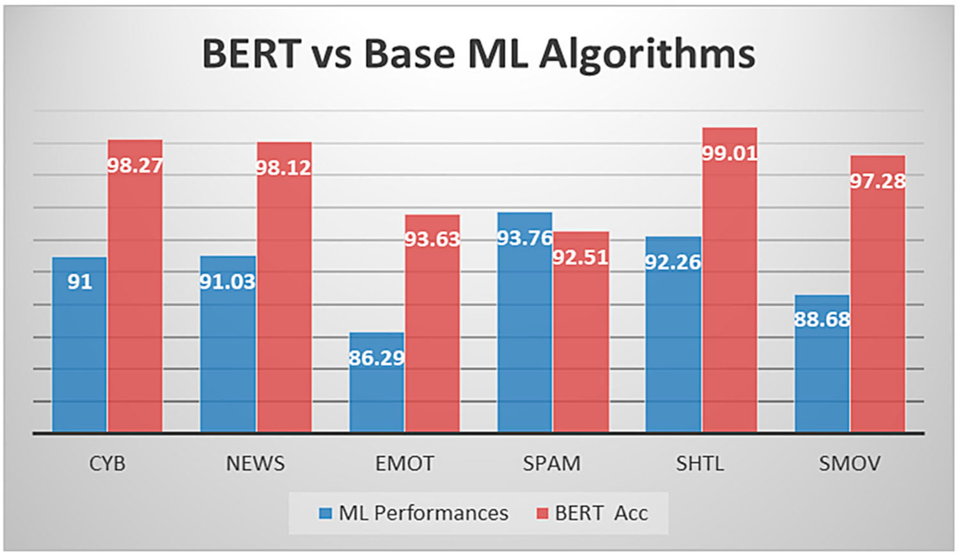

Figure 7. Comparison of BERT and base ML Algorithms.

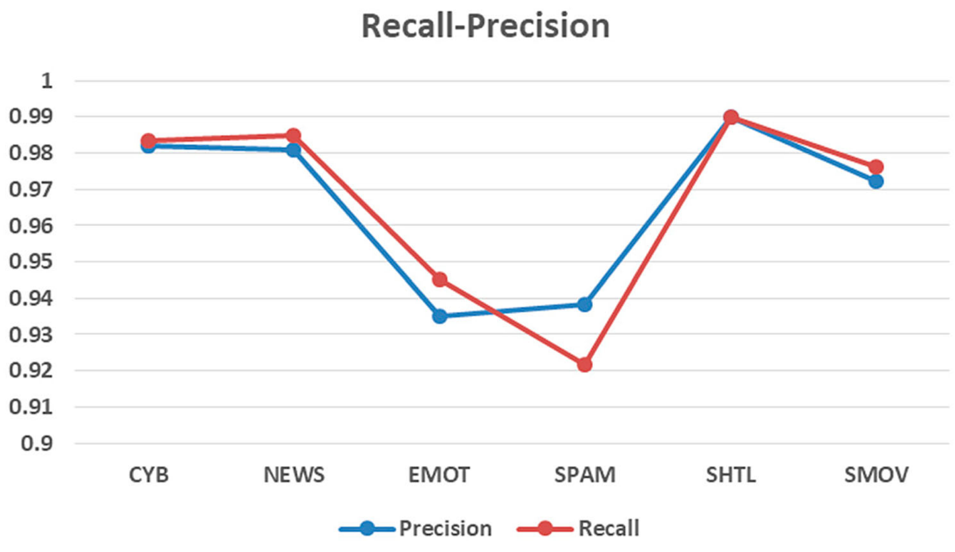

Figure 8. Recall-precision curve for all datasets.

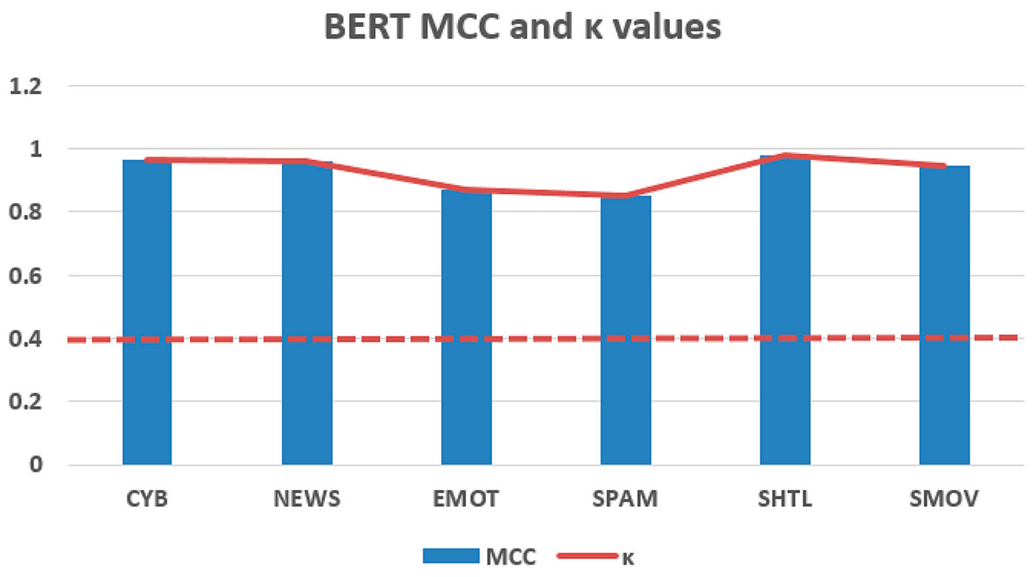

Figure 9. MCC and κ evaluation for BERT architecture.

23 min, SHTL: 7 h 45 min and SMOV: 34 h 26 min. In particular, for relatively larger datasets the training times to complete the tasks become enormous.

7. Conclusion

Morphologically rich languages are analysed with the use of various pre-processing techniques that affect performance of the resultant ML model. The combinations or variations of these steps change the obtained outcomes in terms of prediction efficiency. The mentioned pre-processing tasks need dense software-based steps and they are not applied in a strictly well-defined pipeline. On the other hand, BERT neural model has relatively straightforward application to solve various language tasks. Furthermore, even with its multi-language version, BERT may generate remarkable outcomes in NLP problems.

We used BERT to analyse five types of problems and six datasets from Turkish language domain and compared the obtained results with the *best* predictions of base-line ML algorithms from literature. It was empirically shown that BERT enhanced the prediction performances in all of the sets except spam dataset. And it is observed that the performance increase in BERT predictions are remarkable compared to base-line algorithms. The performance advances from 6.75 to 8.60% numerically.

The empirical evaluation of various datasets from Turkish NLP domain shows that morphologically rich languages may benefit from BERT-multi-language neural model in two significant points: (i) elimination of need for intimidating pre-processing language steps and (ii) enhancing the prediction ability. Although the required training time becomes larger even for a fine-tuning task, it could be possible to decrease the elapsed time with the use of light versions of BERT or some other pre-trained models.

Although we merely conducted a few Turkish language problems, it is expected that BERT multi-language model may similarly be applied to morphologically rich and low-resource languages such as Arabic, Czech, Hungarian, German and Finnish.

As a future direction, we plan to use lighter versions of BERT-like architectures to decrease training time periods while keeping prediction performances high.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Akın Özçift http://orcid.org/0000-0003-2840-1917

References

[1] Batbaatar E, Li M, Ryu KH. Semantic-emotion neural network for emotion recognition from text. IEEE Access. 2019;7:111866–111878.
[2] Wang D, Su J, Yu H. Feature extraction and analysis of natural language processing for deep learning English language. IEEE Access. 2020;8:46335–46345.
[3] Khan W, Daud A, Khan K, et al. Part of speech tagging in Urdu: comparison of machine and deep learning approaches. IEEE Access. 2019;7:38918–38936.
[4] Dong M, Li Y, Tang X, et al. Variable convolution and pooling convolutional neural network for text sentiment classification. IEEE Access. 2020;8:16174–16186.
[5] Kaliyar RK, Goswami A, Narang P, et al. FNDNet–A Deep convolutional neural network for fake news detection. Cogn Syst Res. 2020;61:32–44.
[6] Sailunaz K, Alhajj R. Emotion and sentiment analysis from twitter text. J Comput Sci. 2019;36:101003.
[7] Ren Y, Ji D. Neural networks for deceptive opinion spam detection: An empirical study. Inf Sci (Ny). 2017;385–386:213–224.
[8] Samant SS, Bhanu Murthy NL, Malapati A. Improving term weighting schemes for short text classification in vector space model. IEEE Access. 2019;7:166578–166592.
[9] Lan M, Tan CL, Su J, et al. Supervised and traditional term weighting methods for automatic text categorization. IEEE Trans Pattern Anal Mach Intell. 2009;31(4):721–735.
[10] Rudkowsky E, Haselmayer M, Wastian M, et al. More than bags of words: sentiment analysis with word embeddings. Commun Methods Meas. 2018;12(2–3):140–157.
[11] Rubenstein H, Goodenough JB. Contextual correlates of synonymy. Commun ACM. 1965;8(10):627–633.
[12] Schütze H, Pedersen JO. Information retrieval based on word senses. 1995.
[13] Khattak FK, Jeblee S, Pou-Prom C, et al. A survey of word embeddings for clinical text. J Biomed Inform X. 2019;4:100057.
[14] Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. J Mach Learn Res. 2003;3:1137–1155.
[15] Guo B, Zhang C, Liu J, et al. Improving text classification with weighted word embeddings via a multi-channel text CNN model. Neurocomputing. 2019;363:366–374.
[16] Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. ArXiv:1301.3781 [Cs]. September 6, 2013.
[17] Pennington J, Socher R, Manning C. Glove: global vectors for word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP); Doha, Qatar: Association for Computational Linguistics. 2014:1532–1543.
[18] Mikolov T, Grave E, Bojanowski P, et al. Advances in pre-training distributed word representations. ArXiv: 1712.09405 [Cs]. December 26, 2017.
[19] Devlin J, Chang M-W, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. In: J Burstein, C Doran, T Solorio, editors. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). Association for Computational Linguistics; 2019. p. 4171–4186.
[20] Peters ME, Neumann M, Iyyer M, et al. Deep contextualized word representations. ArXiv:1802.05365 [Cs]. March 22, 2018.
[21] Hao J, Wang X, Yang B, et al. Modeling recurrence for transformer. ArXiv:1904.03092 [Cs]. April 5, 2019.
[22] Li F, Jin Y, Liu W, et al. Fine-tuning bidirectional encoder representations from transformers (BERT)–based models on large-scale electronic health record notes: an empirical study. JMIR Med Inform. 2019;1–13.
[23] Alhaj YA, Xiang J, Zhao D, et al. A study of the effects of stemming strategies on Arabic document classification. IEEE Access. 2019;7:32664–32671.
[24] Demir H, Özgür A. Improving named entity recognition for morphologically rich languages using word embeddings. 2014 13th International Conference on Machine Learning and Applications. 2014:117–122.
[25] Uysal AK, Gunal S. The impact of preprocessing on text classification. Inf Process Manag. 2014;50(1):104–112.
[26] Mulki H, Haddad H, Ali CB, et al. Preprocessing impact on Turkish sentiment analysis. 2018 26th Signal Processing and Communications Applications Conference (SIU). 2018:1–4.
[27] Ebert S, Müller T, Schütze H. LAMB: a good shepherd of morphologically rich languages. EMNLP. 2016. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing: 742–752.
[28] Romanov V, Khusainova A. Evaluation of morphological embeddings for English and Russian languages. Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP; Minneapolis, USA: Association for Computational Linguistics. 2019:77–81.
[29] Belinkov Y, Durrani N, Dalvi F, et al. On the linguistic representational power of neural machine translation models. ArXiv:1911.00317 [Cs]. November 1, 2019.
[30] Jawahar G, Sagot B, Seddah D. What does BERT learn about the structure of language? Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Florence, Italy: Association for Computational Linguistics. 2019:3651–3657.
[31] Zhu Y, Heinzerling B, Vulić I, et al. On the importance of subword information for morphological tasks in truly low-resource languages. ArXiv:1909.12375 [Cs]. September 26, 2019.
[32] Bozyiğit, Alican, Semih Utku, and Efendi Nasiboğlu. Cyberbullying detection by using artificial neural network models. 2019 4th International Conference on Computer Science and Engineering (UBMK). 2019:520–24.
[33] Ucan A, Naderalvojoud B, Sezer EA, et al. SentiWordNet for new language: Automatic translation approach.

2016 12th International Conference on Signal-Image Technology Internet-Based Systems (SITIS). 2016: 308–15.

[34] Özdemir C, Yılmaz K. A new approach to filtering spam SMS: motif patterns. GUJSC. 2018;6(2):436–450.

[35] Kılınç D, Özçift A, Bozyigit F, et al. TTC-3600: A new benchmark dataset for Turkish text categorization. J Inf Sci. 2017;43(2):174–185.

[36] Tocoglu MA, Alpkocak A. TREMO: A dataset for emotion analysis in Turkish. J Inf Sci. 2018.

[37] Sak H, Güngör T, Saraçlar M. Resources for Turkish morphological processing. Lang Resour Eval. 2011;45(2): 249–261.

[38] Vylomova E, Cohn T, He X, et al. Word representation models for morphologically rich languages in neural machine translation. Proceedings of the First Workshop on Subword and Character Level Models in NLP. Copenhagen, Denmark: Association for Computational Linguistics. 2017:103–108.

[39] Hans K, Milton RS. Improving the performance of neural machine translation involving morphologically rich languages. ArXiv:1612.02482 [Cs]. January 8, 2017.

[40] Oflazer K. Turkish and Its challenges for language processing. Lang Resour Eval. 2014;48(4):639–653.

[41] Kışla T, Karaoglan B. A hybrid statistical approach to stemming in Turkish: an agglutinative language. Anadolu Univ J Sci Technol Appl Sci Eng. 2016;401–412.

[42] Abudukelimu H, Liu Y, Chen X, et al. Learning Distributed Representations of Uyghur words and morphemes. CCL. 2015; 202–211.

[43] Wolk K. Machine Learning in Translation corpora processing. 1st ed. CRC Press; 2019.

[44] Nuzumlalı MY, Özgür A. Analyzing stemming approaches for Turkish multi-document summarization. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Doha, Qatar: Association for Computational Linguistics. 2014:702–706.

[45] Mogotsi I, Christopher C, Manning D, et al. Introduction to information retrieval. Inf Retr Boston. 2010;13(2):192–195.

[46] Tantuğ AC, Adali E, Oflazer K. Machine translation between Turkic languages. Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions. Prague, Czech Republic: Association for Computational Linguistics. 2007:189–192.

[47] Vuckovic K, Bekavac B, Silberztein M, et al. Automatic Processing of various levels of linguistic phenomena: selected papers from the NooJ 2011 International Conference.

[48] Akdoğan Ö, Ayşe Özel S. Nitelik Çıkarımı Yöntemlerinin Türkçe Metinlerin Sınıflandırılmasına Etkisi. Çukurova Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi. September 30, 2019:95–108.

[49] Kowsari K, Meimandi KJ, Heidarysafa M, et al. Text classification algorithms: a survey. Information. 2019;10(4):150.

[50] Uysal AK, Gunal S, Ergin S, et al. The impact of feature extraction and selection on SMS spam filtering. 2013.

[51] Tahir M, Haq AU, Asghar S, et al. A classification model for class imbalance dataset using genetic programming. IEEE Access. 2019;7:71013–71037.

[52] Kobayashi VB, Mol ST, Berkers HA, et al. Text classification for organizational researchers: a tutorial. Organ Res Methods. 2018;21(3):766–799.

[53] Gao Z, Feng A, Song X, et al. Target-dependent sentiment classification with BERT. IEEE Access. 2019;7: 154290–154299.

[54] Devlin J, Chang M-W, Lee K, et al. BERT: Pre-training of deep bidirectional transformers for language understanding. ArXiv:1810.04805 [Cs]. May 24, 2019.

[55] Sun C, Qiu X, Xu Y, et al. How to fine-tune BERT for text classification? ArXiv:1905.05583 [Cs]. February 5, 2020.

[56] Taylor WL. 'Cloze procedure': a new tool for measuring readability. Journalism Q. 1953;30(4):415–433.

[57] Lu J, Batra D, Parikh D, et al. ViLBERT: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. ArXiv:1908.02265 [Cs]. August 6, 2019.

[58] Wu X, Lv S, Zang L, et al. Conditional BERT contextual augmentation. December 17, 2018.

[59] McCann B, Bradbury J, Xiong C, et al. Learned in translation: contextualized word vectors. ArXiv:1708.00107 [Cs]. June 20, 2018.

[60] Ma G. Tweets classification with BERT in the Field Of Disaster Management | Semantic Scholar. Accessed May 4, 2020.

[61] Mubarak H, Rashed A, Darwish K, et al. Arabic offensive language on twitter: Analysis and experiments. ArXiv:2004.02192 [Cs]. April 5, 2020.

[62] Asim MN, Ghani MU, Ibrahim MA, et al. Benchmark performance of machine and deep learning based methodologies for Urdu text document classification. ArXiv:2003.01345 [Cs]. March 3, 2020.

[63] Hiew J, Git Z, Huang X, et al. BERT-based financial sentiment index and LSTM-based stock return predictability. ArXiv:1906.09024 [q-Fin]. June 21, 2019.

[64] Erşahin B, Aktaş Ö, Kilinç D, et al. A hybrid sentiment analysis method for Turkish. Turk JElec EngComp Sci. 2019;27(3):1780–1793.

[65] Houlsby N, Giurgiu A, Jastrzebski S, et al. Parameter-efficient transfer learning for NLP. ArXiv:1902.00751 [Cs, Stat]. June 13, 2019.

[66] Huang C, Trabelsi A, Zaïane OR. ANA at SemEval-2019 Task 3: contextual emotion detection in conversations through hierarchical LSTMs and BERT. ArXiv:1904.00132 [Cs]. May 31, 2019.

[67] Botha JA. Probabilistic modelling of morphologically rich languages. ArXiv:1508.04271[Cs]. August 18, 2015.