ISSN 2007-9737

The Impact of Training Methods on the Development of Pre-Trained Language Models

Diego Uribe*, Enrique Cuan, Elisa Urquizo

Tecnológico Nacional de México,
Instituto Tecnológico de La Laguna,
Mexico

{duribea, ecuand, eurquizob}@lalaguna.tecnm.mx

Abstract. The focus of this work is to analyze the implications of pre-training tasks in the development of language models for learning linguistic representations. In particular, we study three pre-trained BERT models and their corresponding unsupervised training tasks (e.g., MLM, Distillation, etc.). To consider similarities and differences, we fine-tune these language representation models on the classification task of four different categories of short answer responses. This fine-tuning process is implemented with two different neural architectures: with just one additional output layer and with a multilayer perceptron. In this way, we enrich the comparison of the pre-trained BERT models from three perspectives: the pre-training tasks in the development of language models, the fine-tuning process with different neural architectures, and the computational cost demanded on the classification of short answer responses.

Keywords. Language models, pre-training tasks, BERT, fine-tuning.

1 Introduction

Currently, the development and deployment of Large Language Models (LLMs) is a common scenario in the sphere of NLP due to the development paradigm known as Self-Supervised Learning (SSL). This learning paradigm, also known as a process of two steps: pre-training and fine-tuning, outlines a generic framework for transferring knowledge [18, 2].

While pre-training a LLM produces semantic representations by processing unlabeled data, fine-tuning makes use of such representations for a particular downstream learning task.

In this way, the performance of this new task depends significantly on the quality of the semantic representations, which in turn depend on the quality of the training methods for the development of a LLM. Thus, how to produce good quality representations? We focus our attention on the analysis of the training methods for producing semantic representations to be transferred to make the definition of a learning model, for a particular downstream language task, a non-complex issue.

As a result of research on representation learning, a semantic vector known as embedding is nowadays the building block for a wide range of NLP tasks.

Since this semantic vector denotes a point in high-dimensional space, modeling similarity between words is straightforward. Two main types of word embeddings have been developed: static and contextual embeddings. Static embeddings are also known as context independent embeddings, as such representations are unique for each word and ignore the word's context.

Glove [15] and word2vec [14] are classic examples of this kind of embeddings. On the other hand, contextual embeddings are also known as context-dependent embeddings, as each word is represented by a different vector for each context in which it is used.

In other words, contextual embeddings allow us to represent multiple senses of a particular word. ELMo [16] and BERT [4] are examples of contextual embeddings. The mechanism for acquiring these embeddings is known as pre-training, a process defined as the computation

of large document collections in order to learn the semantic vectors corresponding to words or sentences. Actually, pre-trained language models denote the mechanism for acquiring these semantic representations that have been developed by using two deep learning architectures: recurrent neural networks (RNNs) and transformer networks.

ELMo is an example of a pre-trained language model based on RNNs, whereas BERT is a classic example of a pre-trained language model based on transformer networks. In this work, we examine BERT, a pre-trained language model based on a bidirectional transformer encoder which is characterized by a bidirectional self-attention mechanism to produce contextual embeddings.

There are many BERT models, all variants on the original BERT, available to perform some downstream task like classification or tagging. From the model collection available at TensorFlow Hub [22], we analyze three BERT models: the original, the universal, and the compact BERT model.

In terms of representation learning, what makes one BERT model better than another? Is there any significant difference in the quality of the contextual embeddings between these three BERT models? To answer these questions, we first analyze the pre-training process of a bidirectional language model as BERT, and then the fine-tuning process to transfer the embedded knowledge to a downstream language task.

2 Motivation Behind the Work

The guide to conducting our study is clearly defined with the following research question: what is the impact of the training methods for each BERT model on the quality of the linguistic representations produced by these models?

Thus, the motivation behind the training methods for each BERT model is to perceive the similarities and differences between the various training techniques to produce semantic knowledge to be embedded via fine-tuning.

Since BERT is a bidirectional encoder, and thus it is able to attend to the whole context of a particular input element (left and right of the current input), the training method is based on a cloze task [21]. Masked Language Model (MLM) is the original unsupervised training method where the model learns to predict the missing words of a text. By learning to predict the masked words, the model produces suitable word-level representations.

Another unsupervised task for the training of BERT is to deal with the relationship between pairs of sentences. Next Sentence Prediction (NSP) is an unsupervised training method where the model learns to predict such connection between pairs of sentences. Now, the pre-training method for the universal BERT model is a bit different.

The purpose is to improve the semantic representations at sentence level by implementing a dual encoder based on the combination of the BERT original training methods: the integration of NSP with MLM training is denominated by its authors as the Conditional Masked Language Model (CMLM) [28].

The third language model studied in this work is the compact BERT model. As LLMs have a high computational cost, this small model was created with the purpose of not only reducing the computational cost but also using the same self-supervised learning paradigm in its development [24].

We then conduct an empirical evaluation via fine-tuning to transfer the embedded knowledge to a downstream language task as classification. The representations obtained from the BERT models are transferred to a classifier model, commonly represented as a simple multiperceptron, to be fine-tuned to the peculiarities of a downstream task as short answer responses classification.

The collection of short answer responses was created with the intention of automated assessment of written responses [3]. Each instance in the collection denotes a short answer corresponding to a particular story of a specific domain where the grade is defined in terms of levels of quality.

In other words, the fine-tuning process performs a downstream task as multi-class classification where a short answer is assigned into one of the multiple rubrics of the responses. Thus, we have described the perspective from which a learning paradigm known as Self-Supervised Learning is

analyzed. The primary contributions of our work are summarised as follows:

– To provide insights about the impact of training methods in the development of pre-trained models. The pre-training process for each BERT model is described to consider similarities and differences between them.
– To conduct an empirical evaluation on semantic linguistic representations. The fine tuning process is implemented on a downstream classification task with a learning model defined in terms of the semantic representations produced by each BERT model.
– To offer additional insight into the computational resources demanded by the language models. The experimentation carried out allows us to detail the computational cost incurred by each BERT model.

3 BERT Pre-training and Language Models

We describe in this section the language models with which BERT has been trained for learning meaning representations for words and sentences: MLM [4], NSP [4], CMLM [28] and Distillation [24]. But we first briefly take a look at BERT and its self-attention mechanism that has impacted the world of NLP.

3.1 BERT: Bidirectional Encoder Representations from Transformers

In its broadest sense, the transformer consists of an encoder-decoder architecture. However, BERT is a transformer model that includes only the encoder component.

Unlike other popular embedding models (e.g., word2vec) that produce static embeddings irrespective of the context, BERT generates dynamic embeddings based on the context so multiple embeddings are produced for the multiple contexts in which a particular word can be used [4]. In order to generate context-based embeddings, the attention mechanism of the transformer plays a crucial role in the encoding process.

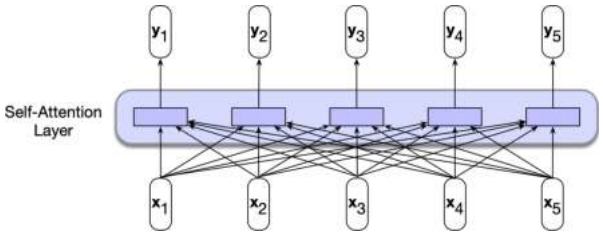

Fig. 1. Bidirectional self-attention model. This figure corresponds to [10]

Self-attention, a special type of attention, emerged as a more efficient alternative to overcome the limitations of the RNNs: capturing long-term dependencies is one of the major challenges with RNNs [25].

Self-attention takes a holistic approach to the analysis of the linguistic elements: instead of considering only the previous elements in the input, self-attention compares each element with all the sequence elements in order to understand how words relate to each other over long distances.

Given a sequence of input elements $(x_1, \ldots, x_n)$, Figure 1 shows how the output of a particular element y_i depends on the comparisons between the input x_i and the preceding and following elements x_j.

In other words, the self-attention mechanism is responsible for considering each element of the entire input sequence and mapping them to contextualized output vectors. A formal description of the output values (vector y) is based on three concepts:

– **Query**: The current focus of attention.
– **Key**: Preceding and following input to be compared with the current focus of attention.
– **Value**: Computation of the output for the current focus of attention.

In this way, each element of the input vector **x** is represented in terms of these concepts and the corresponding weights:

$$\begin{aligned} q_i &= W^Q x_i, \\ k_i &= W^K x_i, \\ v_i &= W^V x_i. \end{aligned} \tag{1}$$

Then, the output y_i corresponding to each input element x_i is:

$$y_i = \sum_{j=i}^{n} \alpha_{ij} v_j, \tag{2}$$

where the alpha weights represent the proportional relevance of each input to the current focus of attention:

$$\alpha_{ij} = \frac{\exp(\text{score}_{ij})}{\sum_{k=1}^{n} \exp(\text{score}_{ik})}, \tag{3}$$

$$\text{score}_{ij} = q_i \, k_j. \tag{4}$$

Thus the comparison of each element with the rest of the sequence elements take place in parallel. This means simultaneous access to all sequence elements and therefore simultaneous computation of the relevance of each sequence element. In this way, the step-by-step processing of intermediate recurrent connections is eliminated.

3.2 BERT Training Techniques

We describe in this section the language models with which BERT has been trained for learning meaning representations for words and sentences: MLM [4], NSP [4], CMLM [28] and Distillation [24].

3.2.1 Masked Language Modeling (MLM)

Masked Language Modeling is the approach to training a deep bidirectional transformer as BERT to learn contextual word-level representations [4].

MLM is basically a cloze task [21]: some percentage of the input tokens are masked in a random way, in order to figure out those masked tokens. More precisely, each token of the sequence can be:

– masked
– replaced with another token from the vocabulary
– left unchanged

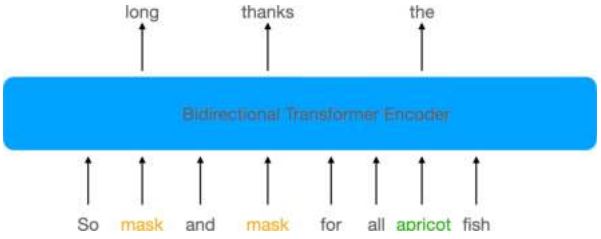

Fig. 2. Masked Language Model training

Figure 2 shows this training task. In this example, three of the input tokens are selected, two of which are masked (long and thanks) and the third (the) is replaced with a tangential token from the vocabulary.

The purpose is to predict the original words for each of the masked tokens as well as the tangential token and in this way to reproduce the original input sequence. MLM is an unsupervised learning method as a large corpus of unannotated input sequences is used for training.

The output vector for each of the masked tokens (h_i) is multiplied by a learned set of classification weights W_v in order to take a softmax to produce a probability distribution over the vocabulary:

$$y_i = \text{softmax}(W_v \, h_i). \tag{5}$$

3.2.2 Next Sentence Prediction (NSP)

Next Sentence Prediction (NSP) is another unsupervised task for the training of BERT on how to deal with the relationship between pairs of sentences [4].

As many applications such as paraphrase detection or entailment demand determining how close or distant two sentences are, NSP is an unsupervised training method where the model learns to predict such connection between pairs of sentences.

In the particular case of BERT, 50% of the training pairs denote adjacent sentences whereas the other 50% of the pairs denote unrelated sentences as the second sentence is randomly selected. In addition to the input elements of the sentences, two new tokens are added to conduct a proper training: the token [CLS] is prepended to the input sentence pair, and the token [SEP] is

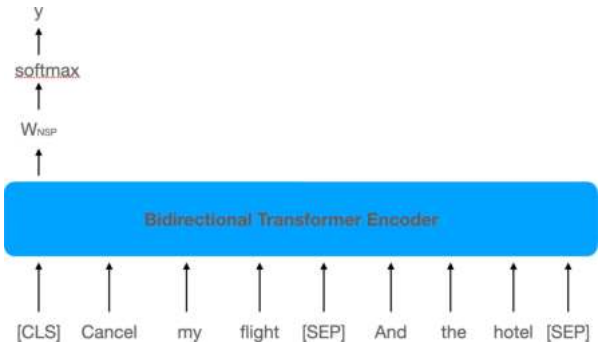

Fig. 3. Next Sentence Prediction training

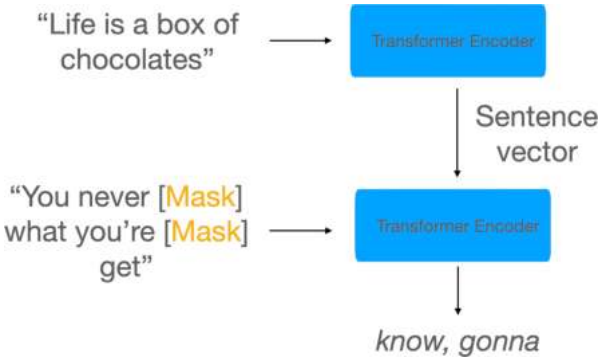

Fig. 4. Conditional MLM training

placed between the sentences and after the final token of the second sentence. Figure 3 shows this training task. While the role of the token [SEP] is obvious, the token [CLS] represents the output vector associated with the final layer of the transformer.

And it is precisely this output vector that denotes the next sentence prediction. The output vector for each training pair (h_i) is multiplied by a learned set of classification weights W_{NSP} in order to take a softmax to produce a two-class prediction:

$$y_i = \text{softmax}(W_{\text{NSP}}\, h_i). \tag{6}$$

This NSP task was inspired by the framework developed by Logeswaran and Lee for learning sentence representations from unlabeled data [13]. The key point of their work was the replacement of a generation objective, that is, the generation of a context sentence given an input sentence.

Instead, they replace the decoder with a classifier to predict the target sentence from a set of candidate sentences. In this way, the NSP training task takes advantage of this antecedent work to allow BERT to be able to produce sentence-level representations.

3.2.3 Conditional Masked Language Modeling (CMLM)

Conditional Masked Language Modeling is an alternative approach to training a deep bidirectional transformer as BERT to learn effective sentence-level representations [28]. Basically, CMLM is a training method that combines two training tasks: Next Sentence Prediction (NSP) and MLM.

The main idea of CMLM is learning sentence representations by optimizing the performance on the MLM task. The architecture of CMLM is based on the use of two transformer encoders and the processing of pair of sentences such as the NSP method does. From each pair of sentences, the first sentence becomes the input into an encoder that produces a sentence vector.

This sentence representation is then provided to the second encoder to perform the MLM task on the second sentence by making use of the learning weights generated by the first encoder to produce the sentence representation. Since the sentence vector is projected into N spaces, the MLM of the second sentence can result from observing more than one representation.

In this way, the optimization of the MLM task depends on the sentence vector representation of the adjacent sentence. Last but not least, this dependency of the MLM task on the sentence vector representation of the adjacent sentence is the reason to include the word "conditional" in the name of this language model: Conditional Masked Language Model. Figure 4 shows the architecture of this training task.

This CMLM training task was inspired by the Skip-Thought work developed by Kiros et al. for learning generic sentence representations from a large training corpus of contiguous text [11]. The key point of their work was the replacement of composition operators based on the mapping

of word embedding to sentence representations. Instead, they replace the composition operator with a sentence encoder to encode a sentence to predict the sentences around it: the previous and the next sentence. In this way, the CMLM training task takes advantage of this antecedent work to allow BERT to be able to improve sentence-level representations.

3.2.4 Knowledge Distillation

Building a compact model revolves around knowledge distillation: the standard technique for model compression [8].

Since LLMs have a high computational cost, research on the development of a small model was guided by not only reducing the computational cost but also by using the same self-supervised learning paradigm in its development.

Indeed, building a compact model proved to be possible by applying the standard pre-training and fine-tuning process but a different training strategy, based on a compression technique known as knowledge distillation, was implemented.

Basically, this distillation technique consists of a student-teacher training method where the teacher, a robust LM, transfers knowledge to the student, a small LM to be developed, through its predictions for unlabeled training examples.

Figure 5 shows the knowledge distillation process incorporated in the development and implementation of a compact BERT model [24]. The training resources demanded by the process are the following:

- **Teacher**: The teacher is a LLM which can be either a BERT-base or a BERT-large pre-trained language model.
- **Student**: The student is the compact model to be built. Whereas the total number of parameters is 110 million in BERT-base, the initial size for a tiny model is 4 million parameters.
- **Label data (D_L)**: A set of N training examples $(x_1, y_1), \ldots, (x_N, y_N)$, where x_i is an input and y_i is a label.

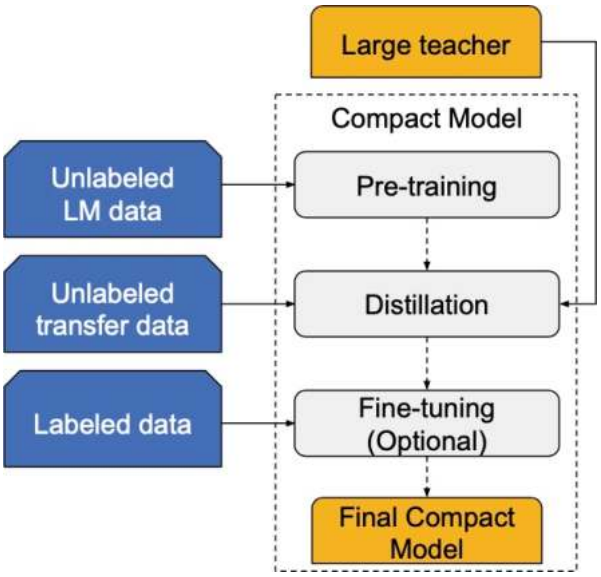

Fig. 5. Knowledge Distillation process. This figure corresponds to [24]

- **Unlabeled training data (D_T)**: A set of M input examples $x_1^{'}, \ldots, x_M^{'}$ obtained from a distribution not necessarily identical to the distribution of the labeled set.

  This dataset is used by the teacher for the transfer of knowledge to the student by making available its predictions for instances $x_m^{'}$.
- **Unlabeled language model data (D_{LM})**: it is an unannotated text collection for unsupervised learning of text representation by using MLM as training method. And a procedure for a sequence of three training operations executed by the algorithm (Figure 1).
- **Pre-training on D_{LM}**: pre-training of the compact model with MLM as training method (Line 1).
- **Distillation on D_T**: transfer knowledge to the student. Once the student is prepared, the teacher transfer its knowledge to the student via its predictions to strengthen the compact model.

  Line 3 shows the estimation of the cross-entropy loss between teacher and student predictions, this loss is then used to update the student model. (Line 4).

Table 1. BERT models and unsupervised training methods

BERT Model	MLM	NSP	CMLM	Distillation
Original	×	×		
Universal	×	×	×	
Compact	×			×

Algorithm 1 Knowledge Distillation algorithm. This figure corresponds to [24]

Require: student θ, teacher Ω, unlabeled LM, data $\mathcal{D}_{LM}$, unlabeled transfer data $\mathcal{D}_T$, labeled data $\mathcal{D}_L$

1: Initialize θ by pre-training and MLM$^+$ on $\mathcal{D}_{LM}$
2: **for each** $x \in \mathcal{D}_T$ **do**
3: Get loss $L \leftarrow -\sum_y P_\Omega(y|x) \log P_\theta(y|x)$
4: Update student $\theta \leftarrow \mathrm{BACKPROP}(L, \theta)$
5: **end for**
6: Fine-tune θ on $\mathcal{D}_L$ ▷ Optional step.
7: **return** θ

– **Fine-tuning on** D_L: Line 6 shows this optional step. The compact model is fine-tuned on end-task labeled data. In other words, the similarity between the distribution of the transfer and labeled datasets is perceived in this step.

This compact model is compared with two contemporary works that also use distillation for transfer knowledge. Both works initialize the student with a BERT model truncated, that is, the bottom layers of a 12-layer BERT model are used for the initialization of the student.

However, the distillation process is different. Whereas Patient Knowledge Distillation performs task-specific distillation [20], DistillBert makes use of a more expensive LM teacher as distillation is performed on general-domain data [19].

3.3 BERT Models

As we previously said, the motivation behind this work is to study three pre-trained BERT models and their corresponding unsupervised training tasks.

The previous section describes each unsupervised training task and Table 1 shows similarities and differences between the BERT models in terms of the training methods used in their development.

As we see in Table 1, MLM and NSP are unsupervised training tasks that characterize the development of the **O**riginal BERT model [5]. This model[1] consists of $L = 12$ encoder layers, a hidden size of $H = 768$, and $A = 12$ attention heads representing a total of 110M parameters.

On the other hand, the development of the **U**niversal BERT model is based on CMLM, an unsupervised training task that integrates MLM and NSP in order to optimize the semantic representations at sentence-level [27].

This model[2], that extends the BERT transformer architecture, maps text into high dimensional vectors to capture sentence-level semantics. Last but no least, we have a very different trained model:

The **C**ompact BERT model based on an initial model trained on MLM (the student) to eventually improve its performance by knowledge distillation from the teacher [23].

This model[3] consists of $L = 4$ encoder layers, a hidden size of $H = 512$, and $A = 8$ attention heads representing a total of 28M parameters.

4 Experimental Evaluation

Once we have described the training methods for each BERT model, we want to know its behavior on a particular text-processing task. So, the experimentation conducted is detailed in this section.

First, we explain the fine-tuning process of the pre-trained language models previously mentioned to perform a downstream task as sequence classification. Then, the dataset characteristics are exposed and the results of each BERT model are exhibited.

[1] bert_en_uncased_L-12_H-768_A-12
[2] universal-sentence-encoder-cmlm/multilingual-base
[3] small_bert/bert_en_uncased_L-4_H-512_A-8

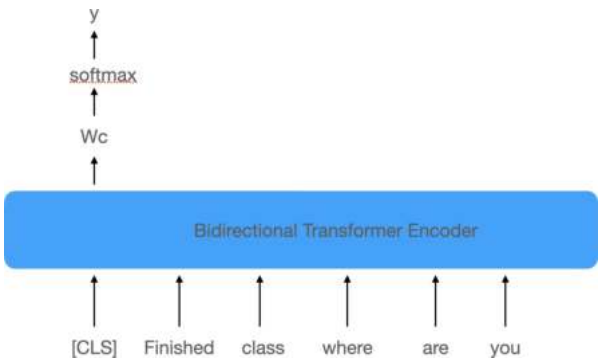

Fig. 6. Sequence classification with a bidirectional transformer encoder

4.1 Fine-Tuning

The process to make use of the representations produced by the pre-trained language models is known as fine-tuning. These semantic representations are helpful to build a sort of pipeline application to cope with NLP tasks such as named entity tagging or sequence classification.

In the case of sequence classification, the key point is the representation of the entire input sequence. Whereas in RNNs the hidden layer corresponding to the last input element denotes the entire sequence, an additional vector in the transformer encoder captures the entire sequence.

This is the reason why this additional vector is called the sentence embedding. The additional vector is symbolized by the [CLS] token which is prepended to the input sequences.

Figure 6 shows the architecture of a transformer encoder for sequence classification where the output of the encoder represented by [CLS] is provided to a neural network classifier that makes the category decision.

By using a labeled dataset, the sequence classification task entails to learn a set of weights (W_C) in order to map the output vector (Y_{CLS}) to a set of categories:

$$y = \text{softmax}(W_C\, Y_{\text{CLS}}). \tag{7}$$

4.2 Data

The dataset used in this experimentation is part of an ambitious research project denominated the Automated Student Assessment Prize (ASAP) [7] for automated grading of student-written responses sponsored by The William and Flora Hewlett Foundation.

The purpose is to explore new forms of testing and grading methods and to reduce the cost of human graders by automating the student assessment. Three stages set up the ASAP project:

- **Phase 1**: Analysis of essays: Long form response.
- **Phase 2**: Analysis of short answers: Short form response.
- **Phase 3**: Analysis of charts/graphs: Symbolic mathematical/logical reasoning.

The focus of our attention is the collection of short-answers corresponding to the phase 2 [3]. Each instance in the collection denotes a short answer corresponding to a reading passage from a broad range of disciplines: from English Language Arts to Science.

More specifically, the dataset is divided into 10 collections, where each one is described by a particular reading passage corresponding to a particular discipline and where the grade is defined in terms of levels of quality or categories.

For instance, the following text is an example of a short answer response where the range of the score is three: 0 (not proficient), 1 (partially proficient), or 2 (proficient).

“Paul is shocked that Mr. Leonard didn’t tell him that he broke all the records he did, and that he won the 400 meter hurdles at nationals when he was only a freshman. Paul also realizes that Mr. Leonard had been trying to help him because he too, was good at something, but couldn’t do it because he didn’t get good enough grades, because he couldn’t read.”

The average length of each answer is approximately 50 words and most training sets contain around 1,800 responses that have been

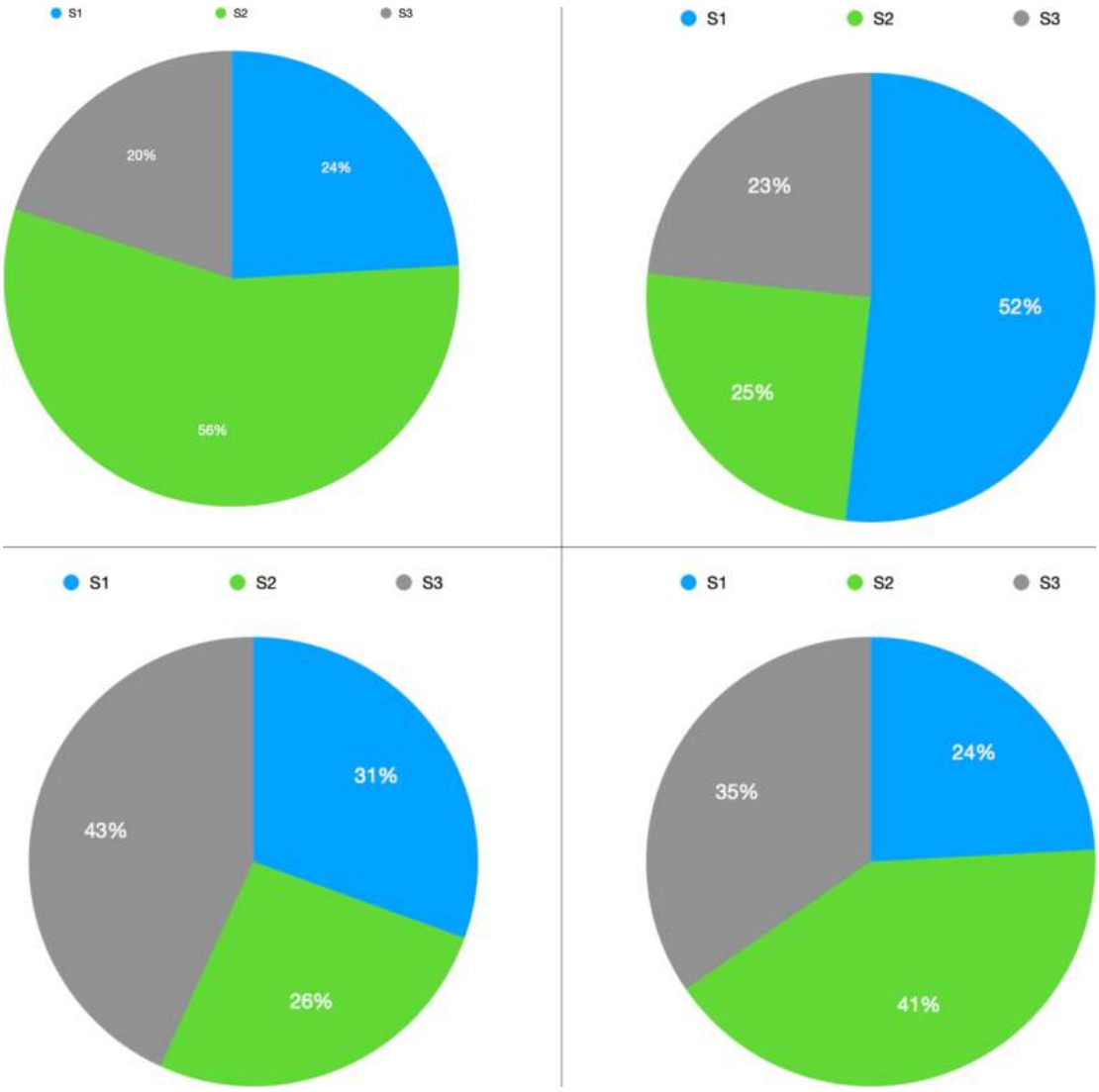

Fig. 7. Data distribution of the short-answers collections 3, 7, 8, 9

randomly selected from a sample of approximately 3,000. From the 10 training collections available in the dataset, we select four training sets where three levels of quality define the grade of each answer.

In other words, the fine-tuning process implemented in our experimentation performs a downstream task as multi-class classification where a short answer is assigned into one of the multiple rubrics of the responses.

The distribution of responses to rubrics corresponding to each training collection is shown in Figure 7.

4.3 Results

In our experiments, we adopt two strategies to the downstream task: the simple use of the embeddings obtained from the pre-trained model, and a more refined optimization of such embeddings via an added classic neural network.

Table 2. Results corresponding to each BERT model for each network architecture and each dataset

BERT model	Architecture	Dataset 3	Dataset 7	Dataset 8	Dataset 9
Compact	Simple	0,64	0,60	0,62	0,65
Compact	Layers	**0,71**	**0,66**	**0,64**	**0,72**
Original	Simple	0,60	0,55	0,56	0,62
Original	Layers	0,61	0,53	0,54	0,63
Universal	Simple	0,69	0,68	0,66	0,72
Universal	Layers	**0,74**	**0,74**	**0,72**	**0,79**

Although there are more sophisticated neural network models such as CNN and RNN, we consider these two simple and basic options as our purpose is to perceive the quality of the embeddings produced by different training methods rather than to obtain a high precision on the downstream task. Thus, the downstream network architectures implemented are:

– Simple: a simple dense layer is used to adjust the pre-trained embeddings obtained from pooled_output. For example, since the number of hidden units of the original BERT model is 768, and our experimentation performs a downstream three-class classification, the number of parameters to be adjusted is 2,307.
– Layers: three dense layers are used to adjust the pre-trained embeddings obtained from pooled_output. The first and second layers contain 64 and 32 hidden units respectively, and since the number of hidden units of the original BERT model is 768, and our experimentation performs a downstream three-class classification, the number of parameters to be adjusted is 51,395.

As the size of the short-answers collections is small, the performance evaluation of the pre-trained models was conducted by the cross-validation method to use all the responses corresponding to a particular domain.

We train our downstream learning models with an Adam optimizer with a learning rate of 0.001, three-fold cross-validation and 25 epochs.

We also apply dropout with $\rho = 0.2$ across layers of the downstream networks to prevent overfitting. Table 2 shows the results obtained in the fine-tuning process where classification of the collection of short-answers is the downstream task implemented for the analysis of the semantic representations obtained from the pre-trained BERT models.

The results are expressed in terms of the F1 score corresponding to each BERT model for each network architecture and each training set. For example, the first row shows a F1 score of 0.64 obtained with the **C**ompact model and a simple network architecture for dataset 3. A deep analysis of the results is carried out in the next section.

5 Discussion

A starting point for our discussion section is the definition of the baseline as a reference point for the obtained results. As it has been described in the data section 4.2, the data collection used in our experimentation is part of a competition for automated grading of student-written responses (ASAP) [7].

Unfortunately, the information available on the competition portal only mentions the winners of the competition but no methodology implemented or obtained results are provided.

But taking into account that our purpose is to perceive the quality of the embeddings produced by different training methods rather than obtain high precision on the downstream task, we define the original BERT model as the baseline model.

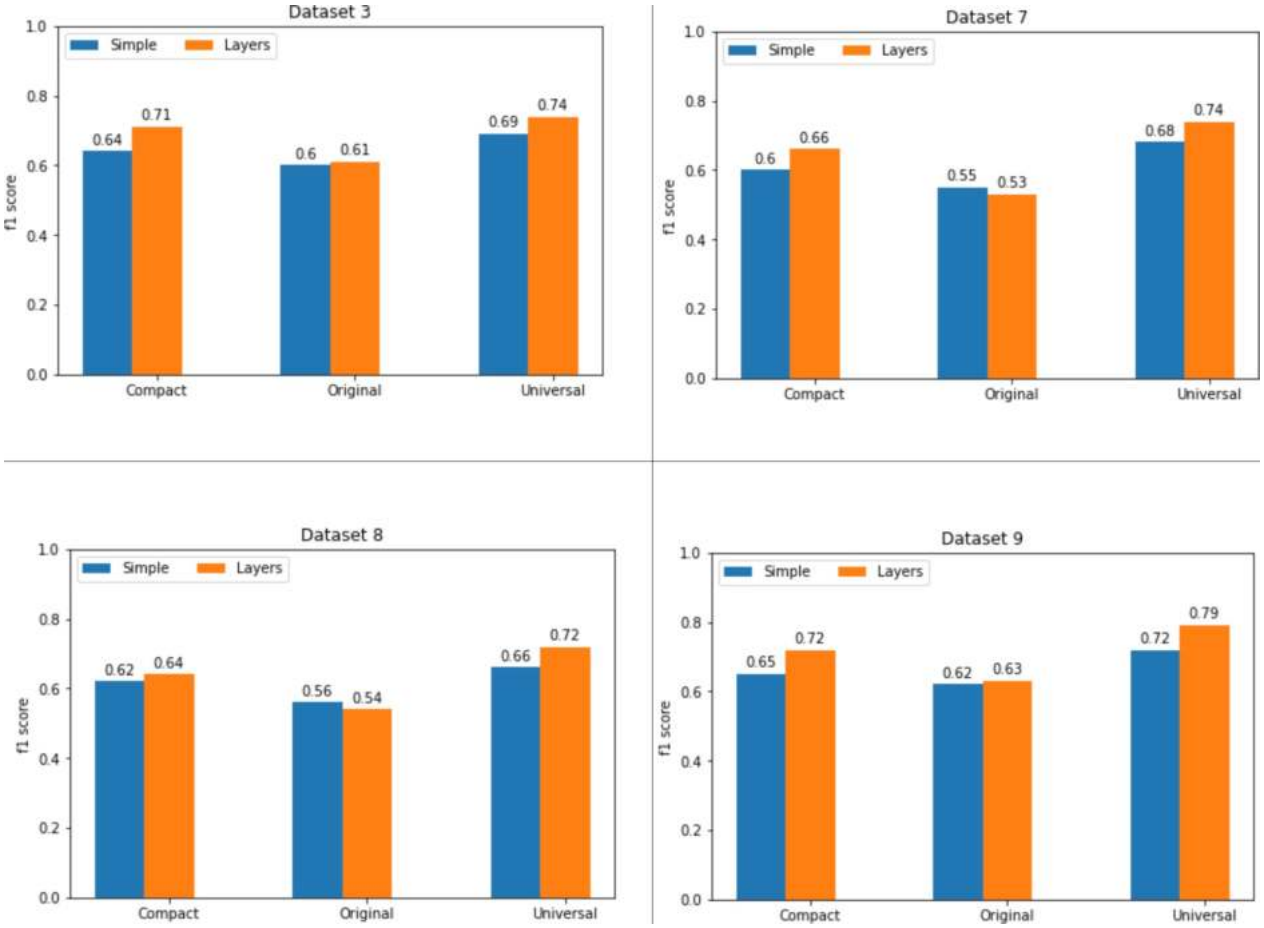

Fig. 8. Results corresponding to each BERT model for each network architecture and each dataset

For the sake of clarity, Figure 8 shows a graphic perspective on the obtained results from Table 2.

5.1 Pre-Trained BERT Models: Unsupervised Training Methods

The research question that guides our work is: what is the impact of the unsupervised training methods on the quality of the semantic representations produced by the pre-trained BERT models? Based on the experimental results, Figure 8 shows how the baseline performance differs from the Compact and Universal models: we can see how the performance of these extended models exceeds that of the original model.

In other words, the obtained results exhibit how the unsupervised training variants contribute to a positive effect on the performance. For example, the highest F1 score obtained for all datasets by the Universal model underpins its argument about the optimization of sentence-level representations.

In fact, the integration of the NSP and MLM training methods, where the MLM task depends on the sentence level representation produced by the NSP task, entails a sort of tradeoff: to perform good MLM, good sentence representations are required.

As we described in section 3.2.3, the CMLM training method of the Universal model makes use of adjacent sentences where the concatenation of the token embeddings of s_2 with the embeddings

of the first sentence s_1, are provided to the transformer encoder for the prediction of the masked tokens in s_2. In this way, this training method proves to be the best option to learn and produce sentence level representations.

As for the results obtained by the Compact model, the use of knowledge distillation as training method proves to be a plausible option for transfer learned knowledge to particular tasks.

The F1 score obtained by this Compact model for all datasets have surpassed the corresponding scores obtained by the baseline model. Thus, these results underpins its argument about the successful development of compact models under the self-supervised pre-training paradigm.

In section 3.2.4, we describe how the pre-trained distillation method of the Compact model defines three training operations: initialize a small model (i.e. the student) by pre-training under the MLM task, transfer learned knowledge (i.e. distillation of the teacher knowledge) and the optional fine-tuning on a particular linguistic task such as classification.

In this way, compared to the use of compression techniques on large language models [20, 19], this distillation training method proves to be a well-performing model developed under the self-supervised pre-training paradigm.

5.2 Pre-Trained BERT Models: Fine-Tuning Model Architectures

As suggested by Goodfellow et al. [6], a good representation is one that makes a subsequent learning task easier.

This is the reason why, in order to know about the strengths and weaknesses of the semantic representations extracted from the pre-trained BERT models studied in this work, we implement the fine-tuning process on a downstream classification task.

In other words, we transfer the acquired knowledge obtained from the pre-trained BERT models to solve automated grading of student written responses. Then, we need to figure out which of these representations demand further training to cope with this classification task.

This is the reason why we implement two fine-tuning model architectures: a simple and a forward neural network named in this work as layers. As we described in previous section 4.3, a simple architecture is just a softmax layer whereas our layers architecture is defined in terms of a small forward neural network to determine whether tuning is worth implementing.

Figure 8 highlights important points to be noticed. First, we see how the tuning of the embeddings produced by the Universal and Compact models has been worth of implementing. For all the observed datasets, the F1 score obtained by the use of the layers architecture is higher than the score obtained by the simple architecture. An average increase of 6 points in the F1 score is observed.

On the other hand, we see how the tuning of the embeddings produced by the Original BERT model has not been worth of implementing. For datasets 3 and 9, the F1 score obtained by the use of the layers architecture is a bit higher than the score obtained by the simple architecture (just one point is the difference).

However, for datasets 7 and 8, the F1 score obtained by the use of the layers architecture is lower than the score obtained by the simple architecture. Thus, two points stand out with the use of the semantic representations produced by the Original BERT model: for all the observed datasets, the lowest F1 score has been obtained, and the tuning of the embeddings has not been worth of implementing.

In summary, the embeddings produced by the extended BERT models, Universal and Compact models, have optimized the downstream task. On the other hand, regardless of the fine-tuning learning model implemented, simple or layers architectures, the F1 score obtained with the Original BERT model was lower than the one obtained with the pre-trained BERT variants.

The use of complex downstream network architectures such as CNN or Bi-LSTM could possibly improve the performance of the Original BERT model, but two previous works do not consider this option as a plausible alternative. Zhao et al., in their work about the use of pre-trained LLMs for toxic comment classification,

prove that using a basic linear downstream architecture outperforms complex ones such as CNN or Bi-LSTM [29]. Also, in their work about the analysis of multiple embeddings methods for text classification, Wang et al. implement CNN and Bi-LSTM as downstream network architectures and the difference in performance was not significant [26]. For the authors, the difference in performance lies in the characteristics of the data rather than the network architectures of the learning model.

5.3 Pre-Trained BERT Models: Computational Resources

What is the computational cost demanded by the pre-trained BERT models? Since determining the runtime and memory requirement of the pre-trained BERT models is highly platform-dependent, we do not describe the computational cost in absolute terms.

We describe rather the computational cost as a degree of runtime. In order to make a viable explanation for the computational cost incurred by each BERT model, we define a baseline as a reference point for the running time demanded by each model in the fine-tuning process.

So, taking into account the longest running time demanded, we define the Universal BERT model as the baseline model. Since the different downstream network architectures (simple or layers architectures) do not show any discrepancy in terms of the time consumed, we attribute the difference in time to the structure and training of each particular BERT model.

For example, the use of the Universal model gives rise to a tradeoff between classification performance and time: the Universal model demands more time but obtains the best F1 score for all datasets.

As we describe in section 3.3, this Large Language Model is based on the BERT transformer architecture that consists of L=12 encoder layers, a hidden size of $H = 768$, and $A = 12$ attention heads representing a total of 110M parameters. By contrast, the running time demanded by the Compact model is really amazing: this model requires only a third of the time required by the Universal model. And the classification performance is also good: this model achieves better F1 score than the Original model.

As we describe in section 3.3, this Small Language Model is based on a knowledge distillation architecture that consists of $L = 4$ encoder layers, a hidden size of $H = 512$, and $A = 8$ attention heads representing a total of 28M parameters.

In summary, and based on the evidence provided by our experimentation, we conclude this discussion section by considering the Compact model as a plausible alternative when the classification task can tolerate slight faults. Otherwise, and despite the running time demanded, the Universal model is the best option.

6 Related Work

Based on the taxonomy proposed by Qiu et al. for a deep examination of pre-trained language models for NLP [17], we focus our attention in this section on the type of pre-training tasks. More specifically, and given that in this work we address the analysis of three pre-trained BERT models and their corresponding pre-training tasks such as MLM, NSP and Distillation, in this section we make a brief description of pre-training tasks related to those previously mentioned. For example, we start with Dynamic MLM as it is a pre-training task closely related to MLM.

6.1 Dynamic MLM

This pre-training task is implemented in the development of a variant of BERT known as RoBERTa [30]. The purpose of this pre-training method is the optimization of the static masking implemented by MLM in which unique and different maskings are generated for each sequence, so each sequence with the same masking is observed more than once.

Instead, Dynamic MLM generates a unique masking every time a sequences is transferred to BERT training. In this way, a wide diversity of masking patterns is available for the training of BERT. Besides this training method optimization, the training of RoBERTa was implemented with

bigger batches, longer sentences and the use of NSP was omitted. In this way, RoBERTa performance achieves state-of-the-art results on a benchmark such as GLUE.

6.2 SOP: Sentence Order Prediction

This pre-training task is implemented in the development of a variant of BERT known as ALBERT [12]. As a sequel to BERT breakthrough, some studies on BERT development suggest the use of next sentence prediction (NSP) as an ineffective training method. In the development of ALBERT, SOP is then introduced to replace NSP. In order to take care of inter-sentence modeling, SOP focuses on coherence between pairs of sentences in a different way to NSP.

Instead of using sentence pairs from different documents as negative examples, SOP makes use of the same two consecutive sentences, used as positive examples in BERT, but with their order swapped. In addition to this new training method, ALBERT implements two parameter reduction techniques to cope with the huge computational resources demanded by BERT.

First, the separation of the hidden layers from the vocabulary embedding to increase the hidden layers without increasing the size of the vocabulary embedding. Second, to share all parameters across layers as a way to improve parameter efficiency. In this way, ALBERT performance achieves state-of-the-art results on a benchmark such as GLUE.

6.3 Transformer Distillation

This pre-training method implements a distillation knowledge technique to reduce the computational overhead of BERT while retaining its performance. A variant of BERT known as TinyBERT is the language model obtained by implementing this transformer distillation technique [9]. Transformer distillation performs layer-to-layer distillation with embedding outputs, hidden states and self-attention distributions. Basically, layer-to-layer distillation consists in choosing M out of N layers from the teacher model where a mapping function is defined for transfer learning from a particular layer of student model to a particular layer of a teacher model. The development of TinyBERT consists of two learning stages: general distillation and task-specific distillation. General distillation makes use of the pre-trained BERT as the teacher to train a smaller student called general TinyBERT with only 4 hidden layers instead of the standard 12.

Because of this significant reduction in the number of hidden layers, general TinyBERT performance is lower than BERT. Now, the purpose of the task-specific distillation is to strengthen the power of TinyBERT by applying again transformer distillation but now having as teacher the knowledge of fine-tuned BERT.

This process makes use of a data augmentation method on a task dataset in order to expand the task-specific training dataset. In this way, TinyBERT performance achieves state-of-the-art results on a benchmark such as GLUE.

7 Conclusion and Future Work

In this paper, we analyze the influence of unsupervised training methods on the development of pre-trained language models for learning linguistic representations. In particular, we study three pre-trained BERT models and their corresponding unsupervised training tasks such as MLM, NSP, CMLM and Distillation.

A broad outline of the pre-training process for each BERT variant allows to consider similarity and differences between them. We conduct fine-tuning as an empirical evaluation on a downstream classification task with a learning model defined in terms of the semantic representations produced by each BERT model. In this way, our experimentation provides empirical evidence of the quality of the embeddings produced by these pre-trained language models.

For example, the results show how the tuning of the embeddings produced by the Universal and Compact models has been worth of implementing as the F1 score obtained by the use of the layers architecture is higher than the score obtained by the simple architecture whereas the tuning of the embeddings produced by the Original BERT model has not been worth of implementing.

Finally, we obtain insight into the computational resources demanded by the BERT models analyzed in this work. The efficiency of the Compact model was rather astonishing. Based on the work about the identification of linguistic properties of data for which contextual embeddings contribute with a significant improvement on performance [1], our future work will explore the linguistic properties of data for which pre-trained models improve performance during downstream task. Said in another way, we will identify linguistic properties of data for which pre-trained models will exhibit the strengths and weaknesses of their corresponding unsupervised training methods.

References

1. **Arora, S., May, A., Zhang, J., Ré, C. (2020).** Contextual embeddings: When are they worth it? Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 2650–2663. DOI: 10.18653/v1/2020.acl-main.236.
2. **Balestriero, R., Ibrahim, M., Sobal, V., Morcos, A., Shekhar, S., Goldstein, T., Bordes, F., Bardes, A., Mialon, G., Tian, Y., Schwarzschild, A., Wilson, A. G., Geiping, J., Garrido, Q., Fernandez, P., Bar, A., Pirsiavash, H., LeCun, Y., Goldblum, M. (2023).** A cookbook of self-supervised learning. DOI: 10.48550/ARXIV.2304.12210.
3. **Barbara, Hamner, B., Morgan, J., lynnvandev, L., Shermis, M. (2012).** The Hewlett foundation: Short answer scoring.
4. **Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019).** BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 4171–4186.
5. **Devlin, J., Chang, M. W., Lee, K., Toutanova, K. (2019).** bert_en_uncased.
6. **Goodfellow, I., Bengio, Y., Courville, A. (2016).** Deep learning. MIT Press.
7. **Hamner, B., Morgan, J., lynnvandev, L., Shermis, M., Vander-Ark, T. (2012).** The Hewlett foundation: Automated essay scoring.
8. **Hinton, G., Vinyals, O., Dean, J. (2015).** Distilling the knowledge in a neural network. DOI: 10.48550/ARXIV.1503.02531.
9. **Jiao, X., Yin, Y., Shang, L., Jiang, X., Chen, X., Li, L., Wang, F., Liu, Q. (2020).** TinyBERT: Distilling BERT for natural language understanding. Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing, pp. 4163–4174. DOI: 10.48550/ARXIV.1909.10351.
10. **Jurafsky, D., Martin, J. H. (2023).** Speech and language processing.
11. **Kiros, R., Zhu, Y., Salakhutdinov, R., Zemel, R. S., Torralba, A., Urtasun, R., Fidler, S. (2015).** Skip-thought vectors. Proceedings of the 28th International Conference on Neural Information Processing Systems, Vol. 2, pp. 3294–3302. DOI: 10.48550/ARXIV.1506.06726.
12. **Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R. (2019).** ALBERT: A lite BERT for self-supervised learning of language representations. Proceedings of the 8th International Conference on Learning Representations, pp. 1–17.
13. **Logeswaran, L., Lee, H. (2018).** An efficient framework for learning sentence representations. Proceedings of the 6th International Conference on Learning Representations, pp. 1–16. DOI: 10.48550/ARXIV.1803.02893.
14. **Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J. (2013).** Distributed representations of words and phrases and their compositionality. Proceedings of the 26th International Conference on Neural Information Processing Systems, Vol. 2, pp. 3111–3119. DOI: 10.48550/ARXIV.1310.4546.
15. **Pennington, J., Socher, R., Manning, C. (2014).** GloVe: Global vectors for

word representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543. DOI: 10.3115/v1/d14-1162.

16. **Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L. (2018).** Deep contextualized word representations. Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Vol. 1, pp. 2227–2237. DOI: 10.48550/ARXIV.1802.05365.
17. **Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X. (2020).** Pre-trained models for natural language processing: A survey. Science China Technological Sciences, Vol. 63, No. 10, pp. 1872–1897. DOI: 10.1007/s11431-020-1647-3.
18. **Rani, V., Nabi, S. T., Kumar, M., Mittal, A., Kumar, K. (2023).** Self-supervised learning: A succinct review. Archives of Computational Methods in Engineering, Vol. 30, No. 4, pp. 2761–2775. DOI: 10.1007/s11831-023-09884-2.
19. **Sanh, V. (2019).** Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT.
20. **Sun, S., Cheng, Y., Gan, Z., Liu, J. (2019).** Patient knowledge distillation for BERT model compression. DOI: 10.48550/ARXIV.1908.09355.
21. **Taylor, W. L. (1953).** Cloze procedure: A new tool for measuring readability. Journalism and Mass Communication Quarterly, Vol. 30, No. 4, pp. 415–433. DOI: 10.1177/107769905303000401.
22. **TensorFlow (2023).** Tensorflow hub.
23. **Turc, I., Chang, M. W., Lee, K., Toutanova, K. (2019).** small_bert/bert_en_uncased.
24. **Turc, I., Chang, M. W., Lee, K., Toutanova, K. (2019).** Well-read students learn better: On the importance of pre-training compact models. DOI: 10.48550/ARXIV.1908.08962.
25. **Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017).** Attention is all you need. Neural Information Processing Systems 17: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 6000–6010. DOI: 10.48550/ARXIV.1706.03762.
26. **Wang, C., Nulty, P., Lillis, D. (2020).** A comparative study on word embeddings in deep learning for text classification. Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, pp. 37–46. DOI: 10.1145/3443279.3443304.
27. **Yang, Z., Yang, Y., Cer, D., Law, J., Darve, E. (2021).** universal-sentence-encoder-cmlm.
28. **Yang, Z., Yang, Y., Cer, D., Law, J., Darve, E. (2021).** Universal sentence representation learning with conditional masked language model. Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 6216–6228. DOI: 10.18653/v1/2021.emnlp-main.502.
29. **Zhao, Z., Zhang, Z., Hopfgartner, F. (2021).** A comparative study of using pre-trained language models for toxic comment classification. Companion Proceedings of the Web Conference, pp. 500–507. DOI: 10.1145/3442442.3452313.
30. **Zhuang, L., Wayne, L., Ya, S., Jun, Z. (2021).** A robustly optimized BERT pre-training approach with post-training. Proceedings of the 20th Chinese National Conference on Computational Linguistics, pp. 1218–1227.

Article received on 14/10/2023; accepted on 14/12/2023.
** Corresponding author is Diego Uribe.*