

Implementasi Fine-Tuning BERT untuk Indo News Classification

BERT PAPER :

<https://arxiv.org/pdf/1810.04805.pdf>

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova

Google AI Language

{jacobdevlin, mingweichang, kentonl, kristout}@google.com

Abstract

We introduce a new language representation model called **BERT**, which stands for **Bidirectional Encoder Representations from Transformers**. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a re-

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters and is trained on the

A little history

- RNN is invented by Hopfield (1982)
- LSTM is invented by Hochreiter & Schmidhuber (1997)
- Transformer is invented by Vaswani (2017)
(<https://arxiv.org/pdf/1706.03762.pdf>) - "Attention Is All You Need" for machine translation task (seq2seq architecture)

Resources :

- [LSTM is dead. Long Live Transformers!](#)
- [Transformer Neural Networks - EXPLAINED! \(Attention is all you need\)](#)
- [The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time.](#)

Attention Is All You Need

Ashish Vaswani*
Google Brain
avaswani@google.com

Noam Shazeer*
Google Brain
noam@google.com

Niki Parmar*
Google Research
nikip@google.com

Jakob Uszkoreit*
Google Research
usz@google.com

Llion Jones*
Google Research
llion@google.com

Aidan N. Gomez* †
University of Toronto
aidan@cs.toronto.edu

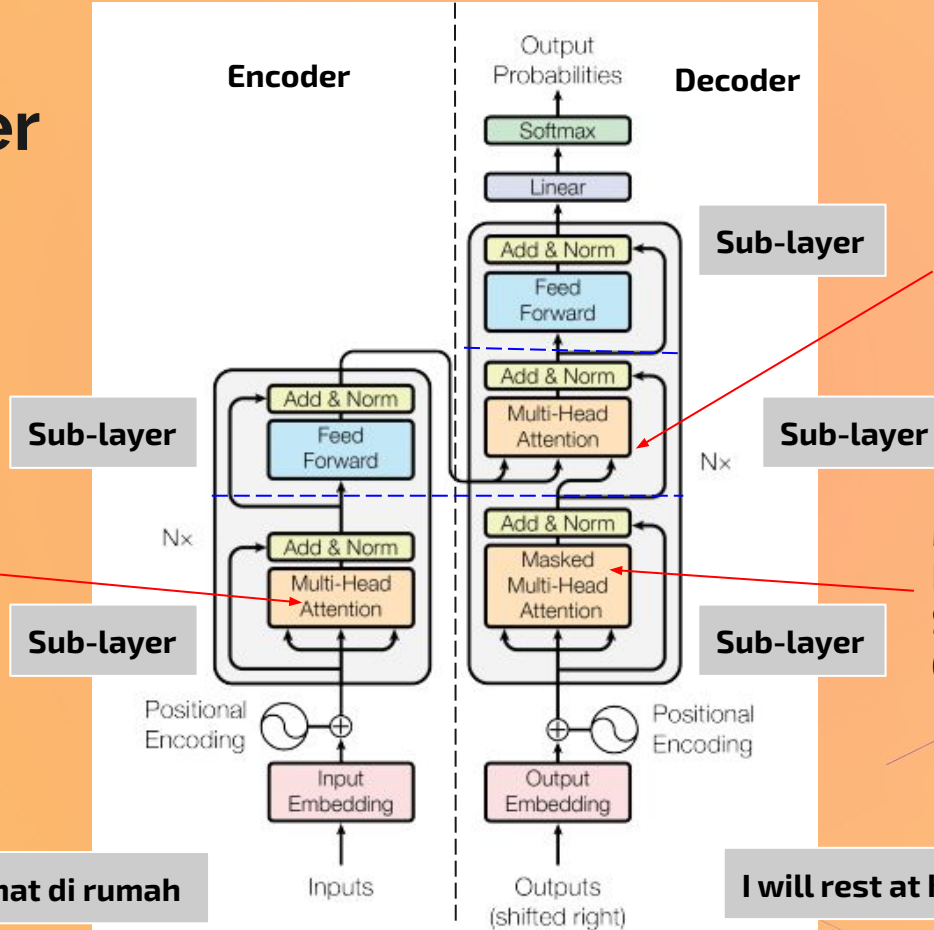
Łukasz Kaiser*
Google Brain
lukaszkaizer@google.com

Illia Polosukhin* ‡
illia.polosukhin@gmail.com

Transformer

Encoder
Self-Attention
(input-input)

saya akan beristirahat di rumah



Encoder-Decoder
Attention
(input-output)

Masked
Decoder
Self-Attention
(output-output)

I will rest at home

Figure 1: The Transformer - model architecture.

Transformer Component

1. Input Embedding (word embedding)
2. Positional Encoding
3. Encoder Block - Multi Head Attention
4. Encoder Block - Position-wise FFN
5. Encoder Block - Add & Norm
6. Decoder Block
7. Linear Transformations, Softmax

Kita tahu kalau RNN punya kemampuan untuk meng-capture informasi pada sequence data, karena hidden state bisa menggabungkan representasi kata/vektor sebelumnya dengan representasi kata saat ini.

Agar transformer dapat melakukan hal yang kurang lebih sama maka muncullah **positional encoding** dan **attention**

2. Positional Encoding

“Vektor yang memberikan konteks berdasarkan posisi suatu kata pada kalimat”

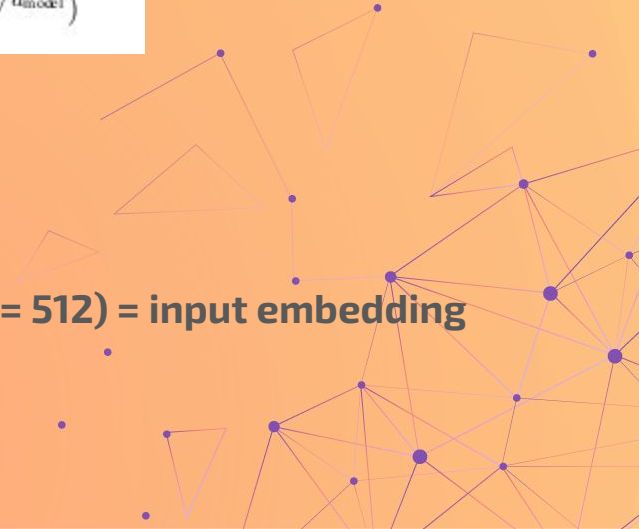
In this work, we use sine and cosine functions of different frequencies:

$$PE_{(pos, 2i)} = \sin(pos/10000^{2i/d_{model}})$$
$$PE_{(pos, 2i+1)} = \cos(pos/10000^{2i/d_{model}})$$

keterangan : pos = posisi kata dalam sebuah kalimat

i = dimensi

d_model = dimensi model (default transformer = 512) = input embedding



2. Positional Encoding

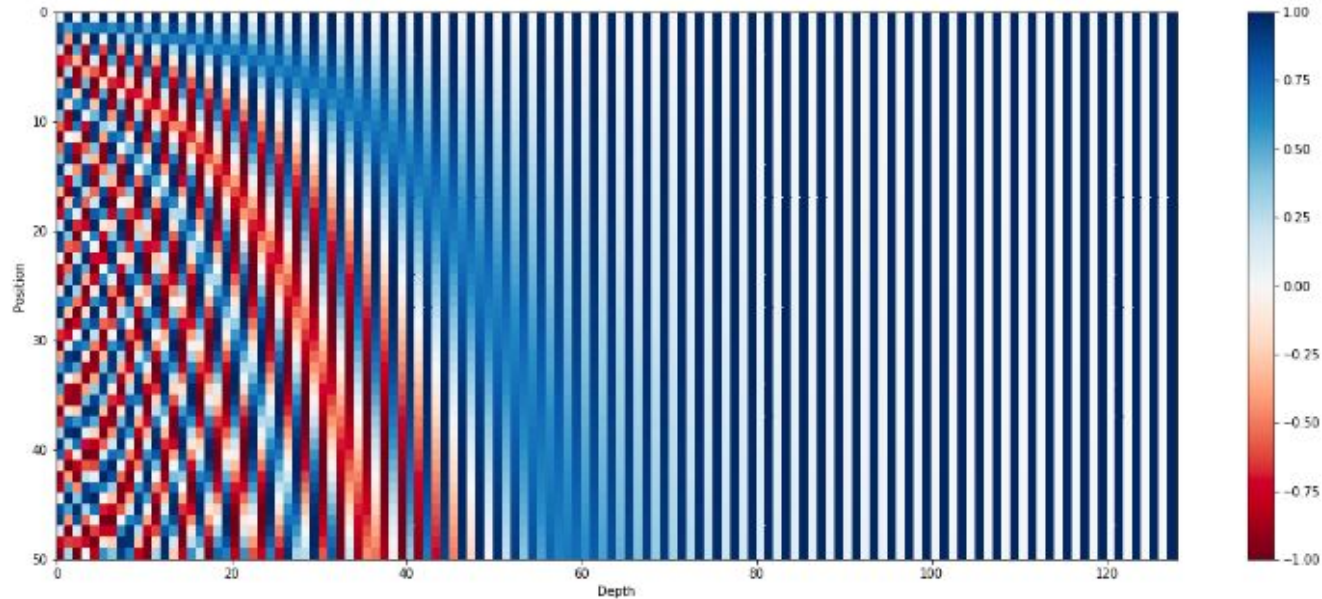


Figure 2 - The 128-dimensional positional encoding for a sentence with the maximum length of 50. Each row represents the embedding vector \vec{p}_t

3. Attention

Attention = perhatian atau “Fokus pada sesuatu”

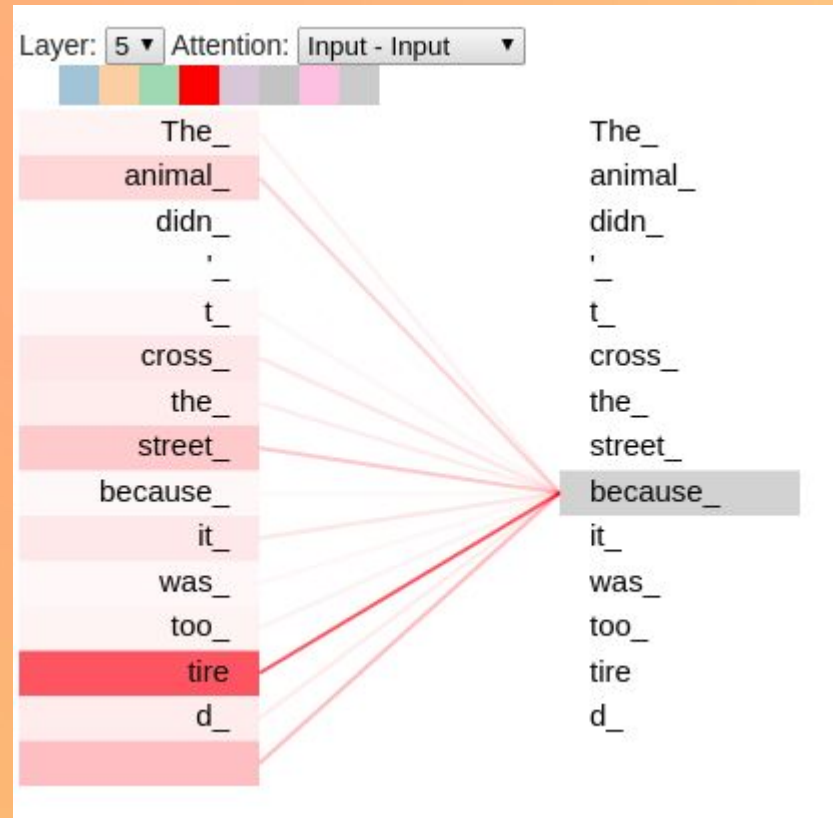
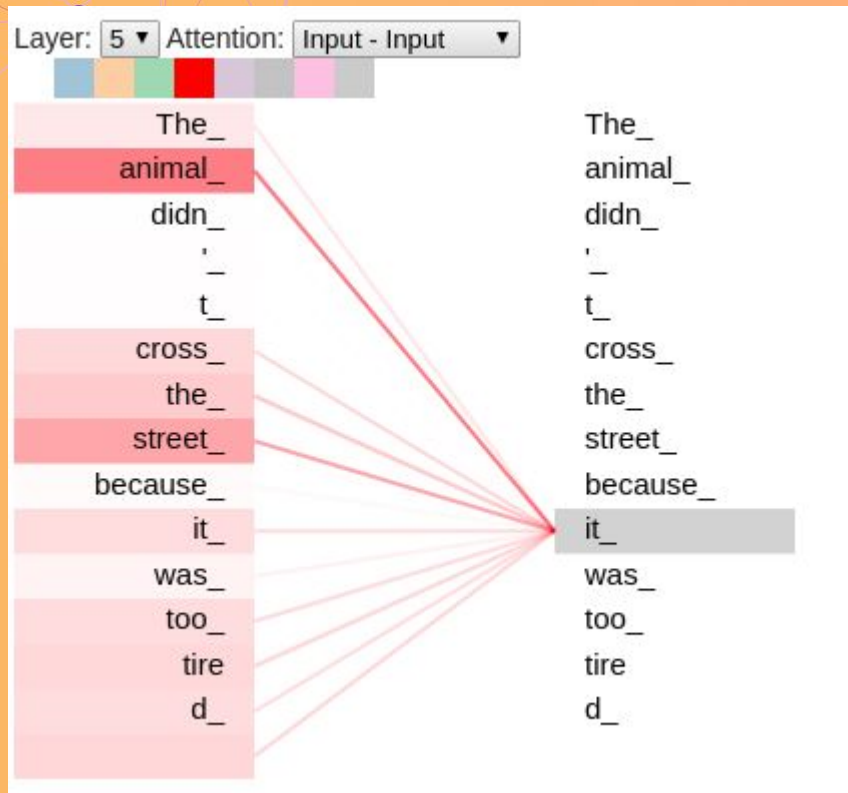
Misalnya kita punya sebuah input kalimat seperti ini :

"The animal didn't cross the street because it was too tired"

Apa yang dimaksud "it" dalam kalimat ini? Apakah ini menunjukkan “street” atau “animal” ? Ini pertanyaan sederhana bagi manusia, tetapi tidak sederhana untuk komputer.

Ketika model memproses setiap kata (setiap posisi dalam urutan input), self attention memungkinkannya untuk melihat posisi lain dalam urutan input untuk memberikan informasi yang dapat membantu mendapatkan “better encoding” suatu kata.

3. Attention ([tensor2tensor by tensorflow](#))



3. Attention

Attention is all you need; Attentional Neural Network Models | Łukasz Kaiser |
Masterclass (Author - Google Brain)

Dot-Product Attention

$$A(Q, K, V) = \text{softmax}(QK^T)V$$

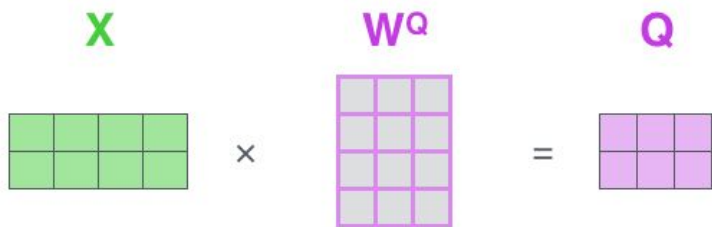
Apa itu Q, K, V ? The general idea is

Q (query) = vektor yang merepresentasikan kata

K (key) dan V(value) = memory, semua kata yang telah di-generate sebelumnya

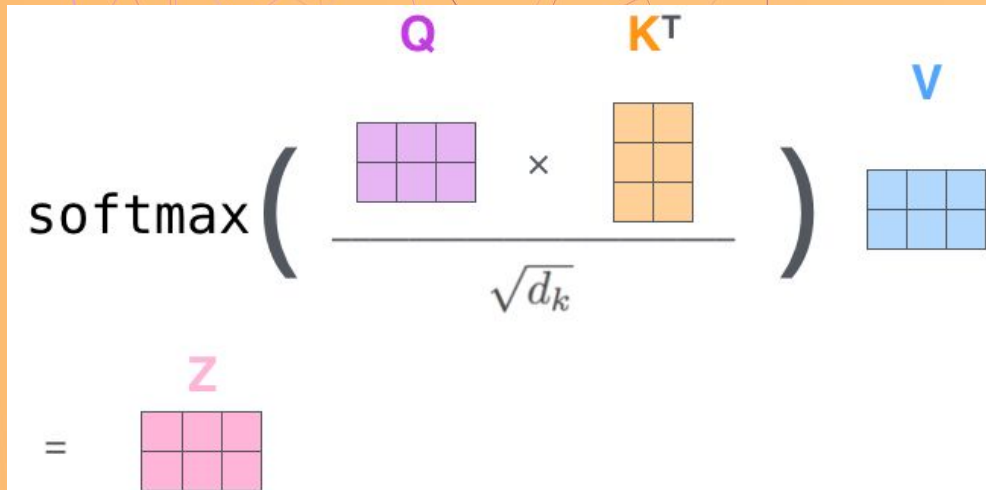
Note : K dan V bisa bernilai sama

3. Attention

$$X \times W^Q = Q$$


$$X \times W^K = K$$


$$X \times W^V = V$$


$$\text{softmax} \left(\frac{Q \times K^T}{\sqrt{d_k}} \right) V = Z$$


3. Attention

Apa yang dilakukan Attention ?

Note : d_k = dimension of key (input sequence length)

“Take the query (Q)(word in most cases), then find the most similar key (K), and then get the values (V) that correspond to these similar key (K)”

Hasil dari
keys

$$\text{softmax}(QK^T)$$

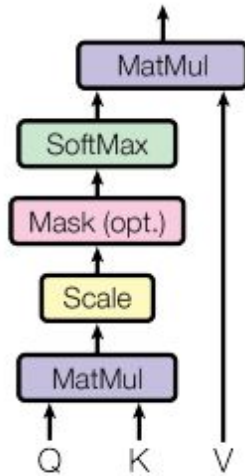
akan memberikan “*probability distribution over*”

which is peaked at the one that are similar to the query”

dot product attention without scaling for larger values of d_k [3]. We suspect that for large values of d_k , the dot products grow large in magnitude, pushing the softmax function into regions where it has extremely small gradients [4]. To counteract this effect, we scale the dot products by $\frac{1}{\sqrt{d_k}}$.

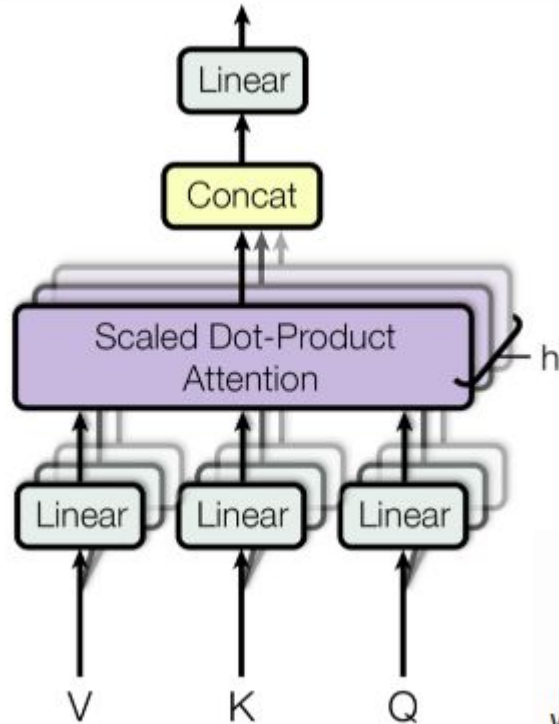
3. Attention

Scaled Dot-Product Attention



$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

3. Multi-Head Attention



Menurut paper : “multi-head attention allows the model to jointly attend to information from different representation subspaces at different positions. With a single attention head, averaging inhibits this.”

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = [\text{head}_1; \dots; \text{head}_h] \mathbf{W}^O$$

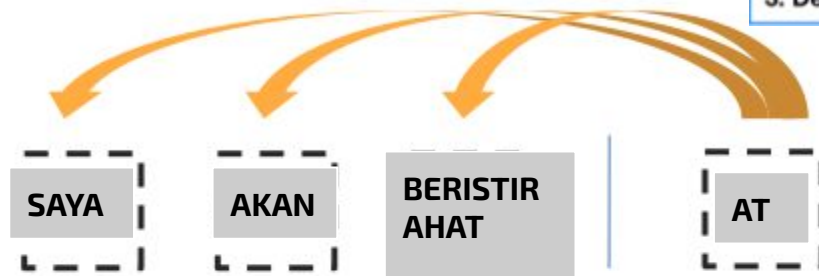
where $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V)$

where \mathbf{W}_i^Q , \mathbf{W}_i^K , \mathbf{W}_i^V , and \mathbf{W}^O are parameter matrices to be learned.

Three ways of attention

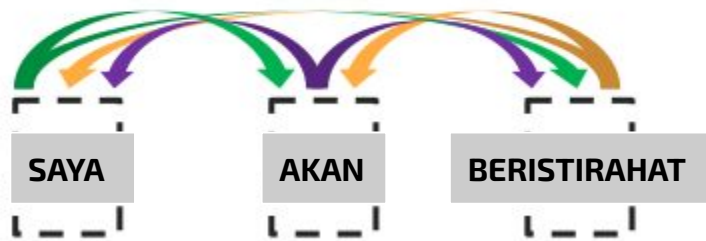
encoder-decoder attention

1. Multi-head Attention
2. Encoder Self attention= $\text{Key}=\text{Value}$
3. Decoder Self attention= Query



encoder self attention

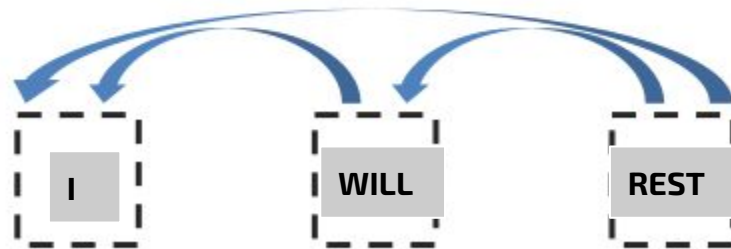
1. Multi-head Attention
2. $\text{Query}=\text{Key}=\text{Value}$



Encoder Self-Attention

decoder self attention

1. **Masked** Multi-head Attention
2. $\text{Query}=\text{Key}=\text{Value}$



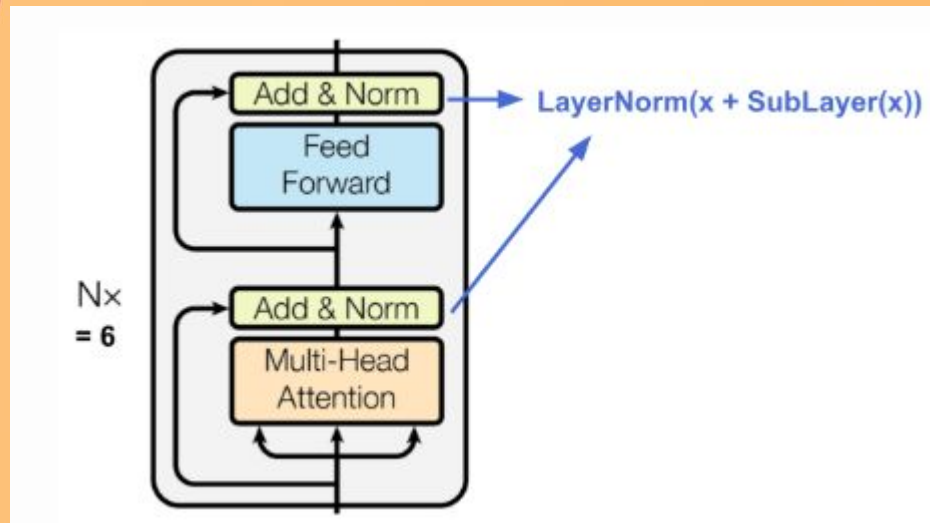
MaskedDecoder Self-Attention

4. Position-wise Feed Forward Network (FFN)

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2$$

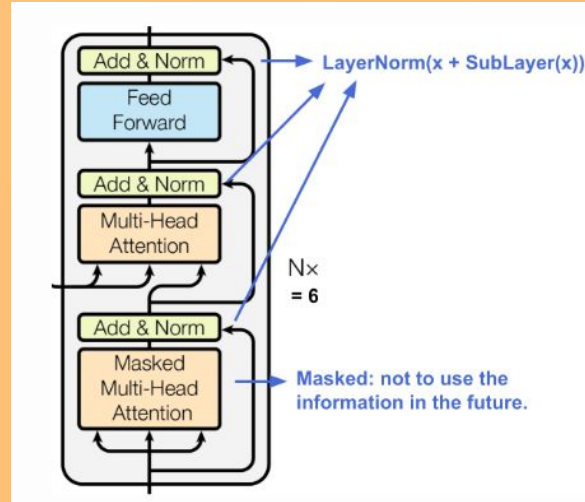
Linear transformation (1) -> ReLu activation -> Linear transformation (2)

Encoder



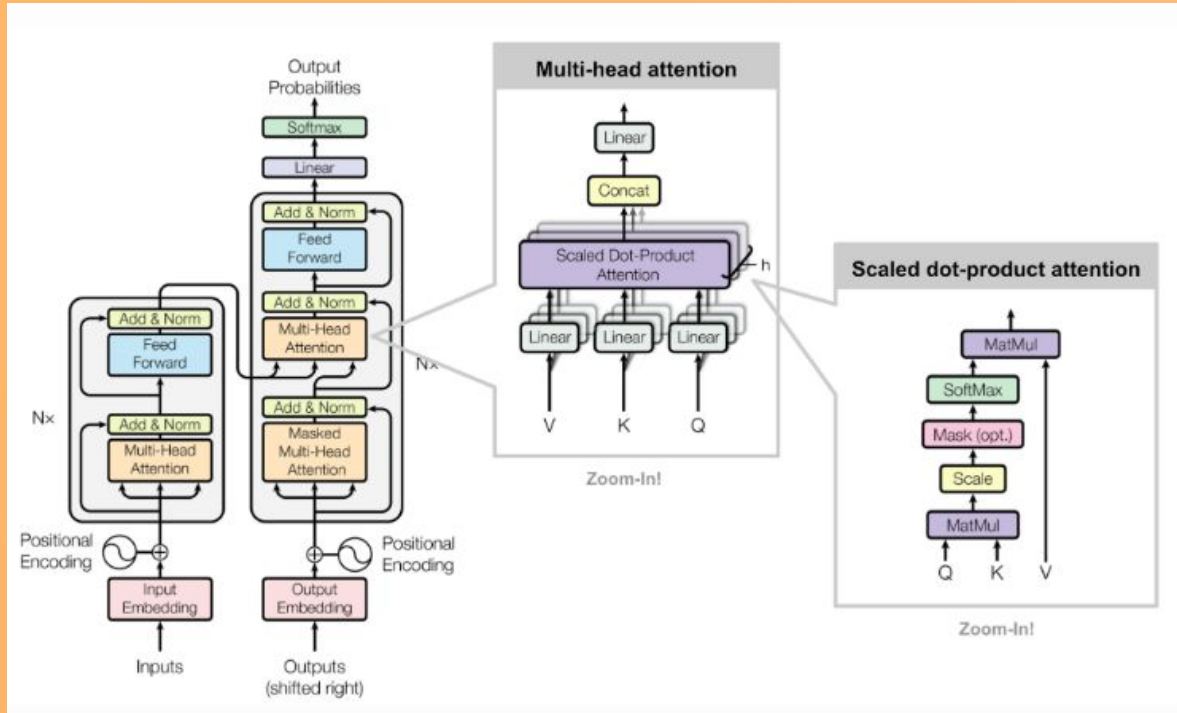
- **$N = 6$ stack encoder**
- **dimensi model = 512**

Decoder



- **$N = 6$ stack encoder**
- **dimensi model = 512**
- **Masked Multi-Head Attention -> hide the future decoder inputs (the next words of a translated sentence)**

Full Architecture of Transformer



Optimizer : Adam

Regularization :

- Residual Dropout (apply dropout for the output of each sub-layer) = 0.1
- Label Smoothing = 0.1

Let's start BERT

Bidirectional Encoder Representations from Transformers (BERT)

BERT is a method of pre-training language representations, meaning that we train a general-purpose "language understanding" model on a large text corpus (like Wikipedia), and then use that model for downstream NLP tasks that we care about (like question answering).

Ada 2 tahapan pada framework BERT :

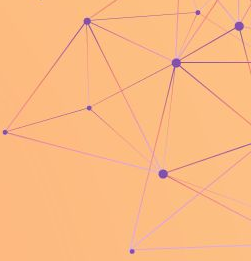
1. **Pre-training** (train on unlabeled data for different task)
2. **Fine-tuning** (transfer learning, train classification layer)

Pre-training data menggunakan BookCorpus (800M words) dan English Wikipedia (2500M words)



Let's start BERT

BERT is . . .

- **Unsupervised** -> BERT dilatih hanya menggunakan plain text corpus
 - **Contextual representations** -> satu kata bisa memiliki multiple "word embedding" representasi yang bergantung pada kalimat (ex: tahu)
 - **Bidirectional** -> representasi kata berdasarkan left dan right context
 - **Can learn relationships between sentences**
- 

Advantages of Fine Tuning

- **Quicker Development (only need 2-4 epoch for fine-tuning BERT on specific dataset)**
- **Small dataset / less data**
- **Better results**



BERT Architecture

BERT base (L=12, H=768, A=12, vocab around 30k token , Total params = 110M, english text)

BERT large (L=24, H=768, A=16, vocab around 30k token, Total params = 340M, english text)

BERT base-multilingual (L=12, H=768, A=12, vocab around 110k token , Total params = 110M, 104 languages text)

Original Transformer (L=6, H=512, A=8)

L : jumlah transformer blok

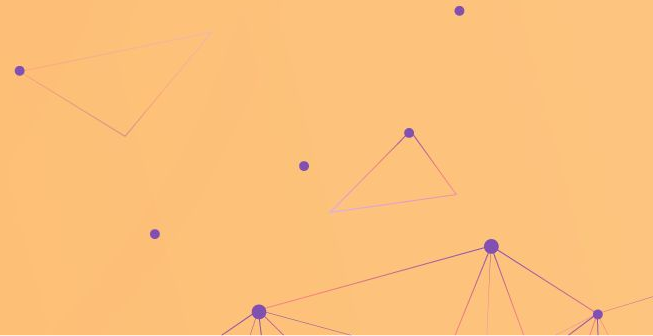
H : hidden size

A : jumlah attention head



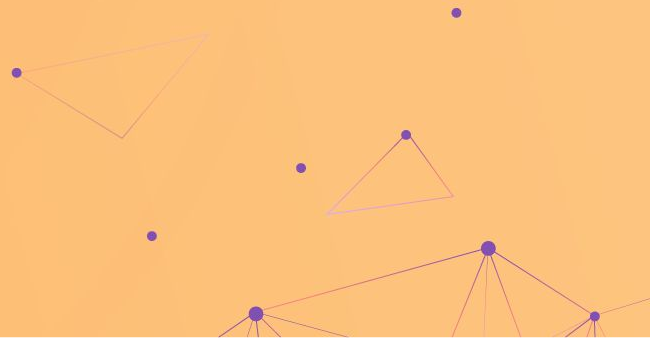
Special Tokens

- **[CLS]** : token pertama untuk setiap sequence, dan token untuk classification task
- **[SEP]** : token delimiter atau pemisah antar sequence yang digunakan untuk pre-training pada sequence-pair tasks
- **[MASK]** : token yang digunakan untuk "mask" sebuah kata, token ini hanya digunakan ketika pre-training



Pre-training BERT

- **Task #1 : Masked LM (MLM)**
-> give bidirectional ability
- **Task #2 : Next Sentence Prediction (NSP)**
-> give ability to learn relationships between sentences



Task #1 : Masked LM

Cara ini terinspirasi dari Cloze Test

Ours was the marsh (1) _____, down by the river, (2) _____, as the river wound, (3) _____ miles of the sea. (4) _____ first most vivid and (5) _____ impression of the identity (6) _____ things, seems to me (7) _____ have been gained on (8) _____ memorable raw afternoon towards (9) _____. At such a time (10) _____ found out for certain, (11) _____ this bleak place overgrown (12) _____ nettles was the churchyard; (13) _____ that Philip Pirrip, late (14) _____ this parish, and the (15) _____ wife of the above, (16) _____ dead and buried.

These are the words to choose from:

I were that My to within with a of broad twenty and Georgiana
of evening country

Table 1: Example of a Fixed-Rate Cloze Test.

Task #1 : Masked LM

Contoh kita punya sebuah kalimat : "my dog is hairy", lalu saat prosedur random masking kita pilih token ke-4

80% : Ganti kata dengan [MASK] token (my dog is hairy -> my dog is [MASK])

10% : Ganti kata dengan kata random (my dog is hairy -> my dog is apple)

10% : Pertahankan kata (tidak berubah) (my dog is hairy -> my dog is hairy)

Berdasarkan paper : "The advantage of this procedure is that the Transformer encoder does not know which words it will be asked to predict or which have been replaced by random words, so it is forced to keep a distributional contextual representation of every input token."

Task #2 : Next Sentence Prediction (NSP)

Next Sentence Prediction The next sentence prediction task can be illustrated in the following examples.

Input = [CLS] the man went to [MASK] store [SEP]
 he bought a gallon [MASK] milk [SEP]

Label = IsNext

Input = [CLS] the man [MASK] to the store [SEP]
 penguin [MASK] are flight ##less birds [SEP]

Label = NotNext

Task #2 : Next Sentence Prediction (NSP)

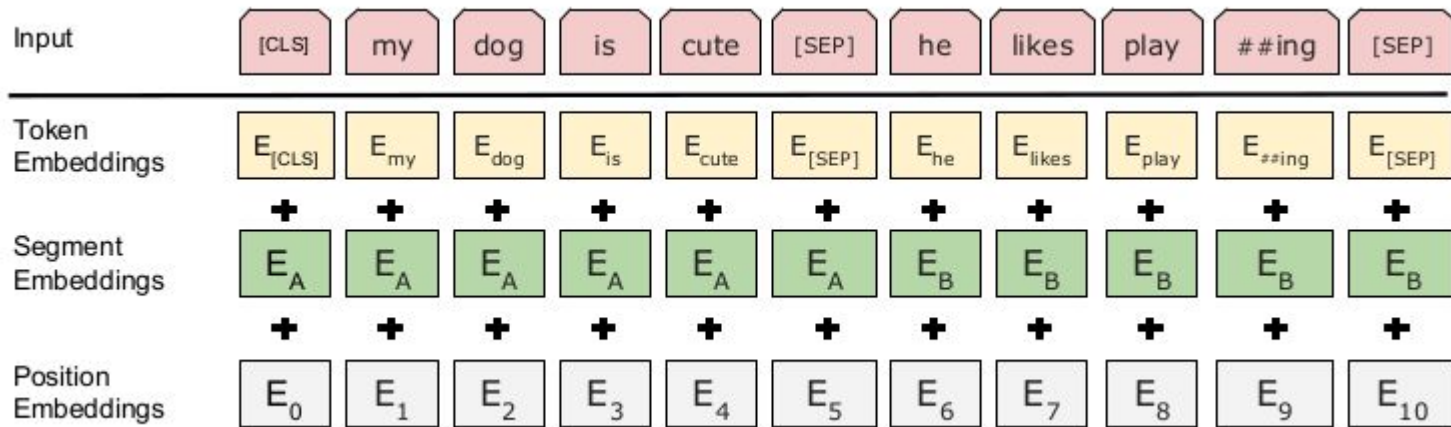
Berguna untuk case Question-Answering, Natural Language Inference, etc

Berapa persentase sebuah kalimat merupakan “actually next sentence” ?

50% dari kalimat dipasangkan dengan kalimat yang berdekatan / berhubungan sedangkan 50% nya dipasangkan dengan random kalimat dari korpus

Input Embeddings

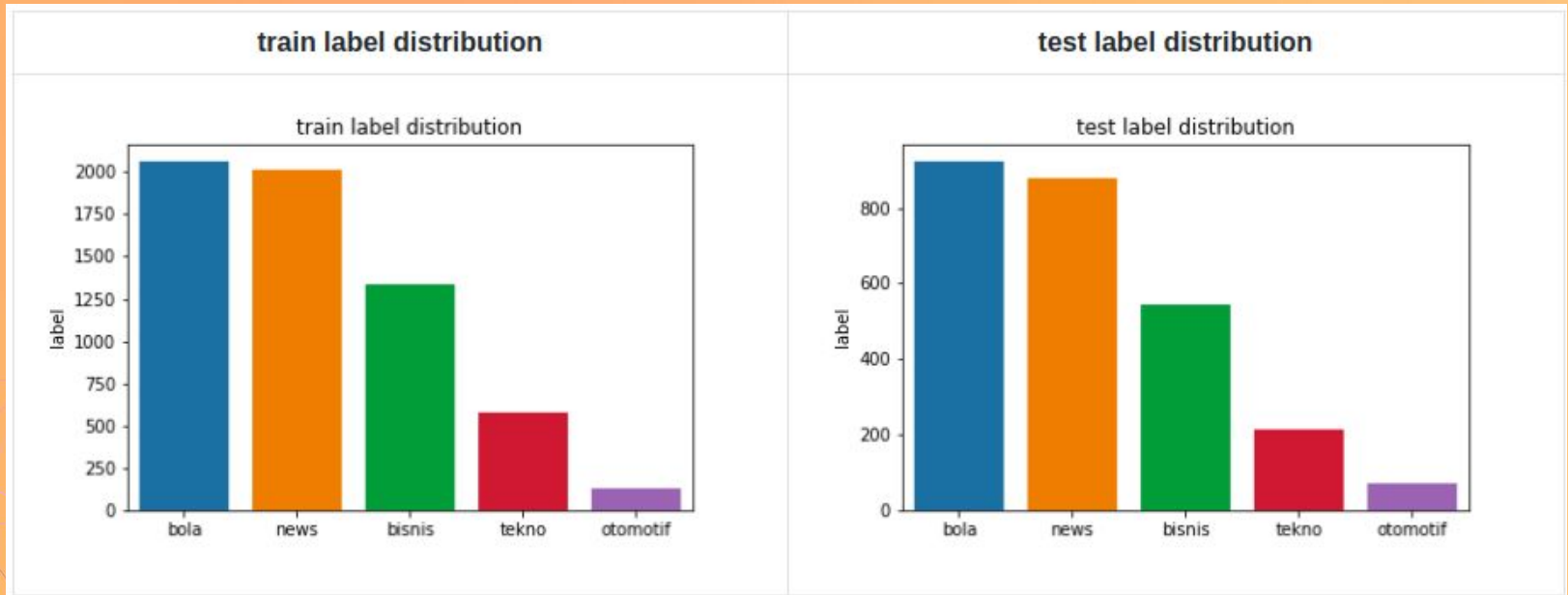
Pertama - tama, input text di tokenize menggunakan metode **WordPiece (sub-word)**



Fine-tuning BERT for sentence classification

Indonesian News Dataset by Andreas Chandra :

<https://github.com/andreaschandra/indonesian-news>



Fine-tuning BERT for sentence classification

<https://colab.research.google.com/drive/1LljYzckl9z55PA4AVkIKOaxjPp0Q5YJ0?authuser=1>





Resources :

[BERT Research Series](#)

[Kaggle Coffee Chat: Jacob Devlin \(Google Researcher, BERT author\) | Kaggle](#)
[Smaller, faster, cheaper, lighter: Introducing DilBERT, a distilled version of BERT](#)

~ THANK YOU ~

rubentea16.github.io