ISSN 1870-4069

Transfer Learning: Shallow and Deep Neural Models

Diego Uribe, Enrique Cuan, Elisa Urquizo

Tecnológico Nacional de México,
Instituto Tecnológico de La Laguna,
Departamento de Sistemas y Computación,
Mexico

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

Abstract. Since developing general-purpose large language models to handle a variety of tasks can be expensive for many businesses, transfer learning plays a crucial role in many natural language tasks nowadays. The pre-trained and fine-tuning framework for transferring and tuning the generic knowledge produced by a large language model (LLM) is a common ground for many natural language tasks. In this work, we focus our attention on how much tuning of the semantic representations (i.e. generic knowledge) obtained from BERT is required to perform downstream language tasks. We analyze the impact of some deep learning models, such as recurrent neural networks, on the fine-tuning process for a downstream classification task. To consider similarities and differences between the deep learning models and the corresponding optimization of the classification task, a dataset of four different categories of short answer responses is used in our empirical experimentation. In this way, we enrich the comparison of the tuning required to optimize the semantic representations obtained from a pre-trained BERT model.

Keywords: Transfer learning, pre-trained language model, BERT.

1 Introduction

In this paper we explore whether there is a meaningful contrast in the use of different deep learning models when transferring the generic knowledge produced by a pre-trained language model. In the context of natural language processing, learning textual representations, the generic knowledge to be transferred, plays a crucial role in multiple language tasks like question answering or classification or natural language generation. These textual representations are learned from unsupervised learning methods and transferred to a supervised learning task. Said in another way, transfer learning occurs when generic representations have been learned to make a subsequent learning task easier [5]. And it is precisely that the pre-trained and fine-tuned framework has emerged as a powerful technique to facilitate transfer learning. Building a pre-trained language model relies on self-supervised learning (SSL), a type of unsupervised learning.

By making use of large amounts of unlabeled text data, SSL learns generic representations useful across many linguistic tasks [2]. The most crucial aspect in the

development of a pre-trained model is the definition of the unsupervised learning tasks such as masked language modeling (MLM) [18], next sentence prediction (NSP) [12], sentence order prediction (SOP) [9], etc. These pre-training tasks make the difference between the large number of pre-trained models available to be used across a range of downstream language tasks [22]. In this work we focus our attention on the bidirectional transformer encoder known as BERT: Bidirectional Encoder Representations from Transformers [3], a pre-trained language model based on a bidirectional transformer encoder which is characterized by a bidirectional self-attention mechanism to produce contextual embeddings. More precisely, among the wide range of available variants of BERT, in this work we make use of the compact BERT model [20, 19].

Since a large language model such as the original BERT has a high computational cost, the compact BERT 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. 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.

Briefly, this distillation technique consists of a student-teacher training method where the teacher transfers knowledge to the student through its predictions for unlabeled training examples. A deep description is given in section 3. As we initially said, we explore in this work whether there is a meaningful contrast in the use of different deep learning models when taking the generic knowledge produced by a pre-trained language model. In other words, we explore the impact of taking and further tuning the linguistic representations obtained from the compact BERT model via deep learning models such as simple recurrent networks (SimpleRNN), long short term memory networks (LSTM) and Bi-directional networks.

In fact, we perform a downstream task as classification by implementing a classifier learning model with each variant of recurrent neural networks for tuning the obtained representations 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. Each instance in the collection denotes a short answer corresponding to a particular story of a specific domain where the range of the score is three: 0, 1, or 2. 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, the primary contributions of our work are summarized as follows:

– Our work provides insights about the impact of deep transfer learning with recurrent neural networks. The transfer learning for each variant of recurrent neural networks is described to consider similarities and differences between them.
– We conduct an empirical evaluation on the use of recurrent neural networks for transfer learning. The fine tuning process is implemented on a downstream classification task with a deep learning model defined in terms of the semantic representations produced by the compact BERT model.
– To enrich the experimentation, we try two different configurations for each variant of RNNs.

2 Related Work

In this section we briefly describe some interesting works about transfer learning and pre-trained language models. Developing a pre-trained language model to produce general-purpose knowledge leads to developing modern techniques for transfer learning in NLP. Azunre has written a fully comprehensive guide about transfer learning techniques for the customization of pre-trained language models.

How to use transfer learning to reproduce state-of-the-art results for downstream tasks, especially when limited resources, such as the availability of label data, are a common scenario in many language understanding tasks [1]. Another important work about the relevance of transfer learning is displayed by Raffel et al. [14]. Given that transferring general knowledge produced by a pre-trained language model is nowadays a common practice in many language tasks, transfer learning emerges as an important research topic. The purpose of this work is to propose a text-to-text framework for the systematic study of the multiple affairs that transfer learning entails:

– Pre-training objectives,

– Unlabeled datasets,

– Benchmarks,

– Fine-tuning methods.

In other words, this work proposes a unified approach to compare the effectiveness of various transfer learning objectives and fine-tuning methods, as well the use of unlabeled datasets. In summary, the purpose is to provide insights about transfer learning from a systematic framework to determine where the field stands.

An excellent paper about the impact of pretrained language models in NLP is the work developed by Qiu et al. [13]. The major contribution of this work is a comprehensive and exhaustive review of pretrained models for NLP. For a better description of the pretrained models, the authors built a taxonomy which categorize existing pretrained models from four different perspectives:

– **Representation Type**: Contextual and non-contextual models for downstream tasks.

– **Architectures**: The network structure and its components such as Transformer encoder and decoder.

– **Task Types**: Type of pre-training tasks.

– **Extensions**: Design of pretrained models for diverse scenarios.

The problem of adapting the general language knowledge to downstream tasks is also contemplated from the perspective of transfer learning and fine-tuning strategies. There have been numerous works on improving BERT such as RoBERTa [11]. This encoder is more robust than BERT, and is trained using much more training data.

ALBERT [9] is another work that lowers the memory consumption and increases the training speed of BERT. These variants have also been fine-tuned for various NLP tasks. Lin et al. used self-attention to extract interpretable sentence embeddings [10]. They use

a 2-D matrix to represent the embedding, with each row of the matrix attending on a different part of the sentence. In this way, different aspects of the sentence are extracted into multiple-vector representation. Experimental results over 3 different tasks show that the model outperforms other sentence embedding models by a significant margin.

3 BERT: The Pre-trained Language Model

Here we explain the pre-trained language model used in our research work, a variant of BERT: the compact BERT model. In fact, a fair description of knowledge distillation, the pre-training task used in the development of this small language model, is given in this section. 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 [3]. In order to generate context-based embeddings, the attention mechanism of the transformer plays a crucial role in the encoding process.

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 [21]. 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. In fact, 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. 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:

$$q_i = W^{Qx_i}; \qquad k_i = W^{Kx_i}; \qquad v_i = W^{Vx_i}. \tag{1}$$

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 Compact Model

The basic idea of the **C**ompact BERT model is to start with an initial model trained on MLM (the student) to eventually improve its performance by knowledge distillation from a large language model (the teacher) [20]. This model consists of $L = 4$ encoder layers, a hidden size of $H = 512$, and $A = 8$ attention heads representing a total of 28M parameters[1]. Since this small language model is based on knowledge distillation, a fair description of this pre-training task is given next. Building a compact model revolves around knowledge distillation: the standard technique for model compression [7]. 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. Fig. 1 shows the knowledge distillation process incorporated in the development and implementation of a compact BERT model [20]. 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), ..., (x_N , y_N), where x_i is an input and y_i is a label. Unlabeled training data (D_T): a set of M input examples $x_1^{'}$, ..., $x_M^{'}$ obtained from a distribution not necessarily identical to the distribution of

[1] _bert/bert_en_uncased_L-4_H-512_A-8

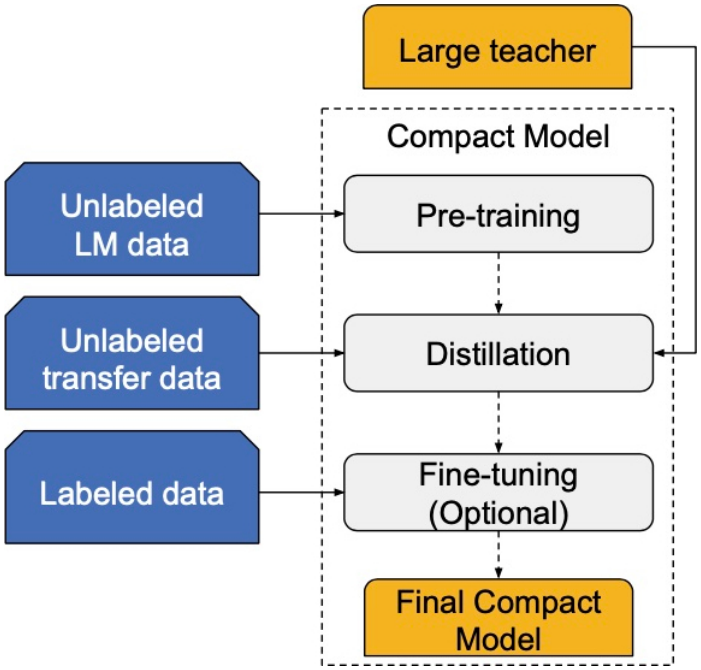

Fig. 1. Knowledge Distillation process. This figure corresponds to [20].

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 the kernel of the compact model is defined for a sequence of three training operations: Pre-training on D_{LM}: pre-training of the compact model with MLM as training method. 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. The estimation of the cross-entropy loss between teacher and student predictions is then used to update the student model.

Fine-tuning on D_L: this operation is an 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 [17], DistillBert makes use of a more expensive LM teacher as distillation is performed on general-domain data [15].

4 Experimental Evaluation

The experimentation conducted in this work is the description of deep transfer learning for a particular text-processing task with recurrent neural networks architectures. So, we first explain the fine-tuning process of the pre-trained Compact model previously mentioned to perform a downstream task as sequence classification. As a fundamental part of the tuning process, the downstream network architectures implemented are also detailed. Then, the dataset characteristics are exposed and the results of each recurrent neural architectures are exhibited.

4.1 Fine-tuning

The process to make use of the generalizations produced by the pretrained language models is known as fine-tuning. These generalizations are helpful to build a sort of pipeline applications to cope with particular NLP tasks such as sequence classification or named entity tagging. In our classification experiments, we adopt two strategies to the downstream task: the use of the embeddings obtained from the pre-trained model as the input to a classic neural network, and a more refined optimization of such embeddings via deep recurrent neural networks.

So, how to obtain the embeddings from the compact BERT model? The BERT models return a map with three important keys: pooled_output, sequence_output and encoder_outputs. For the neural network architectures implemented in our tuning process, the first two keys are of interest to us:

- **pooled_output**: represents each input sequence as a whole so the embedding denotes the entire sequence. The shape is [batch_size, H].
- **sequence_output**: represents each input token in the sequence so in this case we have a contextual embedding for every token. The shape is [batch_size, seq_length, H].

In the case of pooled_output, Fig. 2 shows how there is an additional vector symbolized by the **[CLS]** token which is prepended to the input sequences. This additional vector [CLS] captures the entire sequence so it is provided to a classic 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) in order to map the output vector (Y_{CLS}) to a set of categories:

$$y = \text{softmax}(WY_{\text{CLS}}). \tag{5}$$

On the other hand, the fine-tuning process with deep recurrent neural networks requires the use of sequence_output. Recurrent neural networks represent the temporal nature of language as each element of a sequence (each token in Fig. 2) is processed at a time [4].

The key point in this model is the computation of the hidden layer as the last input element denotes the entire sequence. To activate the current hidden layer is necessary the value obtained in the previous hidden layer corresponding to a preceding point in time. Equation (6) expresses the computation of the hidden layer **h** where **x** denotes

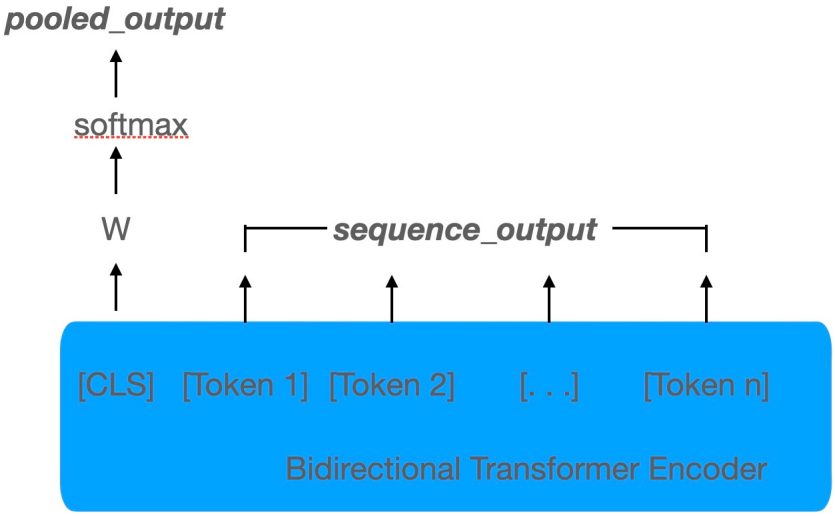

Fig. 2. BERT's output: pooled_output and sequence_output.

the sequence (i.e. input) and **g** an activation function. **W** denotes the weight matrix corresponding to the input x_t whereas **U** denotes the weight matrix corresponding to the hidden layer of the previous timestep h_{t-1}. In this way, this connectionist model is concerned with the context corresponding to each element of the sequence:

$$h_t = f(Uh_{t-1} + Wx_t). \tag{6}$$

4.2 Data

The dataset used in this experimentation is part of an ambitious research project denominated the Automated Student Assessment Prize (ASAP) [6] 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. 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: **0** (not proficient), **1** (partially proficient), or **2** (proficient).

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

4.3 Results

As we previously said, we adopt ,in our classification experiments, two strategies to the downstream task: the use of the embeddings as the input to a classic neural network, and a more refined optimization of such embeddings via deep recurrent neural networks. Thus, the downstream network architectures implemented are:

- **Classic**: 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 compact BERT model is 512, and our experimentation performs a downstream three-class classification, the number of parameters to be adjusted is 35,011.
- **SimpleRNN**: as traditional neural networks are unable to represent the temporal nature of language [4], the embeddings are transferred step by step. A simple dense layer is used to adjust the pre-trained embeddings obtained from sequence_output. For example, since the number of hidden units of the compact BERT model is 512, a SimpleRNN layer with dimensionality of 50 (number of units), and a downstream three-class classification task, the number of parameters to be adjusted is 28,303.
- **LSTM**: as SimpleRNN cannot keep track of long-term dependencies, we try LSTM networks to include the consideration of distant constituents [8]. A simple dense layer is also used to adjust the pre-trained embeddings obtained from sequence_output. In this case, as the number of hidden units of the compact BERT model is 512, a LSTM layer with dimensionality of 50 (number of units), and a downstream three-class classification task, the number of parameters to be adjusted is 112,753.
- **Bidirectional**: as in many cases we need information from the context to the right of the current token, we try Bidirectional networks to include the consideration of the constituents from the start to the end of the input sequence and vice versa [16]. A simple dense layer is also used to adjust the pre-trained embeddings obtained from sequence_output. In this case, as the number of hidden units of the compact BERT model is 512, a Bidirectional network having LSTM layers with dimensionality of 50 (number of units), and a downstream three-class classification task, the number of parameters to be adjusted is 225,503.

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.

Table 1. ASAP-RNNs-50-units.

Architecture	Dataset 3	Dataset 7	Dataset 8	Dataset 9
Classic	0.71	0.66	0.64	0.72
SimpleRNN	0.79	0.79	0.78	0.83
LSTM	**0.84**	**0.84**	**0.82**	**0.86**
Bidirectional	0.84	0.84	0.82	0.86

Table 2. ASAP-RNNs-100-units.

Architecture	Dataset 3	Dataset 7	Dataset 8	Dataset 9
Classic	0.71	0.66	0.64	0.72
SimpleRNN	0.78	0.76	0.73	0.80
LSTM	**0.84**	**0.84**	**0.81**	**0.87**
Bidirectional	0.84	0.84	0.80	0.84

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. Tables 1 and 2 show the results of the transfer of knowledge, obtained from the compact BERT model to deep recurrent neural networks. Table 1 shows the results for deep network architectures with a dimensionality of 50 units, whereas the results with a dimensionality of 100 units are exhibited in Table 2. The results are expressed in terms of the F1 score corresponding to each network architecture and each domain. For example, the second row of the Table 1 shows a F1 score of 0.79 obtained with a SimpleRNN 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, the data collection used in our experimentation is part of a competition for automated grading of student-written responses (ASAP) [6]. Unfortunately, the information available on the competition portal only mentions the winners of the competition but no methodology implemented or obtained results are provided.

However, taking into account that our purpose is to perceive the impact of deep transfer learning with recurrent neural networks, we define the classic neural network model as the baseline model. For the sake of clarity, Fig. 3 and Fig. 4 show a graphic perspective of the obtained results corresponding to the Tables 1 and 2 respectively.

Recurrent Neural Networks. Based on these figures, it is possible to observe whether there is a meaningful contrast in the use of different recurrent neural networks when transferring the generic knowledge produced by a pre-trained language model such as the compact BERT model.

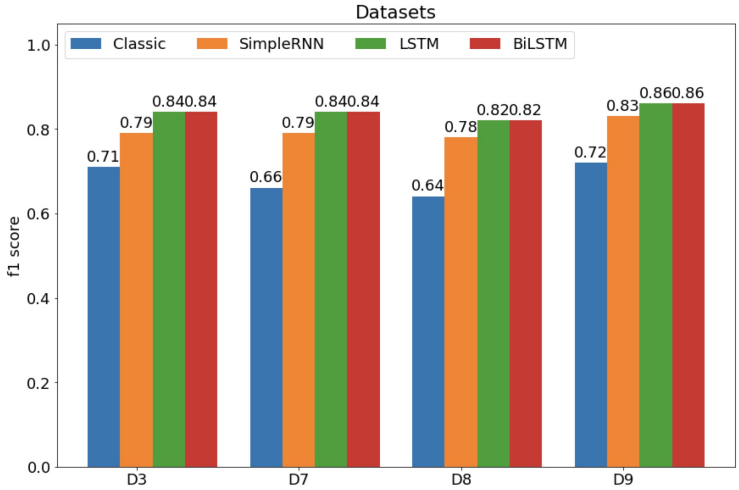

Fig. 3. ASAP-RNNs-50-units.

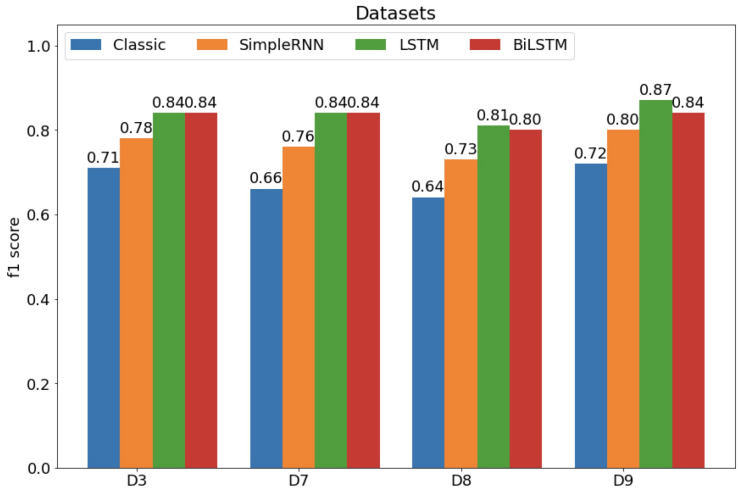

Fig. 4. ASAP-RNNs-100-units.

First of all, we see how the baseline performance denoted by a classic neural network model has been surpassed by the recurrent neural networks. A significant increase in the performance can be observed for all datasets. Now, the use of diverse recurrent neural networks exhibits important points to be noticed. First, we see how the transfer of the embeddings produced by the Compact model has been worth of implementing with recurrent neural networks. For all the observed datasets in Fig. 3 or Table 1, the F1 score obtained by the use of the SimpleRNN architecture is higher than the score obtained by the classic architecture. An average increase of 11 points in the F1 score is observed.

Fig. 3 or Table 1 also show how the highest performance has been obtained with the use of the LSTM architecture. For all the observed datasets, the transfer of knowledge with this recurrent architecture proved to be the best option. On the other hand, we see how the transfer of the embeddings has not been worth of implementing with a Bidirectional architecture such as BiLSTM (see Fig. 3 and Fig. 4). At its best score, this Bidirectional architecture achieves the same performance as the one obtained with the LSTM architecture. We attribute this result to the size of the texts: a short-answer text does not seem to demand information from the context to the right of the current token being analyzed.

Dimensionality. We have presented our results in two figures corresponding to the implementation of diverse recurrent neural networks with two dimensions: 50 and 100 units. We configured in this way taking into account that the average length of each answer is approximately 50 words. Regardless of the dataset observed, the F1 score obtained with an architecture of 100 units is equal or lower than the one obtained with an architecture of 50 units. Thus, the transfer of knowledge with a recurrent neural architecture of 50 units proved to be the best option.

A side effect of dimensionality is the increase in the number of parameters to be adjusted.For example, the number of parameters to be trained for a SimpleRNN architecture of 50 units is 28,303, whereas the number of parameters to be trained for a SimpleRNN architecture of 100 units is 61,603. And of course, given that the complexity of the LSTM and BiLSTM architectures is greater than the complexity of a SimpleRNN architecture, the number of parameters to be considered is much greater.

6 Conclusions

In this paper, we have analyzed the impact of deep transfer learning with RNNs. To consider similarities and differences between diverse RNNs, the transfer learning for each variant is described in terms of the fine-tuning process implemented on a downstream classification task. Our experimentation based on the classification of short-answer texts, provides empirical evidence of how the tuning of the embeddings obtained from a compact BERT model is worth of implementing with RNNs. Compared with a classic neural network, a SimpleRNN architecture improves the results, whereas the best results were obtained with the LSTM architecture.

References

1. Azunre, P.: Transfer learning for natural language processing. Manning Publisher (2021)
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.: A cookbook of self-supervised learning (2023) doi: 10.48550/ARXIV.2304.12210
3. Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, vol. 1, pp. 4171–4186 (2019)

4. Elman, J. L.: Finding structure in time. Cognitive Science, vol. 14, no. 2, pp. 179–211 (1990) doi: 10.1016/0364-0213(90)90002-e
5. Goodfellow, I., Bengio, Y., Courville A.: Deep learning. The MIT Press (2016)
6. Hamner, B., Morgan, J., Lynnvandev., Shermis, M., Vander-Ark, T.: The Hewlett foundation: Automated essay scoring (2012) www.kaggle.com/competitions/asap-aes
7. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network (2015) doi: 10 .48550/arXiv.1503.02531
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation, vol. 9, no. 8, pp. 1735–1780 (1997) doi: 10.1162/neco.1997.9.8.1735
9. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: A lite BERT for self-supervised learning of language representations. In: International Conference on Learning Representations, pp. 1–17 (2020)
10. Lin, Z., Feng, M., Nogueira-dos-Santos, C., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding (2017) doi: 10.48550/arXiv.1703.03130
11. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettle-Moyer, L., Stoyanov, V.: Roberta: A robustly optimized BERT pretraining approach (2019) doi: 10 .48550/arXiv.1907.11692
12. Logeswaran, L., Lee, H.: An efficient framework for learning sentence representations (2018) doi: 10.48550/ARXIV.1803.02893
13. Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., Huang, X.: Pre-trained models for natural language processing: A survey (2020) doi: 10.48550/arXiv.2003.08271
14. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of Machine Learning Research. vol. 21, no. 1, pp. 1–67 (2020)
15. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Smaller, faster, cheaper, lighter: Introducing DistilBERT, a distilled version of BERT (2019) doi: 10.48550/arXiv.1910.01108
16. Schuster, M., Paliwal, K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing, vol. 45, no. 11, pp. 2673–2681 (1997) doi: 10.1109/78.650093
17. Sun, S., Cheng, Y., Gan, Z., Liu, J.: Patient knowledge distillation for BERT model compression (2019) doi 10.48550/arXiv.1908.09355
18. Taylor, W. L.: Cloze procedure: A new tool for measuring readability. Journalism Quarterly, vol. 30, no. 4, pp. 415–433 (1953) doi: 10.1177/107769905303000401
19. TensorFlow: BERT (2020) tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-4_H-512_A -8/2
20. Turc, J., Chang, M. W., Lee, K., Toutanova, K.: Well-read students learn better: On the importance of pre-training compact models. In: Proceedings of the International Conference on Learning Representations, pp. 1–13 (2019) doi: 10.48550/arXiv.1908.08962
21. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, pp. 1–11 (2017)
22. Wei, C., Xie, S. M., Ma, T.: Why do pretrained language models help in downstream tasks? An analysis of head and prompt tuning (2021) doi: 10.48550/ARXIV.2106.09226