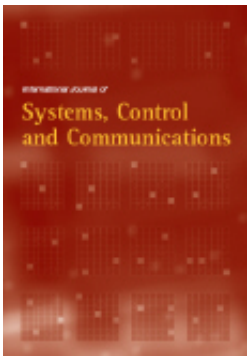

International Journal of Systems, Control and Communications

ISSN online: 1755-9359 - ISSN print: 1755-9340
https://www.inderscience.com/ijscc

Review and analysis for state-of-the-art NLP models

Subrit Dikshit, Rahul Dixit, Abhiram Shukla

DOI: 10.1504/IJSCC.2024.10060461

Article History:

Received:	17 May 2023
Last revised:	13 July 2023
Accepted:	15 July 2023
Published online:	01 December 2023

Review and analysis for state-of-the-art NLP models

Subrit Dikshit* and Rahul Dixit

Department of Computer Science and Engineering,
Indian Institute of Information Technology,
Pune, India
Email: subrit@gmail.com
Email: rahuldixit@iiitp.ac.in
*Corresponding author

Abhiram Shukla

Department of Civil Engineering,
Indian Institute of Technology,
Kanpur, India
Email: abhirams@iitk.ac.in

Abstract: Natural language processing (NLP) is an area of research and study that makes it possible for computers to comprehend human language by utilising software engineering concepts from computer science and artificial intelligence. This work presents seven classifications and 21 state-of-the-art models with a survey of the operating principles. We also provide a comparative study of all models based on metrics such as accuracy, F1 score, exact match, squad scores, glue and superglue dataset scores.

Keywords: artificial intelligence; AI; natural language processing; NLP.

Reference to this paper should be made as follows: Dikshit, S., Dixit, R. and Shukla, A. (2024) 'Review and analysis for state-of-the-art NLP models', *Int. J. Systems, Control and Communications*, Vol. 15, No. 1, pp.48–78.

Biographical notes: Subrit Dikshit is a PhD student at the Department of Computer Science and Engineering of Indian Institute of Information Technology, Pune, India. His research interests focus in the field of artificial intelligence, with specific emphasis on natural language processing. He currently serves as product manager at reputed organisation and has more than 18 years of information technology experience in data analytics and artificial intelligence.

Rahul Dixit is an Assistant Professor at the Department of Computer Science and Engineering of Indian Institute of Information Technology, Pune, India. He earned his PhD degree from National Institute of Technology, Rourkela, India. He received his MTech and BTech degrees in Computer Science and Engineering from Indian School of Mines Dhanbad (IIT Dhanbad) and Uttar Pradesh Technical University, respectively. His major areas of his research interest include digital image and video forensics, multimedia security, image processing, information retrieval and natural language processing, in which he has published books, reputed journals, book chapters and conferences.

Abhiram Shukla is a PhD student at the Department of Civil Engineering at the Indian Institute of Technology Kanpur, Kanpur, India. His research areas include the rheological properties of cementitious suspension, self-compacting concrete, environmental sustainability in the construction and durability of concrete structures. Prior to joining IIT Kanpur, He obtained his BTech from National Institute of Technology Raipur, Raipur, India.

1 Introduction

Artificial intelligence (or AI) enables machines to imitate human intellect. AI can be applied to many intellectual tasks (Russell and Norvig, 2003; Poria et al., 2017; Domingos, 2015; Crevier, 1993) and is pervasive (European Commission, 2020). With the help of current state-of-art AI models, information can be accessed quickly using optimisation (Russell and Norvig, 2003; Poole et al., 1998; Luger and Stubblefield, 2004; Nilsson, 1998; Burke and Kendall, 2013) and search techniques that make AI even surpass human intelligence in many modern-day use cases. AI offers cognitive abilities and perception to computerised machines by using input from sensors to deduce aspects of the world (Russell and Norvig, 2003; Nilsson, 1998).

Natural language processing (NLP) is an area of research and study that enables computers to read, interpret and realise a sense of data (structured/unstructured) and comprehend the intent and context of words used. NLP helps to precisely derive insights and information from documents and classify and arrange them (Lenat and Guha, 1989; Cambria and White, 2014).

In this paper, we provide and detailed survey by creating a readable synthesis of the best resources available in the NLP area of research. We succinctly progress and summarise the current state of knowledge, interpret prior research and reveal the gaps in the literature. We present our understanding by classifying the models and then converse comparative novel NLP architectures, performance, and different hyper-parameter configurations for each state-of-art NLP model. Today's scenario is that in each walk of life, from office, school, and home to war spaces, everywhere, it is being managed and controlled by AI and NLP where the manual approach is impossible to comprehend and implement. Roots of NLP can be found around the 1940s, and so much has been done in this field. However, there still lies a gap, and even after referring to the best possible sources and literature available online and offline, it is very challenging to summarise and understand the state-of-the-art situations in the field, which lies as the key motivation for this study and review that we present in this paper.

We have divided the article into five sections. Section 2 contains the literature review. Section 3 discusses various classifications and reviews of models. Section 4 talks about the metrics used. Section 5 is about experimental results and comprehensive analysis with comparison. At the close, we deliver our inferences in Section 6, while references are documented in Section 7.

2 Literature review

In the previous decades, much attention to NLP has been prescribed. This paper deals with classifications of NLP methods – generalised linear method, neural networks (NNs), recurrent neural networks (RNNs), transformers (Vaswani et al., 2017), autoencoding (AE) transformers, autoregressive (AR) transformers, and human-level performance. However, before discussing these models in detail, let us summarise the theoretical background required for this discussion.

- 1940–1950: Famous researchers Warren McCullock and Walter Pitts published McCullock-Pitts (MCP) neurons in 1943. An artificial neuron is a scientific method in which each neuron receives input, evaluates it separately, adds it up, and then passes the result of this process through a nonlinear function to produce inference.
- 1950–1960: One of the critical research developments in the 1950s was towards generalised linear models (GLMs) (Nelder and Wedderburn, 1972) about various statistical models such as – linear regression, logistic regression (Chao et al., 2022), and Poisson regression. The logistic (or logit) regression estimates the probability of the dependent parameters of a given event occurring by combining its log odds with one or more independent variables in a linear manner. However, this approach to NLP tasks containing large corpus results in various erroneous matches. Frank Rosenblatt introduced the perceptron in 1957, based on the innovative MCP neuron.
- 1980–1990: RNN (Abiodun et al., 2018) – RNN creates directed or undirected connections between nodes graph in the order of time. RNN process input sequences with varied lengths using inner state (remembrance). However, this procedure works differently because of slow computation and difficulties faced during training, such as exploding or gradient vanishing. Encoder-decoder (Seq-2-Seq or sequence-to-sequence) models use together fragments of the Transformer architecture. The attention layers of the encoder can access every word in the first sentence at every step. Still, the attention layers of the decoder can only access the input words that come before a specific term.
- 1990–2000: Long short-term memory (LSTM) (Sak et al., 2014) is a category of RNN that became the most significant mentioned NN in the 20h century. A general LSTM unit comprises a forget gate, a cell, an input, and an output gate.
- 2000–2010: Transformer employs the self-attention method, differently weighing each input's point significance. Attention is modelled using the cognitive attention technique, which makes some input data visible better while making supplementary sections inferior.
- 2010–2020: Transformers were made public in 2017 by a group at Google Brain in a paper by Vaswani and have developed the most amazing broadly utilised architecture in the NLP problems.

  AE transformers are a category of transformer network used to teach unlabelled data efficient coding (unsupervised learning). The encoding is authenticated and enhanced through an initiative to revive the input from the encoding. BERT (Devlin et al., 2018), bidirectional encoder representations from transformers is centred around transformers. Google NLP created a pre-training program that was released

in 2018. BERT become a worldwide baseline in NLP experiments. Google unveiled their model ALBERT (Lan et al., 2019), a Light BERT, in 2019 to prevail GPU/TPU memory restriction issues and stretched training periods by parameter reduction techniques. DistilBERT (Sanh et al., 2019), distilled BERT-like model, follows the process of distillation of coaching a small student model to nearly bear a semblance to a bigger teacher model. RoBERTa (Liu et al., 2019), can perform as well as or better than any post-BERT approaches. T5 (Raffel et al., 2020), text-to-text transfer transformer practices a text-to-text style. Each test is designed so that the model text is fed as input and trained to produce some target text, enabling its application across our varied activities.

ARs are decoder-based models which create the value at the present moment step through using data from the decoder's prior time steps. A mathematical system is AR if it forecasts forthcoming output upon historical values. BART (Lewis et al., 2020) model is an autoencoder that denoises data for sequence-to-sequence. BART is prepared and taught by tampering with text using various noise-reduction techniques and then developing a model to restore the original content. BART works well for knowledge tasks as well as the generation of text, where it is most effective. GPT (Radford et al., 2018, 2019; Brown et al., 2020) (or the generative model) – can pre-train on a diversified corpus with lengthy stretches of continuous text, enabling it to learn global knowledge and understand long-range dependencies. Generative pretrained transformer 2 (GPT-2) – is the descendant of GPT and an unsupervised transformer linguistic model. GPT-2 was initially revealed in February 2019, with the public only seeing a few demonstrator copies. In November 2019, the complete version was released. Transformer-XL (Dai et al., 2019) (extra-long) brings the idea of repetition to the profound self-attention system through a Transformer design. Transformer-XL styles take advantage of concealed conditions collected in former pieces relative to computing the concealed conditions from nothing with every new occurrence. Transformer-XL employs a novel comparative positional encoding design that extends beyond the training attention length to more considerable attention lengths. XLNet (Yang et al., 2019) uses the most remarkable features of AR language modelling and AE while trying to remove the restrictions. XLNet optimises the estimated log-likelihood of a sequence concerning all potential factoring order arrangements.

- Beyond 2020s: XLM (Lample and Conneau, 2019) shows the efficiency of NLP in multiple languages or cross-lingual pretraining. It is used in one of three language modelling objectives – causal language modelling (CLM), translation language modelling (TLM), and masked language modelling (MLM). Longformer (Beltagy et al., 2020), is based on the transformer model to process and execute lengthy sequences. It is an attention instrument that linearly scales along phrase length, allowing it effortless to procedure artefacts of thousands of tokens or even more prolonged. LUKE (Yamada et al., 2020) stands for language understanding with knowledge-based embeddings. LUKE is built on a transformer skilled with substantial entity-annotated data and then uses linear classifiers to classify the output illustrations. Denoising transformers utilise a method where binary NNs are trained – (G) the generator and the discriminator or (D). ELECTRA (Clark et al., 2020) use replaced token discovery as a specimen efficacious pre-training activity. By substituting some input tokens with credible substitutes obtained from a tiny

generator grid, the technique corrupts the input rather than disguising it. The process then trains a discriminative model that forecasts whether every individual token in the corrupted load was substituted by a generator specimen or not, in contrast to building a model which forecasts the actual individualities of the token that is corrupted. METRO (Bajaj et al., 2022) (abbreviation for model generated denoising training objective), consists of up to 5.4 billion parameters. Metro references the plain training arrangement of the ELECTRA model via an MLM model of language as the supplementary, and replaced token detection (RTD) activity to prepare the actual model. Generative pre-trained transformer 3 (GPT-3) is the descendant of GPT-2 and an unsupervised transformer linguistic model. It comprises 175 billion variables, greater by binary orders of magnitude 1.5 billion variables in the complete release for GPT-2. Microsoft was licensed exclusively on September 2020 with GPT-3 model. BLOOM (Scao et al., 2022) is about announcing the ecosphere's biggest unlocked MLM. The model is offered in many different configurations, and the design of BLOOM is principally alike to GPT3. BLOOM is available in 13 software development languages and was trained in 46 different languages.

3 Classification and review of NLP models

This section presents a thorough review of operational principles for many categories of state-of-art NLP models. We deliver seven classifications of NLP models according to operational principles. Then, utilising typical algorithms from each class, we deliver exhaustive scrutiny and operation of each classification. The different arrangements have been discussed as follows:

1 Human performance – Human performance is utmost beneficial for advancing AI and NLP products. It is the statistical value about the performance of humans on said NLP task. In practice, it allows us to estimate Bayes error (BE) (Tumer and Ghosh, 2003) which is the greatest error that any function, past, present, or future, could ever produce.

2 GLMs –GLMs are a group of models that expand on linear regression by enabling the modelling of many additional distributions for the response output by applying the link function. Linear and logistic regression is the most popular among GLM candidates.

3 NNs (Mcculloch and Walter, 1943) or just neural network, is a group of procedures that intend to detect concealed relationships in a source of information by means of a technique that bear similitude to in what way the intellect of human happens. NNs are proficient in becoming accustomed to varying feeds and can yield superlative probable outcomes without altering the output norms.

4 RNN – RNN is an abbreviation for RNN that is a class of neural nets wherever association amongst units can lead to a sequence where the yield of one unit can impact the participation of additional units. It can demonstrate progressive complex comportment as a consequence of this. RNNs that are resultant of feedforward neural nets, can exercise arrangements of varying measurement by using internal conditions (also called memory).

5 Transformers – the transformer NN is a unique design that tries to handle long-range relationships while solving sequence-to-sequence challenges.

6 AE transformers – AE models are pre-trained by attempting to recreate the original sentence after distorting the input. AE are comparable to the transformer's encoder as AE models possess unobstructed all sequence admittance. AE models frequently create a two-way illustration of the whole arrangement. Although AE models might be tweaked and excel at a diversity of assignments, like the generation of text, but has the utmost natural use case for sequence classification or token cataloguing.

7 ARs – Often called de-coders or AR models. AR model uses an attention mask and the decoder portion of the transformers so that the model can solitarily see the tokens earlier to the attention heads at each location.

Next, we will go over each class in great detail, as mentioned in this part. Section 5 that follows presents a performance study and comparison of those various classes in detail.

3.1 Human performance

Because some of the activities that people accomplish are nearly 'perfect', machine learning aims to perform at a level that is comparable to humans. Human-level performance is instrumental in machine learning projects and estimates BE, i.e., the best possible error that could be achieved. In a mathematical sense, BE is the least likely fault degree for any allocator of an arbitrary consequence. The BE can be equated by:

$$BE = 1 - \sum_{i=1}^{L} ZP(C_i)\, p(x \mid C_i) dx \tag{1}$$

where x is the specified vector, L cataloging classes, C_i is the area wherever i class ($1 \le i \le L$) has the maximum posterior, $P(C_i)$ is the a priori prospect of i class, $p(x \mid C_i)$ is the possibility of the class (assuming class i membership, the conditional probability density of x).

As shown in Figure 1, machine learning performs better than humans and advances slowly. One of the reasons is that, particularly for any natural perception problems, human-level performance can be very close to Bayes optimal error. Developments in deep learning have beat human beings' achievements in numerous more activities like – approval for loans, recommendations for products, and many others. We can improve the machine learning performance when it is lower than the human performance via-

- Collecting more labelled data from individuals.
- Checking intuitions from manual error analysis.
- Bias versus variance analysis – If the variance between training and human-level error is larger than the variance between development and training error, we can use the bias-specific reduction procedures. However, if the variance between development and training error is higher than the variance between training and the human level error, we should be using variance reduction techniques.

3.2 Generalised linear models (Nelder and Wedderburn, 1972)

GLMs are the class of models that expand on linear regression by allowing a link function to model a large number of additional distributions for the response variable. Linear and logistic regression is the most popular among GLM candidates.

3.2.1 Logistic regression (Chao et al., 2022)

Logistic regression is used in supervised tasks and reduces the difference in forecast and training data error. The method will assess the likelihood of a result for an issue denoted by feature vector x. Given input features vector x, the predicted output $\hat{y}$ is expressed as:

$$\hat{y} = P(y = 1 \mid x) = \sigma(w^T x + b) \tag{2}$$

where $\hat{y}(0 \le \hat{y} \le 1)$, s is the sigmoid function, b is the bias or threshold, and w is the weights.

As shown in Figure 2, the probability is constrained between [0, 1] and we can observe that:

- if z is a large, then $\sigma(z) = 1$
- if z is small or negative number, then $\sigma(z) = 0$
- if $z = 0$, then $\sigma(z) = 0.5$.

Loss function estimates the variance (or inaccuracy) amongst the anticipated output ($y(i)$) and the prediction ($\hat{y}(i)$), which is expressed as:

$$L(\hat{y}(i), y(i)) = \frac{1}{2}(\hat{y}(i) - y(i))^2 \tag{3}$$

$$= \log(1 - \hat{y}^{(i)})(1 - y^{(i)}) - (\log \hat{y}^{(i)} y^{(i)}) \tag{4}$$

We can observe that:

- When $\log(\hat{y}(i))$ and $\hat{y}(i)$ are near to 1 and $y(i) = 1$ then $L(\hat{y}(i), y(i)) = -\log(\hat{y}(i))$.
- When $(1 - \hat{y}(i))$ and $\hat{y}(i)$ are near to 0 and $y(i) = 0$ then $L = -\log(1 - \hat{y}(i))$.

The overall cost function J is the whole loss function's average over the training set and is calculated as:

$$J(w, b) = \frac{1}{m} \sum_{i=0}^{m} L(\hat{y}(i), y(i)) \tag{5}$$

$$= \frac{1}{m} \sum_{i=0}^{m} \left[\left(\log(\hat{y}(i) y(i)) + \left(\log(1 - \hat{y}(i)(1 - y(i))) \right) \right) \right] \tag{6}$$

where $x(i)$ is the i^{th} training example, w and b are the weight and bias parameters. At the same time, one can use the logistic regression approach to NLP tasks, but NLP tasks that contain large corpus results in various erroneous matches. This is the reason one needs to model with better approaches that are discussed further.

3.3 Neural networks (Mcculloch and Walter, 1943)

Neural nets, likewise acknowledged by artificial NNs, are an assembly of associated calculating schemes of elements, also referred to as artificial neurons. When related to a real neuron, an artificial neuron uses the mathematical function and has inputs, outputs, units (or nodes), and weights that are similar to the dendrites, axon, cell nucleus, and synapse correspondingly.

As shown in Figure 3, each neuron takes inputs (1, x_1 … x_m), weighs (w_0, w_1 … w_m) them separately, sums them up, and passes this sum through a nonlinear function which produces output. Moving ahead from a single neuron, a deep NN (assembly of interconnected neurons) or comprehensive network representation can be shown in Figure 4.

Vectorised forward propagation equations for the network are formulated by:

$$z^{[l]} = b^{[l]} + \left(w^{[l]} a^{[l-1]}\right) \tag{7}$$

$$a^{[l]} = \left(z^{[l]}\right) g^{[l]} \tag{8}$$

Vectorised backward propagation equations of the network are calculated as:

$$da^{[l-1]} = w^{[l-1]T} .dz^{[l]} \tag{9}$$

$$db^{[l]} = dz^{[l]} \tag{10}$$

$$dw^{[l]} = dz^{[l]} .a^{[l-1]T} \tag{11}$$

$$dz^{[l]} = da^{[l]} * g'^{[l]}\left(z^{[l]}\right) \tag{12}$$

Vectorised loss function (L) and cost function (J) are calculated as:

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = \log\left(1 - y^{(i)}\right)\left(1 - y^{(i)} - \log\left(\hat{y}^{(i)} y^{(i)}\right)\right) \tag{13}$$

$$J(x, w, b, y) \text{ or } J(y, \hat{y}) = \frac{1}{m}\sum_{i=0}^{m} L\left(y(i), \hat{y}(i)\right) \tag{14}$$

$$= -\frac{1}{m}\sum_{i=0}^{m} \log\left(\hat{y}(i) y(i)\right) \tag{15}$$

Using single layer NN approach for NLP tasks generate results with erroneous matches similar to logistic regression. On the other hand, using multiple layered NN leads to the problem of overfitting because network treats each input separately and ignores spatial relationships. This leads us in search of stronger approaches that we will discuss next.

3.4 RNN (Abiodun et al., 2018)

RNN is an abbreviation for recurrent neural network that is a class of neural net wherever association amongst units can lead to a sequence where the yield of one unit can impact the participation of additional unit. It can demonstrate progressive complex comportment as a consequence of this. RNNs that are resultant of feedforward neural nets, can exercise arrangements of varying measurement by using internal condition (also called memory).

For period slice t and sequence $x(t)$, the concealed state $h(t)$ which acts as 'memory' and is calculated as:

$$h(t) = f\left(Wh(t-1) + Ux(t)\right) \tag{16}$$

Here, f represents a non-linear conversion function (or activation function) like ReLU, tanh and, $o(t)$ representing the output. For weights (U, V, W) that are common across period, V is the hidden-output association weight matrix, U is the input-hidden association weight matrix, and W is the hidden-hidden recurrent association weight matrix.

The activation $a(t)$ and output $y(t)$ are calculated as:

$$a^{(t)} = \tanh\left(b_a + W_a.x^{(t-1)} + W_x.x^{(t)}\right) \tag{17}$$

$$y^{(t)} = S_{\max}\left(b_y + W_y.a^{(t)}\right) \tag{18}$$

where t is the timestep, b_a, b_y, W_x, W_a, W_y are the coefficients, while the activation functions that can be used are softmax ($S_{\max}$) or tanh.

Backward pass, also known as back-propagation through time (BPTT). The gradient at each output hinge on computations of the preceding period states along with the up-to-date period state due to shared variables in the system. L (or loss function) is expressed by:

$$L\left(y, \hat{y}\right) = \sum_{t=1}^{Ty} L\left(y(t), \hat{y}(t)\right) \tag{19}$$

The derivative of the loss L at timestep T and weight matrix W is expressed by:

$$\frac{\partial L(T)}{\partial W} = \sum_{t=1}^{Ty} \left.\frac{\partial L(T)}{\partial W}\right|_{(t)} \tag{20}$$

3.4.1 LSTM (Sak et al., 2014)

LSTM which abbreviates long short-term memory network is a category of RNN which can process sequential data and consists of feedback connections. A regular LSTM unit is made up of a forget gate, a cell, an input and output gate. The three gates control the knowledge flow, and the cell retains values over random time intervals. LSTM has both 'short-term memory' and 'long-term memory'. The architecture of LSTM is such that 'short-term memory' remains for thousands of timesteps and is thus referred to as 'LSTM'. This property of LSTM is very useful for time series data predictions.

LSTM unit can be represented by at as hidden layer vectors, c_t as cell state vectors, x_t as input vector, y_t as output vector, b as bias, W as parameter matrice and σ, tanh as activation functions. For Γ_u (update gate), tanh = Γ_r (relevance gate), Γ_f (Forget gate), Γ_o (Output gate) and $\star$ denoting the multiplication of two vectors element-wise, characterising equations for LSTM architecture are stated by:

$$c \sim^{<t>} = \tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c) \tag{21}$$

$$c^{<t>} = \Gamma_u \star c \sim^{<t>} + \Gamma_f \star c^{<t-1>} \tag{22}$$

$$a^{<t>} = \Gamma_o \star c^{<t>} \tag{23}$$

$$\Gamma_u = \sigma\left(W_u\left[a^{<t-1>},\ x^{<t>}\right] + b_u\right) \tag{24}$$

$$\Gamma_f = \sigma\left(W_f\left[a^{<t-1>}, x^{<t>}\right] + b_f\right) \tag{25}$$

$$\Gamma_o = \sigma\left(W_o\left[a^{<t-1>},\ x^{<t>}\right] + b_o\right) \tag{26}$$

L (or loss function) is expressed by:

$$L\left(y, \hat{y}\right) = \sum_{t=1}^{Ty} L\left(y(t), \hat{y}(t)\right) \tag{27}$$

Backpropagation is carried out at all times. The derivative of the loss L at timestep T and weight matrix W is expressed by:

$$\frac{\partial L(T)}{\partial W} = \sum_{t=1}^{Ty} \frac{\partial L(T)}{\partial W}\bigg|_{(t)} \tag{28}$$

Using simple RNNs lead to vanishing gradients and exploding gradients problem. On the other hand, using LSTMs has its own problems such as we need very large amount of data for training and they are not very good candidates of online learning tasks. Also, LSTM exerts overfitting issues and using dropouts with LSTM is very difficult. Next, transfer learning is not possible when we use LSTM and hence, we need better approaches like Transformers that will be discussed now.

3.5 Transformers (Vaswani et al., 2017)

The transformer NN is a unique design that tries to handle long-range relationships while solving sequence-to-sequence challenges by using self-attention. Attention is modelled by utilising the cognitive attention technique which allows some of the input data to be visible better while making supplementary sections inferior. The gradient descent technique is utilised to learn part of the data that is more important than other parts of the remaining data. Most of the transformers are encoder-decoder, stacked encoder or stacked decoder configurations.

For an input sequence denoted by (x_1, …, x_n) the encoder translates it to continuous sequence z = (z_1, …, z_n). The decoder then produces one section at a time an output sequence (y_1, …, y_m) using z as input. The model is auto-regressive and customs the yield of the former step as additional input for the next step output generation. Transformer practices layered self-attention and point-wise wholly linked layers as shown in Figure 8 for the decoder and encoder both.

The encoder part of the transformer is composed of an encoder stack where each slab is made up of dual sub-slabs-self-attention multi-head mechanism and feed-forward wholly linked position-wise grid. For each of the two sub-slabs a residual link is used that is trailed by layer normalisation. For each SubSlab(x), the yield is NormLayer (SubSlab(x) + x).

The decoder part of the transformer is composed of a decoder stack of alike layers. The decoder adds a third sub-slab to the encoder's two sub-slabs, which executes multi-head attention on the yield of the encoder stack. Residual contacts are utilised around each sub-layers that is trailed by layer normalisation same as the encoder block. Decoder uses a modified version of the self-attention sub-layer to avoid spots from

joining to subsequent spots. The forecasts for position i only rest on the recognised outputs at locations fewer than i due to masking and offset of output embeddings by single spot.

Dot-product attention (has scaling factor of $\frac{1}{\sqrt{d_k}}$) which is must faster, space-efficient than the additive attention (uses a single hidden layer in a feed-forward network to calculate the compatibility function) are the most widely used functions. The system might mutually participate to information from numerous depiction subspaces at numerous places thanks to multi-head attention. For any i^{th} head attention (QW_i^Q, KW_i^K, VW_i^V) where i $(0 \le i \le h)$, Q set of queries, S is softmax activation function, C is concatenation function, keys K, V and W_i^Q, W_i^K, W_i^V W^O are learned matrices. The attention function (A) and the multi-head attention (M) is expressed as:

$$A(Q, K, V) = S\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{29}$$

$$M(Q, K, V) = C\left(h_1, .., h_h\right)W^O \tag{30}$$

3.6 AE transformers

AE uses the encoder only or stack of encoders. For every stage, unrestricted access to all inputs is available to the attention layers and often has a 'bi-directional' attention mechanism.

3.6.1 BERT (Devlin et al., 2018)

BERT is based on a transformer network and abbreviation for bidirectional encoder representations by Google NLP created in a pre-training program that was released in 2018.

While the same BERT architecture can be utilised in diverse jobs the framework is divided into two steps, first is pre-training followed by fine-tuning (FT) activity. BERT is skilled on unlabelled data in pre-training activity and for FT pre-trained parameters are adjusted using labelled data generating fine-tuned parameters for the downstream job.

BERT is available in two model configurations as described in Table 5. BERT utilises WordPiece vocabulary with 30,000 tokens and [CLS] is the first token and [SEP] is the differentiator between the sentences. As shown in Figure 10, a token is represented by summating the matching position embedding with segment and the token.

3.6.2 ALBERT (Lan et al., 2019)

A light BERT (ALBERT) model was announced in 2019 by Google to prevail GPU/TPU memory restriction issues in BERT and stretched training periods by parameter reduction techniques such as inter sentence coherence loss, factorisation of embedding matrix and parameter sharing are used.

ALBERT has three key deviations from BERT but majorly utilised architecture is same with encoder layers using Gaussian error linear unit (GELU) activation function. ALBERT makes use of two different embedding matrices but input slab embeddings and

concealed layer embeddings of the BERT are the matching magnitude. Concealed embedding (H) involves situation-reliant education, whereas input embedding (E) is refined for context-independent learning as a result, with a slight performance reduction of 80% in parameters drop is achieved. Following, three forms of parameters allocation (all parameters sharing, attention parameters sharing, feed forward network parameter sharing) provide a 70% drop in overall parameter count. For training ALBERT model masked language model is practiced. ALBERT uses sentence order prediction (SOP) loss that individually focuses on sentence coherence.

ALBERT base model has 9× lesser parameters than corresponding BERT base model and ALBERT large model has 18× lesser parameters than the corresponding BERT large model. ALBERT uses 16 GB of uncompressed Book CORPUS and English Wikipedia dataset as used the in BERT model. Using ALBERT can be problematic in situations where we cannot pad the data as it needs absolute position embeddings. Also, computational cost of ALBERT is similar to BERT as it uses similar hidden layers.

3.6.3 DistilBERT (Sanh et al., 2019)

Training a large NLP model like BERT process can be excruciatingly long and drag on for days. DistilBERT is a distilled version of BERT-like model. A BERT model can be made 60% faster and 40% smaller while still preserving 97% of its language processing abilities. Distillation is the process of preparing a little student model to imitate a big teacher model as close as possible.

Distillation is very useful to migrate a model onto compact hardware, like a smartphone or a small laptop, because distilled models are more efficient and use a reduced amount of space.

DistilBERT's distillation approach first duplicates the teacher's architecture but by preparing the student's layers from the teacher's from count of layers for initialisation. DistilBERT switches within cloning and discarding a single layer at a time. The distillation routine is based on the loss, to minimise the classic loss function which the teacher trained on and mimic the teacher. The job that the instructor was optimised has a unique function of loss. We must enter the labels and embeddings of learners in order to calculate that loss because the framework is constructed comprising attention layers containing the identical problem-specific head that the instructor.

The teacher-student cross-entropy loss effects on two 3D vectors loss aim to reduce the gap between teacher and student probability distributions. For a given x as an input, the outputs of instructor and the learner are:

$$T(x) = \left(t_1, ..., t_n\right) = soft\max\left(\hat{t}_1, ..., \hat{t}_n\right) \tag{31}$$

$$S(x) = \left(s_1, ..., s_n\right) = soft\max\left(\hat{s}_1, ..., \hat{s}_n\right) \tag{32}$$

If T and S are brought closer together, the teacher-student cross-entropy loss is applied to S with T as the goal and given by:

$$L_{cross} = -\sum_{i=1}^{n} t_i * \log\left(s_i\right) \tag{33}$$

A cosine loss is a secondary loss that assists the learner in becoming the instructor. In both educator and learner models, co-sine loss aligns the vectors and is equivalent to:

$$L_{cosine} = 1 - \cos\left(T(x), S(x)\right) \tag{34}$$

Temperature is a concept that DistilBERT employs to dampen Softmax. The temperature is equivalent to the following for class i, Z_i as score of model and T that pedals the output distribution's softness, temperature is equated as:

$$p_i = \frac{\exp\left(Z_i / T\right)}{\sum_j \exp\left(Z_j / T\right)} \tag{35}$$

DistilBERT can be problematic in situations where we do not have much time to train student networks and when one wants to use pretrained DistilBERT model can face sequence length problem that is set to 512.

3.6.4 RoBERTA (Liu et al., 2019)

RoBERTA can perform as well as or better than all of the post-BERT methods with modifications in model as mentioned below:

- practice longer successions
- eliminating the goal of the subsequent sentence prediction
- altering the masking pattern used on the training data dynamically
- for massive data assemblies educating the model for extensive time spans.

RoBERTA utilises transformer architecture with hyperparameter tuning and few changes in magnitude of training set. RoBERTA discovers that BERT was undertrained and improves with Adam with weight decay $L2 = 0.01$, ϱ = 1e-6, $\beta 1 = 0.9$ and $\beta 2 = 0.999$. The learning rate is linearly decayed over 10,000 steps to the maximum value of 1e-4. Model are pre-trained mini-batches of $B = 256$ sequences, S = 1,000,000 updates, and a maximum of token length $T = 512$.

RoBERTA has limitations of requirement of large FT time and computational cost for zero shot learning. It is also prone to biased outputs based on the nature of training data.

3.6.5 T5 (Raffel et al., 2020)

Text-to-text transfer transformer (T5) practices a text-to-text style on transformer architecture which permits same model, loss function, hyperparameters to be used in various NLP tasks. T5 uses a variety of other pre-training activities in place of the fill-in-the-blank cloze task and bidirectional architecture includes an additional causal decoder.

T5 utilises transfer learning to indulgence in text handling task as 'text-to-text' issue (that is, the input to the model is text generated output is also new text) which allows T5 to directly apply the model to all NLP tasks.

In T5, arrangement of input tokens is charged with an embedding arrangement and propagated into the encoder. Each encoder stack unit comprises of self-attention layer and feed-forward tiny grid. A simplified form of layer normalisation with rescaled activations and no additive favouritism is applied to input trailed by residual skip connection which joins every module's input with output and dropout is applied.

The decoder shares a structure with the encoder, with the exception that it also has a standard attention mechanism that pays to the encoder's output after each layer of self-attention that uses form of causal or AR self-attention enabling the model to join past outputs. The concluding decoder unit output is propagated into a softmax dense layer and weights are made common with the embedding matrix of input. Comparative spot embeddings are used to yield a different cultured embedding based on 'query' to 'key' offset. For all models, T5 uses 32 embeddings with ranges that get progressively larger logarithmically up to an offset of 128 after which all comparative spots are assigned to the matching embedding.

T5 model uses 12 blocks of encoder and decoder (containing feed-forward network, self-attention with discretionary encoder-decoder). T5 base contains about 220M hyperparameters almost twice the number of parameters of BERT. Another problem with T5 model is it has fixed length for inputs.

3.6.6 XLM (Lample and Conneau, 2019)

XLM shows efficiency of NLP to multiple languages or cross-lingual pretraining. It is used in one of three language modelling objectives: TLM – an objective for (new) TLM to enhance cross-lingual pre-training. MLM – The goal of BERT's MLM. CLM – models the likelihood of a word given the words that came before it in a phrase.

XLM practices byte pair encoding (BPE) and using a multinomial distribution with probability, sampled sentences $\{q_i\}_{i=1\ldots N}$ and $\alpha = 0.5$ are represented as:

$$q_i = \frac{p_i^{\alpha}}{\sum_{j=1}^{N} p_j^{\alpha}} \tag{36}$$

$$p_i = \frac{n_i}{\sum_{k=1}^{N} n_k} \tag{37}$$

The quantity of tokens related with low-resource languages is amplified with this distribution sampling while diminishing the bias in favor of high-resource languages preventing character-level word splitting of low-resource languages in particular.

For the purpose of CLM objective, transformer language model that has been skilled to forecast the likelihood of a word specified the words before it in a sentence for the given task $P(w_t \mid w_1, \ldots, w_{t-1}, \theta)$.

Aimed at the purpose of MLM objective, arbitrarily selecting 15% BPE tokens from the manuscript, substituting it 80% with [MASK] tokens, 10% of the time with random tokens, and 10% leaving them untouched. Text stream tokens are fragmented based on the multinomial spread, for which weights are relational to square root of the inversion rate of recurrence, to address the imbalance between uncommon and frequent tokens.

The purpose of TLM is to refine cross-lingual pretraining. Combining analogous text sentences and arbitrarily masked terms from the basis and goal text sentences, TLM improves on MLM. XLM could pay attention to nearby English terms else on the French conversion in order to forecast a word that is hidden in an English text sentence, which inspires XLM to line up the English and French illustrations. When the English setting is insufficient to deduce disguised English words, XLM can use the French setting. The placements of the final sentences were adjusted to make placement easier. Mostly XLM

is one of the best candidates currently available but it has limitation and problem while dealing with low-resource languages and representations.

3.6.7 Longformer (Beltagy et al., 2020)

Longformer establishes its efficiency in large document sequence-to-sequence task. Longformer has an altered transformer design and attention procedure to process lengthy document sets and excels over BERT like models that has 512 token restriction. Longformer syndicates its attention using sliding window attention, dilated window attention, and self-attention. Longformer eliminates the requirement for task-specific structures by leveraging several levels of attention to construct contextual illustrations of the whole context.

The attention (A) scores for a transformer (Vaswani et al., 2017) and Longformer (Beltagy et al., 2020) are equated by:

$$A(Q, K, V) = S\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{38}$$

where Q is the query, K is the key, V is the value, and S is the softmax activation function. Here, for the Longformer model, two sets of attention vectors are used – Q_{swa}, K_{swa}, V_{swa} scores for sliding window attention and Qga, Kga, Vga scores for global attention. In the start, vector values for Q_{ga}, K_{ga}, V_{ga} are set alike to Q_{swa}, K_{swa}, V_{swa} vector values. Longformer is a very good candidate to process long documents but drawback is model implementation operations still be needs to be more accessible directly in deep learning commonly used libraries.

3.6.8 LUKE (Yamada et al., 2020)

LUKE is built on a transformer that was skilled with a substantial amount of entity-annotated data and then used linear classifiers to classify the output illustrations. LUKE is trained on the masked language model for BERT that includes forecasting arbitrarily masked words and entities in a huge entity-annotated corpus extracted from Wikipedia. When calculating scores of attentions, LUKE uses entity-aware self-attention that extends self-attention with also observing the types (entities or words) of tokens which allows LUKE to accomplish remarkable performance on the NLP tasks.

By the use of [MASK] input entities, LUKE can calculate depictions of any random text entities. The model may generate entity representations grounded on the rich entity-centric data stored in the equivalent entity embeddings if entity annotation is made accessible in the job. The primary and last words in the sequence are replaced with the special tokens [CLS] and [SEP]. For unknown entity in the vocabulary, [UNK] special token is used. By adding the position and token embeddings with the position, the token, and the embeddings of entity type, the input illustration of a word and an entity are calculated.

LUKE configuration follows large configuration of RoBERTa model with 16 self-attention heads, 64 attention head dimensions, 1,024 hidden dimensions, 24 hidden layers, 64 hidden layers, and dimensions as $H = 256$ embedding of entity token. LUKE has an overall number of parameters as 483M (Entity embeddings: 128M + RoBERTa: 355M) and the vocabulary of 50K words. By arbitrary masking entities and words, 15%

of the training time is reduced. While using LUKE one need to be aware of fairness and bias issues as it can have disturbing typecasts for endangered classes and groups like (occupational or social).

3.6.9 ELECTRA (Clark et al., 2020)

ELECTRA is an abbreviation of efficiently learning an encoder that classifies token replacements accurately. The framework recommends swapped out detection of token as a sample-effective pre-training activity. By substituting some input tokens with credible substitutions drawn from tiny generator grid, the technique corrupts the input rather than disguising it. The process then trains a discriminative model that forecasts whether each token in the corrupted input is substituted by a generator sample or not, as contrasting to building a model which forecasts the initial personalities of the manipulated tokens. This tactic is suggestive of training the discriminator with GAN but is not 'adversarial' because of the struggle of text usage for GAN, the generator creating corrupted tokens is trained using maximum likelihood.

ELECTRA small model takes four days to train on a single GPU. Given the same model size, data, and computation, ELECTRA greatly outperforms MLM-based techniques like BERT and XLNet.

ELECTRA consists of two NNs – (G) a generator and (D) a discriminator that are essentially made up of an encoder of the transformer that maps contextualised vector illustrations for the sequence $h(x) = [h_1, \ldots, h_n]$ with the sequence of input tokens $x = [x_1, \ldots, x_n]$. The generator softmax layer output for the token x_t with position t is expressed as:

$$p_G(x_t \mid x) = \frac{\exp\left(e(x_t)^T h_G(x)_t\right)}{\sum_{x'} \exp\left(e(x')^T h_G(t)_t\right)} \tag{39}$$

As shown in Figure 15, model generates output from generator and is trained along with the discriminator. Subsequently pre-training, generator is not employed; instead, just the discriminator's FT on subsequent jobs is utilised.

For the token x_t at position t and token embeddings e, the discriminator prediction of actuality 'fake', not from the data distribution but originates from the generator is expressed as:

$$D(x, t) = sigmoid\left(w^T h_D(x)_t\right) \tag{40}$$

Minimising combined loss:

$$\min_{\theta_G, \theta_D} \sum_{x \in X} L_{MLM}(x, \theta_G) + \lambda L_{Disc}(x, \theta_D) \tag{41}$$

After pre-training task, the generator is not used and FT is achieved by utilising the discriminator on downstream tasks. Further, ELECTRA is enhanced weight sharing that is the process of refining the pre-training efficiency by sharing weights between the generator and discriminator. Next, utilising smaller generators is very beneficial which can be achieved by diminishing the size of layer and keeping the hyperparameters constant. If G and D have the same size, training time is twice as much processing power as training just using MLM in each phase. ELECTRA work best with $\underline{G}$ as 1/4-1/2 the size of D.

Training in ELECTRA trains *G* and *D* together. First, the generator is trained with LMLM for *n* steps. *D* weights are set with the weights of *G*. Next, *D* is trained with L_{Disc} for *n* steps and *G* weights are kept cold. *G* and *D* must be the same size preceding to weight initialisation. *D* might occasionally be unable to acquire anything beyond the mainstream class lacking weight initialisation, maybe because *G* progressed much earlier than *D*. Similarly, combined training offers prospectus for *D* where *G* initially performs poorly but improves with time.

3.6.10 METRO (Bajaj et al., 2022)

METRO is a very effective utilising training indicators produced by an auxiliary model, a denoising pretraining technique for large-scale AE models. Conditional to the input sequence is manipulated by means of machine or rules skilled models, are two sorts of denoising training approaches.

The original input sequence X^o is altered (or corrupted) to generate noisy sequence X^n which is used to train to find the initial sequence. The pretraining denoised arrangement would be expressed as:

$$X^o \xrightarrow{alter} X^n \xrightarrow{model} X^o \tag{42}$$

Rule-based manipulation can be achieved easily to produce noisy sequence X^n via guideline set arbitrarily altering few tokens and is expressed as:

$$X^o \xrightarrow{arbitrarily\ alter\ using\ rules} X^n \tag{43}$$

Model-based technique is more accurate because in the rule-based technique arbitrarily produced noises are insufficiently detailed to allow for successful modelling. Model-based technique uses a supplementary transformer to produce X^n and train the main model to (partially) retrieve the actual text:

$$X^o \xrightarrow{Supplementary\ model} X^n \xrightarrow{main\ model} \hat{X}^o \tag{44}$$

The supplementary model is a language model (e.g., BERT) pretrained using MLM. By altering few of original tokens, the alteration is achieved with samples x_i^{mlm} that produces $X^n\left\{x_1^o, ..., x_i^{mlm}, ..., x_n^o\right\}$. RTD, which trains the primary model, and ELECTRA, which is used for MLM as the supplementary model, are the first reference training arrangements. For every token, the supplementary model g_{aux} generates h_i^{aux} contextualised illustration, that is utilised by head of the model p_{mlm} in order to create softmax dispersion for vocabulary. L_a (language modelling loss) is used in pre-training the supplementary model to regenerate *M* masked tokens. By means of p_{rtd}, binary classification head h_i, the contextualised illustration is formed by g_{main}. It is possible to determine whether token x_i^n is being substituted (noise) or substitution is not achieved (actual). For L_m^r is main model loss RTD loss and λ that stabilises velocity of learning for both main and supplementary model, pretraining together is expressed as:

$$L = \lambda L_m^r + L_a \tag{45}$$

For better efficiency of the reference model several modelling practices such as using shallow auxiliary model, relative position embedding, large vocabulary and Respecting

Document Boundary are utilised. To entirely exclude dropout regularisation, the model also tests the zero-dropout approach along the supplementary axis. As the training of auxiliary model and interpretation (to generate main learning sequences) can together be achieved in single forward-backward step, it considerably lowers the processing cost. A robust auxiliary model could style the denoising activity demanding for the core model to educate, whereas a weak auxiliary model might not produce thought-provoking sufficient training to enhance the core model.

3.6.11 BART (Lewis et al., 2020)

BART is a model uniting two-directional and AR Transformers. BART is an autoencoder that denoises data for seq-to-seq models. BART is skilled via tampering with text using various noise-reduction techniques and then developing a model to restore the original content. BART works well for knowledge tasks as well as generation of text, where it is most effective.

BART presents a novel approach to machine translation that deposits BART unit at the uppermost transformer deposits. To improve efficiency, promulgating over the model and utilising BART as a target-side linguistic model, these layers are skilled to principally modernise the overseas linguistic to noised English. BART employs sequence-to-sequence transformer design, but modifies ReLU with GeLU and initialises parameters after GPT. Encoder and decoder layers for the basic model are six layers each, whereas they are twelve layers each for the large model. With the subsequent exemptions, the design is carefully associated to that of BERT. Firstly, final hidden layer of the encoder is given cross-attention by decoder's each layer. Secondly, beforehand word prediction, BERT employs a separate feed-forward network which is not part of BART. Overall, BART employs 10% extra parameters than similar dimensions of the BERT model.

3.7 AR transformers

AR transformers are decoder-based models which creates the value at the present period unit using data from the decoder's prior time steps. That is, an analytical blueprint is said to be AR if it forecasts upcoming sequence based on historical sequences.

3.7.1 TRANSFORMER-XL (Dai et al., 2019)

Transformer-XL (extra-long) brings the idea of repetition to profound self-attention grid through a Transformer design. Transformer-XL styles with the concealed states collected in former pieces instead of calculating the concealed states from scrape for separately novel parts. Transformer-XL employs a novel comparative positional encoding design that extends beyond the training attention length to larger attention lengths. A relative positional encoding method used by the model that extends to attention spans larger than those used in the training.

Transformer-XL acquires reliance 80% longer than RNNs and 450% longer than transformer. Transformer-XL attains great efficiency on long and short both sequences. While through assessments, Transformer-XL is almost 1800+ times quicker transformer. Model is trained on WikiText-103 and is able to produce comprehensible, fresh text articles with thousands of tokens. The model utilises sinusoidal positional embeddings

while padding can be applied on the right or left. Transformer-XL has no boundary for the length of sequence.

3.7.2 GPT models (Radford et al., 2018, 2019; Brown et al., 2020)

GPT (or generative pre-training) is able to pre-train on a diversified corpus with lengthy stretches of continuous text, enabling it to learn global knowledge and understand long-range dependencies.

- GPT1 (Radford et al., 2018): GPT1 (or GPT-1) was released by Open AI in June-2018 and is an abbreviated form for generative pre-trained transformer 1. Training of GPT1 involves of two phases. Firstly, educating model from big text dataset. Secondly, the supervised FT phase by labelled data to adapt GPT1 with the discriminative task. For a specified unsupervised token dataset $U = \{u_1, \ldots, u_n\}$ goal is to achieve highest likelihood which is expressed as:

$$L_1(U) = \sum_i \log P\left(u_i \mid u_i - k, ..., u_{i-1}; \Theta\right) \tag{46}$$

where Θ represents parameters of the NN, P is the conditional probability and k is context window size. Stochastic gradient descent is utilised for training parameters while the target tokens output distribution for $(0 \le I \le n)$ is expressed as:

$$h_0 = W_p + UW_e \tag{47}$$

$$h_l = T\left(h_{l-1}\right) \tag{48}$$

$$P(u) = S\left(W_e^T h_n\right) \tag{49}$$

where T is transformer block, S is softmax, the count of layers is represented by n, tokens context vector is represented by $U = (u_{-k}, \ldots, u_{-1})$ while W_e and W_p is the token and position embedding matrixes.

Subsequently, the supervised task target is to regulate the hyperparameters. Used for C labelled dataset, input tokens sequence $(x_1, \ldots, x_m)$ along y label. To get last activation of the transformer block h_l^m, inputs are navigated through the pre-trained model. Predicted y is achieved by is passing h_l^m to linear yield layer and constraint W_y expressed as:

$$P\left(y \mid x^1, ..., x^m\right) = soft\max\left(W_y h_l^m\right) \tag{50}$$

As a result, we can optimise the following goal:

$$L_2(C) = \sum_{(x,y)} \log P\left(y \mid x^1, ..., x^m\right) \tag{51}$$

Improving the output (by regularisation λ):

$$L_3(C) = L_2(C) + \lambda * L_1(C) \tag{52}$$

- GPT2 (Radford et al., 2019): GPT2 (or GPT-2) is an abbreviated form for generative pre-trained transformer 2 and the successor to GPT1. Publicly was first announced in February 2019 with limited capabilities, while the final version was released in

November 2019. GPT2 training utilises WebText corpus consisting of approximately 40 GB text with 8 million documents from Reddit URLs with three plus upvotes.

For the set of examples ($x_1, x_2, \ldots, x_n$) and sequences of symbols ($s_1, s_2, \ldots, s_n$) with adjustable length language model is framed. We can express, factorising symbols joint probability over conditional probability product as:

$$p(x) = \prod_{i=1}^{n} p\left(s_n \mid s_1, ..., s_{n-1}\right) \tag{53}$$

GPT2 model principally follows the GPT1 model decoder-only blocks with few modifications. Except for the elimination of the second self-attention layer, the blocks are identical to the original decoder blocks. Analogous with pre-activation outstanding system, layer normalising shifted every sub-unit input. After the final self-attention block, a further layer normalisation was added. Lexis is extended to 50,257, context dimension to 1,024 instead of 512 and magnitude of lot to 512. GPT2 can address up to 4,000 segment tokens.

- GPT3 (Brown et al., 2020): GPT3 (or GPT-3) is an abbreviated form for GPT-3 and the successor to GPT-2, which was released in May 2020. Compared to the 150 crore constraints of the complete type of GPT-2, the 175 billion parameters in GPT-3 are two orders of magnitude more numerous. The 'meta-learning' activities that GPT-3 is successful at. A single input-output pair's function can be generalised by using it. Benchmark performance between GPT-2 and GPT-3 was significantly better. GPT-3 was exclusively licensed to Microsoft on September 2020.

Model scaling up involves relatively simple increases in model size, dataset quantity and diversity, and training duration. The model uses a basic pre-training technique that includes the model, data, and training. Although comparable, in-context learning systematically investigates various settings for learning within the context. A pre-trained model's weights are updated by FT, which comprises training on supervised datasets relevant to the target job. Usually, tens of thousands to millions of instances with labels are employed. A key benefit of refined calibration is an excellent accomplishment, but the chief drawback is a requirement of a new, substantial dataset for each task.

Few-shot (FS) is a configuration in which the model receives a limited number of task demos at the time of interpretation as training, but no weight updates are permitted. A setting and a desired conclusion FS entail presenting K examples of the setting and conclusion, trailed by one last example of the setting, with the model being anticipated to propose the conclusion. 2,048 is the extreme quantity of examples that are suitable to the context space of the model, K is typically set to be between 10 and 100. Few-principal shot's benefits include a significant decrease in the need for task-specific data and a decreased risk of learning an excessively slender spread from huge but limited refined calibration database. The vital downside to this approach crops significantly inferior outputs than refined calibrated model. There is still a limited quantity of task-specific data needed.

One-shot (1S) is similar to a FS but precisely single demo is allowed and a natural language explanation of the task is permitted. Since it most closely resembles the manner in which particular activities are explained to humans, 1S is preferred over FS and zero-shot. With the exclusion of discontinuous compressed and nearby hooped sparse attention designs in the layers, model architecture closely resembles that of GPT-2 with a few

minor alterations, such as initialisation, pre-normalisation, and alterable tokenisation. With the context window of 2048 tokens, the feedforward layer is four times as large as the bottleneck layer. GPT models can be prone biases and also misuse and errors. While dealing with larger passage pieces, GPT-2 also has problems maintaining coherence for context. GPT-3 on the other hand has a problem that it cannot and always constantly learn because of no long-term memory and need of pre training requirements.

3.7.3 XLNET (Yang et al., 2019)

XLNET capitalises on the projected likelihood across all variations of the factoring directive, XLNet overpowers BERT and allows bidirectional contexts learning and adopts Transformer-XL pretraining concepts. While minimising their drawbacks, XLNet utilises the strongest aspects of AE and AR language modelling, and on 20 tasks, XLNet beats BERT in experiments with similar experimental conditions.

XLNet peaks sequence predicted logarithmic probability relative to all feasible factorisation permutations. Data corruption is not a prerequisite for XLNet. As a result, XLNet is not affected by the pretrain-finetune. XLNeT parametrises the Transformer-XL network to remove the ambiguity. Higher efficiency for prolonged text sequences is achieved by with Transformer-XL segment recurrence mechanism and encoding strategy and are used into XLNet. XLNET method is to pretrain on sizable unlabelled corpora of text and FT the models on subsequent tasks.

As presented in Figure 18, there are self-attention based two streams used to produce target aware outputs. The first attention stream is content based that is exactly same as self-attention. The second attention is a query based that will need more information access about the content. That is, diagonal members of query are zero. Initially, g and h are set to w and $e(x_n)$. Query and Content mask outputs g_1 and h_1 for layer one g_2 and h_2 for layer 2. Input has one order while diverse attention masks can be utilised to achieve many factorisation orders.

3.7.4 Megatron-LM (Shoeybi et al., 2019)

Megatron-LM model utilises parallel model training billions of hyperparameters. Megatron-LM utilises transformer models convergence on 512 GPUs and 8.3 billion parameters by implementing intra-layer model-parallelism. For a single NVIDIA V100 32 GB GPU, a basic training model with 1.2 billion parameters may support 39 TeraFLOPs. 15.1 PetaFLOPs per second are sustained when the model is scaled to 8.3 billion parameters on 512 GPUs. Comparing the scaling efficiency to a single GPU case, this is 76% better. Figure 19 below demonstrates the detailed scaling results of the approach. The eight-way parallel model weak scaling of 1 billion parameters per GPU is shown by long dashed lines. The model with data parallel similar configurations as model parallel combined with 64-way data-parallel is indicated by a dotted line. The regular black line depicts linear performance.

A single transformer slab entails a self-attention unit and dual-slab numerous-slab perceptron with parallelism. The first part is GEMM followed by GeLU for MLP block and expressed as:

$$Y = GeLU(XA) \tag{54}$$

A is divided along the columns $A = [A_1, A_2]$ and due to this partitioning GeLU nonlinearity could be individualistically registered to the yield of every divided GEMM and expressed as:

$$[Y_1, Y_2] = [GeLU(XA_1), GeLU(XA_2)] \tag{55}$$

The first GEMM along columns while the second GEMM along its rows is the partitioning strategy. Second GEMM's yield is scaled to the GPUs and sent to the dropout layer. This method needs one all-reduce operation (g operator) in the forward pass and one all-reduce operation (f operator) in the backward pass, splitting both GEMMs in the MLP block across GPUs. In order to accomplish parallelism for multi-headed self-attention block, GEMMs get separated parallel column style related to key (K), query (Q), and value (V) each attention head matrix multiplied locally achieved on a single GPU is fragmented across GPUs per attention head parameters and workload. Weights are shared by the transformer's input and output embedding layer. To achieve parallelism, E_w (the weight matrix for the input embedding) with column-wise E vocabulary dimension is done, where $E = [E_1, E_2]$. The parallel GEMM output $[Y_1, Y_2]$ (output embedding,) along with cross-entropy loss is diminishes the dimensions. Scalar losses are communicated rather than logits, which greatly reduces communication and boosts the effectiveness of the parallel model approach. Since Megatron-LM model utilises parallel models training billions of hyperparameters it can be very computational costly and not suitable for general purpose single GPU user uses.

3.7.5 Bloom (Scao et al., 2022)

Bloom, launched in 2022, introduces the world's largest multilingual language models to date over 176 billion parameters, and has an open collaboration model. BLOOM architecture is analogous to GPT3 and is trained with complete transparency, where AI researchers have ever collaborated across the globe on a single project, as a product. BLOOM is programmable to 13 programming languages, and can produce output in 46 natural tongues where 1.6 TB of pre-processed text is transformed to 350 B exclusive tokens, and the vocabulary size is of 250,680.

Over 1,000 research practitioners participating from 250+ institutions and 70+ countries contributed to the final run of 117 days BLOOM training located in France on Jean Zay supercomputer with and projected cost of €3M from GENCI and CNRS French research agencies. Research practitioners may use BLOOM to analyse the efficiency and behaviour by downloading, running, and studying it. BLOOM model training was started on 11/03/2022 and ended on 05/07/2022. A vocabulary size of 250,680.

BLOOM has multiple configuration availability such as: bloom (176B parameters-70 layers, 112 attention heads, 14,336-dimensional hidden layers with the 2,048 token sequence length), bloom-6b3, bloom-760m, bloom-2b5, bloom-350m, and bloom-1b3. Model is trained on massive quantities of text dataset consuming industrial-scale computations. BLOOM is analogous to GPT3 and revised from Megatron-LM GPT2, where architecture follows decoder-only blocks. Cross entropy with mean reduction is the objective function used by the model while activation function GeLU and positional encoding ALiBI is used. To the word embeddings layer model applies layer normalisation. For preprocessing tokenisation, a learned sub-word tokeniser is trained using byte-level BPE algorithm with no normalisation for simple pre-tokenisation rule is

used. Since BLOOM has very large number of parameters to be trained it has very high currency and computational costs. Also, model can have bias since it is open model.

After reviewing and classifying NLP models, let us now confidently talk about the potential applications.

- Analysis of sentiment: Process the piece of text or sequence and identify the sentiment.
- Translations: Automatically detect and translate different languages.
- Entity recognition: Extract and process most useful information in provided text like names, people, places, organisations and measures.
- Spam protection: Identify and detect unwanted information and protect user from it.
- Automatic corrections: Check spelling and grammar in piece of text and provide user with solution and rectifications.
- Topic identification: Process text and identify topics.
- Generation of text: Also known as natural text generation (NLG) can be used to autocomplete words, sentences or generate texts based on model learning patterns.
- Chatbots: Used to converse with users when human agents are not available.
- Recommender systems: Provide most useful recommendations-based query.
- Summarisations: Find most useful piece on information in the essay or provided text.
- Question and answers: Model uses learnt patterns to answer questions based on input queries.
- Optical character recognition (OCR): For an input image provide equivalent text output.
- Speech, image, video and text applications: speech-to-text, image-to-text or video-to-text for a given input generate the equivalent text output. text-to-speech, text-to-image, text-to-video for a given text generate equivalent desired output.
- Segmentation: For a given text of speech segment identify and extract most important segments.

4 Evaluation criteria

This section includes the important metrics and the performance evaluation outcomes.

1 Accuracy (A) – Accuracy is the percentage of accurate predictions made compared to the total cases examined and expressed as:

$$Accuracy = (TN + TP)/(FP + FN + TN + TP) \tag{56}$$

where

TN: True negative – The token count shared between the prediction and the incorrect response

TP: True positive – The token count shared between the prediction and the right response

FP: False positive – The token count shared between the prediction and but not in the right response

FN: False negative – The token count shared between the correct response but not in the prediction

2 F1-score (F1) – This statistic calculates the typical variance amongst the projected and the genuine answer. We compute their F1 by treating the prediction and the ceiled veracity as token circles. For each question, we take the maximum F1 over all of the ground truth responses, and we then average that value across all of the questions. The harmonic average of recall (R) alongside precision (P) is referred to F1-score. It can be computed with the equation:

$$F1 = 2\frac{RP}{R+P} \tag{57}$$

$$P = \frac{TP}{FP+TP} \tag{58}$$

$$R = \frac{TP}{FN+TP} \tag{59}$$

3 Exact match (EM) – This statistic counts the proportion of forecasts that perfectly match any given ground truth response. When a system's expectation and the empirical evidence are congruent positive responses, then for each Q&A duo, EM = 1, if not EM = 0. Due to the rigorous none-to-all nature of this scheme of measurement, errors for even one character result in a score of 0.

4 Matthew's correlation coefficient (MCC) – Matthew's correlation computes MCC and phi coefficient is another term for it in statistics. It serves as a quality benchmark for binary and multiclass classification quality in machine learning. MCC measure are centred which could be employed even after units have enormously dissimilar magnitudes since it contemplates together correct and erroneous corrects and wrongs. Average random prediction by 0 and MCC value ranges between +1 and –1. Faultless forecast is characterised by a factor of 1 and inverse forecast with –1. MCC is expressed as:

$$MCC = \frac{(TN*TP)-(FN*FP)}{\left((FP+TP)(FN+TP)(FP+TN)(FN+TN)\right)^{1/2}} \tag{60}$$

5 Pearson correlation coefficient (PCC) –Pearson and Spearman correlation is a mathematical metric called correlation that delivers evidence around the connection amongst dual factors. It illustrates in what way single variable perform when the supplementary variable deviates. +1 relationship occurs when variables concurrently rise or lessen. On the other hand, –1 association is present when one variable is growing while other is dipping. 0 relationship occurs if altering either does not influence the next. PCC and Spearman correlation coefficient (SCC) can be expressed as:

$$PCC = \frac{\sum_{i=1}^{S}(a_i - \overline{a})(b_i - \overline{b})}{\left(\sum_{i=1}^{S}(\alpha_i - \overline{a})^2\right)^{1/2}\left(\sum_{i=1}^{S}(b_i - \overline{b})^2\right)^{1/2}} \tag{61}$$

$$SCC = \frac{\sum_{i=1}^{S}\left(r[a_i] - \overline{r[a]}\right)\left(r[b_i] - \overline{r[b]}\right)}{\left(\sum_{i=1}^{S}\left(r[a_i] - r[a]\right)^{2^{1/2}}\right)\left(\sum_{i=1}^{S}\left(r[b_i] - \overline{r[b]}\right)^2\right)^{1/2}} \tag{62}$$

where S is the count of observations, the sample set is $(a, b) = \{(a_1, b_1), \ldots, (a_s, b_s)\}$, $\overline{a}$ and $\overline{b}$ are sample means, $r[a_i]$ is rank for a_i i^{th} element, $r[b_i]$ is rank for b_i i^{th} element, while $\overline{r[a]}$ and $\overline{r[b]}$ are means of ranks.

6 GLUE (Wang et al., 2018) benchmark score – GLUE, a set of tools for developing, testing, and assessing systems of natural language understanding is called general language understanding evaluation standard. Following is the list of GLUE tasks and used metric for each one of the tasks and each task dataset is described in next Section 5. CoLA: MCC, SST-2: A, MRPC: F1/A, STS-B: PSC, QQP: F1 /A, MNLI: A, QNLI: A, RTE: A, WNLI: A are respective task/metrics.

7 SuperGLUE (Wang et al., 2019) benchmark score – SuperGLUE is modelled on GLUE and has a set of extra challenging language comprehension problems, better resources, and a new public scoreboard. Following is the list of SuperGLUE tasks and used metric for each one of the tasks and each task dataset is described in next Section 5. BoolQ: A, CB: F1/A, COPA: A, MultiRC: F1/EM, RTE: A, WiC: A, WSC: A, ReCoRD: F1/A, AX: A are respective tasks/metrics.

5 Experimental results

Here we first discuss the diverse repository utilised for the trial and results. SQuAD (Rajpurkar et al., 2016) dataset stands for Stanford Question Answering Dataset. A portion of the phrase from the applicable analysis material serves as the response to each enquiry in SQuAD, an understanding knowledge database encompassed of queries succumbed by crowd workforces Wikipedia pages assembly.

SQuAD2.0 (Rajpurkar et al., 2018) integrates the 100,000 issues from SQuAD1.1 with more than 50,000 problems that can be solved but were produced by crowd workers to appear to be answered. In order for computers to perform effectively on SQuAD2.0, they should not only respond to inquiries when they can, but also recognise when a response is not justified by the text and refrain from responding.

GLUE (Wang et al., 2018) dataset consists of first single-sentence tasks, second as similarity and paraphrase tasks while third are inference tasks.

Single-sentence tasks: the corpus of linguistic acceptability, or COLA, compiles evaluations of English permissibility from language concept journal articles and books. Evaluates between –1 to 1, with 0 being an uneducated guess as the effectiveness of classification algorithm.

SST-2, the Stanford Sentiment Treebank comprises of ratings of movies with emotion comments added by people. Predicting the tone of a statement is the job and utilises only sentence-level labels in the two-way (positive/negative) class.

Similarity and paraphrase tasks: The Microsoft Research Paraphrase Corpus, or MRPC, is a catalogue of sentence pairs that have been mechanically retrieved from online sources of news and have had the semantic equivalency of the phrases annotated by humans while unbalanced classes (68% in favor).

The Quora question pairings database, often known as QQP, is a collection of question pairs from the Quora community forum. Identifying linguistic equivalency between two queries is the problem at hand. The class distribution in QQP is also lopsided (63% negative), much like in MRPC.

The semantic textual similarity benchmarking, or STS-B, is a set of phrase sets culled from media stories, subtitles for videos and images, and information from natural language interpretation.

Inference tasks: A library of phrase twos with text-based entailment labels were crowdsourced and called the multi-genre natural language inference corpus (MNLI). Estimate if a premise implies a hypothesis (entailment), opposes a hypothesis (contradiction), or neither when provided a premise statement and a hypothesis phrase (neutral).

The Stanford Question Answering Database, or QNLI, is a question-answering collection of question-paragraph tuples, each of which includes a phrase from Wikipedia that answers the accompanying question (written by an annotator).

The recognising textual entailment (RTE) datasets are the result of several text entailment competitions that take place every year. Records from RTE1, RTE2, RTE3, and RTE5 are pooled. Samples are created using information from Wikipedia and the media.

The Winograd Schema Challenge, often known as WNLI, requires a computer to interpret a sentence, including a pronoun, and choose the pronoun's referent from a variety of options.

SuperGLUE (Wang et al., 2019) dataset consists of: Boolean questions (BoolQ) is a question-answer exercise in which each sample comprises a brief excerpt and a related yes/no inquiry. Private Google search engine members submit the queries and are matched with a Wikipedia page passage containing a solution. The CommitmentBank (CB) is a collection of brief writings wherein engrained clauses may be found in at least one phrase. Each of these embedded sentences has a note indicating how strongly the author of the text seems to believe it to be true.

Choice of plausible alternatives (COPA) is a causal thinking job where a computer must decide between two options to figure out the origin or consequence of a particular premise.

Multi-sentence reading comprehension (MultiRC) entails a situation passage, an inquiry regarding the same passage, and potential answers set for every case in a quality assurance work. The computer must foretell which responses are accurate and those that are unreliable.

The reading comprehension with commonsense reasoning dataset (ReCoRD) is a QA job with many choices. Every specimen entails news story and Cloze-style query regarding story that leaves out one element. The computer must choose the masked-out entity from a set of potential candidates in the given paragraph, in which the same item

could be represented in a variety of acceptable surface forms. CNN and Daily Mail content are used.

RTE is a database of various yearly contests in word-based deductions.

Word-in-context (WiC) is a binary classifier of phrase tuples challenge for disambiguation of word sense. The aim is to assess if a polysemous word is employed in both phrases with the same sense when offered two message samples and the word. Wiktionary, WordNet, and VerbNet databanks are used to create the phrases.

Winograd schema challenge (WSC) is a coreference-solving job, and the samples are a pronoun-filled phrase and a table of the statement's noun phrases. Out of the options given, the algorithm must choose the proper pronoun referent. Winograd schemas are intended to be solved using expertise that is commonplace and practical wisdom. The initial WSC information, as well as those made available by the related group commonsense reasoning, were used to create the training and validation samples. The authors of the original dataset used fiction novels as the source for the test instances.

5.1 Comprehensive analysis and comparison

We did a thorough comparative examination of the cutting-edge NLP methods provided in part in this Section 3. Here, we will showcase the comparison of diverse models discussed earlier and the performance of these models for each evaluation criteria. Findings of this detailed investigation are shown in a table format and bar charts.

In Tables 1 and 2, we provide efficiency assessment findings for SQuAD1 (Rajpurkar et al., 2016) and SQuAD2 (Rajpurkar et al., 2018). We observe that for SQuAD1, Megatron-LM performance is the finest, and logistic regression is the model that performs the worst. While for SQuAD2, ELECTRA performance is the strongest, and logistic regression is the model that performs the worst.

In Table 3, we present performance evaluation results for GLUE (Wang et al., 2018). Observe, T5 has the finest performance and Logistic Regression is the worst performing model.

In this table, we display the outcomes of the performance review and results for SuperGLUE (Wang et al., 2019). We observe, that METRO has the finest performance and logistic regression is the model that performs the worst.

This section, we performed a relative comprehensive scrutiny of the state-of-art NLP techniques presented in Section 3 and the consequences are showcased in the tabular form Table 5. Column features of Table 5 are expressed as – model name: name of the model, timeline: year when the model was released, classification category: grouping of model class – AE transformers vs. AR transformers, model configuration: different pre-trained models formations available, total parameters: total number of the configurable hyper-parameters in each configuration of the model, total layers: total count of layers in the model, hidden layers: total count of hidden layers in the model, count of heads: total count of the attention head mechanisms. Our conclusion in this is that the most crucial element affecting how accurately any algorithm performs is majorly the total number of parameters that enable each model to retain knowledge. Apart from this, the other major attribute is number of heads which allow model to encompass a wider variety of word associations, while the count of neurons in concealed layers affects the model performance as models having fewer neurons in the concealed layers effect in underfitting.

It is evident from the discussion in Section 3, that, the computational efficiency of all techniques is majorly influenced by the architecture as well. The distillation and teacher-student architectures demonstrates that performance of large models can be achieved in smaller ones by using these smarter architecture approaches. Also, from our analysis, the idea of which model to use between AE transformers vs. ARs depends on whether we want to learn an encoded representation of the inputs by corrupting inputs and generating the original variants, then we should be using AE model whereas if we use all previous predictions for generating the next one, in a cyclical fashion, we need AR model.

From our preformed analysis of SQuAD1 (Table 1 – tabular representation, Figure 20 – pictorial representation) and SQuAD2 (Table 2 – tabular representation, Figure 21 – pictorial representation), we observe that logistic regression is the worst performing model even below the human baselines for SQuAD1 and SQuAD2 tasks. The reason being that logistic regression can have good accuracy for simple data sets where the dataset is linearly separable, but it does not perform well in complex NLP tasks of SQuAD1 and SQuAD2 as the target labels have no linear correlation with the features, and thus model cannot predict targets with very good accuracy even on the training data. While for SQuAD1, Megatron-LM has the best performance and for SQuAD2, ELECTRA is the best performance model. Megatron-LM mostly performs well at SQuAD1 due to its massive number of parameters and ability to retain knowledge. In contrast, ELECTRA performs well in SQuAD2 because of model extensions such as weight sharing, smaller generators, and the ability to replace tokens with a plausible alternative word using a generator (RTD).

Our analysis also shows that for GLUE (Table 3 – tabular representation, Figure 22 – pictorial representation), T5 has the finest performance and logistic regression is the worst performing model, while for SuperGLUE (Table 4 – tabular representation, Figure 23 – pictorial representation), METRO has the most remarkable performance and logistic regression is the worst performing model. The reason for the lousy performance of logistic regression remains the same, as we stated in the discussion above. However, the success of the T5 model is due to the fact that model is prepared beforehand on a variety of assignments that combine unsupervised and supervised activities before being transformed to texts. T5 also employs related scalar embeddings with left and right source padding. As contrasting to that, the success of METRO is majorly due to efficient denoising pretraining and the massive number of parameters allowing model to retain the knowledge.

6 Conclusions

In this paper, we delivered a reader 360-view for high-quality state-of-art NLP research efforts by investing minimal time and efforts. In this study, we provide a comprehensive evaluation of various state-of-art models of NLP. While all models described in the text are advantageous and powerful and each one has its strengths and suited for different use cases.

In our study, for GLUE, T5 is the finest performing and logistic regression is the worst performing model. For SuperGLUE, METRO has the finest efficiency, and logistic regression is the lowest performing model. For SQuAD1, Megatron-LM has the most incredible efficiency, Logistic Regression is the lowest performing model, and for

SQuAD2, ELECTRA has the finest performance, and logistic regression is the worst performing model.

This study was conducted using MSI RTX 3070 8 GB GPU, so when the reader executes the models on their systems, their performances might be slightly different. Our work can be used further to examine the effectiveness of various models for different configurations of batch magnitudes, dropout values, and a variety of hyper-parameters. Additionally, these models can be optimised to boost their performance based on the need to get better performance and results. However, this paper will assist readers in selecting the top pre-trained models in a shorter time frame.

Figures and Tables are available on request by emailing the corresponding author or can be obtained under https://drive.google.com/drive/folders/1fE_G8FGSMSvsQBmE4bqNySgdbEc4mn_4?usp=drive_link.

References

Abiodun, O.I., Jantan, A., Omolara, A.E., Dada, K.V., Mohamed, N.A. and Arshad, H. (2018) 'State-of-the-art in artificial neural network applications: a survey', *Heliyon*, Vol. 4, No. 11, pp.15–17.

Bajaj, P., Xiong, C., Ke, G., Liu, X., Di, D.H., Tiwary, S., Liu, T-Y., Bennett, P., Song, X. and Gao, J. (2022) *METRO: Efficient Denoising Pretraining of Large-Scale Autoencoding Language Models with Model Generated Signals*, arXiv preprint arXiv:2204.06644.

Beltagy, I., Peters, M.E. and Cohan, A. (2020) *Longformer: The Long-Document Transformer*, arXiv preprint arXiv:2004.05150.

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplany, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. and Amodei, D. (2020) *Language Models are Few-Shot Learners*, arXiv preprint arXiv:2005.14165.

Burke, E.K. and Kendall, G. (2013) *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, Springer Science & Business Media, London, UK.

Cambria, E. and White, B. (2014) 'Jumping NLP curves: a review of natural language processing research', *IEEE Computational Intelligence Magazine*, No. 2, pp.48–57.

Chao, Y., Peng, J., Lee, K.L. and Ingersoll, G.M. (2022) 'An introduction to logistic regression analysis and reporting', *The Journal of Educational Research*, Vol. 96, No. 1, pp.3–14.

Clark, K., Luong, M-T., Le, Q.V. and Manning, C.D. (2020) 'ELECTRA: pre-training text encoders as discriminators rather than generators', *ICLR 2020*, arXiv preprint arXiv:2003.10555.

Crevier, D. (1993) *AI: The Tumultuous Search for Artificial Intelligence*, pp.271–279, Basic Books, New York, USA.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J. and Salakhutdinov, R. (2019) 'Transformer-XL: attentive language models beyond a fixed-length context', *57th Annual Meeting of the Association for Computational Linguistics*, pp.19–1285.

Devlin, J., Chang, M-W., Lee, K. and Toutanova, K. (2018) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv preprint arXiv:1810.04805.

Domingos, P. (2015) *The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World*, Basic Books, Chap. 9, New York, NY, USA.

European Commission (2020) *White Paper: On Artificial Intelligence – A European Approach to Excellence and Trust*, p.1, Brussels, Belgium.

Lample, G. and Conneau, A. (2019) 'Cross-lingual language model pretraining', *NIPS'19: International Conference on Neural Information Processing Systems*, No. 33, pp.7059–7069.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R. (2019) 'ALBERT: a Lite BERT for self-supervised learning of language representations', *ICLR 2020 Conference*, arXiv preprint arXiv:1909.11942.

Lenat, D. and Guha, R.V. (1989) *Building Large Knowledge-Based Systems*, Addison-Wesley Chap. Introduction, Denver, Colorado, USA.

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V. and Zettlemoyer, L. (2020) 'BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension', *58th Annual Meeting of the Association for Computational Linguistics*, pp.7871–7880.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. (2019) *RoBERTa: A Robustly Optimized BERT Pretraining Approach*, arXiv preprint arXiv:1907.11692.

Luger, G. and Stubblefield, W. (2004) *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, No. 5, pp.127–133, 509–530, 530–541, Benjamin/Cummings, Boston, USA.

Mcculloch, W.S. and Walter, P.A. (1943) 'Logical calculus of the ideas immanent in nervous activity', *Bulletin of Mathematical Biophysics*, No. 5, pp.115–133.

Nelder, J. and Wedderburn, R. (1972) 'Generalized linear models', *Journal of the Royal Statistical Society*, Vol. 135, No. 135, pp.370–384.

Nilsson, N. (1998) *Artificial Intelligence: A New Synthesis*, Chap. 4.26, Morgan Kaufmann, San Francisco CA, USA.

Poole, D., Mackworth, A. and Goebel, R. (1998) *Computational Intelligence: A Logical Approach*, pp.56–163, Oxford University Press, New York, USA.

Poria, S., Cambria, E., Bajpai, R. and Hussain, A. (2017) 'A review of affective computing: from unimodal analysis to multimodal fusion', *Information Fusion*, Vol. 37, No. 37, pp.98–125.

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018) *Improving Language Understanding by Generative Pre-Training*, Vol. 12, OpenAI Assets, Pre Print arXiv:2012.11747.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. and Sutskever, I. (2019) *Language Models are Unsupervised Multitask Learners*, OpenAI Technical Report, ePrint arXiv:2005.14165.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Wei, L.P. and Liu, J. (2020) 'Exploring the limits of transfer learning with a unified text-to-text transformer', *The Journal of Machine Learning Research*, Vol. 21, No. 21, pp.5485–5551.

Rajpurkar, P., Jia, R. and Liang, P. (2018) 'Know what you don't know: unanswerable questions for SQuAD', *56th Annual Meeting of the Association for Computational Linguistics*, No. 2, pp.18–2124.

Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. (2016) 'SQuAD: 100,000+ questions for machine comprehension of text', *2016 Conference on Empirical Methods in Natural Language Processing Association for Computational Linguistics*, pp.2383–2392.

Russell, S.J. and Norvig, P. (2003) *Artificial Intelligence: A Modern Approach*, No. 2, pp.1110–1129, 537–581, 863–898, Prentice Hall, New Jersey, USA.

Sak, H., Senior, A. and Beaufays, F. (2014) 'Long short-term memory recurrent neural network architectures for large scale acoustic modeling', *Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp.338–342.

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019) *DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter*, arXiv preprint arXiv:1910.01108.

Scao, T.L., Wang, T., Hesslow, D., Saulnier, L., Bekman, S., Bari, M.S., Biderman, S., Elsahar, H., Muennighoff, N., Phang, J., Press, O., Raffel, C., Sanh, V., Shen, S., Sutawika, L., Tae, J., Yong, Z.X., Launay, J. and Beltagy, I. (2022) 'What language model to train if you have one million GPU hours', *EMNLP*, arXiv preprint arXiv:2210.15424.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J. and Catanzaro, B. (2019) *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*, arXiv preprint arXiv:1909.08053.

Tumer, K. and Ghosh, J. (2003) 'Bayes error rate estimation using classifier ensembles', *International Journal of Smart Engineering System Design*, Vol. 5, No. 5, pp.95–109.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. and Polosukhin, I. (2017) *Attention Is All You Need*, arXiv preprint arXiv:1706.03762.

Wang, A., Nangia, N., Pruksachatkun, Y., Bowman, S.R., Singh, A., Michael, J., Hill, F. and Levy, O. (2019) 'SuperGLUE: a stickier benchmark for general-purpose language understanding systems', *NIPS'19 33rd International Conference on Neural Information Processing Systems*, pp.3266–3280.

Wang, A., Singh, A., Bowman, S.R., Levy, O., Michael, J., Paul, G. and Hill, F. (2018) 'GLUE: a multi-task benchmark and analysis platform for natural language understanding', *EMNLP 2018 Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP Association for Computational Linguistics*, pp.353–355.

Yamada, I., Asai, A., Shindo, H., Takeda, H. and Matsumoto, Y. (2020) 'LUKE: deep contextualized entity representations with entity-aware self-attention', *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp.6442–6454, Association for Computational Linguistics, arXiv Preprint arXiv:2010.01057.

Yang, Z., Yang, Y., Carbonell, J., Salakhutdino, R., Dai, Z. and Quoc, V. (2019) 'XLNet: generalized autoregressive pretraining for language understanding', *NIPS'19 33rd International Conference on Neural Information Processing Systems*, pp.5753–5763.