DIBERT: Dependency Injected Bidirectional Encoder Representations from Transformers

Abdul Wahab [1] and Rafet Sifa [2]

[1]RWTH Aachen
[2]Affiliation not available

October 30, 2023

Abstract

In this paper, we propose a new model named DIBERT which stands for Dependency Injected Bidirectional Encoder Representations from Transformers. DIBERT is a variation of the BERT and has an additional third objective called Parent Prediction (PP) apart from Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). PP injects the syntactic structure of a dependency tree while pre-training the DIBERT which generates syntax-aware generic representations. We use the WikiText-103 benchmark dataset to pre-train both BERT- Base and DIBERT. After fine-tuning, we observe that DIBERT performs better than BERT-Base on various downstream tasks including Semantic Similarity, Natural Language Inference and Sentiment Analysis.

DIBERT: Dependency Injected Bidirectional Encoder Representations from Transformers

Abdul Wahab
Fraunhofer IAIS
St. Augustin, Germany
abdul.wahab@rwth-aachen.de

Rafet Sifa
Fraunhofer IAIS
St. Augustin, Germany
rafet.sifa@iais.fraunhofer.de

***Abstract*—Prior research work shows that including the syntactic structure of a sentence using a dependency parse tree while training a model improves the performance of a model on downstream tasks. However, most of the prior research work makes use of the dependency parse tree of a sentence for learning task-specific word representations rather than generic representations. In this paper, we propose a new model named DIBERT which stands for Dependency Injected Bidirectional Encoder Representations from Transformers. DIBERT is a variation of the BERT and has an additional third objective called Parent Prediction (PP) apart from Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). PP injects the syntactic structure of a dependency tree while pre-training the DIBERT which generates syntax-aware generic representations. We use the WikiText-103 benchmark dataset to pre-train both BERT-Base and DIBERT. After fine-tuning, we observe that DIBERT performs better than BERT-Base on various downstream tasks including Semantic Similarity, Natural Language Inference and Sentiment Analysis.**

I. Introduction

Progress in the domain of Natural Language Processing (NLP) shows that including relational structure or non-linear structure that makes use of a dependency tree of a sentence generates better word embeddings which improve model performance. As opposed to Skip-gram and CBOW (Continious bag-of-word) based embeddings which consider the context of a word in a sliding window [1], dependency based embeddings make use of the dependency-tree based context rather than window-based context due to which a target word can reach the relevant words which are not accessible in n-gram window-based context [2], [3]. CBOW and Skip-Gram-based models are only good for learning short text representations. To overcome this, Dependency Tree based Convolution Neural Network (D-TBCNN) utilizes a dependency parse tree of a sentence by using fixed-depth tree based convolution window as a feature extractor [4].

Earlier work is based on task-specific dependency-based word embeddings, which means word embeddings can only be used for the task for which embeddings are learned. Most of the state-of-the-art models are based on the attention mechanism [5], [6] to avoid the long-term dependencies issue. This research work makes the following contributions:

- Dependency tree based objective function for injecting syntactic structure
- Utilizing an attention-based model rather than sequential recurrence or convolution model
- Learning fixed-length generic word embeddings which can later be directly fine-tuned

DIBERT model is an extension of the BERT-Base model with an additional objective function called Parent Prediction (PP). PP objective predicts the parent of the i_{th} word in the dependency parse tree which is the main contribution of this paper. We have shown that adding an additional linguistic feature such as syntactic structure learns better word representations by comparing the performance of DIBERT with the BERT-Base model.

The remaining work is organized as follows: We continue with a related work section where we discuss relevant approaches used to include dependency parse tree in the training of models. Following that, we present our model in detail along with evaluations. The paper concludes with an outline of further development.

II. Related Work

Since our main focus is on using dependency tree information, we next briefly review the most relevant prior work. The first work of injecting dependency tree information while learning context based word embeddings is the modification of the original Word2Vec [1] in which the model is asked to predict the surrounding words or nodes in a dependency tree of a sentence [2], [3].

Subject, Verb and Predicate of a sentence are the primary structure of a sentence. Dependency Based Siamese LSTM (DLSTM) for learning sentence-based embeddings which capture the primary structure of a sentence (Subject, Verb, Predicate) improves the performance on a downstream task [7].

While the standard LSTM creates its hidden state using the input at the current time step and the hidden state of the LSTM at the previous time step, the Tree-LSTM composes its current hidden state using the input vector and hidden state(s) of the child unit according to the tree structure [8].

1-dimensional CNN can capture non-linear features of the sentence but it fails to capture the inherent structural features. Tree-Based Convolution Neural Network (TBCNN) includes a tree-based feature extractor that slides over the whole tree which improves the performance on classification

and sentiment analysis [4]. To catch long-term dependencies Ma et al. proposed the dependency-based convolution model (DCNN) [9]. DCNN can capture the tree-based bi-grams or tri-grams which are more meaningful than normal n-gram using convolution on ancestors path and siblings path.

Encoding constituency tree using hierarchical accumulations in a bottom-up fashion and then using the encoded tree representation in transformer architecture improves the machine learning task performance [10].

Earlier work supports utilizing dependency tree structure but it is only limited to task-specific word embeddings. This work proposes a way to learn dependency-based generic representations. To our knowledge, this paper is the first in providing evidence on how to successfully inject dependency tree information in the pre-training of a language model.

III. DIBERT (PROPOSED SOLUTION)

DIBERT is an extension of the BERT-Base model with an additional pre-training objective called Parent Prediction (PP). In the PP task, the model is trained to predict the parent of the word which is extracted from the dependency tree of a sentence. We pre-train DIBERT on three objectives: MLM, NSP and PP. MLM objective enables the model to capture word-level relations and enables a bidirectional flow of information unlike in GPT-1 [11] and prevents a word to see itself indirectly like in ELMO [12]. NSP objective gives the model an understanding of relations between sentences. Whereas PP allows the model to understand the syntactic and structural information of a sentence. Adding a piece of linguistic information in the form of PP objective achieves higher accuracy on downstream tasks as compared to the BERT-Base and it is the key contribution of this research.

A. Parent Prediction (PP)

In the PP task, the model is trained to predict the parent of each word in a sentence. The parent of each word is extracted from the dependency parse tree of a sentence. We used spaCy library [13] for annotating data for Parent Prediction task rather than utilizing commonly used Stanford parser [14]. The reason for selecting spaCy over Stanford parser and many other libraries is that it is much faster and accurate [15]. Figure 1 shows the simple flow of annotating a sentence with parent labels. BERT model has its WordPiece tokenizer to handle out-of-vocabulary words and it can split a word into multiple in-vocabulary subwords[1]. This leads to an inconsistency between tokenized input and parse tree. One problem with this is that we can't parse a sentence after passing it through the tokenizer because the dependency parser does not work with subword tokenized words, and small chunks of subword inputs are not understandable. To solve this, we adopted the idea [16] and did the following:

- First pass a sentence to the parser and get the parent of each word in a sentence

[1]After the word *embeddings* passes through bert-tokenizer it is divided into 3 subwords: *emb, ##bed, ##dings*.

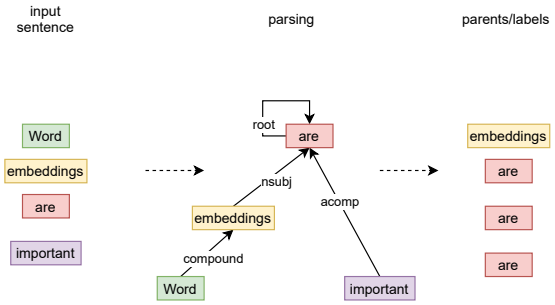


Fig. 1: The input sentence is first passed to the dependency tree parser. The parser generates a dependency tree that represents each word as a node and the link between two words is a labeled edge. It gives PP labels for the input sentence.

- Tokenize both parents and their corresponding words using BERT tokenizer and treat each subword as a separate token
- Assign each token the middle position of the tokenized parent hence generating parent labels for PP objective

The BERT tokenizer takes the input sentence and decides whether to keep every word as a whole or split it into subwords (in case of the out-of-vocabulary word). Figure 2 shows how an input sentence is first parsed and the parent of each word is extracted. After this, each word in the input sentence and each parent word in the parents are tokenized using a tokenizer and all of the subwords are considered as separate tokens. In 2 the word *Word* has a parent *embeddings* and it is mapped to the middle position *##bed* because it contains multiple subwords (*emb ##bed ##dings*). We assign each subword token of a word like *em ##bed ##dings* the parent of the word *embeddings* which in this case is *are*, according to the parse tree. Parent mapping is important because every word in a sentence should have exactly one parent in PP objective. In pre-training, we only predict the parents for the input tokens which are not selected as masked tokens because the [MASK] token is not going to be seen during fine-tuning. Apart from this, we also ignore [CLS] and [SEP] token positions for the PP task. The objective can be explained in the following steps:

$$X_{out} = dibert(X_{in})$$

where DIBERT can be written as a function *dibert* and X_{in} is input to the model and X_{out} is token-level output representation from DIBERT.

$$P = PP(X_{out})$$

$$PP = softmax(\text{FFNN}(X_{out}))$$

FFNN is a feed-forward neural network that generates logits values which are passed to the $softmax$ over vocabulary size because the parent token of each input token is drawn from the whole vocabulary. P is the predicted probability distribution. Cross-Entropy (CE) loss is used to determine

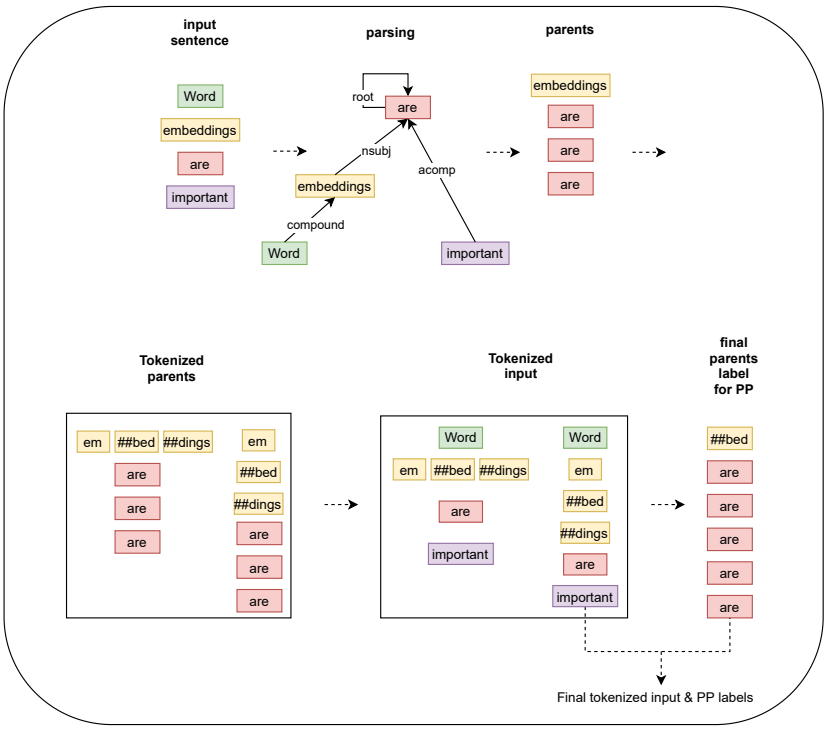


Fig. 2: The input sentence is first passed to the dependency tree parser. The dependency tree parser generates the parent of each word. After that, both parents and words are tokenized using Bert tokenizer and each subword is represented as a separate token (Both boxes show the conversion of subwords into separate tokens). For a parent word having multiple subword tokens e.g. *embeddings* $->$ *emb ##bed ##dings*, only middle subword token is considered like *##bed* for the word *embeddings*.

the disparity between predicted probability distribution P and actual distribution Q. CE loss is written as follows:

$$CE = -\sum_{i=1}^{n} t_i log(p_i) \quad \text{for n classes}$$

where t_i is the truth value and p_i is the probability of the ith class.

B. Pre-training

BERT-Base is originally pre-trained on BookCorpus (800M words) [17] and Wikipedia (25,00M words). To compare DIBERT with BERT-Base, we instead use the WikiText-103 (100M words with pre-defined train, test and validation splits) [18] dataset for pre-training both models from scratch because of the lack of computational resources. In the WikiText-103 dataset, each line represents a paragraph. We select paragraphs having at least two sentences because we need sentence pairs for the NSP task. We use a period as a delimiter for simplicity. This leads to training data of size around 3M sentence pairs.

Based on the size of the data and available GPU power, we pre-train both models with the batch size of 32 and 512 tokens per sequence (16384 tokens per batch). To check the effectiveness of the PP objective, BERT-Base is pre-trained with MLM and NSP objective while DIBERT used MLM, NSP and PP (additional objective). We pre-train for around 900,000 training steps. Since the dataset is small, 2 heads and 2 hidden layers are used for both DIBERT and BERT-Base. We use AdamW with a learning rate of 1e-4 and a learning rate warm-up over the first 90,000 steps. The rest of the parameters are the same as the BERT-Base. The models are trained on an Nvidia Tesla V100-SXM2-32 GB GPU. Pre-training is done separately for both models using the above-mentioned parameters. The rest of the training parameters are the same as the configuration[2] provided by hugging face library.

C. Fine-tuning

We use both models separately for fine-tuning on multiple tasks including Classification, Sentiment Analysis and Natural Language Inference. Most of the hyperparameters are the same as in pre-training except learning-rate and dropout probability. During hyperparameter tuning we use grid-search (16 trails) on the following search space: **learning rate**: [5e-5, 4e-5, 3e-5, 2e-5] , **dropout**: [0.0, 0.1, 0.2, 0.3]. Hyperparameter values depend on the task and we use the following range of values for other hyperparameters: **Batch size**: 64, 128 , **Epochs**: 5, 10, 15. We set the maximum number of epochs for most of the tasks to 15 but for some tasks during fine-tuning, we set to 5 or 10 based on how much time the model is taking to converge. We use pre-trained BERT-Base and DIBERT as a backbone for the additional classifier added on top of the pre-trained models. Similar to the pre-training, fine-tuning is also done separately for both models.

[2]https://huggingface.co/transformers/model_doc/bert.html

IV. Datasets

We fine-tune both models on multiple datasets for classification, sentiment analysis, semantic similarity and natural language inference tasks. The IMDB dataset [19] consists of 50k movie reviews from imdb labeled as positive or negative. We use train, validation and test split of 50%, 25%, 25%.

SciTail dataset [20] is a crowd-sourced entailment dataset from science and web corpus. This dataset consists of around 27k sentences with two labels *entails* and *neutral*. with around 10k *entails* labels and around 16k *neutral* labels.

LIAR dataset [21] is a fake-news detection dataset containing around 12.8k news statements. This dataset is very challenging in terms of labels since it contains six fine-grained labels for truthfulness: labels: *pants-fire, false, barely-true, half-true, mostly-true and true*. The sentences in the dataset are political speeches and it also contains information about the speaker, venue, title of the speech and party information, which we utilize by simple concatenation. DIBERT achieves better performance than what authors have reported.

The Microsoft Research Paraphrase Corpus (MRPC) [22] is a part of the General Language Understanding (GLUE) benchmark which is a group of various natural language understanding tasks [23]. MRPC dataset is annotated for detecting whether a sentence pair is semantically equivalent or not.

Question Natural Language Inference (QNLI) is also a GLUE benchmark dataset. It consists of question-sentence pairs and tells whether a question is answerable from the sentence or not. It is derived from Standford Question Answering Dataset (SQUAD) [24] where question answering task (token level task) is converted into a sentence-level task. It consists of around 108k training sentences and 11k validation and test sentences each.

The Stanford Sentiment Treebank (SST-2) [25] contains human annotated movie reviews. It uses sentence level labels which are positive and negative.

V. Results

All models and scripts used are implemented in Python 3.8.3. We did not do a lot of text preprocessing except punctuation removal using regex because in the original implementation of the BERT authors directly used corpora because preprocessing might cause loss of important context detail. All deep neural networks such as Transformer and BERT are implemented using Pytorch [26] and hugging face [27]. We use Optuna [28] for hyperparameter tuning. Scikit-learn (https://scikit-learn.org/stable/) is used for evaluation metrics. Furthermore, we use built-in Python libraries such as matplotlib for visualization and NumPy matrix operations. The implementation of the DIBERT can be found here[3].

In this section, the experimental results are discussed. We show the learning curves for both DIBERT and BERT-Base to get an idea about how each model is behaving while pre-training and fine-tuning. We also did a comparison of both models on downstream tasks by plotting f1 scores of validation sets. Our main evaluation metric is both, accuracy and weighted F1 score.

A. Pre-training results

Figure 3 shows the pre-training loss curves for BERT-Base model on WikiText-103 dataset. It is observed that MLM (Masked Langauge Modeling) objective requires more pre-training data and converges slower than NSP (Next Sentence Prediction) objective. The reason for MLM to converge slower is that doing 15% token masking makes it difficult for the model to converge. Pre-training using BERT-Base took around **114.5 hours** and validation loss for both MLM and NSP was **3.063** and **0.275** after 900k training steps (10 epochs). Pre-training loss curves for DIBERT on WikiText-103 dataset

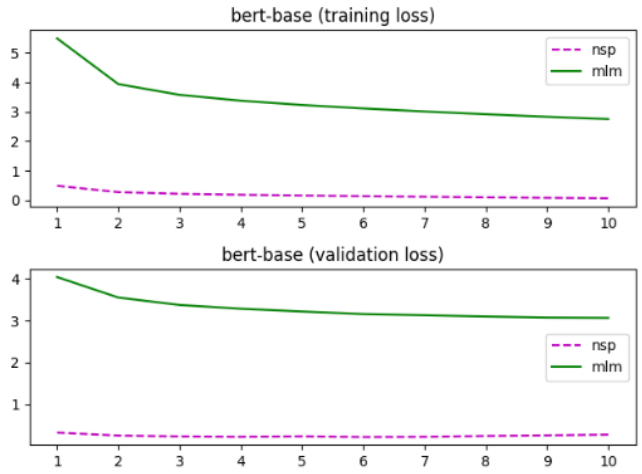


Fig. 3: During the pre-training of the BERT model, training and validation loss is shown for both MLM and NSP objectives.

are shown in figure 4. Pre-training took around **160.7 hours**. After 10 epochs, the validation loss for MLM, NSP and PP was **3.075**, **0.265** and **1.154** respectively. In the figure 5, It is observed that DIBERT needs more steps to converge as compared to BERT-Base and it is explainable because the DIBERT model has an additional objective. Despite taking more time to converge, DIBERT has outperformed BERT-Base on various downstream tasks. In both 3 and 4, we observe that convergence has not happened yet. This would suggest that training for more epochs could improve the performance given the time and powerful GPU or it could be because we did not use a very huge dataset for pre-training.

B. Fine-tuning results

In this section, we first show the comparison of validation f1-score for both BERT-Base and DIBERT. This comparison helps in understanding the overall performance of both models. We do this comparison for every downstream task's dataset using both models. The figure 6 shows the weighted-f1 score for both models over 15 epochs on all datasets. It is seen

[3]https://github.com/wahab4114/dibert

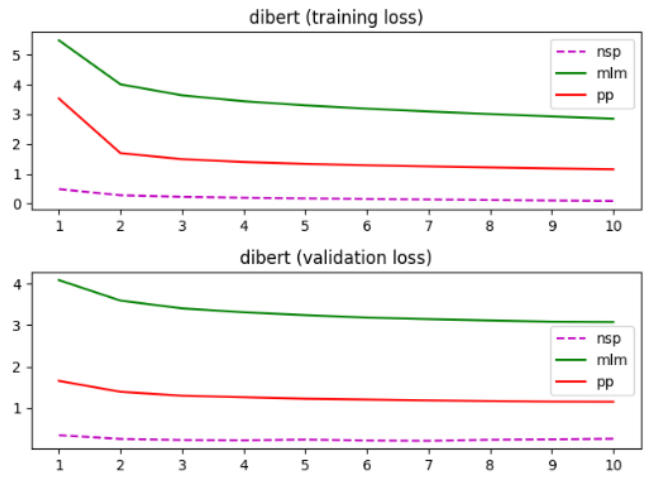


Fig. 4: During the pre-training of the DIBERT model, training and validation loss are shown for the NSP, MLM and PP objectives. Here PP is an additional objective to leverage the performance of DIBERT.

that the highest peaks of the f1 score are for the DIBERT checkpoints[4].

This trend is the same for all of the datasets and it is observed that the DIBERT model has performed (in terms of accuracy and f1-score) well as compared to BERT-Base. We also observe that the validation accuracy is fluctuating and the reason could be that all downstream tasks have less data as compared to the dataset required for training a deep learning model.

C. Comparing the performance on downstream tasks

For the evaluation, after training models using the best hyperparameters, we select the best performing model's checkpoint on the validation set and note both the accuracy and weighted-f1 score on the test set using that checkpoint. We repeat this step for 5 different random restarts and report the average accuracy and f1 score of the test set. For Glue Benchmark datasets, true labels of the test set are not publicly available, we submit the results on their server[5] to get scores on test data. Table I shows the result of DIBERT and BERT-Base after fine-tuning on multiple datasets. It shows that DIBERT performs better than BERT-Base. The overall improvement is around **1%** and for some tasks, it is more than that.

We show that the DIBERT model outperforms the BERT-Base model despite having a slow convergence rate, this can most likely be due to the additional objective in DIBERT. This work is also supported by the prior work which shows that adding the syntactic structure of the sentence improves the model performance on the downstream task. But in this work, we extended the idea of injecting syntactic information

[4]A checkpoint is an intermediate dump of a model's entire internal state so that the training can be resumed whenever required.

[5]https://gluebenchmark.com/

	Result					
	IMDB acc	**Scitail** acc	**LIAR** acc	**MRPC** acc/f1	**QNLI** acc	**SST-2** acc
BERT-Base	86.90	80.15	27.71	70.78/80.44	74.48	86.12
DIBERT	**87.56**	**80.98**	**28.16**	**71.0**/80.44	**76.76**	**86.26**

TABLE I: Comparison of the DIBERT with BERT-Base on downstream tasks. In addition to the accuracy, the weighted f1-score is used for the tasks having an imbalanced dataset.

of a sentence by pre-training DIBERT with an additional objective named PP. Till now we have been focusing on more quantitative analysis of the models but in the next section, we show a little bit of the qualitative evaluation.

D. Analysis of the predictions

In this section, We do a qualitative analysis of the predictions obtained by the DIBERT in comparison with BERT-Base. We show the predictions of DIBERT and BERT-Base on datasets for which we had an access to the test labels. Some examples of the prediction on the LIAR dataset are shown in I. Example 2 shows that even though both BERT and DIBERT did not predict the actual true label but the prediction of the DIBERT (*'true'*) is more close to the actual true label (*'mostly-true'*). In example 3, the prediction of DIBERT (*'mostly-true'*) is closer to the true label (*'true'*) and it is the same for example 4 and 5. For cases when BERT-Base predicts the actual true label and DIBERT prediction is not correct (see examples 6 and 7), we note that the prediction of the DIBERT is still close to the true label. We observe these trends by manually checking the predictions and we report the most occurring trends.

Given enough amount of resources (GPU & Dataset), we can definitely show that DIBERT performs way better than the normal BERT-Base model. Results also show that DIBERT has the potential to completely outperform the BERT-Base model.

VI. Conclusion

We show that adding syntactic information of a sentence in the form of an objective (Parent Prediction (PP) defined in the section III-A) by pre-training a language model can improve the performance of the model on downstream tasks. It utilizes syntax aware or we can call dependency aware pre-trained word embeddings for further fine-tuning. We show that DIBERT III outperforms the BERT model by pre-training both models on the same data and later compare the performance by fine-tuning on various NLP tasks. To our knowledge, this research work is the first in providing evidence on how to successfully inject dependency tree information in the pre-training of a model. Pre-training a language model comes with a cost of resource-intensive computing due to which we decided to use less amount of the pre-training dataset as opposed to what the BERT model was actually pre-trained on. We reported the averaged accuracy/f1-score of both models on various downstream tasks by performing multiple random restarts. DIBERT shows the overall improvement of around

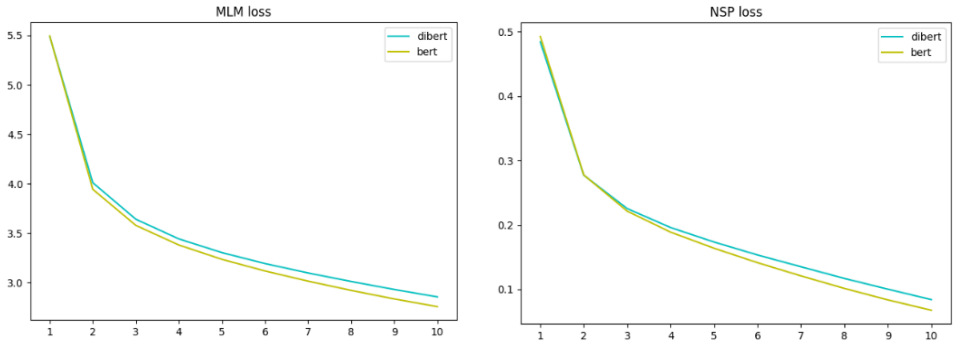
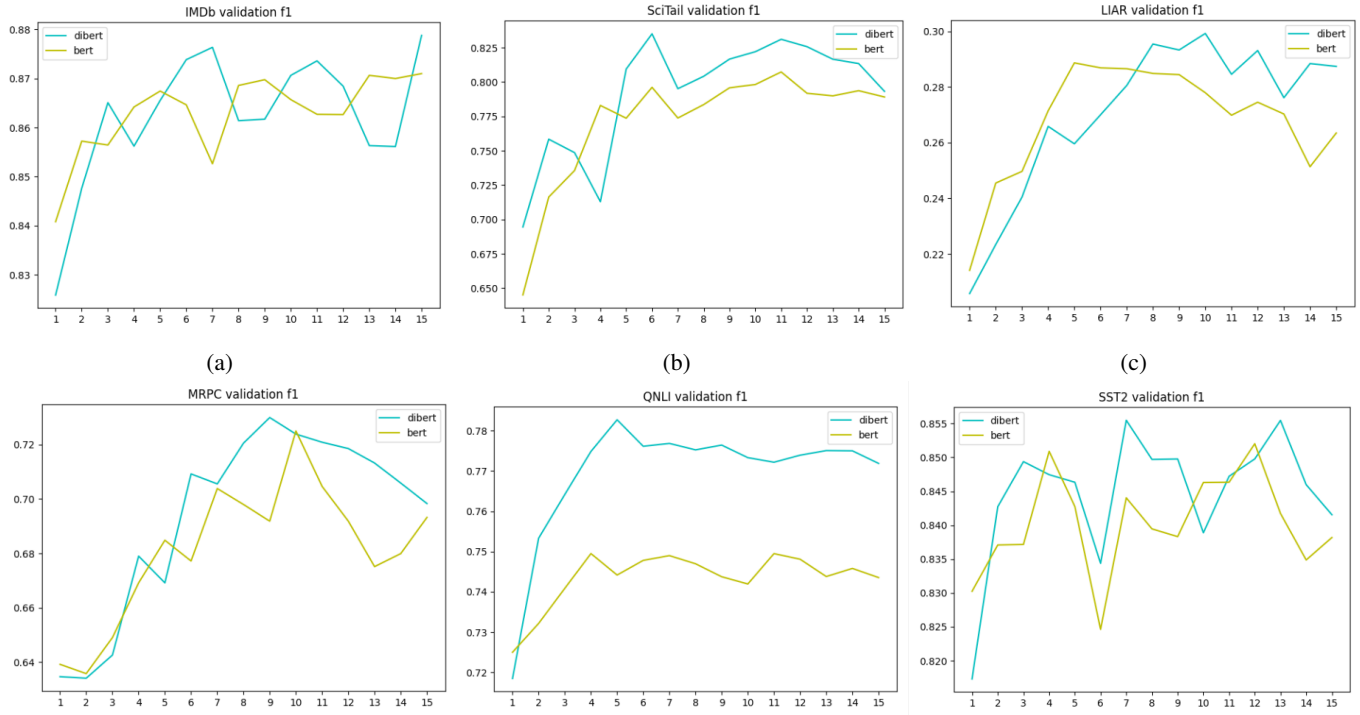


Fig. 5: DIBERT takes more time to converge for both NSP and MLM as compared to BERT-Base.



(a) (b) (c)

(d) (e) (f)

Fig. 6: Learning curves of the validation weighted f1 score for both DIBERT and BERT-Base on various downstream tasks. It shows that the highest weighted-f1 is achieved for DIBERT checkpoints.

1% in the accuracy as compared to the BERT-Base model. We believe that this improvement can be increased given more resources.

VII. Future work

Increasing the size of pre-training data would lead to more solid comparison between the DIBERT and BERT-Base. We would want to extend PP objective by incorporating relationship labels between nodes in dependency tree rather than simply utilizing nodes. We would also be interested in investigating the constituency parse tree and trying constituency tree-based objective rather than dependency tree-based objective. It could be worthwhile to do the comparison of both approaches. We also believe that adding an objective that makes sense linguistically, could also be studied or researched.

References

[1] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International conference on machine learning*. PMLR, 2014, pp. 1188–1196.

[2] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2014, pp. 302–308.

[3] A. Komninos and S. Manandhar, "Dependency based embeddings for sentence classification tasks," in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1490–1500.

	LIAR Dataset			
	statement	**BERT prediction**	**DIBERT prediction**	**True label**
1	"Says Charlie Crist is embroiled in a fraud case for steering taxpayer money to a de facto Ponzi scheme."	barely-true	false	false
2	"Marijuana is less toxic than alcohol."	half-true	true	mostly-true
3	"Now, there was a time when someone like Scalia and Ginsburg got 95-plus votes."	false	mostly-true	true
4	"Every child born today inherits a $30,000 share in a national debt that stands at more than $13 trillion."	half-true	true	mostly-true
5	"Says Charlie Crist voted against raising the minimum wage."	barely-true	half-true	half-true
6	"Each year, 18,000 people die in America because they don't have health care."	true	mostly-true	true
7	"Two million more Latinos are in poverty today than when President Obama took his oath of office less than eight years ago."	half-true	mostly-true	half-true

TABLE II: Prediction examples of LIAR dataset by the pre-trained BERT-Base, and by our DIBERT model with additional PP objective, in comparison. An MLP classifier with one hidden layer was used to obtain the results. Labels of the dataset are *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true* and *true*. It is like 0 to 5, where 0 being the most false and 1 being the most true.

[4] L. Mou, H. Peng, G. Li, Y. Xu, L. Zhang, and Z. Jin, "Discriminative neural sentence modeling by tree-based convolution," *arXiv preprint arXiv:1504.01106*, 2015.

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

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[7] W. Zhu, T. Yao, J. Ni, B. Wei, and Z. Lu, "Dependency-based siamese long short-term memory network for learning sentence representations," *PloS one*, vol. 13, no. 3, p. e0193919, 2018.

[8] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," *arXiv preprint arXiv:1503.00075*, 2015.

[9] M. Ma, L. Huang, B. Xiang, and B. Zhou, "Dependency-based convolutional neural networks for sentence embedding," *arXiv preprint arXiv:1507.01839*, 2015.

[10] X.-P. Nguyen, S. Joty, S. C. Hoi, and R. Socher, "Tree-structured attention with hierarchical accumulation," *arXiv preprint arXiv:2002.08046*, 2020.

[11] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[12] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[13] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd, "spaCy: Industrial-strength Natural Language Processing in Python," 2020. [Online]. Available: https://doi.org/10.5281/zenodo.1212303

[14] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit," in *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, 2014, pp. 55–60.

[15] F. N. A. Al Omran and C. Treude, "Choosing an nlp library for analyzing software documentation: a systematic literature review and a series of experiments," in *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*. IEEE, 2017, pp. 187–197.

[16] E. Bugliarello and N. Okazaki, "Enhancing machine translation with dependency-aware self-attention," *arXiv preprint arXiv:1909.03149*, 2019.

[17] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[18] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," *arXiv preprint arXiv:1609.07843*, 2016.

[19] A. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 2011, pp. 142–150.

[20] T. Khot, A. Sabharwal, and P. Clark, "Scitail: A textual entailment dataset from science question answering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[21] W. Y. Wang, "" liar, liar pants on fire": A new benchmark dataset for fake news detection," *arXiv preprint arXiv:1705.00648*, 2017.

[22] W. B. Dolan and C. Brockett, "Automatically constructing a corpus of sentential paraphrases," in *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

[23] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[24] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[25] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[26] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[27] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6

[28] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.