Effects of Architecture and Training on Embedding Geometry and Feature Discriminability in BERT

Maksim Podkorytov*, Daniel Biś†, Jinglun Cai‡, Kobra Amirizirtol§, Xiuwen Liu¶
*†§¶Department of Computer Science, ‡Department of Mathematics
Florida State University, Tallahassee, FL, USA
{*maksim, †bis, §amirizir, ¶liux}@cs.fsu.edu ‡jcai@math.fsu.edu

***Abstract*—Natural language processing has improved substantially in the last few years due to the increased computational power and availability of text data. Bidirectional Encoder Representations from Transformers (BERT) have further improved the performance by using an auto-encoding model that incorporates larger bidirectional contexts. However, the underlying mechanisms of BERT for its effectiveness are not well understood. In the paper we investigate how the BERT architecture and its pre-training protocol affect the geometry of its embeddings and the effectiveness of its features for classification tasks. As an auto-encoding model, during pre-training, it produces representations that are context dependent and at the same time must be able to "reconstruct" the original input sentences. The complex interactions of the two via transformers lead to interesting geometric properties of the embeddings and subsequently affect the inherent discriminability of the resulting representations. Our experimental results illustrate that the BERT models do not produce "effective" contextualized representations for words and their improved performance may mainly be due to fine-tuning or classifiers that model the dependencies explicitly by encoding syntactic patterns in the training data.**

I. Introduction

With the availability of parallel computation powered by graphical processing units and massive amounts of data, natural language processing techniques have improved the performance of many challenging tasks significantly and surpassed human performance in a number of areas. As vast human knowledge exists in texts and language is the most effective human-to-human communication interface, machines that could understand natural language and communicate with humans naturally could transform many of the services in modern societies. A fundamental problem in natural language processing is to capture rich semantics of natural language computationally. Clearly, the meaning of a sentence depends on its constituents (such as words) and their relationships via syntax rules. By capturing regular patterns in large corpora, computational models can be used to help resolve a number of problems. One such model is the language model (LM) [1], which captures the probability distribution of the next word given its context. The model can be used to predict the next word(s) and resolve ambiguities (such as in speech recognition).

A distinctive feature of language models is the ability to be trained through self-supervision in that one can use the actual next word as the target. This feature enables training models using larger and larger datasets with better performance on a number of benchmarks [2] [3] [4]. Traditional language models are auto regressive; the context has consisted of either the words before the target word or after the target word, but not the entire sentence. However, representations based on the entire sentence or even sentences are useful for at least some NLP tasks [5]. For example, in order to predict the emotion in a sentence accurately, one needs to read the entire sentence as negation and implicit words (such as "not" and "but") can change its meaning completely.

To overcome the limitation, BERT [3] builds on an auto-encoding model, rather than an auto regressive one. More specifically, a BERT model is pre-trained by being able to predict some of the masked words via transformers [6]. Since it was introduced, BERT has set the new state of arts on a number of NLP benchmarks [3]. While there is ample empirical evidence that BERT models work well on a number of NLP tasks, it is not very well understood why they are effective. A common explanation is that BERT computes contextualized representations, where the representations of words depend on their contexts. However, this is not supported by the underlying auto-encoding model. For example, even though the word "bank" may have multiple meanings, the objective function of the pre-training task requires the same word embedding index to be predicted under different contexts. In this paper, we attempt to understand BERT by **analyzing the geometry of its word embeddings, and the intrinsic discriminability of the context-dependent representations produced from the embeddings via transformers**. Such analyses provide new insights that are necessary to further improve BERT models.

II. BERT Architecture and pre-training

BERT is essentially an encoder from the encoder-decoder architecture of Transformer [6]. By overcoming the limitations imposed by unidirectional processing, BERT and its variations are able to process a bidirectional context by masking the target words with a special [MASK] token, thus adding noise and making the prediction of the target word more difficult. Consequently, the training objective is to predict the original vocabulary id of the masked token based only on its noisy context. The introduction of the [MASK] token causes a mismatch between the pre-training and fine-tuning input. To alleviate this, the training data generator chooses 15% of the token positions at random for prediction; if the token t is chosen for prediction, it is replaced with [MASK] 80%

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

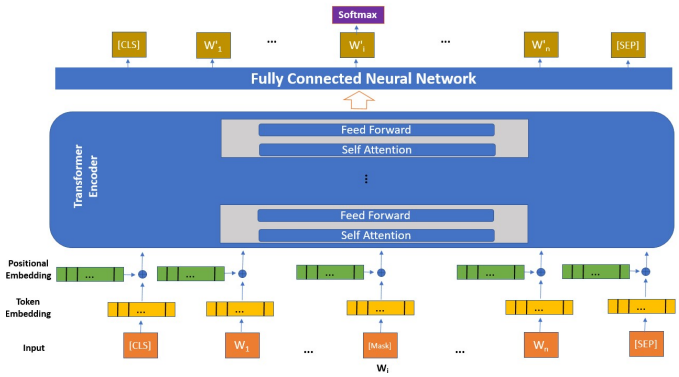

Fig. 1: Bert pre-training architecture

of the time, a random token 10% of the time and kept unchanged 10% of the time. Effectively, the model does not know which tokens it will be asked to predict and is forced to keep a distributional contextual representation for every token, although the cross-entropy loss is calculated only for the chosen 15% of the tokens. The second pre-training objective is the binary Next Sentence Prediction (NSP). An input sequence of BERT consists of two segments *A* and *B*, 50% of time *B* is the actual segment that follows *A* in the corpus and 50% of time it is a random segment. BERT is asked to predict whether *A* and *B* are indeed a consecutive segments. The pre-training loss is the sum of the mean masked LM likelihood and the mean next sentence prediction likelihood. For a detailed description of the structure and the objective function, we refer the reader to [6] and BERT [3].

By looking at BERT from its input and output relationship, BERT is an auto-encoding model as shown in Fig. 1. Given a sentence, consisting of tokenized word pieces [7], the word pieces are first translated into embedding vectors which can be modeled as matrix multiplications. The embedding vectors are first transformed by a number of multi-head attention and feed-forward neural network layers, producing transformed vectors as the output from the BERT architecture. The matching of query and key vectors performed by the attention heads via inner product is inherently pair-wise; this resembles the product of the input and output vectors used in the skip-gram model [8], and may be a source of similarities between the geometry of the representations produced by the two models.

During pre-training, the output vectors are used as "contextualized" representation of the word pieces and a softmax function is applied to the result of multiplying each output vector and the input embedding matrix to produce a probability distribution over the word pieces in the BERT dictionary. Note that while the transformers in BERT encourage contextualized representations through query, key, and value matrices and feed-forward neural networks, the use of cross entropy is to "recover" the indices of the original word pieces, where the optimal solution would be the average vector in different positions and different contexts.

III. BERT Embedding Geometry and Effects of Transformers

The goal of this section is to study how the BERT embedding vectors are affected by its architecture and pre-training protocol by probing them. The parameters of a BERT model include the embedding vectors for word pieces along with the parameters for self-attention and feed-forward transformer layers. First, the embedding vectors were analyzed, then different layers of transformers were probed using simplified inputs. All of the experiments and analysis in this paper were done using the uncased base BERT model, with 12 transformer layers, 12 attention heads, and the hidden dimension equal to 768[1]. When BERT is given an input sequence of tokens, the embedding layer, as well as all 12 transformer layers have a rank-2 tensor as their activations; with the shape of $(len\_sequence, hidden\_dim)$.

A. Embeddings Geometry Analysis

To study the distribution of the representations in the embedding space, pairwise correlations of weights of BERT embedding layer were computed. Since there are 30522 768-dimensional embedding vectors in the pre-trained BERT model, the correlations are 30522×30522 values in $[-1, 1]$ range. The histogram of correlations in Fig. 2 shows that the embeddings tend to have somewhat strong pairwise correlations, implying that the representations occupy a narrow region in the embedding space. This is one of the geometric effects of pre-training, as vectors sampled from a normal distribution would have correlation histogram centered around 0.

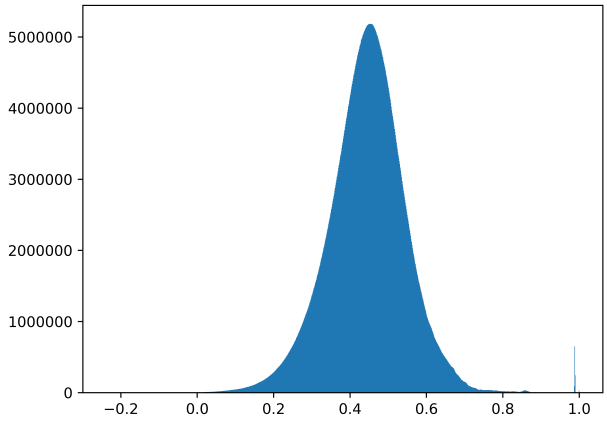

Fig. 2: Histogram of BERT embedding pairwise correlations. Note that the peak near 1.0 is due to some vectors are similar to each other.

Next, the magnitudes of the representations were analyzed. The magnitude of an embedding is important since it directly affects the result of the dot product of the BERT output and the embedding matrix; this product is subsequently used as an

[1]Pre-trained models based on Transformers [9] are publicly available at https://huggingface.co/transformers/

# most freq	token	count	magnitude
1	the	960941	**0.821649**
2	of	537171	0.843784
3	and	377328	0.898151
4	one	363157	1.014323
5	in	340397	0.866191
6	a	294527	0.870349
7	to	287473	0.907536
8	zero	234036	**1.244975**
9	nine	219649	0.967104
10	two	170537	0.938169

TABLE I: 10 most frequent tokens in Text8 dataset [10]. The most frequent token ('the') has the smallest Euclidean norm.

input to the softmax function to produce a probability distribution over the model's vocabulary. In order to analyze the magnitudes of the representations, the text8 dataset was used [10], which is a concatenation of English Wikipedia articles text truncated at 10^8 characters. The dataset was tokenized and the word-part tokens were filtered out. The remaining tokens were sorted by their frequency and the euclidean norm was computed for each of the remaining tokens' embedding. The plot of embedding euclidean norm with respect to token frequency is displayed in Figure 3. The norms of 10 most frequent tokens are in the Table I. The more frequent words tend to have smaller norms, with the most frequent word in the dataset ("the") having the smallest norm. Many of the most frequent tokens are *function* words that appear in a large variety of contexts; such a context diversity may be responsible for pushing the tokens embeddings towards the coordinates origin during pre-training.

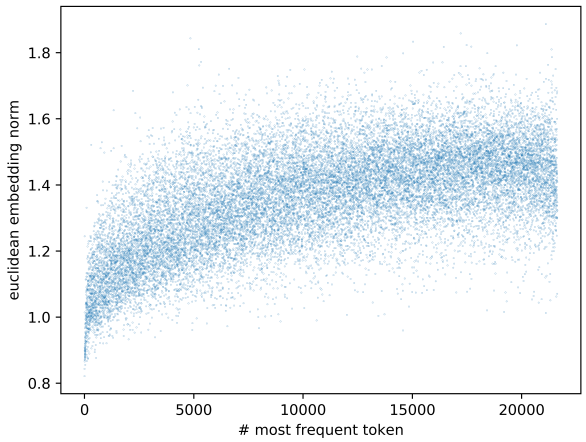

Fig. 3: Euclidean embedding norms for tokens sorted by their occurrence frequency in Text8 dataset [10]. The more frequent tokens tend to have a smaller Euclidean norm.

B. BERT Transformer Analysis

We probed the BERT transformers layers using simplified inputs. In the following experiment, a single token ("cat") was used as an input for BERT in order to compute all hidden states; then, the pairwise correlations of the hidden states were computed. The correlation matrix heatmap is displayed in Fig. 4; we observed, that in the matrix there is an emergent block structure with bottom layers and top layers having high pairwise correlations within their group. The correlation with the embedding layer with each of the following layers decreases up to layer 7 and starts increasing after that. Being an auto-encoder model, BERT encodes patterns in the data in the first few layers and the upper layers try to "reconstruct" the original masked words, by recovering information from the patterns.

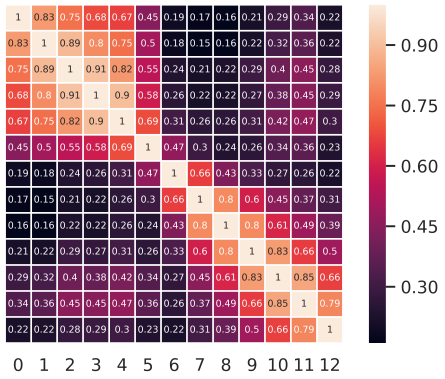

Fig. 4: The correlation coefficients between outputs of the embedding layer (#0) and all the transformer layers (#1-#12) of pre-trained BERT [3] for the input token sequence consisting of a single token 'cat'.

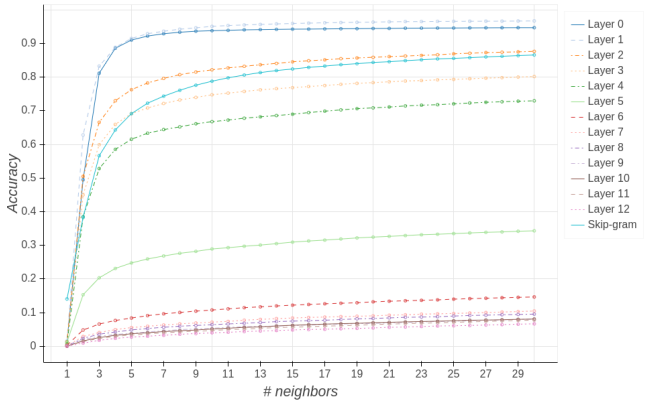

Fig. 5: Analogy query results: the number of dataset entries for which the answer to analogy query (e.g. *king - man + woman*) belongs to correct analogy's (*queen*) K nearest neighbors. Different lines correspond to different BERT transformer layers that are used to get embeddings from. Skip-gram is used as a baseline.

In the next experiment, the analogy dataset [2] was used, where each sample is a quadruple (A, B, C, D) organized so that A is related to B in the same way as C is related to D, e.g. (Paris France Helsinki Finland). The dataset was

filtered, so that each word corresponds to a single token. BERT activations were computed when each of these tokens was the only input to the model, and the activations were used for the *analogy* query formulated in terms of vector arithmetic as follows: with respect to (B - A + C), how many tokens embeddings are closer in terms of cosine similarity than the correct answer D? The results are shown in Fig. 5; y-axis shows the percentage of samples for which the correct answer D belongs to k neighbor embeddings to the *analogy* expression, while x-axis stands for the number of neighbors k. The results show that the activations in all transformer layers are able to capture the notion of similarity formulated in terms of *analogy* query. However, this ability is gradually lost along with increasing depth. The rate of accuracy decrease is steepest between the middle layers. [11] show that the activations from higher layer occupy a smaller region in the embedding space, this compressed distribution may negatively impact the performance of the higher layers on this task. Also, surprisingly, the first layer demonstrates a better accuracy than the embedding one. Skip-gram (SG)[2] [8] was used as a baseline. The same set of filtered queries was used; for a fair comparison the vocabulary of SG was limited to the 30,000 most frequent English words and the words needed for the task, resulting in a dictionary of 30,378 words. While SG performs better when only one nearest neighbor is used, it is surpassed by BERT activations when more neighbors are considered. One of the reasons for this may be the higher dimensionality of BERT vectors (768) than the SG vectors (300), resulting in a exponentially larger embedding space and increased difficulty when only one neighbor is used. The experiment was repeated by evaluating (A - B + D) against C. The reformulated task results show the same patterns as the original ones.

IV. Discriminability of BERT Features

As pointed in [3], BERT models can be used both with feature-based approaches and fine-tuning approaches. In this section, the inherent discriminability of representations produced by BERT models is evaluated by analyzing whether words in semantic classes will cluster together first. This forms the basis for such features to generalize well for tasks. Our systematic analysis shows the features are not inherently discriminant.

Then, we investigated whether the sequence using computed embeddings still contains the information necessary for classification if we model the dependencies among the embeddings.

A. Discriminability analysis

We compute all transformers layer activations of pre-trained BERT on a labeled dataset and compute two distinct metrics that summarize how well the activations are clustered within the same label and separated across different labels.

We borrow the idea for the first metric from [12]. For activation $x \in \mathbf{R}^n$ with label y, let L designate the relation between x and y ($L(x) = y$), let $\rho(x)$ be a ratio of the distance to the nearest activation in another class $y' \neq y$ to the distance to the nearest activation in the same class y incremented by a small positive number ϵ for numerical stability:

$$\rho(x) = \frac{\min\limits_{L(x')\neq y} d(x, x')}{\min\limits_{L(x)=y, x'\neq x} d(x, x') + \epsilon}. \tag{1}$$

Intuitively, if an activation x is within a cluster of activations with the same label, its nearest neighbor has the same label; the numerator is greater than the denominator and $\rho(x) > 1$. In a converse case, the nearest neighbor has a different label and $\rho(x) < 1$. Next, we compute the summary of ρ values for the entire dataset; since ρ can range from 0 to $+\infty$, we squash it into a finite range using the following transformation:

$$f(x) = \frac{1}{1 + exp(1 - x)} \tag{2}$$

After that, we compute the arithmetic mean over the entire dataset:

$$F = \frac{1}{|\boldsymbol{X}|} \sum_{x \in \boldsymbol{X}} f(\rho(x)) \tag{3}$$

as well as the empirical distribution of $f(\rho(x))$ and its statistics. Hence we are able to reason about discriminability of the activations; for in-cluster activations, $1 > f(\rho(x)) > 0.5$, whereas for outliers $0 < f(\rho(x)) < 0.5$.

The second metric is based on *silhouette* analysis [13]. Following the convention used during the first metric definition, let $a(x)$ be the mean of in-class distances:

$$a(x) = \frac{1}{|\{x' : L(x') = y, x' \neq x\}|} \sum_{x' : L(x')=y, x'\neq x} d(x, x') \tag{4}$$

Let $b(x)$ be the smallest mean distance to a set of activations having the same label distinct from y:

$$b(x) = \min_{y' : y' \neq y} \frac{1}{|\{x' : L(x') = y'\}|} \sum_{x' : L(x')=y'} d(x, x') \tag{5}$$

The silhouette $s(x)$ is defined as follows:

$$s(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))} \tag{6}$$

If x is a sole sample in its class, $s(x)$ is defined to be 0.

Values of $s(x)$ close to 1 imply that $a(x) \ll b(x)$, that is, the mean distance between x and samples within its class is much less than the minimal mean distance between x and samples in other classes, contributing to a high discriminability of chosen feature. Similarly, values of $s(x)$ close to -1 mean that the sample x is closer to some set of samples with a label distinct from its own. Values of $s(x)$ close to 0 imply that x is as close to set of samples with its own label as to the samples with another label. The latter two cases contribute to a low discriminability of chosen feature.

Similarly to the first metric, we computed the empirical distribution of the *silhouette* over the dataset and its statistics to analyze the discriminability.

[2]Pre-trained vectors used for this experiment are publicly available at https://code.google.com/archive/p/word2vec/.

B. Two approaches for sentiment analysis using BERT

We have a labeled dataset of tweets, where each tweet is a short text and the label is the sentiment expressed in the tweet. For classification, instead of training a model from scratch, we want to leverage knowledge from a pre-trained model. We consider two ways of doing that: The first one is to use BERT activations as input features for a simpler model that is trained *from scratch*, while the second one is to augment the pre-trained BERT model with a classifier subnetwork and *fine-tune* the composite network. Each considered BERT activation is a rank-2 tensor of shape $(num\_tokens, bert\_hd)$, where num_tokens is the number of tokens in the input to BERT and $bert\_hd$ is the *hidden dimension* BERT hyperparameter. We will refer to each activation as a length-num_tokens sequence of $bert\_hd$-dimensional vectors in this section. For each model, a cross-entropy loss is computed based on final linear layer outputs (logits) and true data labels; it is optimized using a gradient-descent based optimizer.

1) Feature-based classification: For the feature-based approach, two different network architectures were used. The first architecture is a shallow network that consists of 2 linear layers with CELU nonlinearity between the layers. The input vector is the first element of a BERT hidden state; for such state we use the output of each transformer layer as well as the embedding layer output.

The second architecture is a two-layer BiLSTM with a classifier head on top; the classifier head is a single linear layer that takes as input a concatenation of last forward and backward outputs of the BiLSTM. The BiLSTM input is the *entire* sequence of BERT hidden state elements (as opposed to using only *the first* element of such sequence). We use the same hidden states of BERT as input as for the first architecture.

2) Fine-tuning: In this case, the model consists of a pre-trained BERT model and a classifier head. The classifier head is a linear layer; its input is *the first* element of a BERT hidden state; the hidden states are again the embedding layer output and each of the 12 transformer layer outputs.

V. Experiments and Results

In this paper, we used two datasets: text8 and an emotion dataset. The text8 was used to study the word frequency and the emotion dataset was used for classification.

A. Emotion Classification

In each classification scenario, we perform stratified 10-fold validation and report mean accuracy, balanced accuracy and adjusted balanced accuracy [14] for classifiers based on each BERT hidden state in its according figure.

For feature-based classification using shallow network, we used the entire emotions dataset with 340540 tweets labeled into 6 classes. We use Adam optimizer [15] with learning rate of 2^{-8}, constant learning rate schedule and no weight decay. We set the batch size to 64 during training and to 1 during validation. The dimension of middle hidden layer activation is 384; the classifier input is 768-dimensional and the network output is 6-dimensional. The classifier is trained for 5 epochs. We report the accuracy metrics in Figure 6. The worst accuracy is at the embedding hidden state for the same reason as we have observed in our fine-tuning experiment. In contrast to fine-tuned and BiLSTM-based models, we observe worse accuracy; it seems that this architecture has less capabilities in capturing temporal patterns present in text.

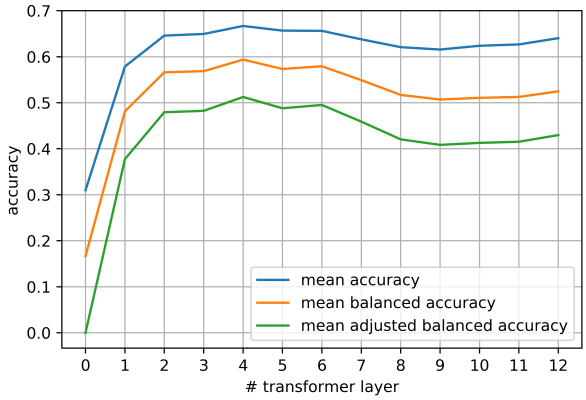

Fig. 6: Accuracy metrics of feature-based classifier based on a shallow network for each of the BERT hidden states as the classifier input.

For feature-based classification using BiLSTM as well as for fine-tuning based classification, we used a stratified sample of 8964 tweets of the original emotion dataset as a trade-off between training time and accuracy. We also pad/truncate the input sequences to 32 tokens for enabling batch processing. Having the same dataset size allows to perform a fairer comparison of these models' accuracy scores. For feature-based classification using BiLSTM, we use Adam optimizer [15] with learning rate of 2^{-11}, constant learning rate schedule and no weight decay. We set the batch size to 64 during training and to 1 during validation; the LSTM hidden dimension is 384, input dimension is 768 and there are 2 bidirectional layers. The dropout probability is 0.2 for dropout layers inside LSTM as well as between LSTM and the linear classifier head. The classifier is trained for 12 epochs. The accuracy metrics are displayed in Figure 7.

For fine-tuning based classification, we use AdamW optimizer [16] with learning rate of 2^{-17}, constant learning rate schedule and no weight decay. The batch size is 16 during training and 1 during validation; the dropout layer between BERT and linear layer is set to have probability 0.1. The classifier is trained for 6 epochs. Figure 8 contains the accuracy metrics. The lowest accuracy is at the embedding output. Since the first element is always the '[CLS]' token embedding. the model fails to generalize as it never sees other tokens from the input sentence.

We aggregate the results in Table II. According to the classification results, the performance of shallow network is the worst among the models, while BiLSTM and fine-tuned

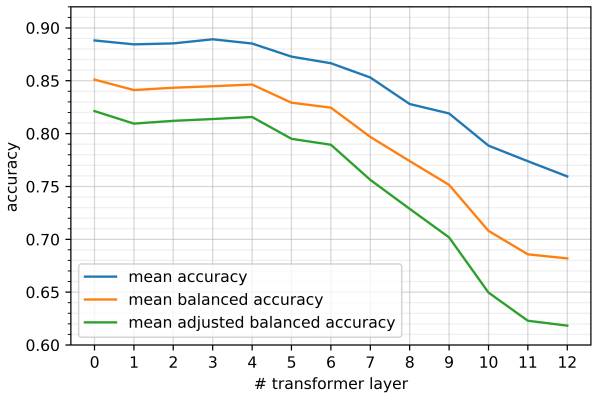

Fig. 7: Accuracy metrics of feature-based classifier with BiLSTM processing the BERT hidden states. We observe that it is possible to achieve a comparable performance to a fine-tuned model using output of a pre-trained model and a simpler architecture. In contrast to fine-tuned and shallow network based models, here we achieve the best performance in the embedding layer and a few bottom transformer layers.

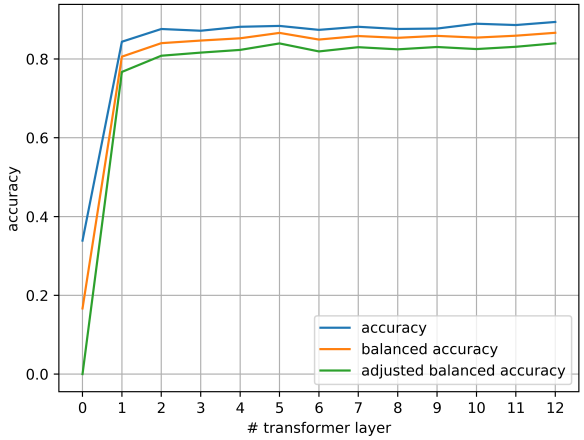

Fig. 8: Accuracy metrics of fine-tuned model for each of the BERT hidden states as the classifier head attachment point.

networks have comparable best performances among BERT hidden states. The original BERT paper [3] also reported that the accuracy of BiLSTM feature-based approach is comparable to fine-tuning one; however, they did not provide accuracies for *all* layers, while we do.

The fine-tuned and shallow networks have almost 0 adjusted balanced accuracy on the embedding layer. In these cases, a model does not take into account all tokens in the input token sequence, as we have set up the model input as the first element of the BERT hidden state; hence, the performance is predictably random. BiLSTM-based network does not have this property as its recurrent subnetwork processes all input tokens before feeding into a linear classifier, and this is reflected in its accuracy. The fine-tuned and shallow networks using the first element of a deeper BERT state as input do not have this property as well, as there is a connection to all tokens in the input through attention [6] layers of BERT.

There is an interesting pattern that we cannot explain yet; the graph of accuracy with respect to BERT hidden state is not monotonous for each classifier architecture. Both feature-based classifiers have the accuracy peak near 4th BERT hidden state. The fine-tuning based classifiers have the accuracy peak in the 4th to last BERT hidden state.

B. Embeddings Discriminability

We study the embeddings of a few selected emotion words and their 10 nearest neighbors (in terms of Euclidean norm) from text8 dataset. A 2-dimensional projection of BERT activations are demonstrated via PCA [17]. The results in Figures 9 and 10 show that the Bert embeddings do not display well-formed semantic clustering. For instance, in Figure 9 “sadness” and “happiness” are very close to each other, while both of them are far from “pleasure” and “joy”. In Figure 10, “sadness” is somewhere between “pleasure” and “happiness”. As we can see, intrinsic discriminability of BERT’s pre-trained embeddings is limited for direct usage.

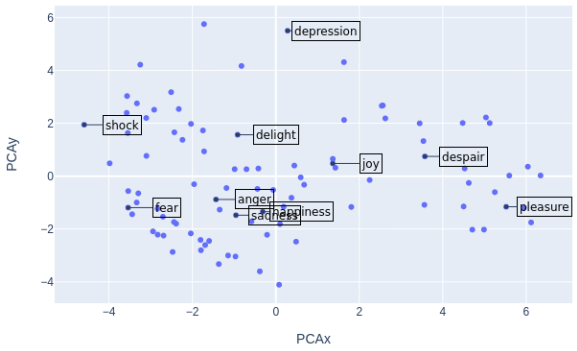

Fig. 9: PCA of emotion words, averaged through all layers

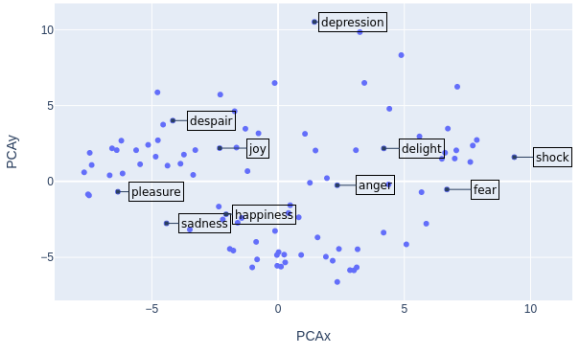

Fig. 10: PCA of emotion words in the 12th layer

Additionally, we used the emotion dataset to study the raw BERT activations discriminability. We computed two discriminability metrics for each sample in the dataset based on sequence-wise average activation of each BERT transformer layer as well as the embedding layer; the distributions of each discriminability metric are in Figures 11 and 12. The results show that both metrics values tend to be in the neighborhood of the decision boundary (0.5 and 0, respective), hence, the raw activations in any BERT layer are not clustered based on the dataset labels.

VI. Related Work and Discussion

Neural distributed representations of words were initially introduced by [1], who proposed to learn these representation using Neural Network based LM (NNLM). Later, Mikolov et al. [18] introduced RNN based LM (RNNLM) to handle variable length sequences and a better context modeling. Distributed representations of words have gained immense popularity after the introduction of an efficient way of estimating them, in the form of skip-gram and CBOW models by [2], [8] as well as the Glove [19]. However, these approaches use limited contexts. To model dependencies in larger contexts, LSTM networks [20] were used to model the language [21]. The potential of LSTM's memory cell was further utilized by [5] who proposed deep contextualized word representations, called ELMo. In contrast to having a fixed representation, in ELMo words are assigned a representation that is a function of the entire input sentence, where the function is modeled using bi-directional LSTM.

	Fine-tuning		BiLSTM		Shallow NN	
L.	Acc.	Bal. acc.	Acc.	Bal. acc.	Acc.	Bal. acc.
E	0.3382	0.1667	0.8881	0.8511	0.3099	0.1667
1	0.8551	0.8020	0.8844	0.8412	0.5787	0.4812
2	0.8859	0.8371	0.8853	0.8434	0.6459	0.5662
3	0.8976	0.8531	**0.8892**	0.8448	0.6495	0.5687
4	0.9003	0.8608	0.8852	**0.8464**	**0.6668**	**0.5937**
5	0.8978	0.8606	0.8728	0.8292	0.6568	0.5734
6	0.8986	0.8538	0.8666	0.8245	0.6562	0.5793
7	0.8993	0.8535	0.8531	0.7969	0.6379	0.5491
8	0.9006	0.8513	0.8280	0.7740	0.6208	0.5170
9	**0.9028**	**0.8626**	0.8191	0.7515	0.6157	0.5070
10	0.9003	0.8513	0.7886	0.7080	0.6238	0.5107
11	0.9022	0.8508	0.7739	0.6858	0.6267	0.5127
12	0.9016	0.8488	0.7595	0.6819	0.6404	0.5247

TABLE II: Emotion classification accuracies.

Vaswani et al. [6] introduced Transformer, the first sequence transduction model based entirely on attention (described in detail in the architecture section). Transformer eliminates recursion, allowing for modeling the long-distance dependencies with a constant number of operations. First of the Transformer based LM, GPT [22], is an auto-regressive model based on Transformer decoder (without encoder-decoder attention). At the time of its introduction GPT improved the state-of-the-art result on multi-task NLP benchmark GLUE [23]. Finally, Devlin et al. [3] introduced BERT, further improving the state-of-the-art on GLUE.

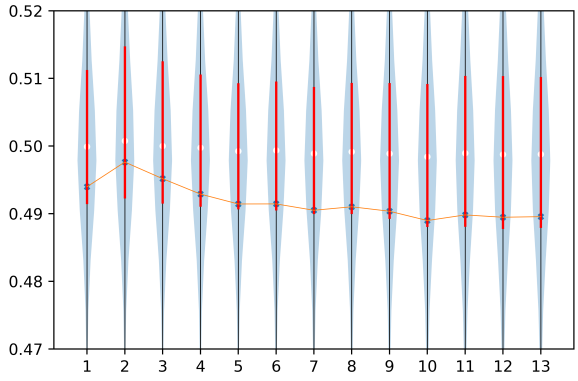

Fig. 11: Discriminability distributions for each transformer layer.

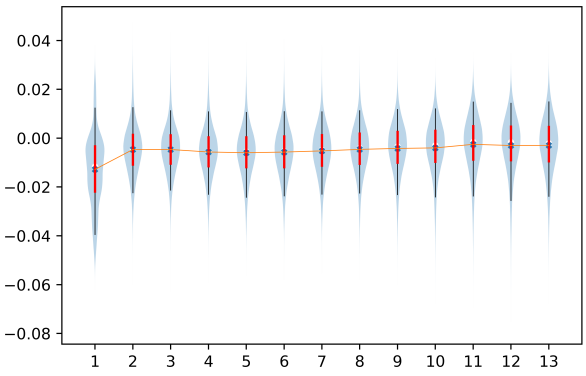

Fig. 12: Silhouette distributions for each transformer layer.

Due to BERT's superior performance, a number of studies were performed in order to understand its underlying mechanisms [24] [25]. For example, there are a number of studies on analyzing attention heads in BERT and their roles of capturing syntax and semantic patterns [26] [27] [28] [29] [30]. In contrast, the focus of this paper is on the characteristics of the learned representations, not the patterns captured by the attention mechanisms. The closest to this work are the papers of Voita et al. [31] and Ethayarajh [11]. Voita et al. [31] analyzed the evolution of the word representations across the layers of a masked language model. By using correlation analysis, it reports patterns similar to the one in Figure 2, to show that masked language models are auto-encoders which create contextual representations in the early layers and try to recover the tokens identities in the later layers in order to predict the correct ids of the target tokens. The evolution of the embeddings across the layers is not the focal point of this study; rather, we performed a thorough analysis of key geometric properties learned by masked language models and their roots in the Transformer's architecture, using BERT as

a case study. Ethayarajh [11] studied geometry of representations produced by BERT, GPT-2 and ELMo, focusing on evaluating how much context is captured by the respective models using self-similarity - the average cosine similarity between the contextualized representations of the same word in different context across the layers. The author reported that the self-similarities in BERT are consistently higher than that in GPT-2. This can be attributed to the different objectives of the models; while representations in GPT-2 are used to predict the next token in the sequence, those in BERT must predict their own id. Similarly to [11], we found that the words in BERT occupy a narrow cone in the embedding space, however we used a different methodology. In this work we focused on the discriminability of the representations produced by BERT and their influence on sentence classification tasks; we conducted a broader study of the geometry of the representations grounded in the underlying model's architecture.

While BERT is more powerful than the skip-gram [2] model, there are inherent similarities between the two. Consequently, studies on the skip-gram model have shown similar observations to that of BERT embedding geometry. For example, Schakel and Wilson [32] show a similar inverse relationship between word frequency and the embedding vector length of words using the skip-gram model.

There are also a number of studies which show that the BERT models are capable of discovering features in commonly used NLP pipeline [33] [34]. However, as the BERT models are inherently an auto-encoder model, the underlying mechanisms of the observed features must be rooted in the auto-encoder model and data. An effective multiple-layer auto-encoder model would naturally lead to a hierarchical representation, where more common features are discovered first and then other less common features will be built on the features. Generally, since more complex relationships such as semantic relationships are less common, they are developed in higher layers. This is fundamentally different from the patterns in ELMo [5], where the local dependencies need to form first before longer dependencies.

VII. Conclusion and Future Work

In this paper, we have analyzed how the BERT architecture and its pre-training protocol affect its embedding vectors and its induced feature vectors. Due to its similarity to the architecture of the skip-gram and CBOW models, the geometric properties of BERT embeddings are similar in many ways to the vectors given by Word2Vec [8]. Our systematic results show the induced features do not lead to clusters for emotion classification in the original embedding space. However, they are effective for classification when temporal dependencies are modeled explicitly (such as BiLSTM). More importantly, the results show that the BERT models do not produce "effective" contextualized representations for words and their improved performance may mainly be due to fine-tuning or classifiers that model the dependencies explicitly by encoding syntactic patterns in the training data. This is being investigated further.

References

[1] Y. Bengio, R. Ducharme *et al.*, "A neural probabilistic language model," in *J. Mach. Learn. Res.*, 2003.

[2] T. Mikolov, K. Chen *et al.*, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[3] J. Devlin, M.-W. Chang *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[4] Y. Liu, M. Ott *et al.*, "Roberta: A robustly optimized bert pretraining approach," in *arXiv:1907.11692*, 2019.

[5] M. Peters, M. Neumann *et al.*, "Deep contextualized word representations," in *NAACL-HLT*, 2018.

[6] A. Vaswani, N. Shazeer *et al.*, "Attention is all you need," in *NIPS*, 2017.

[7] Y. Wu, M. Schuster *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," in *arXiv:1609.08144*, 2016.

[8] T. Mikolov, I. Sutskever *et al.*, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013.

[9] T. Wolf, L. Debut *et al.*, "Huggingface's transformers: State-of-the-art natural language processing," in *ArXiv:1910.03771*, 2019.

[10] M. Mahoney. (2006) Text8 dataset. [Online]. Available: http://mattmahoney.net/dc/textdata.html

[11] K. Ethayarajh, "How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings," in *EMNLP/IJCNLP*, 2019.

[12] X. Liu, A. Srivastava, and K. A. Gallivan, "Optimal linear representations of images for object recognition," in *CVPR*, 2003.

[13] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," in *J. Comput. Appl. Math.*, 1987.

[14] K. H. Brodersen, C. S. Ong *et al.*, "The balanced accuracy and its posterior distribution," in *ICPR*, 2010.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv:1412.6980*, 2014.

[16] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *arXiv:1711.05101*, 2017.

[17] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," in *Philosophical Magazine*, 1901.

[18] T. Mikolov, M. Karafiát *et al.*, "Recurrent neural network based language model," in *INTERSPEECH*, 2010.

[19] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," in *Neural computation*, 1997.

[21] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *INTERSPEECH*, 2012.

[22] A. Radford, "Improving language understanding by generative pre-training," in *preprint*, 2018.

[23] A. Wang, A. Singh *et al.*, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *ICLR*, 2019.

[24] E. Voita, D. Talbot *et al.*, "Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned," in *arXiv:1905.09418*, 2019.

[25] A. Ettinger, "What bert is not: Lessons from a new suite of psycholinguistic diagnostics for language models," in *arXiv:1907.13528*, 2019.

[26] Y. Goldberg, "Assessing bert's syntactic abilities," in *arXiv:1901.05287*, 2019.

[27] T. Niven and H.-Y. Kao, "Probing neural network comprehension of natural language arguments," in *arXiv:1907.07355*, 2019.

[28] J. Du, F. Qi, and M. Sun, "Using bert for word sense disambiguation," in *arXiv:1909.08358*, 2019.

[29] K. Huang, J. Altosaar, and R. Ranganath, "Clinicalbert: Modeling clinical notes and predicting hospital readmission," in *arXiv:1904.05342*.

[30] K. Clark, U. Khandelwal *et al.*, "What does bert look at? an analysis of bert's attention," in *arXiv:1906.04341*, 2019.

[31] E. Voita, R. Sennrich, and I. Titov, "The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives," in *arXiv:1909.01380*, 2019.

[32] A. M. Schakel and B. J. Wilson, "Measuring word significance using distributed representations of words," in *arXiv:1508.02297*, 2015.

[33] E. Alsentzer, J. R. Murphy *et al.*, "Publicly available clinical bert embeddings," in *arXiv:1904.03323*, 2019.

[34] I. Tenney, D. Das, and E. Pavlick, "Bert rediscovers the classical nlp pipeline," in *arXiv:1905.05950*, 2019.