Indonesian Journal of Electrical Engineering and Computer Science
Vol. 22, No. 2, May 2021, pp. 1032~1040
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v22.i2.pp1032-1040

A comparative study of deep learning based language representation learning models

Mohammed Boukabous, Mostafa Azizi
MATSI Research Lab, ESTO, Mohammed First University, Oujda, Morocco

Article Info

Article history:

Received Feb 13, 2021
Revised Mar 24, 2021
Accepted Apr 11, 2021

Keywords:

BERT
Deep learning
GPT-2
Natural language processing
Representation models
Sentiment analysis

ABSTRACT

Deep learning (DL) approaches use various processing layers to learn hierarchical representations of data. Recently, many methods and designs of natural language processing (NLP) models have shown significant development, especially in text mining and analysis. For learning vector-space representations of text, there are famous models like Word2vec, GloVe, and fastText. In fact, NLP took a big step forward when BERT and recently GTP-3 came out. In this paper, we highlight the most important language representation learning models in NLP and provide an insight of their evolution. We also summarize, compare and contrast these different models on sentiment analysis, and thus discuss their main strengths and limitations. Our obtained results show that BERT is the best language representation learning model.

Corresponding Author:

Mohammed Boukabous
MATSI Research Lab, ESTO
Mohammed First University
BP 473 complexe Universitaire Al Qods, Oujda 60000, Morocco
Email: m.boukabous@ump.ac.ma

1. INTRODUCTION

The field of natural language processing (NLP) aims to convert human language into a formal representation using a range of computational techniques to make it easy for computers to manipulate. NLP is rapidly advancing due to the growing interest in human-to-machine communications, the big amount of text data stored in the web, and the powerful computing systems and enhanced algorithms. Deep learning algorithms and architectures have made remarkable advances within the past few years in the field of text analytics. NLP market was valued by Mordorintelligence at 10.93 billion USD in 2019, and it is predicted to reach the worth of 34.80 billion USD by 2025 [1].

Dependently of the objectives, NLP could be processed in two ways: 1) natural language understanding (NLU) and (2) natural language generation (NLG). NLU involves mapping the input data in the natural language form into useful representations, and analyzing the multiple aspects of the natural language [2]. NLG is the process of generating meaningful sentences and phrases in a targeted natural language, that involves text planning (retrieving related content from the knowledge base), sentence planning (selecting required words, forming meaningful phrases, and setting tone of the sentence), and text realization (mapping sentences outlines into sentences structures) [3].

Deep learning algorithms are unable to deal with textual data in their natural language data form which is typically unstructured information; they require special representation of data as inputs instead. Usually, natural language text data needs to be converted into internal representations form that DL

algorithms can read such as feature vectors, hence the necessity to use representation learning models [4]. These models have shown a big leap during the last years. Their set ranges from the methods that embed words into distributed representations and use the language modeling objective to adjust them as model parameters (like Word2vec [5], fastText [6], and GloVe [7]), to recently transfer learning models (like ELMo [8], BERT [9], ULMFiT [10], XLNet [11], and GPT-2 [12]). These last use larger corpora, more parameters, more computing resources, and instead of assigning each word with a fixed vector, they use multilayer neural networks to calculate dynamic representations for the words according to their context, which is especially useful for the words with multiple meanings.

The rest of this paper is organized as follows: in the next section, we briefly introduce sentiment analysis, transfer learning, then the most important language representation learning techniques. In Section 3, we make a comparison between these techniques and discuss our findings.

2. BACKGROUND

2.1. Sentiment analysis

Sentiment analysis (SA) also known as emotion AI or opinion mining is the analysis of feelings from dematerialized textual sources on large amounts of data (big data), or from images [13]. There are three usual granularity levels for opinion mining [14]: the document, the sentence and the aspect levels [15]. These sentiments can express the author's opinion, his emotional state (when writing his text), or a deliberate sense of connection (that the author expects to make with readers). Sentiment analysis is widely used in security intelligence purposes to analyze and synthesize individual reactions to deduce trends and user needs [16]. Indeed, we have already overviewed in [16] learning-based techniques of sentiment analysis for security purposes.

2.2. Transfer learning

Deep learning models necessitates a lot of data and time while training [17], [18]. Transfer learning is a technique that benefits from an already trained weight on big datasets for a long period of time and transfer this knowledge [19] to the targeted model. For instance, the BERT model was trained for 4 days on 16 Cloud TPUs, and the GPT-3 model has 175 billion parameters [20]. The idea of retraining these models with new data is very expensive, both in terms of time and resources, so here implementing transfer learning is more practical than retraining.

2.3. Language representation learning models

One of the important tasks in NLP is the learning of vector representations of text, as deep learning algorithms require representing their input entries in a vector format. For this, we highlight the most important language representation learning models in NLP and we classified them into two categories: neural word embeddings and transfer learning techniques, then we compared them as shown in Table 1.

Table 1. Classification of NLP language representation learning models

		Context Direction		Downstream Model		Base Architecture					
		Unidirectional	Bidirectional	Task-dependent	Task-independent	Shallow neural network	2 layers bi-LSTM	3 layers LSTM	Transformer Encoder	Transformer Decoder	Transformer-XL
Neural Word Embeddings	Word2vec	✔	–	✔	–	✔	–	–	–	–	–
	fastText	✔	–	✔	–	✔	–	–	–	–	–
	Glove	✔	–	✔	–	✔	–	–	–	–	–
Transfer Learning Techniques	ELMo	–	✔	✔	–	–	✔	–	–	–	–
	BERT	–	✔	✔	–	–	–	–	✔	–	–
	RoBERTa	–	✔	✔	–	–	–	–	✔	–	–
	ALBERT	–	✔	✔	–	–	–	–	✔	–	–
	ULMFiT	✔	–	–	✔	–	–	✔	–	–	–
	XLNet	–	✔	✔	–	–	–	–	–	–	✔
	GPT-2	✔	–	–	✔	–	–	–	–	✔	–

2.3.1. Neural word embeddings

a) Word2vec is an unsupervised learning algorithm that consists of a group of related models used for word embeddings generation. It is based on three-layer neural networks and seeks to learn the vector representations of words composing a text, so that words that share similar contexts are represented by close digital vectors [21]. Word2Vec has two neural architectures, called continuous bag-of-words (CBOW) and Skip-Gram. CBOW receives as input the context of a word, i.e., the terms surrounding it in a sentence, and tries to predict the word in question. Skip-Gram does exactly the opposite: it takes a word as input and tries to predict its context [5].
b) fastText is a Facebook's AI library for efficient learning of sentences classification and word embeddings [6], [22]. It supports multiprocessing during training and allows to create an unsupervised or supervised learning algorithm to obtain vector representations of words and sentences. fastText uses a neural network for word embeddings and supports training continuous bag of words (CBOW) or skip-gram model. It can be used as an initializer for transfer learning.
c) Glove is an unsupervised learning algorithm to obtain word vector representations. This is accomplished by mapping words in a meaningful space where the distance between words is related to semantic resemblance. Training is performed using an underlying count-based model on the aggregated global word to word co-occurrence matrix within a text corpus, and the subsequent representations display interesting linear substructures in the word vector space. It combines the features of two sets of models, namely the local context window approaches and the global matrix factorization [7].

2.3.2. Transfer learning techniques

a) Embeddings from language models (ELMo) is a pre-trained biLSTM (bidirectional LSTM) language model. Word embeddings is calculated by taking a weighted score of the hidden states from each layer of the LSTM. Weights are learned with downstream model parameters for a particular task, but LSTM layers are kept constant [8]. Thus, the same word under different contexts can have different word vectors.
b) Bidirectional encoder representations from transformers (BERT) is another language representation learning model that uses an attention transformers mechanism to learn the contextual relations between words in a text instead of bidirectional LSTMs to encode context which shows that pre-training transformer networks on a masked language modeling objective leads to even better performance by precisely adjusting the transformer weights over a wide range of NLP tasks [9].
c) A robustly optimized BERT pretraining approach (RoBERTa) is an optimized model resulting from analysis of Google's BERT training model and the identification of several changes to the training procedure that enhance its performance by Facebook AI and the University of Washington researchers. Specifically, these researchers used a novel and bigger dataset for training, trained the model over far more epochs, and removed the next sequence prediction training objective [23].
d) A lite BERT for self-supervised learning of language representations (ALBERT) is a “Lite” version of BERT, this model architecture includes two parameter-reduction methods: cross-layer parameter sharing and factorized embeddings parameterization. Furthermore, the proposed method contains a self-supervised loss for the sentence-order prediction [24].
e) Universal language model fine-tuning for text classification (ULMFiT) is a transfer learning method that can be applied to NLP. It uses a regular 3-layer LSTM architecture for either pre-training and fine-tuning tasks. ULMFiT consists of three steps: general-domain language model (LM) pertaining (pertaining language model on a large general-domain corpus), target task LM fine-tuning (the LM fine-tuned on the data of the target task), and the target task classifier fine-tuning (fine-tuning the classifier) [10].
f) XLNet is a generalized autoregressive (AR) pertaining method that uses the context word to predict the next word which is constrained to a unidirectional context, either backward or forward. Although, XLNet learns from bidirectional context using permutation language modeling. It also influences the best of both AR language modeling and autoencoders while avoiding their limitations [11].
g) Generative pretrained transformer 2-successor of GPT (GPT-2) follows the OpenAI GPT model with a few architecture modifications. It consists of a big transformer-based language model with 1.5 billion parameters, trained with the objective of the prediction of the next word, given all previous words in a text. And unlike the previous models that require pre-training and fine-tuning, there is no fine-tuning step for GPT-2 [12].

3. BUILDING MODELS

First, dataset used in experiments was combined from CARER-Emotion, DailyDialog, CrowdFlower, and Isear to create a rich dataset with 5 labels: anger (5k sentences), joy (26k sentences), sad

(13k sentences), fear (3.6k sentences), and neutral (94k sentences). The used texts consist of tweets, dialog utterances, and short messages as shown in Table 2.

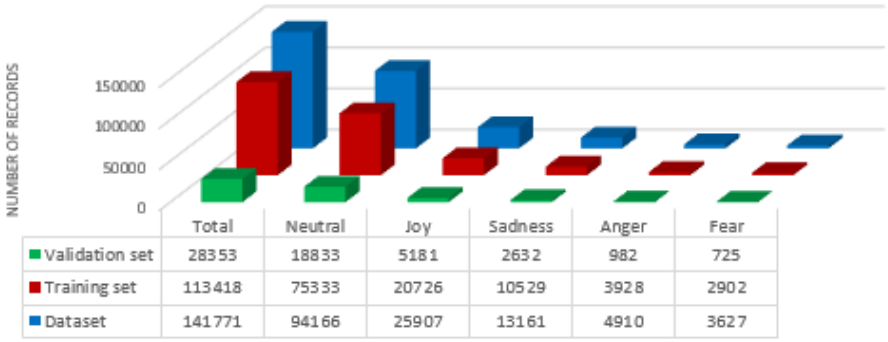
Table 2. Combined datasets for benchmarking

Dataset	Year	Content	Number of sentences	Emotion categories
CARER – Emotion [25]	2018	Tweets	20k	Anger, anticipation, disgust, fear, joy, sadness, surprise, and trust
DailyDialog [26]	2017	Dialogues	102k	Neutral, joy, surprise, sadness, anger, disgust, and fear
CrowdFlower [27]	2016	Tweets	40k	Empty, sadness, enthusiasm, neutral, worry, surprise, love, fun, hate, happiness, boredom, relief, anger
Isear [28]	1994	Emotion situations	7.5k	Joy, fear, anger, sadness, disgust, shame, guilt

3.1. Word2vec and glove

For these algorithms, we started by importing the dataset created previously, then input it into our neural network model, and we do some preprocessing and tokenization using NLTK to double check that sentences are properly split into words. We could also add Stopword removal, but steps like stemming or lemmatization are not needed since words with the same stem can have different meanings. Moreover, we split the data at first into Data X (contains text data) and Label Y (contains the emotions), and also to random training subset and validation subset with 80% for training set and 20% for the validation set. Figure 1 shows the number of data rows for each set and each emotion type. After that, we import pre-trained models (Word2vec: wiki-news-300d-1M, Glove: Stanford glove.twitter.27B) and create embeddings matrix to map each word in our corpus to the existing word vector, then we create our neural network pipeline.

a) The first level creates embeddings of words, using a vocabulary size (36866), a maximum length (300), and a size of embeddings (36867, 300).
b) SpatialDropout1D (0.2).
c) Bi-LSTM layer (128 units) which will receive word embeddings for each token as inputs.
d) Dropout (0.5).
e) A dense layer with a number of neurons equal to the classes of the problem (5 units), a “softmax” activation function for multi-class classification, and because of this, categorical_crossentropy is used as the loss function.
f) Finally, we train our model with a batch size of 32, and 20 epochs.



	Total	Neutral	Joy	Sadness	Anger	Fear
Validation set	28353	18833	5181	2632	982	725
Training set	113418	75333	20726	10529	3928	2902
Dataset	141771	94166	25907	13161	4910	3627

Figure 1. Emotions distribution in training and validation sets

3.2. fastText

We use the ktrain library [29] a lightweight wrapper for TensorFlow 2 with Keras. It is created to make DL easier to apply for domain experts and beginners. We firstly, import the ktrain library and our dataset then preprocessing the data using text.texts_from_array of the ktrain library with the following parameters: class_names=[‘joy’, ‘sadness’, ‘fear’, ‘anger’, ‘neutral’]; preprocess_mode=’standard’ (refers to fastText mode); maxlen=300 (maximum length of the text). Secondly, we create our model using text.text_classifier function, and we use the learning rate finder function to find a good learning rate (0.006 appears to be good in our case). Finally, we train our model for 20 epochs using the fit_onecycle function.

3.3. BERT, RoBERTa, ALBERT, and XLNet

As of last versions ktrain includes a simplified interface to hugging face transformers for text classification, permitting users build, train, and deploy their models with hugging face transformers package in a simple way. Hugging face transformers [30] is a popular Python library that provide pre-trained models accessible to researchers and end users, it is very useful for a variety of NLP tasks. It supports both PyTorch and TensorFlow 2 (from late 2019). Therefore, we use it to train these 4 models, as they are supported by both Hugging Face and ktrain. We used a maximum length of 300, a batch size of 6, 0.00002 learning rate, and 20 epochs for these algorithms. Models used was: bert-base-uncased (BERT), roberta-base (RoBERTa), albert-base-v2 (ALBERT), and xlnet-base-cased (XLNet).

3.4. ELMo

After processing our data like Word2vec and Glove, we create an Elmo embeddings layer by using Tensorflow hub to create a text classifier, then we build a two dense layer (512 units for the first one, and 5 for the second one). Finally, we train our model for 20 epochs using a batch size of 32.

3.5. ULMFiT

After processing our data like Word2vec and Glove, we define the language model and set the learning rates to 0.0479 using lr_find functions. Then, we fit the model for a few cycles by running one epoch and then unfreezing and running more epochs to fine-tune it. Next, we use the encoder from the language model in our classifier with a batch size of 32 and we train it by gradually unfreezing layers and then running an epoch each time with a learning rate related to the result of the lr_find function. Finally, we unfreeze all the layers and run the model for 20 epochs.

3.6. GPT-2

We train this model using PyTorch (as backend), an open-source machine learning python library based on Torch and developed by Facebook [31]. It makes possible to perform necessary tensor calculations in particular for deep learning. For this, we build a custom PyTorch class Dataset, then we initialize and tokenize our data using GPT2ForSequenceClassification and GPT2Tokenizer, then we add_special_tokens to padding it. We also implement the keras CategoricalCrossentropy loss function, due to the very low loss that we got using the pytorch crossentropyloss loss function, because the PyTorch CrossEntropyLoss accepts unnormalized scores for each class (not probability). However, Keras categorical_crossentropy uses from_logits=False by default which means it assumes y_pred contains probabilities (not raw scores). We did that to have the same loss metric between all our models. Finally, we train our model for 20 epochs using a batch size of 6 and a 0.00001 learning rate.

4. EXPERIMENTS AND RESULTS

4.1. Hardware characteristics

We performed our experiments on a MARWAN's high-performance computing (HPC) infrastructure with the following hardware characteristics:

a) CPU: 2x Intel Xeon Gold 6148 (2.4 GHz/20 cores)
b) RAM: 192 Gb
c) GPU: 2x NVIDIA Tesla P100 (12 Gb) with cuda v10.1

4.2. Evaluation metrics

Having more metrics actually makes it harder to compare language models, especially as indicators of how well a language model will perform on a specific task are often unreliable.

a) Accuracy: is the fraction of correct predictions among the total number of predictions as shown in (1).
b) Loss**:** is the difference between the predicted value by the model and the true value. The most common loss function used in deep learning is cross-entropy, where $p(x)$ is the true distribution, and $q(x)$ the estimated distribution, defined over the discrete variable x [32] as shown in (2).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$H(p,q) = -\sum_{\forall x} p(x)\log(q(x)) \quad (2)$$

where:
True Positive (TP): is the number of positive class records correctly classified.

True Negative (TN): is the number of negative class records correctly classified.
False Positive (FP): is the number of negative class records wrongly classified.
False Negative (FN): is the number of positive class record wrongly classified.

a) Precision: is the fraction of correctly identified positive results among all positive results as shown in (3).
b) Recall: is the fraction of correctly identified positive results among the total number of existing positive class as shown in (4).

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

$$Recall = \frac{TP}{TP+FN} \quad (4)$$

c) F1-scor**e**: also called F-score is the harmonic mean of the combination between precision and recall [33], with a value β that can emphasize one or the other as shown in (5). The highest possible value of F1 is 1 and the worst is 0, and guaranteed to be between precision and recall.

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{precision+recall} \quad (5)$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision+recall} \quad (6)$$

where $\beta = 1$, we have the traditional F-measure or balanced F_1-score as shown in (6).

4.3. Experimental results

This section uses the comparative experiment to summarize, compare and contrast the different NLP models and thus discuss their main strengths, using sentiment analysis and deep learning techniques as shown in Table 3. In all our models, we used Tensorboard callbacks, even with the pytorch algorithms (we used TensorboardX an alternative of Tensorboard for pytorch) to plot the validation accuracy as shown in Figure 2, and the validation loss as shown in Figure 3. We also use an early stopping of one epoch based on the validation loss, and we calculate the precision, recall, f1-score of our algorithms using the scikit-learn precision_recall_fscore_support metrics function.

Table 3. Results of the implemented models

Algorithm	Validation Accuracy	Validation Loss	Precision	Recall	F1-score	Training Time	Number of parameters
Word2vec	0.8452	0.4178	0.8410	0.8453	0.8391	Step: 63 ms Epoch: 222s Total: 2472s	Trainable: 440,581 Non-Trainable: 11 060 100
fastText	0.8212	0.4908	0.8167	0.8223	0.8128	Step: 88 ms Epoch: 311s Total: 1567s	Trainable: 4 613 Non-Trainable: 8 640 128
Glove	0.8391	0.4412	0.8362	0.8388	0.8297	Step: 245 ms Epoch: 869s Total: 3576s	Trainable: 338 181 Non-Trainable: 7 373 200
ELMo	0.8152	0.4810	0.8041	0.8102	0.8064	Step: 7 ms Epoch: 844s Total: 3576s	Trainable: 527 369 Non-Trainable: 0
BERT	0.8612	0.3551	0.8589	0.8612	0.8596	Step: 787 ms Epoch: 247 min Total: 495 min	Trainable: 109 361 669 Non-Trainable: 0
RoBERTa	0.8622	0.3629	0.8574	0.8533	0.8548	Step: 609 ms Epoch: 192 min Total: 579 min	Trainable: 125 240 069 Non-Trainable: 0
ALBERT	0.8558	0.3845	0.8514	0.8537	0.8468	Step: 595 ms Epoch: 188 min Total: 567 min	Trainable: 11 687 429 Non-Trainable: 0
ULMFiT	0.8509	0.4315	0.8472	0.8509	0.8476	Step: - Epoch: 192s Total: 1920s	Trainable: 62 805 Non-Trainable: 0
XLNet	0.8574	0.3697	0.8562	0.8583	0.8564	Step: 1s Epoch: 430 min Total: 1293 min	Trainable: 117 312 773 Non-Trainable: 0
GPT-2	0. 8591	0.3796	0.8559	0.8591	0. 8549	Batch: 67 ms Epoch: 18 min Total: 73 min	Trainable: 124 439 808 Non-Trainable: 0

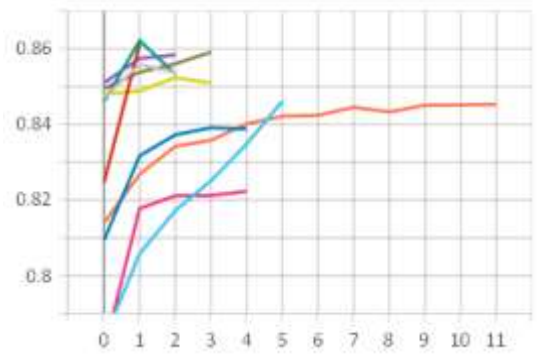


Figure 2. Validation accuracy per epoch scalar generated by Tensorboard

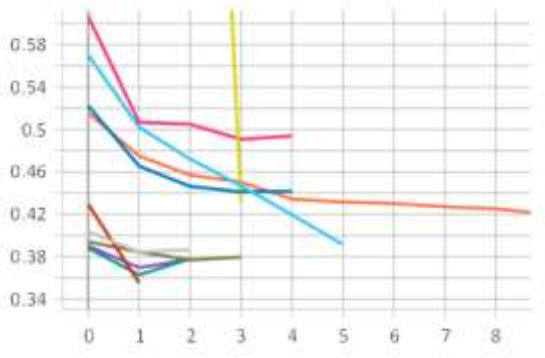


Figure 3. Validation loss per epoch scalar generated by Tensorboard

Word2vec fastText Glove ELMo BERT RoBERTa ALBERT ULMFiT XLNet GPT-2

In this work, we show the application of deep learning-based language representation learning models for the classification of 5 sentiment types based on a combined dataset. We notice that transfer learning approaches reach the best average results using the training and validation data in fewer epochs than word embeddings ones, because it benefits from other base models' knowledge. Nevertheless, it takes more time to train, due to the huge number of parameters used. Among these transfer learning approaches, we conclude that the best one is BERT algorithm because it reaches the best results in almost all our metrics, as shown in Table 3 and Figure 3, with 35.51% as validation loss, 85.89% as precision, 86.12% as recall, and 85.96% as F1-score in 495 min (2 epochs). For the accuracy, RoBERTa model has the best accuracy as shown in Table 3 and Figure 2, with 86.22% in 579 min (3 epochs). On the other hand, transformer-based techniques reach their best result in more time (more than one hour to be trained) compared to the other models.

By examining these results, it is clear that BERT model performed the best results compared to the other methods, since it takes everything into account, in order to predict the true meaning of sentences. This means that transfer learning algorithms can achieve better classification results and learn additional correlations, but in terms of computation time, it consumes more because more parameters are needed as shown in Table 3. In fact, most DL architectures use similar computational elements; therefore, it is a convention to use the number of parameters as a stand-in for complexity, although those networks may have the same number of parameters but require different numbers of operations (ALBERT for example is configured to share all parameters including feed-forward network and attention parameters across layers).

The amount of data in the dataset created is still considerably unbalanced regarding the different types of sentiments. For example, Anger and Fear sentiments in the training and validation sets have very small amounts of data as shown in Figure 1. Therefore, the models have a limited capability to learn accurately these sentiments. Detection average of these sentiments is one of the main factors restricting the overall prediction accuracy.

Despite all efforts, our models tend to overfit. In fact, models trained on text data are subject to overfitting due to the use of out of vocabulary (OOV) token in NLP-based models. OOV is used to handle unseen words. There is a high chance of unseen words in NLP models, and overfitting occurs when the model is trained heavily on the training data but cannot generalize well to unseen data. Those unseen words generate a scenario where the model is strongly tuned to the training set. Hence, we stop at the epoch when each algorithm begins to over-fit.

5. CONCLUSION

Applying NLP and deep learning techniques to sentiment analysis has become a popular research topic lately. Emotions are one of the major aspects of human life that are very useful in various applications. Our work here focused on deep learning-based language representation learning models, and compared them. We are more interested in predicting various types of sentiments. After several experiments, we obtained a reasonable prediction rate for our all models. By analyzing the obtained results, we concluded that BERT is the best one for sentiment analysis. It was able to successfully predict different types of sentiment and showed a very good accuracy (86.12%), and the best performance in terms of validation loss (35.51%),

precision (85.89%), recall (86.12%) and F1-score (85.96%) metrics in comparison to the other models. As future works, we will apply the BERT model to analyze sentiments on online messaging (CHAT or social media) for security purposes.

ACKNOWLEDGEMENT

This research was supported through computational resources of HPC-MARWAN (www.marwan.ma/hpc) provided by the National Center for Scientific and Technical Research (CNRST), Rabat, Morocco.

REFERENCES

[1] "Natural Language Processing Market | Growth, Trends,Forecasts (2020-2025)," *Mordor Intelligence*, 2020. [Online]. Available: https://www.mordorintelligence.com/industry-reports/natural-language-processing-market. [Accessed: 15-Aug-2020].
[2] M. A. Covington, "Building Natural Language Generation Systems (review)," Language (Baltim)., 2001.
[3] M. Bates, "Models of natural language understanding," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 92, no. 22, pp. 9977–9982, Oct. 1995, doi: 10.1073/pnas.92.22.9977.
[4] T. Mikolov, S. Kombrink, L. Burget, J. Černocký, and S. Khudanpur, "Extensions of recurrent neural network language model," in *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing-Proceedings*, 2011, pp. 5528–5531, doi:10.1109/ICASSP.2011.5947611.
[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 2013.
[6] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching Word Vectors with Subword Information," *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 135–146, Jul. 2016.
[7] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.
[8] M. E. Peters *et al.*, "Deep contextualized word representations," in *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2018.
[9] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 2019, doi: 10.18653/v1/N19-1423.
[10] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers)*, 2018.
[11] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," Jun. 2019.
[12] A. Radrof and J. Wu, "language model and unsupervised multitask learning," *OpenAI*, 2018.
[13] M. Berrahal and M. Azizi, "Review of DL-Based Generation Techniques of Augmented Images using Portraits Specification," 2020, pp. 1-8.
[14] B. Liu, "Sentiment analysis and opinion mining," *Synth. Lect. Hum. Lang. Technol.*, 2012.
[15] F. A. Vargas and T. A. S. Pardo, "Aspect Clustering for Sentiment analysis," in *Horizons in Computer Science Research*, pp. 213-224, 2020.
[16] M. Boukabous and M. Azizi, "Review of Learning-Based Techniques of Sentiment Analysis for Security Purposes," *Springer, Cham*, pp. 96–109, 2021.
[17] I. Idrissi, M. Azizi, and O. Moussaoui, "IoT security with Deep Learning-based Intrusion Detection Systems: A systematic literature review," pp. 1-10, 2020.
[18] I. Idrissi, M. Boukabous, M. Azizi, O. Moussaoui, and H. El Fadili, "Toward a deep learning-based intrusion detection system for IoT against botnet attacks," *IAES Int. J. Artif. Intell.*, vol. 10, no. 1, pp. 110-120, Mar. 2021, doi: 10.11591/ijai.v10.i1.pp110-120
[19] L. Torrey and J. Shavlik, "Transfer learning," in *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, IGI global, pp. 242–264, 2010.
[20] G. Brockman, M. Murati, P. Welinder, and OpenAI, "OpenAI API," *2020*. [Online]. Available: https://openai.com/blog/openai-api/. [Accessed: 18-Aug-2020].
[21] N. W. Method *et al.*, "word2vec Explained : Deriving Mikolov *et al*," *arXiv1402.3722 [cs, stat]*, 2014.
[22] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of Tricks for Efficient Text Classification," *15th Conf. Eur. Chapter Assoc. Comput. Linguist. EACL 2017 - Proc. Conf.*, vol. 2, pp. 427–431, Jul. 2016.
[23] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," *arXiv.org*, Jul. 2019.
[24] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations," *arXiv.org*, Sep. 2019.
[25] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen, "CARER: Contextualized Affect Representations for Emotion Recognition," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3687-3697.

[26] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, and S. Niu, "DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset," *arXiv.org*, Oct. 2017.
[27] "Sentiment Analysis in Text-dataset by crowdflower | data.world." [Online]. Available: https://data.world/crowdflower/sentiment-analysis-in-text. [Accessed: 13-Sep-2020].
[28] K. R. Scherer and H. G. Wallbott, "Evidence for Universality and Cultural Variation of Differential Emotion Response Patterning," *J. Pers. Soc. Psychol.*, vol. 66, no. 2, pp. 310–328, 1994, doi: 10.1037/0022-3514.66.2.310.
[29] A. S. Maiya, "ktrain: A Low-Code Library for Augmented Machine Learning," Apr. 2020.
[30] T. Wolf *et al.*, "HuggingFace's Transformers: State-of-the-art Natural Language Processing," arXiv.org, Oct. 2019.
[31] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," *arXiv*, Dec. 2019.
[32] "Cross entropy - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy. [Accessed: 13-Sep-2020].
[33] L. Derczynski, "Complementarity, F-score, and NLP evaluation," in *Proceedings of the 10th International Conference on Language Resources and Evaluation, LREC 2016*, 2016.