Journal of Data Acquisition and Processing
ISSN: 1004-9037
https://sjcjycl.cn/
DOI: 10.5281/zenodo.7740041

BERT TRANSFORMER MODELS: COMPUTATIONAL ANALYSIS FOR INFORMATION EXTRACTION FROM BIOMEDICAL LITERATURE

G. Sasipriya
Department of Computer Science, University of Madras, Guindy, Chennai
msasipriyam@yahoo.co.in

B. Lavanya
Department of Computer Science, University of Madras, Guindy, Chennai
lavanmu@gmail.com

Abstract
In the past decade, information extraction played an essential role in bioinformatics, medical analysis, and the healthcare system, and it's still in the budding stage for nanomaterials. The crucial step in the information extraction process is extracting and analyzing unstructured text documents. The use of transformer models for this task has been rapidly growing in the field of nanomaterials. To facilitate our work, this paper proposes the fundamental task of relation extraction manually and outlines the step-by-step characteristics of a pre-trained Bert Model. BERT (Bi-Directional Encoder Representation from Transformers) symbolized the use of pre-train to comprehend the language and fine-tune it to learn a particular task. The training phase is usually composed of the next sentence prediction model and the MLM (Masked Language Model). The various relation extraction BERT algorithms that have been applied in recent articles are examined in this paper. This article analyzed the data for nanomaterial existing documents, for information extraction, by using BERT models like SciBert, MatSciBert, and BioBert. This paper identifies the relations present in the sentence and labels the entities, using the NER (named entity recognition) model. Finally, use the most recent and cutting-edge deep learning techniques as baseline models and carry out extensive experiments using MatSciBert, which yielded a good results accuracy of 72.79%. In essence, this paper provides detailed information on successfully customizing deep learning for healthcare applications. According to the comprehensive findings from the models, the article has shown that relation extraction in Material Bert gives better accuracy compared to other BERT models.
Keywords — Nanoinformatics, MLM, Relation Extraction, BERT;

I. INTRODUCTION

Named Entity Recognition (NER) and Relation Extraction (RE) are the two conceptual tasks for deep learning in NLP (natural language processing). The process of determining the connections that exist between the many entities discussed in a sentence is known as relation extraction. Consider the following sentence as an example, Minimum pore size required to enable ingrowth of along with the blood supply surrounding the bone is about 100–150 mm for macropores. Here pore size and 100-150 mm are considered an entity and "along with the blood supply surrounding the bone" is the relation between the sentences. Recently increasing

number of pieces of literature, state of art neural network models for both the NER and Relation Extraction task. Natural language processing studies have revealed that recently relation extraction has evolved around neural models that make substantial use of pretraining-based language modeling [1]. From sentence categorization to sequence prediction, BERT had shown significant improvement throughout a broad spectrum of natural language activities [2].
Researchers can use the pre-trained models in relation extraction and semantic labeling state of art methods without syntactic features, but for biomedical, Nanoinformatics domains we must use syntactic or lexical features. We show that simple deep neural BERT achieves state-of-art relation extraction in the Nanoinformatics domain. The following sections of this paper will explain the Encoding and Decoding with attention mechanism for relation extraction, and pretraining and fine-tuning methods [3] [4].

II. NAMED ENTITY RECOGNITION IN BERT

NER is the most standard for preprocessing tasks, and predefined categories for the key information like the person, location, organization, date/time, etc., And detects the entities from the text documents and it can classify them into different categories. For example, Sundar Pichai, the CEO of Google Inc. is walking in the streets of California. In this example, we can classify as [Person: Sundar Pichai] [org: Google Inc.], [location: California]. NER machine and deep learning algorithms are mainly focused on Rule-based Parsing, Dictionary-based methods, POS Tagging methods, and Dependency Parsing for extracting the entities. Given that n Entities can be represented in a relation R, it can be assigned as R (e1, e2, e3, ...en). For example, the nanomaterial sentence will be represented as "*The presence of calcium phosphate in bone takes the shape of needle-like crystals that are nanometer-sized and measure roughly 5-20 nm in width and 60 nm in length, with a poorly crystallized."* In this sentence calcium phosphate, nanometer-sized needle, crystal, approximately 5-20 nm width by 60nm, crystallized are referred to as entities.

III. RELATION EXTRACTION IN BERT

Bidirectional Encoder Representations from Transformers, or BERT, was developed by Google with the aim of natural language processing (NLP) pre-training. BERT is a methodology for machine learning that is built on transformers. Jacob Devlin and the other Google employees he works with are the ones who are accountable for the production and release of BERT in 2018.

A. BERT types

The following table 1 and 2 survey represents BERT architecture usage in the Biomedical, Materials [5], and Mathematics domains and for various languages like Arabic, Dutch, and French [6].

B. Understanding the Transformers

Recently many researchers in the field of text mining have heard about BERT, GPT-3, RoBERT, ALBERT, Bio Bert, XLNet, and so on. The sheer number of new models appears to be enormous, we understand the Transformer architecture, and want to learn the inner workings of all these models. Before Bert, the text mining field exists with Continuous word embeddings

that were developed to better classify the semantic meaning and similarity of words. The sequence-to-sequence approach of the deep neural network is being used to extract semantic and syntactic information from large amounts of unstructured text [3]. Transformer models handle the encoding of the sentences. Pre-trained models handle the complexity of self-supervised learning, the mask of words, we can use a large dataset train to predict the masked words, that way the model can learn. One of the most valuable aspects that deep learning needs to work on word representation, it promotes the same. The embedding layer is another essential critical feature that is utilized in the process of compressing the input feature space into a more reasonable size.

IV. BERT ARCHITECTURE

The model of word representation that is contextualized BERT was developed using bidirectional transformers in its training and is based on a masked language model [2] [4]. BERT can be found in the paper of Devlin. Due to the inherent structure of language modeling, in which it is hard to predict what words will come next, earlier linguistic systems were restricted to a combination of only two language paradigms that only go in one direction. BERT makes use of a model of masked language that predicts the appearance of randomly masked words in a string. This enables it to be utilized for the purpose of learning representations in both directions, which is useful in many applications.

TABLE I: BERT models in the Biomedical and Material Science domain

S.No	Name	Purpose	URL
1.	BioBERT [7]	Biomedical and clinical text	https://github.com/dmis-lab/biobert-pytorch
2.	SciBERT [8]	Biomedical text	https://github.com/allenai/SciSpaCy
3.	BioALBERT [9]	Biomedical text	https://github.com/usmaann/BioALBERT
4.	MATBERT [10]	Material text	https://github.com/lbnlp/MatBERT
5.	PUBMEDBERT[11]	Biomedical Abstract	https://github.com/ncbi-nlp/bluebert
6.	MatSciBERT [12]	Material text	https://github.com/M3RG-IITD/MatSciBERT

TABLE II: BERT models in some other domains

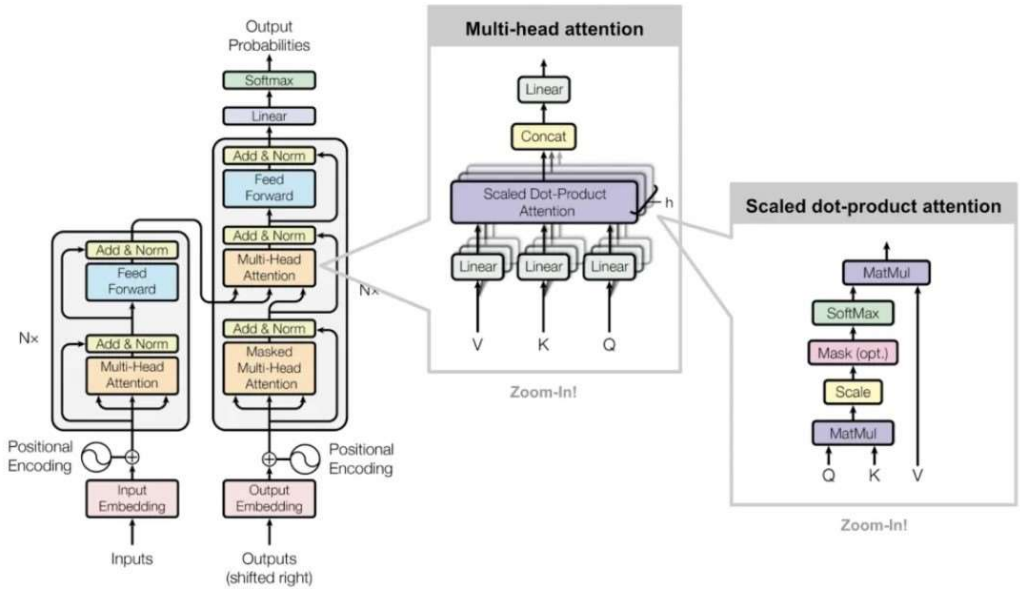
S.No	Name	Purpose	URL
1.	AraBERT [13]	Arabic text	https://github.com/aub-mind/arabert
2.	RoBERT [14]	Common text more than the original BERT model	https://aka.ms/BLURB
3.	AlphaBERT [15]	BERT in corpus level	————

4.	BERTje [16]	Dutch Language DistBERT	https://github.com/wietsedv/bertje
5.	DistBERT [17]	Distilled Compression technique	————-
6.	MobileBERT [18]	Compressing the original BERT model	https://github.com/zaidalyafeai/mobilebert
7.	FlaceBERT [19]	French Corpus	https://github.com/getalp/Flaubert
8.	SqueezeBERT [20]	self-attention layers with grouped convolutions	https://github.com/topics/squeezebert
9.	CamemBERT [21]	monolingual languages	https://github.com/huggingface/transformers
10.	MacBERT [22]	Improved pretraining and fine-tuning	https://github.com/ymcui/MacBERT
11.	Electra [23]	Replaces tokens instead of masking input.	https://github.com/google-research/electra
12.	MathBERT [24]	NLP Tasks in Mathematics Education	https://github.com/tbs17/MathBERT

Furthermore, it achieves cutting-edge performance on many NLP activities while requiring only a modest amount of architectural modification that is task specific. The creators of BERT believe that it is essential to incorporate information obtained from representations that go in both directions rather than information obtained from representations that only go in one direction when it comes to representing words in natural language [25]. The following figure 1 shows the BERT architecture representation. The mechanism that lets the attention-scaled dot product utilize the context is as following.

1. It uses the scalar product to determine how much the input embedding vectors are related to each other.
2. Results are scaled down, and an activation function is used to normalize the result in a nonlinear fashion.
3. Finally, a new contextualized, each token is given an embedding by linearly combining all concerning the input embeddings with SoftMax prepositions as coefficients.

Fig. 1: BERT Architecture



A. Multi-head attention

In the architecture of BERT, attention transforms the default embedding by analyzing the whole sequence of tokens, so that the token represents the context of the sentence. Each token is initially replaced by its default embedding, which in this case is a vector with 768 components (768 in the case of BERT Base and 1024 for BERT Large). We can start by calculating the scalar product between the pairs of embeddings.

When two or more vectors are correlated or aligned, meaning that are generally more similar, the scalar product is higher so we can consider that they have a strong relationship. We apply the SoftMax column by column, what the SoftMax does is exponential, amplifying large values, while low and negative values to zero. It also does normalization, so each column sums up to 1.

In BERT, the updated embedding is distributed to each head that is a part of the Multi-Head Attention Layer, after calculating the positional encoding that has a dimension of n x 512, where n is the number of tokens in the sequence and produces an output that has a shape that is n x 64. This occurs after the calculation of the positional encoding that is n x 512, the number of tokens in the sequence, denoted by n. After that, the outputs from all the heads are joined together into one form, a single output coming from the module that handles multi-headed attention that has dimensions of n x 512.

Each projection can be thought of as focusing on distinct directions of the vector space, which would represent different semantic characteristics. This would be the case because different directions of the vector space have different meanings. One can conceptualize of a key as the projection of an embedding onto the direction of prepositions, and a query as the projection of an embedding along with the directions of locations. Both concepts are imaginable. Values can come from yet another projection that is relevant, for example, the direction of physical places.

Example Sentence: *The nature of the powder particles is polycrystalline, and their average size is from 5 to 90 nanometers.*

Word vectors = 'The' 'nature' 'of' 'the' 'powder' 'particles' 'is' 'polycrystalline' 'and' 'their' 'average' 'size' 'is' 'from' '5 - 90' 'nanometers'. These vectors are represented as $y = \sum w1x1+w2x2+w3x3+... +w16x16$. Calculating weights for a given sentence is $z_1 = x^t{}_1 x_{16}$. Here x is the token. And it can be written as $[w_1, w_2, w_3, ... w_{16}] = SoftMax\ (z_1, z_2, z_3, ... z_{16})$ Self-attention in our example "The nature of the powder particles is polycrystalline, and their average size is from 5 to 90 nanometers." The tokens 'powder', 'particle', 'polycrystalline', 'average', 'size', and'5-90 nm'. So, we have to calculate a new vector for these words. After finding self-attention, we must calculate the vector description. Let n represents num of words in the sentence, $d_i = length\ of\ x_i\ \forall i \in 1,2,3,...,n$.

The weights are computed for query, key, and value are *Query:* $q_i = W_q X_i$ *Keys*: $K_i = W_k X_i$ *Values*: $V_i = W_v X_i$ But self-attention will not compute the weight of input word embedding corresponding value vectors V_i. $\forall i, j \in 1,2,3,...,n$.

$$Z_{ji} = \frac{k_i^t q_i}{\sqrt{\{d\}}} \quad For\ all\ i \in \{1, 2, 3, \ldots, n\}$$

Weights: $[w_{1,i}, w_{2,i}, ..., w_{n,i}]$ = softmax $(z_{1,i}, z_{2,i}, ..., z_{n,i})$ New embedding:

$$y_i = \sum_{j=1}^{n} v_j w_{ij} \qquad (1)$$

For matrix representation the self-attention weights and averages using column-wise softmax as Weights(W)=softmax(z) New embedding: y=vw

$$y_i = \begin{bmatrix} w_{1i} \\ w_{2i} \\ \cdots \\ w_{ni} \end{bmatrix} = \sum_{j=1}^{n} v_j w_{ij} \qquad (2)$$

B. Encoder Overview

The encoder bings N encoder blocks. Same structure but it has different parameters. Each encoder n block maintains the shape of vectors and the vectors' length. The number of input and output vectors never changes. Briefly explain all the components and input embedding with positional encoding. Multi-head self-attention, add and normalize, and feedforward network [26].

Input embedding and positional encoding

Similarly, to self-attention entire encode blocks take the set of vectors as input and another set as output. Encoder blocks map sets to sets. However, word order is not irrelevant. Because there are several ways to encode the position in the vector. For example, input: she is angry? Words in vocabulary: she is angry? One hot encoding: t_1, t_2, t_3, t_4 . t_1 will be zero for every position t_1 at element 123.

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \end{bmatrix} \qquad (3)$$

The positional one is hot: P 1 is represented in the matrix as (1 0 ...), P _2 is represented in the matrix as (0 1 ...), P 3 is represented in the matrix as (0 0 ...), input embedding (vectors), i= 1,2,3,4, $X_i = Et_i$(*word embedding*)+ PP_i(*position embedding*) Where, but dimensional D, fails in a large number network could select E and P, so the position is one dimensional and the word in another dimension. E: d *size of the vocabulary P: d* max length of the sequence.
The Input Embedding layer facilitates the transition of meaningful 512-dimensional embeddings for each token. Hence in our example, we have n tokens so the representation of the input embedding layer would be n * 512, related words calculated like this and vice versa. Adding the residual connection, can yield stronger gradients and also helps remember positional encoding. By normalization, we can reduce covariate and the training was very fast, which helps the attention layers.
After this applying a fully connected network to each word embedding, have the same in and out dimensions, and weights can be shared among all words. The input and output in every single layer in every ad encoder block as the same dimensions as the matrix received as the input to the first encoded block.

V. DECODER OVERVIEW

The Decoder stacks N decoder blocks. Same structure but it has different parameters. Each decoder n block maintains the shape of vectors and the vectors' length. The number of vectors as well as the length of the pattern of the output are both in our possession. Every token from the pattern of the input sequence is attention aware. Every token of the output sequence is now predicted. It has been established that the new token that has been predicted will be the next token.

The decoder could guess the next word in the sequence based on the previous tokens and the attention of the encoder. This is because it has been trained to do so. Because there have been no tokens before this one, this is a problem for the first token. Because of this, it would have been challenging to always guess the initial token correctly. As a direct consequence of this, the output sequence gets altered, and a Beginning of Sentence marker is added at the beginning. When it is necessary for us to make a prediction regarding the initial token, we make use of this BOS as the token that came before it in the output sequence.

Like the Encoder layer in both function and structure, the output embedding layer and the positional embedding layer is responsible in order to same task. The core of the decoder is somewhat different from the core of the encoder because of the inclusion of Masked Multi-Head Attention; nonetheless, like the encoder's core, the decoder's core is repeated for a total of 6 iterations. The Masked Multi-Head Attention Layer only pays attention to tokens that have been processed up to the current position. Attention is not paid to tokens that will be processed in the future. This runs completely counter to the Encoder, which determines attention for the entire sequence all at once.

A. Decoder overview

Immediately following the Normalization layer comes the layer for multi-headed attention, which is responsible for taking in the output regarding the encoder (n x d model; don't forget

that this is the encoder's final output!). This output is referred to as K (key) and V (Value), and by the attention focused on multiple heads of the Decoder; additionally, the Query matrix that was taken from the layer that came before the concentration with multiple heads covered Up layer is utilized. As a result, because it uses values determined by pretraining for the Query(Q), Key(k), and Value(V) matrices, this attention layer does not require any training. This layer employs the Encoder's pre-trained data. This Multi-Head Attention Layer, with masking for multiple heads, will be unable to look further than what isn't anticipated in the output sequence, because the Query vector will only be available for foreseen tokens.

- Following these iterations of the same code blocks, there will be a linear layer, then a SoftMax function that will calculate the likelihood that the token will appear in the sequence that was predicted.
- Following the successful prediction of the most likely token, that token is then placed at the end of the output sequence.

B. Add and Normalize

Adding the residual connection yields a stronger gradient and remembers positional encoding. by normalization, we can reduce covariate shift (faster training). It centers embedding around the origin, which helps attention layers (compared to our products).

C. Feed Forward network

Applies a fully connected network to each word embedding. Embedding has the same in and out dimensions (true for all layers in the encoders). Weights are shared among all words. The specific form of the network is a linear layer. Inside each encode block we have a multi-head attention block followed by adding and normalizing and feeding the forward network. The input and output in every single layer in every encoder block as the same dimensions as the matrix received as the input to the first encoded block.

VI. PRETRAINING

The BERT loss function ignores non-masked word prediction and solely takes into account masked value prediction. Because of this, the model converges at a pace that is significantly slower than that of directional models; however, this is compensated for by the example given improved semantics.

A. Prediction for the Next Sentence (NSP)

During the training phase of BERT, the model is provided with sentence pairs to analyze, and it learns to determine whether the second sentence in each pair represents the next sentence in the original document by determining whether or not the first sentence in each pair is the previous sentence. During the process of training, half of the inputs are paired, with the second sentence being the sentence that comes after the first in the original text. This is done so that the system can learn the proper order of sentences. The remaining portion of the inputs is chosen at random from the corpus. It is presumed that the random sentence has no connection to the sentence that came before it.

Before the input is given to the model, it is processed in the following manner so that it can better aid the model in distinguishing between the two sentences while it is being trained: 1. A [CLS] token has been placed at the beginning of the very first sentence, and a [SEP] token has been placed at the very end of each and every sentence that has followed it. 2. Every token now

has an embedded sentence that specifies whether belongs in Sentence A or Sentence B. Conceptually, token embeddings with a vocabulary size of 2 are quite like sentences. 3. A positional embedding is given to each of the characters to specify where in the essence of it falls. In the transformer paper, both the notion of positional embedding and its implementation are discussed in detail.

To determine whether the second statement is related to the first, the following steps need to be carried out: 1. The Transformer model is used continuously all the way through the input sequence. 2. Using a straightforward knowledge outcome, the output of the [CLS] token is reshaped into a 21-shaped vector. 3. Using SoftMax to compute the likelihood of IsNextSequence.

In the BERT model, Masked LM and Next Sentence Prediction are trained concurrently to minimize the total loss function associated with the application of the two methods. This is done to reduce the overall loss. In the masked language model, BERT starts with a sentence that is filled with masked words that are chosen at random. In this example sentence, *the calcium phosphate that is found in bone is in the form of needle-like crystals that are nanometers in size and measure roughly 5-20 nm in width and 60 nm in length. These crystals are weakly crystallized.* mask 1=" needle-like crystals" mask 2= "5-20 nm width by 60 nm". This kind of masking will help us to understand the bidirectional context.

VII. FINE-TUNING

Even though the existing model might be sufficient for our issue. For the following two reasons, it is frequently better to adjust the pre-trained model: First, to reach an even higher level of precision. Second, our well-calibrated model can create results in the appropriate format. The lower and middle layers of a neural network are responsible for representing general characteristics, meanwhile, the higher levels oversee handling problem-specific characteristics. In general, the bottom and mid-level layers represent the features. Because of a unique challenge from the one we were originally trying to solve; we frequently eliminate the top layers. By adding layers that are unique to our issues, we can get a higher level of accuracy. To accomplish what we set out to do, we must first remove the layers that are on top and then add our own.

VIII. RESULT AND DISCUSSION

The three BERT algorithms related to material and science domains like BioBERT, SciBERT, and MatBERT are analyzed by using material sentences, and the results are described.

A. BioBERT

BioBERT is a representation model that has been pre-trained for biomedical language and biomedical text mining [7]. In this article, the domain-specific language presentation model known as BioBERT (Bidirectional Encoder Representations from Transformers for Biomedical Text Mining) is discussed. The BioBERT model is one that has already undergone preliminary training on a sizeable healthcare corpus. They performed three essential tasks in this paper as follows: named entity recognition, question-answering system, and relation extraction, the is a state-of-an-art method in biomedical text mining. The pretraining of biobert corpus has been

trained by English Wikipedia, Books Corpus, PubMed Abstracts, and PubMed Central. They worked on different models like BERT, biobert with PubMed, biobert with PubMed Central, and BioBert with PubMed and PubMed central. Fine-tuning BioBERT by the same tasks like Relation extraction, Named entity recognition, and question answering. The fundamental task has been evaluated by the algorithm, LSTM, and CRF and evaluated by the precision, recall, and F1 score and 0.62% enhancement in named entity recognition. They used [CLS] for sentence classification of relations and they target the entities of genes and diseases and achieved an f1 score enhanced as 2.80%. They used the BioASQ factoid datasets SQuAD for the question-answering system and achieved an enhancement of the F1 score of 12.24%.

B. SciBERT

SciBERT is a BERT-based pre-trained language model for performing scientific natural language processing (NLP) tasks [8]. Because SciBERT's architecture is based on the BERT model, if you are not familiar with the cutting-edge base model, you should read the BERT research article. The basic BERT model is formed on two tasks: 1. Predict masked tokens randomly. 2. Predict if two sentences follow each other. SciBERT follows the same architectural model as BERT; the only dissimilarity is that scibert is trained on scientific data instead.

The 793 PubMed abstracts with annotated disease entities that make up the used data are part of a collection. Each token entity has a "B-" (Beginning) tag that indicates whether the token is at the beginning of the entity or an "I-" (Inside) tag that indicates whether the token is inside the annotation, while the "O" tag implies that the token is not an entity recognition. In scibert, they used a dataset named entity recognition BC5CDR with 90.1 % accuracy, JNLPBA with 77.28 % accuracy, and NCBI-disease with 88.57 % accuracy. For relation extraction, they used the chemprot dataset and achieved 83.64 %. SCIBERT has been trained using the entire text of Computer science and biomedical papers totaling 1.14M. from the Semantic Scholar corpus. SCIBERT additionally employs an in-domain. whereas the other models mentioned above make use of the original BERT vocabulary (BASEVOCAB), and SCIVOCAB.

C. MatSciBERT

Text mining and information extraction may be accomplished through the utilization of a materials domain language model known as MatSciBERT [12]. We demonstrate that MatSciBERT is an improved version of SciBERT, a language model that was educated using a collection of scientific texts, and we exhibit cutting-edge outcomes on three subsequent tasks including abstract classification, named entity recognition, and relation classification. 1. NER on SOFC, Friedrich, et al.'s SOFC Slot, and the Matscholar dataset. 2. Glass Abstracts of papers that are not covered by glass10. 3. Relation Classification on the MSPT Corpus 46.

About extraction, the dataset consists of synthesis The 230 synthesis procedures in this dataset have been annotated as graphs, where the nodes stand in for the individuals who took part in each step of the synthesis and the edges signify the connections among the nodes. Nine sentences are typically used in a synthesis procedure, and each sentence contains an average of 26 tokens. There are 16 relation labels in the dataset. The authors divided the relation labels into the following three groups:

1. Target for the recipe, ingredients for the solvent and atmosphere, ingredients for the participants, equipment for the recipe, and conditions for the recipe. 2. Relationships between

Non-Operation Entities include Descriptor of, Number of, Amount of, Apparatus-attr-of, Brand of, Core of, Property of, and Type of [12]. 3. Operations are related to one another in the following operation. are done using the synthesis procedures dataset, and examined the results with the models MaxPool and MaxAtt which are considered as the baseline model. MatSciBERT performs better on the test set than SciBERT by more than 2.75% accuracy.

D. Discussion

This paper proposes theanalysis of three essential BERT models for biomedical text BioBERT, SciBERT, and MatSciBERT. Figure 2 and table 3 represent the precision, recall, and F1 scores. The following sections go into detail about the experimental findings, analyses, and comparisons.

The Bio-BERT base was used, which mentions the BioBERT models trained on Pubmed for 1 M steps (training time) as the pertained model. The process was carried out using Pytorch hugging face implementation fine-tuning of the model for the nanomaterial dataset. The detailed investigation of experimental analyses was done by using training data and test data.

TABLE III: Evaluation Metrics of three BERT models

Models	Precision	Recall	F1-Score	Accuracy
BioBERT	0.4972	0.4556	0.4414	0.5794
SciBERT	0.6919	0.5093	0.5867	0.6534
MatSciBERT	0.7031	0.6852	0.6751	0.7279

All the results are based on models trained using nanomaterials properties separately. For tuning the data, the train, test, and dev data are used in the ratio of 70:20:10. The relation extraction training is done with 5 epochs, batch sizes are 2,4,8, and 16, and the learning rate of 0.001. The model is customized using the Hugging face BERT API and BioBERT model as the base model pre-trained using Pubmed vocabulary. Each of the models has been given prior trained and fine-tuned with the BioBERT model.

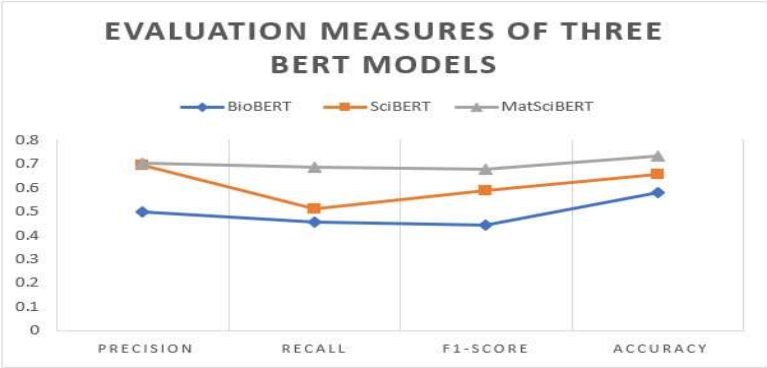


Fig. 2: Evaluation Metrics of three BERT models

Like any NLP framework, SciBERT also requires training with labeled data. The basic BERT model uses masked tokens to predict if two sentences go together or not. SciBERT also works the same way, but the models are trained using scientific data. For training PyTorch AllenNLP-based pre-trained model is used. For tuning the data, the train, test, and dev data are used in the ratio of 70:20:10. The relation extraction training is done with 5 epochs, batch sizes are

2,4,8,16, and 32, and the learning rate of 0.001. The training for relation extraction is worked for a predefined set of data with a predefined set of labels. This can be customized later for a new set of labels in the future as the changes in new datasets.

In MatSciBert, the model for relation extraction is done using the predefined model matscibert, which is specialized for materials-specific scientific data. The text data used here is normalized using text normalization. The same ratio of 70:20:10 is used here also for train, test, and dev data. The model used is uncased. The same set of batch sizes 2,4,8,16,32 and epochs 5 and a learning rate of 0.001 is used. The text data used here is normalized using text normalization, which uses vocabulary mapping. The same ratio of 70:20:10 is used here also for train, test, and dev data. The model used is uncased.

IX. CONCLUSION

In this paper, the transformer model is explored manually to extract the relations between Nanoinformatics text documents from the literature. The research on nanoparticle text mining will benefit from our technique. It is possible to reduce time and money by extracting entity relations for nano informatics research articles using a model that is based on a transformer. The survey for relation extraction found that the transformer model BERT delivers a better outcome for neural networks, particularly on the topic of nano informatics. This was the conclusion reached by the researchers. That is, taking into consideration our findings, the transformer model can produce competitive advantage results with less annotation of input data and less support from external resources such as knowledge bases. BERT dictionary provides competitive advantages results with less annotation of input data and less support from external resources such as knowledge bases. According to the evaluation of three BERT Models BioBERT, SciBERT, and MatSciBERT, material science, BERT gives better performance compared to other models with an accuracy of 72.79%. Integration of NER and RE into a unified model with high precision and recall. the remains to be investigated.

X. ACKNOWLEDGMENT

This research was funded by DST-RUSA via the University of Madras as part of the RUSA 2.0 program.

REFERENCES

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

[2] 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.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[5] M. Yoshitake, F. Sato, H. Kawano, and H. Teraoka, "Materialbert for natural language processing of materials science texts," *Science and Technology of Advanced Materials: Methods*, vol. 2, no. 1, pp. 372– 380, 2022.

[6] A. H. Mohammed and A. H. Ali, "Survey of bert (bidirectional encoder representation transformer) types," in *Journal of Physics: Conference Series*, vol. 1963, p. 012173, IOP Publishing, 2021.

[7] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: a pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[8] I. Beltagy, K. Lo, and A. Cohan, "Scibert: A pre-trained language model for scientific text," *arXiv preprint arXiv:1903.10676*, 2019.

[9] U. Naseem, M. Khushi, V. Reddy, S. Rajendran, I. Razzak, and J. Kim, "Bioalbert: A simple and effective pre-trained language model for biomedical named entity recognition. arxiv 2020," *arXiv preprint arXiv:2009.09223*.

[10] N. Walker, A. Trewartha, H. Huo, S. Lee, K. Cruse, J. Dagdelen, A. Dunn, K. Persson, G. Ceder, and A. Jain, "The impact of domain-specific pre-training on named entity recognition tasks in materials science," *Available at SSRN 3950755*, 2021.

[11] Y. Gu, R. Tinn, H. Cheng, M. Lucas, N. Usuyama, X. Liu, T. Naumann, J. Gao, and H. Poon, "Domain-specific language model pretraining for biomedical natural language processing," *ACM Transactions on Computing for Healthcare (HEALTH)*, vol. 3, no. 1, pp. 1–23, 2021.

[12] T. Gupta, M. Zaki, N. Krishnan, *et al.*, "Matscibert: A materials domain language model for text mining and information extraction," *npj Computational Materials*, vol. 8, no. 1, pp. 1–11, 2022.

[13] W. Antoun, F. Baly, and H. Hajj, "Arabert: Transformer-based model for Arabic language understanding," *arXiv preprint arXiv:2003.00104*, 2020.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[15] Y.-P. Chen, Y.-Y. Chen, J.-J. Lin, C.-H. Huang, F. Lai, *et al.*, "Modified bidirectional encoder representations from transformers extractive summarization model for hospital information systems based on character-level tokens (alphabert): development and performance evaluation," *JMIR medical informatics*, vol. 8, no. 4, p. e17787, 2020.

[16] W. de Vries, A. van Cranenburgh, A. Bisazza, T. Caselli, G. van Noord, and M. Nissim, "Bertje: A dutch bert model," *arXiv preprint arXiv:1912.09582*, 2019.

[17] R. Dasgupta, F. Tom, S. Kumar, M. Das Gupta, Y. Kumar, B. N. Patro, and V. P. Namboodiri, "Visually precise query," in *Proceedings of the 28th ACM International Conference on Multimedia*, pp. 3550–3558, 2020.

[18] Z. Sun, H. Yu, X. Song, R. Liu, Y. Yang, and D. Zhou, "Mobilebert: a compact task-agnostic bert for resource-limited devices," *arXiv preprint arXiv:2004.02984*, 2020.

[19] H. Le, L. Vial, J. Frej, V. Segonne, M. Coavoux, B. Lecouteux, A. Allauzen, B. Crabbe, L. Besacier, and D. Schwab, "Flaubert:´ Unsupervised language model pre-training for french," *arXiv preprint arXiv:1912.05372*, 2019.

[20] F. N. Iandola, A. E. Shaw, R. Krishna, and K. W. Keutzer, "Squeezebert: What can computer vision teach nlp about efficient neural networks?," *arXiv preprint arXiv:2006.11316*, 2020.

[21] L. Martin, B. Muller, P. J. O. Suarez, Y. Dupont, L. Romary,´ E. V.´ de La Clergerie, D. Seddah, and B. Sagot, "Camembert: a tasty french language model," *arXiv preprint arXiv:1911.03894*, 2019.

[22] Y. Cui, W. Che, T. Liu, B. Qin, S. Wang, and G. Hu, "Revisiting pretrained models for chinese natural language processing," *arXiv preprint arXiv:2004.13922*, 2020.

[23] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pretraining text encoders as discriminators rather than generators," *arXiv preprint arXiv:2003.10555*, 2020.

[24] J. T. Shen, M. Yamashita, E. Prihar, N. Heffernan, X. Wu, B. Graff, and D. Lee, "Mathbert: A pre-trained language model for general nlp tasks in mathematics education," *arXiv preprint arXiv:2106.07340*, 2021.

[25] A. Rogers, O. Kovaleva, and A. Rumshisky, "A primer in bertology: What we know about how bert works," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 842–866, 2020.

[26] P. Xia, S. Wu, and B. Van Durme, "Which* bert? a survey organizing contextualized encoders," *arXiv preprint arXiv:2010.00854*, 2020.