To appear as a part of Prof. Ali Ghodsi's material on deep learning.

Attention Mechanism, Transformers, BERT, and GPT: Tutorial and Survey

Benyamin Ghojogh BGHOJOGH@UWATERLOO.CA
Department of Electrical and Computer Engineering,
Machine Learning Laboratory, University of Waterloo, Waterloo, ON, Canada

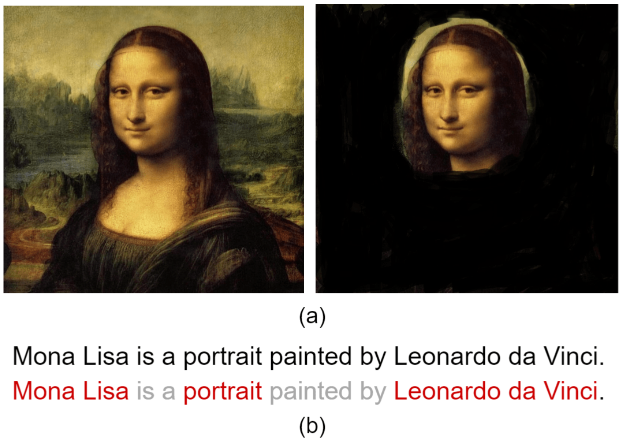
Ali Ghodsi ALI.GHODSI@UWATERLOO.CA
Department of Statistics and Actuarial Science & David R. Cheriton School of Computer Science,
Data Analytics Laboratory, University of Waterloo, Waterloo, ON, Canada

Abstract

This is a tutorial and survey paper on the attention mechanism, transformers, BERT, and GPT. We first explain attention mechanism, sequence-to-sequence model without and with attention, self-attention, and attention in different areas such as natural language processing and computer vision. Then, we explain transformers which do not use any recurrence. We explain all the parts of encoder and decoder in the transformer, including positional encoding, multihead self-attention and cross-attention, and masked multihead attention. Thereafter, we introduce the Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT) as the stacks of encoders and decoders of transformer, respectively. We explain their characteristics and how they work.

1. Introduction

When looking at a scene or picture, our visual system, so as a machine learning model (Li et al., 2019b), focuses on or attends to some specific parts of the scene/image with more information and importance and ignores the less informative or less important parts. For example, when we look at the Mona Lisa portrait, our visual system attends to Mona Lisa's face and smile, as Fig. 1 illustrates. Moreover, when reading a text, especially when we want to try fast reading, one technique is skimming (Xu, 2011) in which our visual system or a model skims the data with high pacing and only attends to more informative words of sentences (Yu et al., 2018). Figure 1 shows a sample sentence and highlights the words to which our visual system focuses more in skimming.


(a)

Mona Lisa is a portrait painted by Leonardo da Vinci.

Mona Lisa is a portrait painted by Leonardo da Vinci.

(b)

Figure 1. Attention in visual system for (a) seeing a picture by attending to more important parts of scene and (b) reading a sentence by attending to more informative words in the sentence.

The concept of attention can be modeled in machine learning where attention is a simple weighting of data. In the attention mechanism, explained in this tutorial paper, the more informative or more important parts of data are given larger weights for the sake of more attention. Many of the state-of-the-art Natural Language Processing (NLP) (Indurkhya & Damerau, 2010) and deep learning techniques in NLP (Socher et al., 2012) use attention.

Transformers are also autoencoders which encode the input data to a hidden space and then decode those to another domain. Transfer learning is widely used in NLP (Wolf et al., 2019b). Transformers can also be used for transfer learning. Recently, transformers were proposed merely composed of attention modules, excluding recurrence and any recurrent modules (Vaswani et al., 2017). This was a great breakthrough. Prior to the proposal of transformers with only attention mechanism, recurrent models such as Long-Short Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and Recurrent Neural Network (RNN) (Kombrink et al., 2011) were mostly used for NLP. Also, other NLP models such as word2vec (Mikolov et al., 2013a;b; Goldberg & Levy, 2014; Mikolov et al., 2015) and GloVe (Pennington et al., 2014) were state-of-the-art before the appearance of transformers. Recently, two NLP methods, named Bidirectional Encoder Representations from Transformers (BERT) and Generative Pre-trained Transformer (GPT), are proposed which are the stacks of encoders and decoders of transformer, respectively.

In this paper, we introduce and review the attention mechanism, transformers, BERT, and GPT. In Section 2, we introduce the sequence-to-sequence model with and without attention mechanism as well as the self-attention. Section 3 explains the different parts of encoder and decoder of transformers. BERT and its different variants of BERT are introduced in Section 4 while GPT and its variants are introduced in Section 5. Finally, Section 6 concludes the paper.

2. Attention Mechanism

2.1. Autoencoder and the Context Vector

Consider an autoencoder with encoder and decoder parts where the encoder gets an input and converts it to a *context vector* and the decoder gets the context vector and converts to an output. The output is related to the input through the context vector in the so-called *hidden space*. Figure 2 illustrates an autoencoder with the encoder (left part of autoencoder), hidden space (in the middle), and decoder (right part of autoencoder) parts. For example, the input can be a sentence or a word in English and the output is the same sentence or word but in French. Assume the word “elephant” in English is fed to the encoder and the word “l’éléphant” in French is output. The context vector models the concept of elephant which also exists in the mind of human when thinking to elephant. This context is abstract in mind and can be referred to any fat, thin, huge, or small elephant (Perlovsky, 2006). Another example for transformer is transforming a cartoon image of elephant to picture of a real elephant (see Fig. 2). As the autoencoder is transforming data from a domain to a hidden space and then to another domain, it can be used for domain transformation (Wang et al., 2020), domain adaptation (Ben-David et al., 2010), and domain generalization (Dou et al., 2019). Here, every context is modeled as a vector in the hidden space. Let the context vector be denoted by $\boldsymbol{c} \in \mathbb{R}^p$ in the p-dimensional hidden space.

2.2. The Sequence-to-Sequence Model

Consider a sequence of ordered tokens, e.g., a sequence of words which make a sentence. We want to transform this sequence to another related sequence. For example, we want to take a sentence in English and translate it to the same sentence in French. This model which transforms a sequence to another related sequence is named the sequence-to-sequence model (Bahdanau et al., 2015).

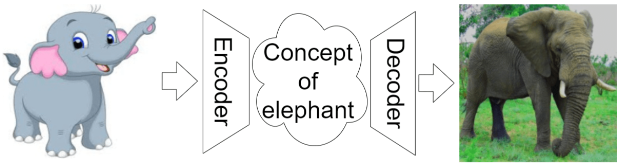


Figure 2. Using an autoencoder for Transformation of one domain to another domain. The used images are taken from the PACS dataset (Li et al., 2017).

Suppose the number of words in a document or considered sentence be n. Let the ordered input tokens or words of the sequence be denoted by $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_{i-1}, \boldsymbol{x}_i, \dots, \boldsymbol{x}_n\}$ and the output sequence be denoted by $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \dots, \boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \dots, \boldsymbol{y}_n\}$. As Fig. 3-a illustrates, there exist latent vectors, denoted by $\{\boldsymbol{l}_i\}_{i=1}^n$, in the decoder part for every word. In the sequence-to-sequence model, the probability of generation of the i-th word conditioning on all the previous words is determined by a function $g(.)$ whose inputs are the immediate previous word $\boldsymbol{y}_{i-1}$, the i-th latent vector $\boldsymbol{l}_i$, and the context vector $\boldsymbol{c}$:

$$\mathbb{P}(\boldsymbol{y}_i | \boldsymbol{y}_1, \dots, \boldsymbol{y}_{i-1}) = g(\boldsymbol{y}_{i-1}, \boldsymbol{l}_i, \boldsymbol{c}). \tag{1}$$

Figure 3-a depicts the sequence-to-sequence model. In the sequence-to-sequence model, every word $\boldsymbol{x}_i$ produces a hidden vector $\boldsymbol{h}_i$ in the encoder part of the autoencoder. The hidden vector of every word, $\boldsymbol{h}_i$, is fed to the next hidden vector, $\boldsymbol{h}_{i+1}$, by a projection matrix $\boldsymbol{W}$. In this model, for the whole sequence, there is only one context vector $\boldsymbol{c}$ which is equal to the last hidden vector of the encoder, i.e., $\boldsymbol{c} = \boldsymbol{h}_n$. Note that the encoder and decoder in the sequence-to-sequence model can be any sequential model such as RNN (Kombrink et al., 2011) or LSTM (Hochreiter & Schmidhuber, 1997).

2.3. The Sequence-to-Sequence Model with Attention

The explained sequence-to-sequence model can be with attention (Chorowski et al., 2014; Luong et al., 2015). In the sequence-to-sequence model with attention, the probability of generation of the i-th word is determined as (Chorowski et al., 2014):

$$\mathbb{P}(\boldsymbol{y}_i | \boldsymbol{y}_1, \dots, \boldsymbol{y}_{i-1}) = g(\boldsymbol{y}_{i-1}, \boldsymbol{l}_i, \boldsymbol{c}_i). \tag{2}$$

Figure 3-b shows the sequence-to-sequence model with attention. In this model, in contrast to the sequence-to-sequence model which has only one context vector for the whole sequence, this model has a context vector for every word. The context vector of every word is a linear combination, or weighted sum, of all the hidden vectors; hence,

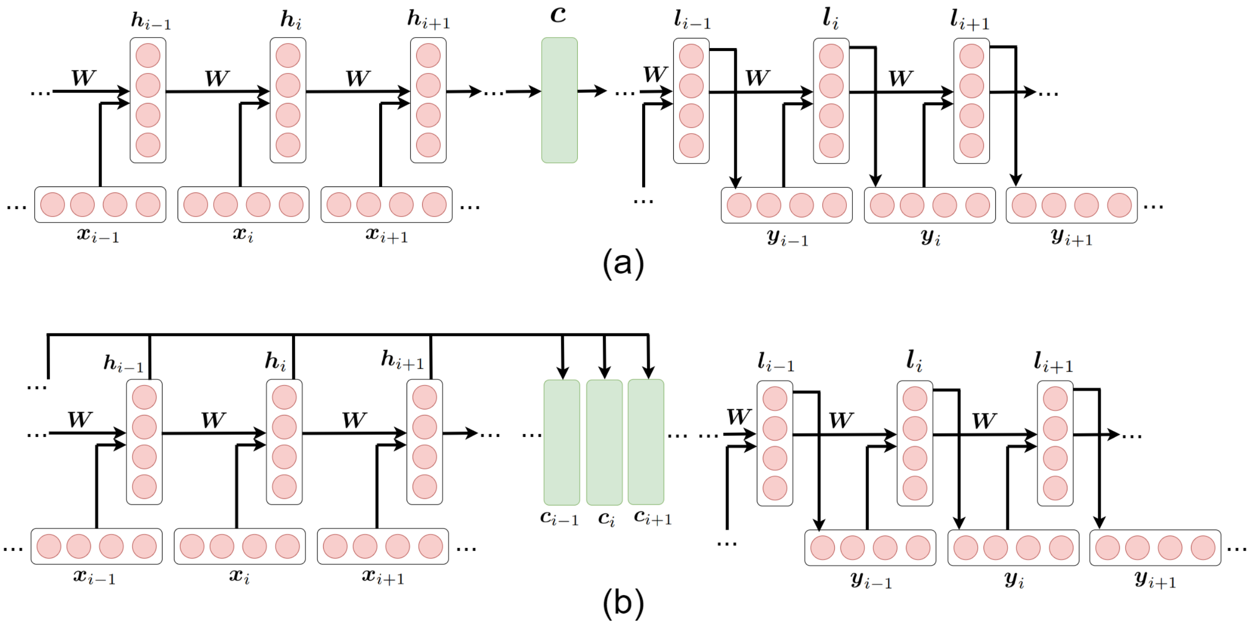


Figure 3. Sequence-to-sequence model (a) without and (b) with attention.

the i-th context vector is:

$$\boldsymbol{c}_i = \sum_{j=1}^{n} a_{ij} \boldsymbol{h}_j, \tag{3}$$

where $a_{ij} \geq 0$ is the weight of $\boldsymbol{h}_j$ for the i-th context vector. This weighted sum for a specific word in the sequence determines which words in the sequence have more effect on that word. In other words, it determines which words this specific word "attends" to more. This notion of weighted impact, to see which parts have more impact, is called "attention". It is noteworthy that the original idea of arithmetic linear combination of vectors for the purpose of word embedding, similar to Eq. (3), was in the Word2Vec method (Mikolov et al., 2013a;b).

The sequence-to-sequence model with attention considers a notion of similarity between the latent vector $\boldsymbol{l}_{i-1}$ of the decoder and the hidden vector $\boldsymbol{h}_j$ of the encoder (Bahdanau et al., 2015):

$$\mathbb{R} \ni s_{ij} := \text{similarity}(\boldsymbol{l}_{i-1}, \boldsymbol{h}_j). \tag{4}$$

The intuition for this similarity score is as follows. The output word $\boldsymbol{y}_i$ depends on the previous latent vector $\boldsymbol{l}_{i-1}$ (see Fig. 3) and and the hidden vector $\boldsymbol{h}_j$ depends on the input word $\boldsymbol{x}_j$. Hence, this similarity score relates to the impact of the input $\boldsymbol{x}_j$ on the output $\boldsymbol{y}_i$. In this way, the score s_{ij} shows the impact of the j-th word to generate the i-th word in the sequence. This similarity notion can be a neural network learned by backpropagation (Rumelhart et al., 1986).

In order to make this score a probability, these scores should sum to one; hence, we make its softmax form as (Chorowski et al., 2014):

$$\mathbb{R} \ni a_{ij} := \frac{e^{s_{ij}}}{\sum_{k=1}^{n} e^{s_{ik}}}. \tag{5}$$

In this way, the score vector $[a_{i1}, a_{i2}, \dots, a_{in}]^\top$ behaves as a discrete probability distribution. Therefore, In Eq. (3), the weights sum to one and the weights with higher values attend more to their corresponding hidden vectors.

2.4. Self-Attention

2.4.1. The Need for Composite Embedding

Many of the previous methods for NLP, such as word2vec (Mikolov et al., 2013a;b; Goldberg & Levy, 2014; Mikolov et al., 2015) and GloVe (Pennington et al., 2014), used to learn a representation for every word. However, for understanding how the words relate to each other, we can have a composite embedding where the compositions of words also have some embedding representation (Cheng et al., 2016). For example, Fig. 4 shows a sentence which highlights the relation of words. This figure shows, when reading a word in a sentence, which previous words in the sentence we remember more. This relation of words shows that we need to have a composite embedding for natural language embedding.

2.4.2. Query-Retrieval Modeling

Consider a database with keys and values where a query is searched through the keys to retrieve a value (Garcia-Molina et al., 1999). Figure 5 shows such database. We

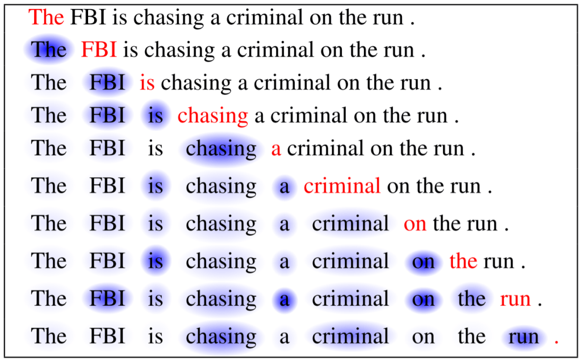
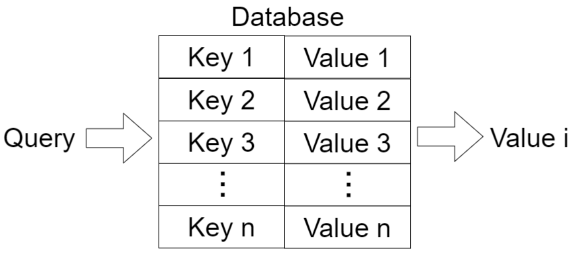

The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .
The FBI is chasing a criminal on the run .

Figure 4. The relation of words in a sentence raising the need for composite embedding. The credit of this image for (Cheng et al., 2016).



Key 1	Value 1
Key 2	Value 2
Key 3	Value 3
⋮	⋮
Key n	Value n

Figure 5. Query-retrieval in a database.

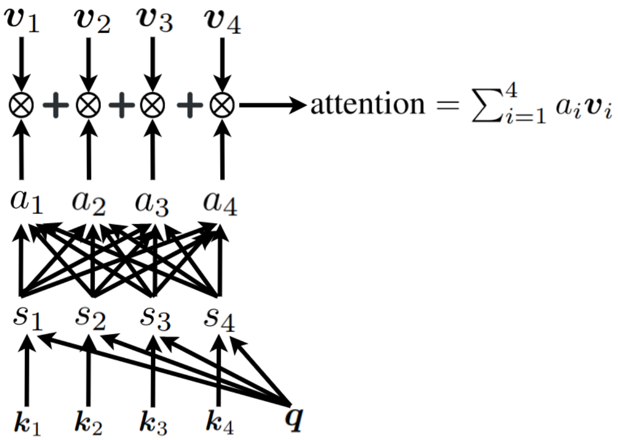


Figure 6. Illustration of Eqs. (6), (7), and (8) in attention mechanism. In this example, it is assumed there exist five (four keys and one query) words in the sequence.

can generalize this hard definition of query-retrieval to a soft query-retrieval where several keys, rather than only one key, can be corresponded to the query. For this, we calculate the similarity of the query with all the keys to see which keys are more similar to the query. This soft query-retrieval is formulated as:

$$\text{attention}(\boldsymbol{q}, \{\boldsymbol{k}_i\}_{i=1}^{n}, \{\boldsymbol{v}_i\}_{i=1}^{n}) := \sum_{i=1}^{n} a_i \boldsymbol{v}_i, \tag{6}$$

where:

$$\mathbb{R} \ni s_i := \text{similarity}(\boldsymbol{q}, \boldsymbol{k}_i), \tag{7}$$

$$\mathbb{R} \ni a_i := \text{softmax}(s_i) = \frac{e^{s_i}}{\sum_{k=1}^{n} e^{s_k}}, \tag{8}$$

and $\boldsymbol{q}$, $\{\boldsymbol{k}_i\}_{i=1}^{n}$, and $\{\boldsymbol{v}_i\}_{i=1}^{n}$ denote the query, keys, and values, respectively. Recall that the context vector of a sequence-to-sequence model with attention, introduced by Eq. (3), was also a linear combination with weights of normalized similarity (see Eq. (5)). The same linear combination is the Eq. (6) where the weights are the similarity of query with the keys. An illustration of Eqs. (6), (7), and (8) is shown in Fig. 6. Note that the similarity s_i can be any notion of similarity. Some of the well-known similarity measures are (Vaswani et al., 2017):

$$\text{inner product: } s_i = \boldsymbol{q}^\top \boldsymbol{k}_i, \tag{9}$$

$$\text{scaled inner product: } s_i = \frac{\boldsymbol{q}^\top \boldsymbol{k}_i}{\sqrt{p}}, \tag{10}$$

$$\text{general inner product: } s_i = \boldsymbol{q}^\top \boldsymbol{W} \boldsymbol{k}_i, \tag{11}$$

$$\text{additive similarity: } s_i = \boldsymbol{w}_q^\top \boldsymbol{q} + \boldsymbol{w}_k^\top \boldsymbol{k}_i, \tag{12}$$

where $\boldsymbol{W} \in \mathbb{R}^{p \times p}$, $\boldsymbol{w}_q \in \mathbb{R}^p$, and $\boldsymbol{w}_k \in \mathbb{R}^p$ are some learnable matrices and vectors. Among these similarity measures, the scaled inner product is used most often.

The Eq. (6) calculates the attention of a target word (or query) with respect to every input word (or keys) which are the previous and forthcoming words. As Fig. 4 illustrates, when processing a word which is considered as the query, the other words in the sequence are the keys. Using Eq. (6), we see how similar the other words of the sequence are to that word. In other words, we see how impactful the other previous and forthcoming words are for generating a missing word in the sequence.

We provide an example for Eq. (6), here. Consider a sentence “I am a student”. Assume we are processing the word “student” in this sequence. Hence, we have a query corresponding to the word “student”. The values are corresponding to the previous words which are “I”, “am”, and “a”. Assume we calculate the normalized similarity of the query and the values and obtain the weights 0.7, 0.2, and 0.1 for “I”, “am”, and “a”, respectively, where the weights sum to one. Then, the attention value for the word “student” is $0.7\boldsymbol{v}_{\text{I}} + 0.2\boldsymbol{v}_{\text{am}} + 0.1\boldsymbol{v}_{\text{a}}$.

2.4.3. Attention Formulation

Let the words of a sequence of words be in a d-dimensional space, i.e., the sequence is $\{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^{n}$. This d-

dimensional representation of words can be taken from any word-embedding NLP method such as word2vec (Mikolov et al., 2013a;b; Goldberg & Levy, 2014; Mikolov et al., 2015) or GloVe (Pennington et al., 2014). The query, key, and value are projection of the words into p-dimensional, p-dimensional, and r-dimensional subspaces, respectively:

$$\mathbb{R}^p \ni \boldsymbol{q}_i = \boldsymbol{W}_Q^\top \boldsymbol{x}_i, \quad (13)$$

$$\mathbb{R}^p \ni \boldsymbol{k}_i = \boldsymbol{W}_K^\top \boldsymbol{x}_i, \quad (14)$$

$$\mathbb{R}^r \ni \boldsymbol{v}_i = \boldsymbol{W}_V^\top \boldsymbol{x}_i, \quad (15)$$

where $\boldsymbol{W}_Q \in \mathbb{R}^{d \times p}$, $\boldsymbol{W}_K \in \mathbb{R}^{d \times p}$, and $\boldsymbol{W}_V \in \mathbb{R}^{d \times r}$ are the projection matrices into the low-dimensional query, key, and value subspaces, respectively. Consider the words, queries, keys, and values in matrix forms as $\boldsymbol{X} := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$, $\boldsymbol{Q} := [\boldsymbol{q}_1, \dots, \boldsymbol{q}_n] \in \mathbb{R}^{p \times n}$, $\boldsymbol{K} := [\boldsymbol{k}_1, \dots, \boldsymbol{k}_n] \in \mathbb{R}^{p \times n}$, and $\boldsymbol{V} := [\boldsymbol{v}_1, \dots, \boldsymbol{v}_n] \in \mathbb{R}^{r \times n}$, respectively. It is noteworthy that in the similarity measures, such as the scaled inner product, we have the inner product $\boldsymbol{q}^\top \boldsymbol{k}_i$. Hence, the similarity measures contain $\boldsymbol{q}^\top \boldsymbol{k}_i = \boldsymbol{x}_i^\top \boldsymbol{W}_Q \boldsymbol{W}_K^\top \boldsymbol{x}_i$. Note that if we had $\boldsymbol{W}_Q = \boldsymbol{W}_K$, this would be a kernel matrix so its behaviour is similar to the kernel for measuring similarity.

Considering Eqs. (7), and (8) and the above definitions, the Eq. (6) can be written in matrix form, for the whole sequence of n words, as:

$$\begin{aligned}\mathbb{R}^{r \times n} \ni \boldsymbol{Z} &:= \text{attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \\ &= \boldsymbol{V}\, \text{softmax}(\frac{1}{\sqrt{p}} \boldsymbol{Q}^\top \boldsymbol{K}),\end{aligned} \quad (16)$$

where $\boldsymbol{Z} = [\boldsymbol{z}_1, \dots, \boldsymbol{z}_n]$ is the attention values, for all the words, which shows how much every word attends to its previous and forthcoming words. In Eq. (16), the softmax operator applies the softmax function on every row of its input matrix so that every row sums to one.

Note that as the queries, keys, and values are all from the same words in the sequence, this attention is referred to as the "self-attention" (Cheng et al., 2016).

2.5. Attention in Other Fields Such as Vision and Speech

Note that the concept of attention can be used in any field of research and not merely in NLP. The attention concept has widely been used in NLP (Chorowski et al., 2014; Luong et al., 2015). Attention can be used in the field of computer vision (Xu et al., 2015). Attention in computer vision means attending to specific parts of image which are more important and informative (see Fig. 1). This simulates attention and exception in human visual system (Summerfield & Egner, 2009) where our brain filters the observed scene to focus on its important parts.

For example, we can generate captions for an input image using an autoencoder, illustrated in Fig. 7-a, with the attention mechanism. As this figure shows, the encoder is a Convolutional Neural Network (CNN) (LeCun et al., 1998) for extracting visual features and the decoder consists of LSTM (Hochreiter & Schmidhuber, 1997) or RNN (Kombrink et al., 2011) modules for generating the caption text. Literature has shown that the lower convolutional layers in CNN capture low-level features and different partitions of input images (Lee et al., 2009). Figure 8 shows an example of extracted features by CNN layers trained on facial images. Different facial organs and features have been extracted in lower layers of CNN. Therefore, as Fig. 7-b shows, we can consider the extracted low-layer features, which are different parts of image, as the hidden vectors $\{\boldsymbol{h}_j\}_{j=1}^{n'}$ in Eq. (4), where n' is the number of features for extracted image partitions. Similarity with latent vectors of decoder (LSTM) is computed by Eq. (4) and the query-retrieval model of attention mechanism, introduced before, is used to learn a self-attention on the images. Note that, as the partitions of image are considered to be the hidden variables used for attention, the model attends to important parts of input image; e.g., see Fig. 1.

Note that using attention in different fields of science is usually referred to as "attend, tell, and do something...". Some examples of applications of attention are caption generation for images (Xu et al., 2015), caption generation for images with ownership protection (Lim et al., 2020), text reading from images containing a text (Li et al., 2019a), translation of one image to another related image (Zhang et al., 2018; Yang et al., 2019a), visual question answering (Kazemi & Elqursh, 2017), human-robot social interaction (Qureshi et al., 2017), and speech recognition (Chan et al., 2015; 2016).

3. Transformers

3.1. The Concept of Transformation

As was explained in Section 2.1, we can have an autoencoder which takes an input data, embeds data to a context vector which simulates a concept in human's mind (Perlovsky, 2006), and generates an output. The input and output are related to each other through the context vector. In other words, the autoencoder transforms the input to a related output. An example for transformation is translating a sentence from a language to the same sentence in another language. Another example for transformer is image captioning in which the image is transformed to its caption explaining the content of image. A pure computer vision example for transformation is transforming a day-time input image to the same image but at night time.

An autoencoder, named "transformer", is proposed in the literature for the task for transformation (Vaswani et al., 2017). The structure of transformer is depicted in Fig. 9. As this figure shows, a transformer is an autoencoder consisting of an encoder and a decoder. In the following, we explain the details of encoder and decoder of a transformer.

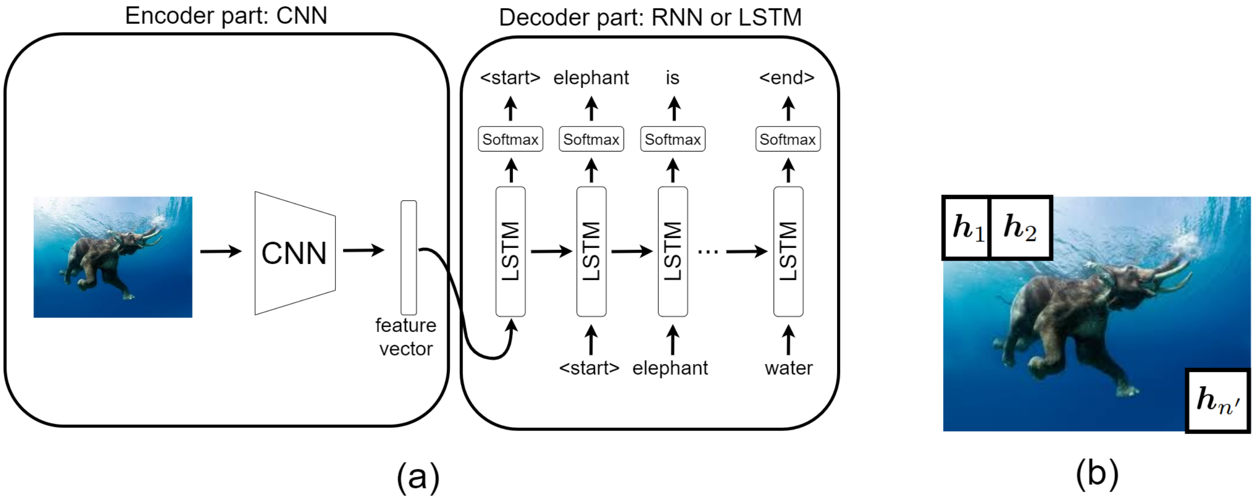


Figure 7. Using attention in computer vision: (a) transformer of image to caption with CNN for its encoder and RNN or LSTM for its decoder. The caption for this image is "Elephant is in water", (b) the convolutional filters take the values from the image for the query-retrieval modeling in attention mechanism.

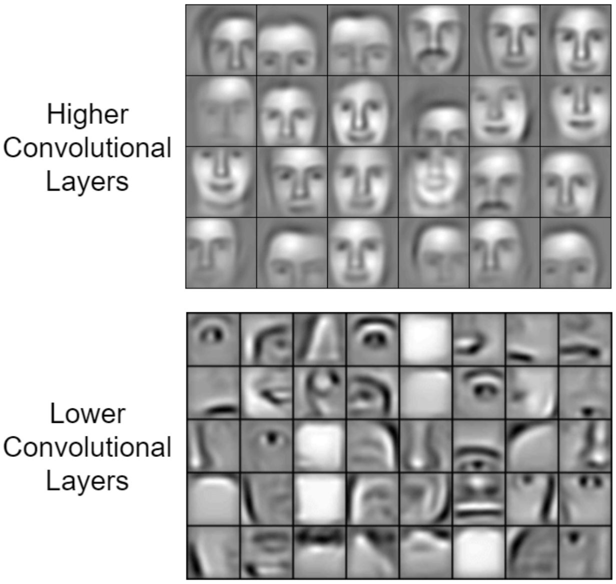


Figure 8. The low-level and high-level features learned in the low and high convolutional layers, respectively. The credit of this image is for (Lee et al., 2009).

3.2. Encoder of Transformer

The encoder part of transformer, illustrated in Fig. 9, embeds the input sequence of n words $\boldsymbol{X} \in \mathbb{R}^{d \times n}$ into context vectors with the attention mechanism. Different parts of the encoder are explained in the following.

3.2.1. Positional Encoding

We will explain in Section 3.4 that the transformer introduced here does not have any recurrence and RNN or LSTM module. As there is no recurrence and no convolution, the model has no sense of order in sequence. As the order of words is important for meaning of sentences, we need a way to account for the order of tokens or words in the sequence. For this, we can add a vector accounting for the position to each input word embedding.

Consider the embedding of the i-th word in the sequence, denoted by $\boldsymbol{x}_i \in \mathbb{R}^d$. For encoding the position of the i-th word in the sequence, the position vector $\boldsymbol{p}_i \in \mathbb{R}^d$ can be set as:

$$\begin{cases} \boldsymbol{p}_i(2j+1) := \cos\Big(\frac{i}{10000^{\frac{2j}{p}}}\Big), \\ \\ \boldsymbol{p}_i(2j) := \sin\Big(\frac{i}{10000^{\frac{2j}{p}}}\Big), \end{cases} \tag{17}$$

for all $j \in \{0, 1, \dots, \lfloor d/2 \rfloor\}$, where $\boldsymbol{p}_i(2j+1)$ and $\boldsymbol{p}_i(2j)$ denote the odd and even elements of $\boldsymbol{p}_i$, respectively. Figure 10 illustrates the dimensions of the position vectors across different positions. As can be seen in this figure, the position vectors for different positions of words are different as expected. Moreover, this figure shows that the difference of position vectors concentrate more on the initial dimensions of vectors. As Fig. 9 shows, for incorporating the information of position with data, we add the positional encoding to the input embedding:

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_i + \boldsymbol{p}_i. \tag{18}$$

3.2.2. Multihead Attention with Self-Attention

After positional encoding, data are fed to a multihead attention module with self-attention. The multihead attention is

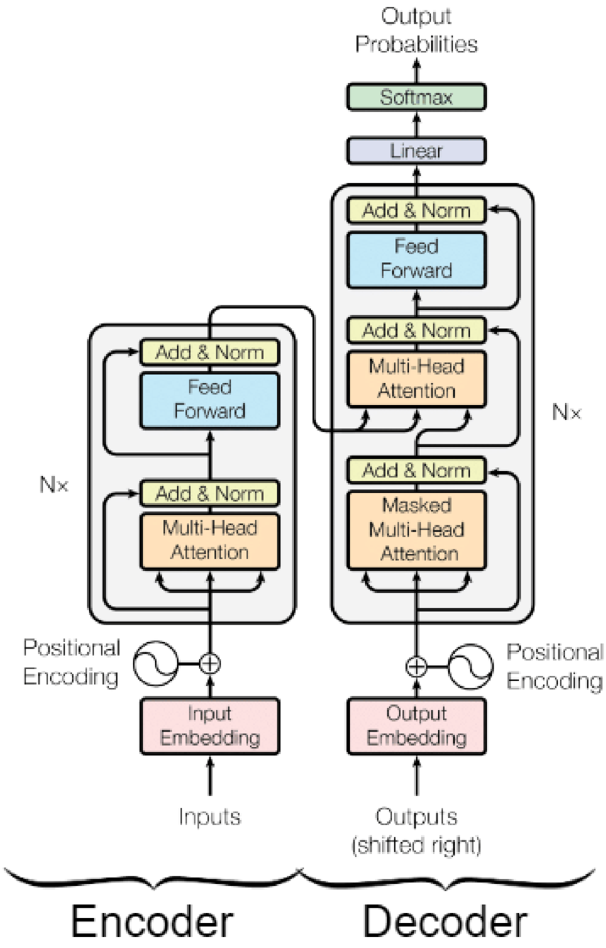


Figure 9. The encoder and decoder parts of a transformer. The credit of this image is for (Vaswani et al., 2017).

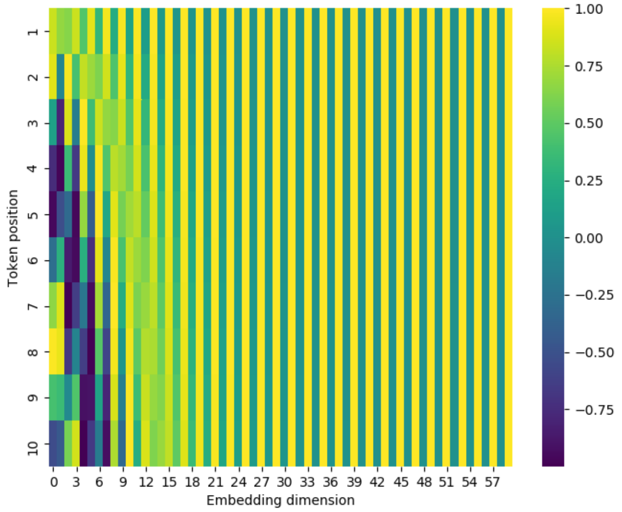


Figure 10. The vectors of positional encoding. In this example, it is assumed that $n = 10$ (number of positions of words), $d = 60$, and $p = 100$.

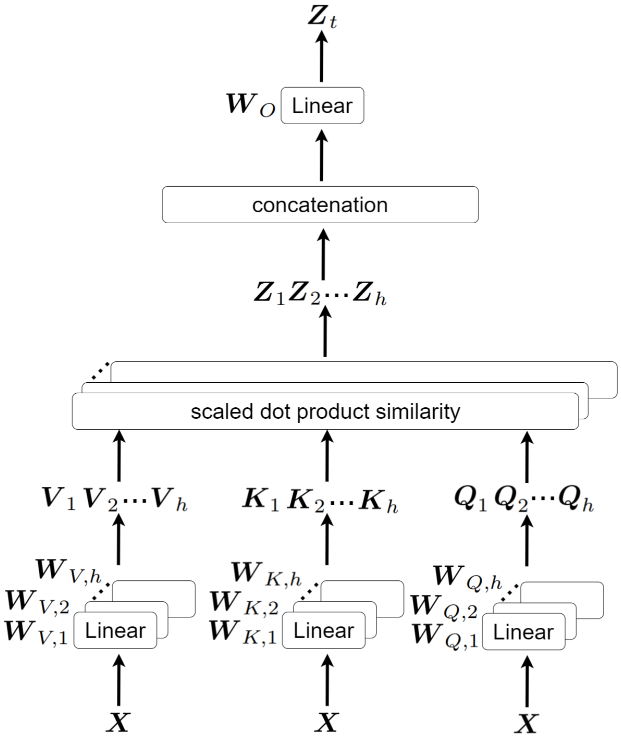


Figure 11. Multihead attention with h heads.

illustrated in Fig. 11. This module applies the attention mechanism for h times. This several repeats of attention is for the reason explained here. The first attention determines how much every word attends to other words. The second repeat of attention calculates how much every pair of words attends to other pairs of words. Likewise, the third repeat of attention sees how much every pair of pairs of words attends to other pairs of pairs of words; and so on. Note that this measure of attention or similarity between hierarchical pairs of words reminds us of the maximum mean discrepancy (Gretton et al., 2007; 2012) which measures similarity between different moments of data distributions.

As Fig. 11 shows, the data, which include positional encoding, are passed from linear layers for obtaining the queries, values, and keys. These linear layers model linear projections introduced in Eqs. (13), (15), and (14), respectively. We have h of these linear layers to generate h set of queries, values, and keys as:

$$\mathbb{R}^{p\times n} \ni \boldsymbol{Q}_i = \boldsymbol{W}_{Q,i}^\top \boldsymbol{X}, \quad \forall i \in \{1,\dots,h\}, \tag{19}$$

$$\mathbb{R}^{p\times n} \ni \boldsymbol{V}_i = \boldsymbol{W}_{V,i}^\top \boldsymbol{X}, \quad \forall i \in \{1,\dots,h\}, \tag{20}$$

$$\mathbb{R}^{r\times n} \ni \boldsymbol{K}_i = \boldsymbol{W}_{K,i}^\top \boldsymbol{X}, \quad \forall i \in \{1,\dots,h\}. \tag{21}$$

Then, the scaled dot product similarity, defined in Eq. (10)

or (16), is used to generate the h attention values $\{\boldsymbol{Z}_i\}_{i=1}^h$. These h attention values are concatenated to make a new long flattened vector. Then, by a linear layer, which is a linear projection, the total attention value, $\boldsymbol{Z}_t$, is obtained:

$$\boldsymbol{Z}_t := \boldsymbol{W}_O^\top \text{concat}(\boldsymbol{Z}_1, \boldsymbol{Z}_2, \dots, \boldsymbol{Z}_h). \tag{22}$$

3.2.3. Layer Normalization

As Fig. 9 shows, the data (containing positional encoding) and the total attention value are added:

$$\boldsymbol{Z}_t' \leftarrow \boldsymbol{Z}_t + \boldsymbol{X}. \tag{23}$$

This addition is inspired by the concept of residual introduced by ResNet (He et al., 2016). After this addition, a layer normalization is applied where for each hidden unit h_i we have:

$$h_i \leftarrow \frac{g}{\sigma}(h_i - \mu), \tag{24}$$

where μ and σ are the empirical mean and standard deviation over H hidden units:

$$\mu := \frac{1}{H}\sum_{i=1}^{H} h_i, \tag{25}$$

$$\sigma := \sqrt{\sum_{i=1}^{H}(h_i - \mu)^2}. \tag{26}$$

This is a standardization which makes the mean zero and the variance one; it is closely related to batch normalization and reduces the covariate shift (Ioffe & Szegedy, 2015).

3.2.4. Feedforward Layer

Henceforth, let $\boldsymbol{Z}_t'$ denote the total attention after both addition and layer normalization. We feed $\boldsymbol{Z}_t'$ to a feedforward network, having nonlinear activation functions, and then like before, we add the input of feedforward network to its output:

$$\boldsymbol{Z}_t'' \leftarrow \boldsymbol{R} + \boldsymbol{Z}_t', \tag{27}$$

where $\boldsymbol{R}$ denotes the output of feedforward network. Again, layer normalization is applied and we, henceforth, denote the output of encoder by $\boldsymbol{Z}_t''$. This is the encoding for the whole input sequence or sentence having the information of attention of words and hierarchical pairs of words to each other.

3.2.5. Stacking

As Fig. 9 shows, the encoder is a stack of N identical layers. This stacking is for having more learnable parameters to have enough degree of freedom to learn the whole dictionary of words. Through experiments, a good number of stacks is found to be $N = 6$ (Vaswani et al., 2017).

3.3. Decoder of Transformer

The decoder part of transformer is shown in Fig. 9. In the following, we explain the different parts of decoder.

3.3.1. Masked Multihead Attention with Self-Attention

A part of decoder is the masked multihead attention module whose input is the output embeddings $\{\boldsymbol{y}_i\}_{i=1}^n$ shifted one word to the right. Positional encoding is also added to the output embeddings for including the information of their positions. For this, we use Eq. (18) where $\boldsymbol{x}_i$ is replaced by $\boldsymbol{y}_i$.

The output embeddings added with the positional encodings are fed to the masked multihead attention module. This module is similar to the multihead attention module but masks away the forthcoming words after a word. Therefore, every output word only attends to its previous output words, every pair of output words attends to its previous pairs of output words, every pair of pairs of output words attends to its previous pairs of pairs of output words, and so on. The reason for using the masked version of multihead attention for the output embeddings is that when we are generating the output text, we do not have the next words yet because the next words are not generated yet. It is noteworthy that this masking imposes some idea of sparsity which was also introduced by the dropout technique (Srivastava et al., 2014) but in a stochastic manner.

Recall Eq. (16) which was used for multihead attention (see Section 3.2.2). The masked multihead attention is defined as:

$$\begin{aligned}\mathbb{R}^{r\times n} \ni \boldsymbol{Z}_m &:= \text{maskedAttention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \\ &= \boldsymbol{V}\,\text{softmax}\Big(\frac{1}{\sqrt{p}}\big(\boldsymbol{Q}^\top\boldsymbol{K} + \boldsymbol{M}\big)\Big),\end{aligned} \tag{28}$$

where the mask matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$ is:

$$\boldsymbol{M}(i,j) := \begin{cases} 0 & \text{if } j \leq i, \\ -\infty & \text{if } j > i. \end{cases} \tag{29}$$

As the softmax function has exponential operator, the mask does not have any impact for $j \leq i$ (because it is multiplied by $e^0 = 1$) and masks away for $j > i$ (because it is multiplied by $e^{-\infty} = 0$). Note that $j \leq i$ and $j > i$ correspond to the previous and next words, respectively, in terms of position in the sequence.

Similar to before, the output of masked multihead attention is normalized and then is added to its input.

3.3.2. Multihead Attention with Cross-Attention

As Fig. 9 illustrates, the output of masked multihead attention module is fed to a multihead attention module with cross-attention. This module is not self-attention because

Table 1. Comparison of complexities between self-attention and recurrence (Vaswani et al., 2017).

	Complexity per layer	Sequential operations	Maximum path length
Self-Attention	$\mathcal{O}(n^2 p)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
Recurrence	$\mathcal{O}(np^2)$	$\mathcal{O}(n)$	$\mathcal{O}(n)$

all its values, keys, and queries are not from the same sequence but its values and keys are from the output of encoder and the queries are from the output of the masked multihead attention module in the decoder. In other words, the values and keys come from the processed input embeddings and the queries are from the processed output embeddings. The calculated multihead attention determines how much every output embedding attends to the input embeddings, how much every pair of output embeddings attends to the pairs of input embeddings, how much every pair of pairs of output embeddings attends to the pairs of pairs of input embeddings, and so on. This shows the connection between input sequence and the generated output sequence.

3.3.3. Feedforward Layer and Softmax Activation

Again, the output of the multihead attention module with cross-attention is normalized and added to its input. Then, it is fed to a feedforward neural network with layer normalization and added to its input afterwards. Note that the masked multihead attention, the multihead attention with cross-attention, and the feedforward network are stacked for $N = 6$ times.

The output of feedforward network passes through a linear layer by linear projection and a softmax activation function is applied finally. The number of output neurons with the softmax activation functions is the number of all words in the dictionary which is a large number. The outputs of decoder sum to one and are the probability of every word in the dictionary to be the generated next word. For the sake of sequence generation, the token or word with the largest probability is the next word.

3.4. Attention is All We Need!

3.4.1. No Need to RNN!

As Fig. 9 illustrates, the output of decoder is fed to the masked multihead attention module of decoder with some shift. Note that this is not a notion of recurrence because it can be interpreted by the procedure of teacher-forcing (Kolen & Kremer, 2001). Hence, we see that there is not any recurrent module like RNN (Kombrink et al., 2011) and LSTM (Hochreiter & Schmidhuber, 1997) in transformer. We showed that we can learn a sequence using the transformer. Therefore, attention is all we need to learn a sequence and there is no need to any recurrence module. The proposal of transformers (Vaswani et al., 2017) was a breakthrough in NLP; the state-of-the-art NLP methods are all based on transformers nowadays.

3.4.2. Complexity Comparison

Table 1 reports the complexity of operations in the self-attention mechanism and compares them with those in recurrence such as RNN. In self-attention, we learn attention of every word to every other word in the sequence of n words. Also, we learn a p-dimensional embedding for every word. Hence, the complexity of operations per layer is $\mathcal{O}(n^2 p)$. This is while the complexity per layer in recurrence is $\mathcal{O}(np^2)$. Although, the complexity per layer in self-attention is worse than recurrence, many of its operations can be performed in parallel because all the words of sequence are processed simultaneously, as also explained in the following. Hence, the $\mathcal{O}(n^2 p)$ is not very bad for being able to parallelize it. That is while the recurrence cannot be parallelized for its sequential nature.

As for the number of sequential operations, the self-attention mechanism processes all the n words simultaneously so its sequential operations is in the order of $\mathcal{O}(1)$. As recurrence should process the words sequentially, the number of its sequential operations is of order $\mathcal{O}(n)$. As for the maximum path length between every two words, self-attention learns attention between every two words; hence, its maximum path length is of the order $\mathcal{O}(1)$. However, in recurrence, as every word requires a path with a length of a fraction of sequence (a length of n in the worst case) to reach the process of another word, its maximum path length is $\mathcal{O}(n)$. This shows that attention reduces both sequential operations and maximum path length, compared to recurrence.

4. BERT: Bidirectional Encoder Representations from Transformers

BERT (Devlin et al., 2018) is one of the state-of-the-art methods for NLP. It is a stack of encoders of transformer (see Fig. 9). In other words, it is built using transformer encoder blocks. Although some NLP methods such as XLNet (Yang et al., 2019b) have slightly outperformed it, BERT is still one of the best models for different NLP tasks such as question answering (Qu et al., 2019), natural language understanding (Dong et al., 2019), sentiment analysis, and language inference (Song et al., 2020).

BERT uses the technique of masked language modeling. It masks 15% of words in the input document/corpus and asks the model to predict the missing words. As Fig. 12 depicts,

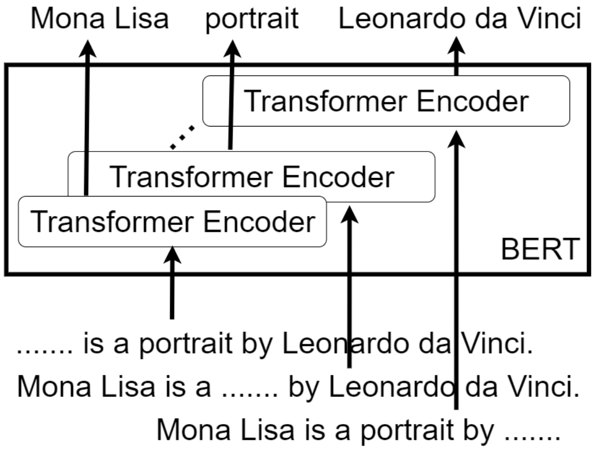

Mona Lisa
portrait
Leonardo da Vinci
Transformer Encoder
Transformer Encoder
Transformer Encoder
BERT
.... is a portrait by Leonardo da Vinci.
Mona Lisa is a .... by Leonardo da Vinci.
Mona Lisa is a portrait by ....

Figure 12. Feeding sentences with missing words to the BERT model for training.

a sentence with a missing word is given to every transformer encoding block in the stack and the block is supposed to predict the missing word. 15% of words are missing in the sentences and every missing word is assigned to every encoder block in the stack. It is an unsupervised manner because any word can be masked in a sentence and the output is supposed to be that word. As it is unsupervised and does not require labels, the huge text data of Internet can be used for training the BERT model where words are randomly selected to be masked.

Note that BERT learns to predict the missing word based on attention to its previous and forthcoming words so it is bidirectional. Hence, BERT jointly conditions on both left (previous) and right (forthcoming) context of every word. Moreover, as the missing word is predicted based on the other words of sentence, BERT embeddings for words are context-aware embeddings. Therefore, in contrast to word2vec (Mikolov et al., 2013a;b; Goldberg & Levy, 2014; Mikolov et al., 2015) and GloVe (Pennington et al., 2014) which provide a single embedding per each word, every word has different BERT embeddings in various sentences. The BERT embeddings of words differ in different sentences based on their context. For example, the word "bank" has different meanings and therefore different embeddings in the sentences "Money is in the bank" and "Some plants grow in bank of rivers".

It is also noteworthy that, for an input sentence, BERT outputs an embedding for the whole sentence in addition to giving embeddings for every word of the sentence. This sentence embedding is not perfect but works well enough in applications. One can use the BERT sentence embeddings and train a classifier on them for the task of spam detection or sentiment analysis.

During training the BERT model, in addition to learning the embeddings for the words and the whole sentence, paper (Devlin et al., 2018) has also learned an additional task. This task is given two sentences A and B, is B likely to be the sentence that follows A or not?

The BERT model is usually not trained from the scratch as its training has been done in a long time on huge amount of Internet data. For using it in different NLP applications, such as sentiment analysis, researchers usually do transfer learning and add one or several neural network layers on top of a pre-trained BERT model and train the network for their own task. During training, one can either freeze the weights of the BERT model and just train the added layers or also fine tune BERT weights by backpropagation.

The parameters of encoder in transformer (Vaswani et al., 2017) are 6 encoder layers, 512 hidden layer units in the fully connected network and 8 attention heads ($h = 8$). This is while BERT (Devlin et al., 2018) has 24 encoder layers, 1024 hidden layer units in the fully connected network and 16 attention heads ($h = 16$). Usually, when we say BERT, we mean the large BERT (Devlin et al., 2018) with the above-mentioned parameters. As the BERT model is huge and requires a lot of memory for saving the model, it cannot easily be used in embedded systems. Hence, many commercial smaller versions of BERT are proposed with less number of parameters and number of stacks. Some of these smaller versions of BERT are small BERT (Tsai et al., 2019), tiny BERT (Jiao et al., 2019), DistilBERT (Sanh et al., 2019), and Roberta BERT (Staliūnaitė & Iacobacci, 2020). Some BERT models, such as clinical BERT (Alsentzer et al., 2019) and BioBERT (Lee et al., 2020), have also been trained on medical texts for the biomedical applications.

5. GPT: Generative Pre-trained Transformer

GPT, or GPT-1, (Radford et al., 2018) is another state-of-the-art method for NLP. It is a stack of decoders of transformer (see Fig. 9). In other words, it is built using transformer decoder blocks. In GPT, the multihead attention module with cross-attention is removed from the decoder of transformer because there is no encoder in GPT. Hence, the decoder blocks used in GPT have only positional encoding, masked multihead self-attention module and feedforward network with their adding, layer normalization, and activation functions.

Note that as GPT uses the masked multihead self-attention, it considers attention of word, pairs of words, pairs of pairs of words, and so on, only on the previous (left) words, pairs of words, pairs of pairs of words, and so on. In other words, GPT is not bidirectional and conditions only on the previous words and not the forthcoming words. As was explained before, the objective of BERT was to predict a masked word in a sentence. However, GPT model is used for language model (Rosenfeld, 2000; Jozefowicz et al., 2016; Jing & Xu, 2019) whose objective is to predict the next word, in an incomplete sentence, given all of the previous words. The predicted new word is then added to the sequence and is fed to the GPT as input again and the other next word is predicted. This goes on until the sentences get complete with their next coming words. In other words, GPT model takes some document and continues the text in the best and related way. For example, if the input sentences are about psychology, the trained GPT model generates the next words and sentences also about psychology to complete the document. Note that as any text without label can be used for predicting the next words in sentences, GPT is an unsupervised method making it possible to be trained on huge amount of Internet data.

The successors of GPT-1 (Radford et al., 2018) are GPT-2 (Radford et al., 2019) and GPT-3 (Brown et al., 2020). GPT-2 and GPT-3 are extension of GPT-1 with more number of stacks of transformer decoder. Hence, they have more learnable parameters and can be trained with more data for better language modeling and inference. For example, GPT-2 has 1.5 billion parameters. GPT-2 and especially GPT-3 have been trained with much more Internet data with various general and academic subjects to be able to generate text in any subject and style of interest. For example, GPT-2 has been trained on 8 million web pages which contain 40GB of Internet text data.

GPT-2 is a quite large model and cannot be easily used in embedded systems because of requiring large memory. Hence, different sizes of GPT-2, like small, medium, large, Xlarge, and DistilGPT-2, are provided for usage in embedded systems, where the number of stacks and learnable parameters differ in these versions. These versions of GPT-2 can be found and used in the HuggingFace transformer Python package (Wolf et al., 2019a).

GPT-2 has been used in many different applications such as dialogue systems (Budzianowski & Vulić, 2019), patent claim generation (Lee & Hsiang, 2019), and medical text simplification (Van et al., 2020). A combination of GPT-2 and BERT has been used for question answering (Klein & Nabi, 2019). It is noteworthy that GPT can be seen as few shot learning (Brown et al., 2020). A comparison of GPT and BERT can also be found in (Ethayarajh, 2019).

GPT-3 is a very huge version of GPT with so many number of stacks and learn-able parameters. For comparison, note that GPT-2, NVIDIA Megatron (Shoeybi et al., 2019), Microsoft Turing-NLG (Microsoft, 2020), and GPT-3 (Brown et al., 2020) have 1.5 billion, 8 billion, 17 billion, and 175 billion learnable parameters, respectively. This huge number of parameters allows GPT-3 to be trained on very huge amount of Internet text data with various subjects and topics. Hence, GPT-3 has been able to learn almost all topics of documents and even some people are discussing whether it can pass the Turing's writer's test (Elkins & Chun, 2020; Floridi & Chiriatti, 2020). Note that GPT-3 has kind of memorized the texts of all subjects but not in a bad way, i.e., overfitting, rather in a good way. This memorization is because of the complexity of huge number of learnable parameters (Arpit et al., 2017) and not being overfitted is because of being trained by big enough Internet data.

GPT-3 has had many different interesting applications such as fiction and poetry generation (Branwen, 2020). Of course, it is causing some risks, too (McGuffie & Newhouse, 2020).

6. Conclusion

Transformers are very essential tools in natural language processing and computer vision. This paper was a tutorial and survey paper on attention mechanism, transformers, BERT, and GPT. We explained attention mechanism, the sequence-to-sequence model with and without attention, and self-attention. The different parts of encoder and decoder of a transformer were explained. Finally, BERT and GPT were introduced as stacks of the encoders and decoders of transformer, respectively.

Acknowledgment

The authors hugely thank Prof. Pascal Poupart whose course partly covered some of materials in this tutorial paper. Some of the materials of this paper can also be found in Prof. Ali Ghodsi's course videos.

References

Alsentzer, Emily, Murphy, John R, Boag, Willie, Weng, Wei-Hung, Jin, Di, Naumann, Tristan, and McDermott, Matthew. Publicly available clinical BERT embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

Arpit, Devansh, Jastrzebski, Stanisław, Ballas, Nicolas, Krueger, David, Bengio, Emmanuel, Kanwal, Maxinder S, Maharaj, Tegan, Fischer, Asja, Courville, Aaron, Bengio, Yoshua, and Lacoste-Julien, Simon. A closer look at memorization in deep networks. In *International Conference on Machine Learning*, 2017.

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

Ben-David, Shai, Blitzer, John, Crammer, Koby, Kulesza, Alex, Pereira, Fernando, and Vaughan, Jennifer Wortman. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

Branwen, Gwern. GPT-3 creative fiction. `https://www.gwern.net/GPT-3`, 2020.

Brown, Tom B, Mann, Benjamin, Ryder, Nick, Subbiah, Melanie, Kaplan, Jared, Dhariwal, Prafulla, Neelakantan, Arvind, Shyam, Pranav, Sastry, Girish, Askell, Amanda, et al. Language models are few-shot learners. In *Advances in neural information processing systems*, 2020.

Budzianowski, Paweł and Vulić, Ivan. Hello, it's GPT-2–how can I help you? Towards the use of pretrained language models for task-oriented dialogue systems. *arXiv preprint arXiv:1907.05774*, 2019.

Chan, William, Jaitly, Navdeep, Le, Quoc V, and Vinyals, Oriol. Listen, attend and spell. *arXiv preprint arXiv:1508.01211*, 2015.

Chan, William, Jaitly, Navdeep, Le, Quoc, and Vinyals, Oriol. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4960–4964. IEEE, 2016.

Cheng, Jianpeng, Dong, Li, and Lapata, Mirella. Long short-term memory-networks for machine reading. In *Conference on Empirical Methods in Natural Language Processing*, pp. 551–561, 2016.

Chorowski, Jan, Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. End-to-end continuous speech recognition using attention-based recurrent NN: First results. *arXiv preprint arXiv:1412.1602*, 2014.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dong, Li, Yang, Nan, Wang, Wenhui, Wei, Furu, Liu, Xiaodong, Wang, Yu, Gao, Jianfeng, Zhou, Ming, and Hon, Hsiao-Wuen. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pp. 13063–13075, 2019.

Dou, Qi, Coelho de Castro, Daniel, Kamnitsas, Konstantinos, and Glocker, Ben. Domain generalization via model-agnostic learning of semantic features. *Advances in Neural Information Processing Systems*, 32:6450–6461, 2019.

Elkins, Katherine and Chun, Jon. Can GPT-3 pass a writer's Turing test? *Journal of Cultural Analytics*, 2371:4549, 2020.

Ethayarajh, Kawin. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

Floridi, Luciano and Chiriatti, Massimo. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*, pp. 1–14, 2020.

Garcia-Molina, Hector, D. Ullman, Jeffrey, and Widom, Jennifer. *Database systems: The complete book*. Prentice Hall, 1999.

Goldberg, Yoav and Levy, Omer. word2vec explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

Gretton, Arthur, Borgwardt, Karsten, Rasch, Malte, Schölkopf, Bernhard, and Smola, Alex J. A kernel method for the two-sample-problem. In *Advances in neural information processing systems*, pp. 513–520, 2007.

Gretton, Arthur, Borgwardt, Karsten M, Rasch, Malte J, Schölkopf, Bernhard, and Smola, Alexander. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Indurkhya, Nitin and Damerau, Fred J. *Handbook of natural language processing*, volume 2. CRC Press, 2010.

Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Jiao, Xiaoqi, Yin, Yichun, Shang, Lifeng, Jiang, Xin, Chen, Xiao, Li, Linlin, Wang, Fang, and Liu, Qun. TinyBERT: Distilling BERT for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

Jing, Kun and Xu, Jungang. A survey on neural network language models. *arXiv preprint arXiv:1906.03591*, 2019.

Jozefowicz, Rafal, Vinyals, Oriol, Schuster, Mike, Shazeer, Noam, and Wu, Yonghui. Exploring the limits of language modeling. *arXiv preprint arXiv:1602.02410*, 2016.

Kazemi, Vahid and Elqursh, Ali. Show, ask, attend, and answer: A strong baseline for visual question answering. *arXiv preprint arXiv:1704.03162*, 2017.

Klein, Tassilo and Nabi, Moin. Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*, 2019.

Kolen, John F and Kremer, Stefan C. *A field guide to dynamical recurrent networks*. John Wiley & Sons, 2001.

Kombrink, Stefan, Mikolov, Tomáš, Karafiát, Martin, and Burget, Lukáš. Recurrent neural network based language modeling in meeting recognition. In *Twelfth annual conference of the international speech communication association*, 2011.

LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, Honglak, Grosse, Roger, Ranganath, Rajesh, and Ng, Andrew Y. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, pp. 609–616, 2009.

Lee, Jieh-Sheng and Hsiang, Jieh. Patent claim generation by fine-tuning OpenAI GPT-2. *arXiv preprint arXiv:1907.02052*, 2019.

Lee, Jinhyuk, Yoon, Wonjin, Kim, Sungdong, Kim, Donghyeon, Kim, Sunkyu, So, Chan Ho, and Kang, Jaewoo. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Li, Da, Yang, Yongxin, Song, Yi-Zhe, and Hospedales, Timothy M. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Li, Hui, Wang, Peng, Shen, Chunhua, and Zhang, Guyu. Show, attend and read: A simple and strong baseline for irregular text recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8610–8617, 2019a.

Li, Yang, Kaiser, Lukasz, Bengio, Samy, and Si, Si. Area attention. In *International Conference on Machine Learning*, pp. 3846–3855. PMLR, 2019b.

Lim, Jian Han, Chan, Chee Seng, Ng, Kam Woh, Fan, Lixin, and Yang, Qiang. Protect, show, attend and tell: Image captioning model with ownership protection. *arXiv preprint arXiv:2008.11009*, 2020.

Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

McGuffie, Kris and Newhouse, Alex. The radicalization risks of GPT-3 and advanced neural language models. *arXiv preprint arXiv:2009.06807*, 2020.

Microsoft. Turing-NLG: A 17-billion-parameter language model by Microsoft. Microsoft Blog, 2020.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*, 2013a.

Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013b.

Mikolov, Tomas, Chen, Kai, Corrado, Gregory S, and Dean, Jeffrey A. Computing numeric representations of words in a high-dimensional space, May 19 2015. US Patent 9,037,464.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher D. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing*, pp. 1532–1543, 2014.

Perlovsky, Leonid I. Toward physics of the mind: Concepts, emotions, consciousness, and symbols. *Physics of Life Reviews*, 3(1):23–55, 2006.

Qu, Chen, Yang, Liu, Qiu, Minghui, Croft, W Bruce, Zhang, Yongfeng, and Iyyer, Mohit. BERT with history answer embedding for conversational question answering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1133–1136, 2019.

Qureshi, Ahmed Hussain, Nakamura, Yutaka, Yoshikawa, Yuichiro, and Ishiguro, Hiroshi. Show, attend and interact: Perceivable human-robot social interaction through neural attention q-network. In *2017 IEEE International Conference on Robotics and Automation*, pp. 1639–1645. IEEE, 2017.

Radford, Alec, Narasimhan, Karthik, Salimans, Tim, and Sutskever, Ilya. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018.

Radford, Alec, Wu, Jeffrey, Child, Rewon, Luan, David, Amodei, Dario, and Sutskever, Ilya. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9, 2019.

Rosenfeld, Ronald. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8):1270–1278, 2000.

Rumelhart, David E, Hinton, Geoffrey E, and Williams, Ronald J. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Sanh, Victor, Debut, Lysandre, Chaumond, Julien, and Wolf, Thomas. DistilBERT, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Shoeybi, Mohammad, Patwary, Mostofa, Puri, Raul, LeGresley, Patrick, Casper, Jared, and Catanzaro, Bryan. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Socher, Richard, Bengio, Yoshua, and Manning, Chris. Deep learning for nlp. *Tutorial at Association of Computational Logistics (ACL)*, 2012.

Song, Youwei, Wang, Jiahai, Liang, Zhiwei, Liu, Zhiyue, and Jiang, Tao. Utilizing BERT intermediate layers for aspect based sentiment analysis and natural language inference. *arXiv preprint arXiv:2002.04815*, 2020.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Staliūnaitė, Ieva and Iacobacci, Ignacio. Compositional and lexical semantics in RoBERTa, BERT and DistilBERT: A case study on CoQA. *arXiv preprint arXiv:2009.08257*, 2020.

Summerfield, Christopher and Egner, Tobias. Expectation (and attention) in visual cognition. *Trends in cognitive sciences*, 13(9):403–409, 2009.

Tsai, Henry, Riesa, Jason, Johnson, Melvin, Arivazhagan, Naveen, Li, Xin, and Archer, Amelia. Small and practical BERT models for sequence labeling. *arXiv preprint arXiv:1909.00100*, 2019.

Van, Hoang, Kauchak, David, and Leroy, Gondy. AutoMeTS: The autocomplete for medical text simplification. *arXiv preprint arXiv:2010.10573*, 2020.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Wang, Yong, Wang, Longyue, Shi, Shuming, Li, Victor OK, and Tu, Zhaopeng. Go from the general to the particular: Multi-domain translation with domain transformation networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 9233–9241, 2020.

Wolf, Thomas, Debut, Lysandre, Sanh, Victor, Chaumond, Julien, Delangue, Clement, Moi, Anthony, Cistac, Pierric, Rault, Tim, Louf, Rémi, Funtowicz, Morgan, et al. HuggingFace's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019a.

Wolf, Thomas, Sanh, Victor, Chaumond, Julien, and Delangue, Clement. TransferTransfo: A transfer learning approach for neural network based conversational agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019b.

Xu, Jun. On the techniques of English fast-reading. In *Theory and Practice in Language Studies*, volume 1, pp. 1416–1419. Academy Publisher, 2011.

Xu, Kelvin, Ba, Jimmy, Kiros, Ryan, Cho, Kyunghyun, Courville, Aaron, Salakhudinov, Ruslan, Zemel, Rich, and Bengio, Yoshua. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015.

Yang, Chao, Kim, Taehwan, Wang, Ruizhe, Peng, Hao, and Kuo, C-C Jay. Show, attend, and translate: Unsupervised image translation with self-regularization and attention. *IEEE Transactions on Image Processing*, 28(10):4845–4856, 2019a.

Yang, Zhilin, Dai, Zihang, Yang, Yiming, Carbonell, Jaime, Salakhutdinov, Russ R, and Le, Quoc V. XLnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5753–5763, 2019b.

Yu, Keyi, Liu, Yang, Schwing, Alexander G, and Peng, Jian. Fast and accurate text classification: Skimming, rereading and early stopping. In *International Conference on Learning Representations*, 2018.

Zhang, Honglun, Chen, Wenqing, Tian, Jidong, Wang, Yongkun, and Jin, Yaohui. Show, attend and translate: Unpaired multi-domain image-to-image translation with visual attention. *arXiv preprint arXiv:1811.07483*, 2018.