



NLP with Transformers



ABD RAHMAN



Table of contents

01 Pengantar

Bagaimana awal mula NLP dan model Transformers?

02 Mekanisme

Bagaimana arsitektur dan mekanisme Transformers?

03 Implementasi

Bagaimana implementasi Transformers dalam melaksanakan tugas-tugas NLP?

04 Training

Bagaimana proses pelatihan dan evaluasi model Transformers?

05 Aplikasi

Bagaimana aplikasi nyata Transformers dan tantangannya di masa depan?

06 BERT

Bagaimana menerapkan Transformers dengan BERT di Python?



Sumber Pustaka

Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022)

**Natural Language Processing with
Transformers:**
Building Language Applications with
Hugging Face,

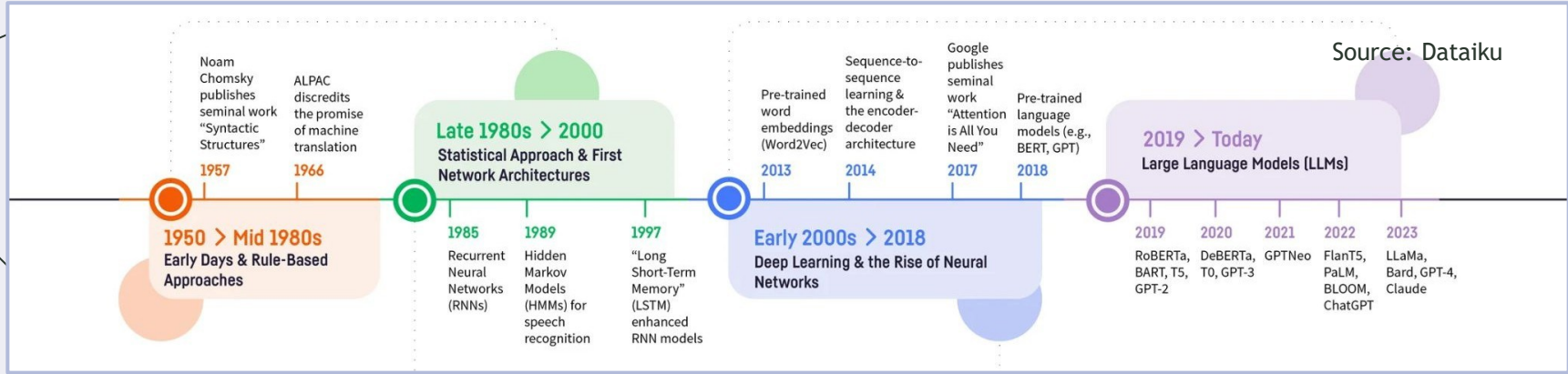
O'Reilly Media



01

Pengantar

Sejarah dan Evolusi NLP



Pertengahan 1930-an mesin translasi bahasa pertama dibuat di Jerman bernama "Enigma". Pada tahun 1966, program "Eliza" dibuat untuk mensimulasikan percakapan manusia dengan aturan-aturan sederhana. Pada periode ini, dikembangkan pula model dengan grammar dan model translasi.

Pada periode sebelumnya, aturan-aturan dibuat untuk membuat mesin mengenali kalimat hanya berdasarkan penstrukturan frasa. Periode ini, pendekatan **statistik** digunakan untuk memahami kalimat berdasarkan rangkaian katanya juga.

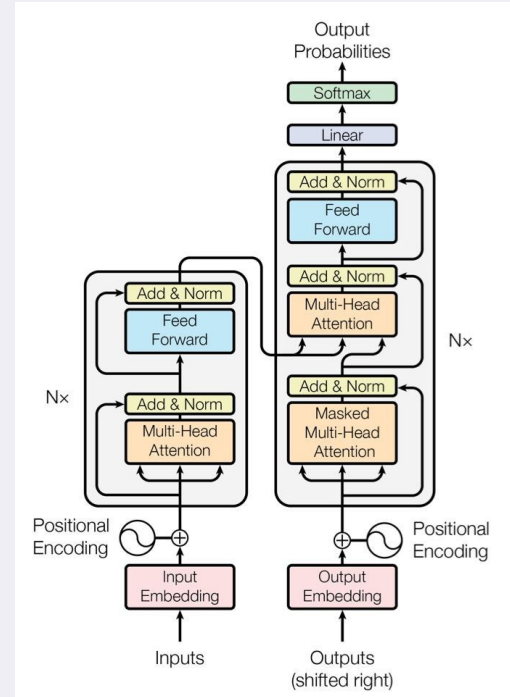
Kapasitas komputasi ditingkatkan pada periode ini, di mana digunakan beberapa pendekatan **ML** dan juga feature engineering seperti POS Tagging, NER, dan Dependency Parsing dalam peningkatan kinerja model.

Di era Transformers dan LLM, pengembangan dilakukan pada data pre-train yang berukuran sangat **besar** untuk melakukan berbagai tugas NLP lainnya.

Pengenalan Model Transformers

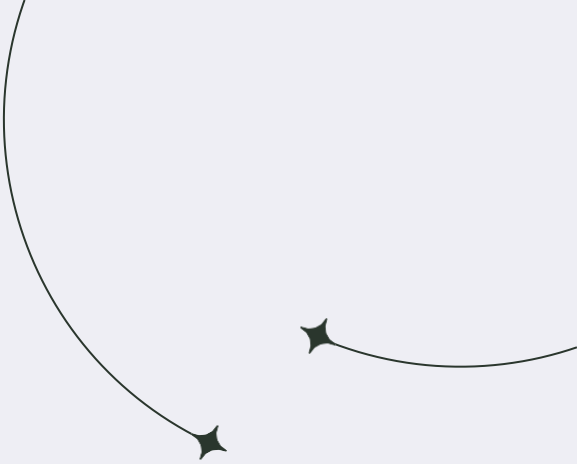
Transformers diperkenalkan pada tahun 2017 oleh Vaswani pada “[Attention is All You Need](#)”. Transformer adalah arsitektur model jaringan saraf yang sangat sukses dalam tugas-tugas pemrosesan bahasa alami dan tugas-tugas machine learning lainnya. Model ini digunakan untuk menangani data-data sekuensial seperti NLP. Arsitektur utamanya terdiri dari lapisan Encoder dan Decoder.

Pada intinya, transformer bergantung pada prinsip self-attention, memungkinkan mereka memproses data masukan secara paralel, menjadikan mereka sangat efisien dan mampu menangkap hubungan kompleks dalam data.



Transformers vs Last Models

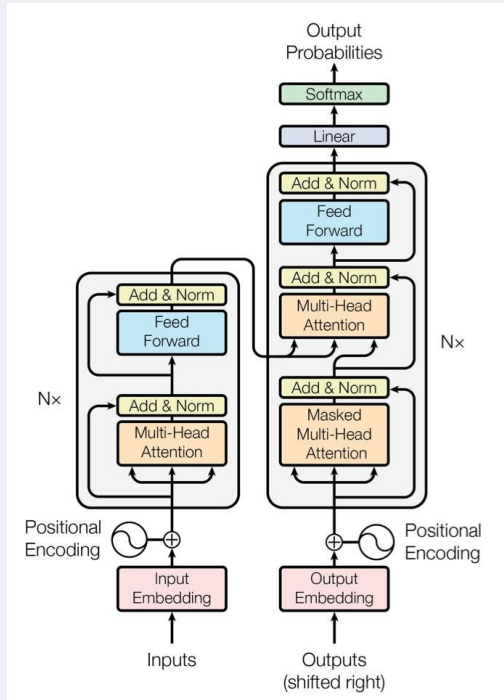
Model sebelumnya (RNN & LSTM)	Transformers
<ul style="list-style-type: none">• Arsitektur berurutan, di mana state waktu saat ini menunggu hasil dari pemrosesan state sebelumnya• Mekanisme attention merupakan mekanisme tambahan• Implicit Positional Information, di mana urutan posisi informasi dalam model RNN dan LSTM dijelaskan secara implisit• Training lebih lama dan kurang mampu memahami konteks pada dependensi jangka panjang• Sangat baik untuk tugas urutan pendek dan menengah	<ul style="list-style-type: none">• Arsitektur paralel, di mana pemrosesan data dilakukan secara bersama/paralel dalam satu waktu• Mekanisme attention sebagai inti dari arsitektur• Explicit Positional Encoding, di mana posisi informasi dalam urutan diperoleh dari positional encoding• Training lebih cepat dan dependensi jangka panjang ditangani lebih efektif• Unggul pada berbagai tugas NLP termasuk pada pemrosesan urutan yang panjang



02

Mekanisme Kerja

Arsitektur Transformers



Dua lapisan utama:

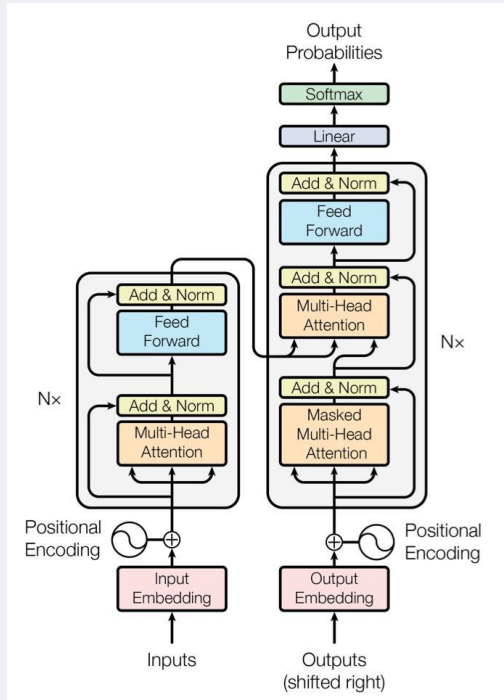
- **Lapisan Encoder**
Encoder bertanggung jawab untuk memproses input dan menghasilkan representasi yang kaya dan terstruktur yang dapat digunakan oleh decoder.
- **Lapisan Decoder**
Decoder menghasilkan urutan output dengan mempertimbangkan representasi yang diberikan oleh encoder dan konteks output sebelumnya.

Karena transformers memproses seluruh urutan secara paralel dan tidak memiliki urutan intrinsik, **positional encoding** ditambahkan ke input embeddings untuk memberi model informasi tentang urutan posisi kata dalam teks. Positional encoding biasanya berupa vektor sinus dan kosinus yang ditambahkan ke embedding kata:

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

Di mana pos adalah posisi kata dalam urutan dan i adalah dimensi embedding.

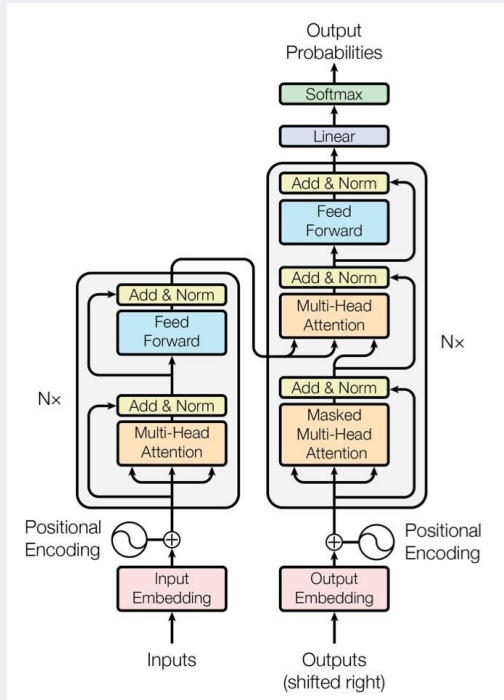
Arsitektur Transformers



Pada Lapisan Encoder:

- **Self Attention**
Memungkinkan setiap posisi dalam input untuk berinteraksi dengan setiap posisi lainnya, sehingga memungkinkan model untuk mempertimbangkan konteks global dari seluruh urutan input.
- **Multi-Head Attention**
Menggunakan beberapa "heads" untuk memungkinkan model menangkap berbagai aspek dari hubungan antara kata-kata dalam urutan.
- **Feed-Forward Neural Network (FFNN)**
Lapisan FFNN adalah jaringan saraf dengan dua lapisan linear yang terpisah oleh aktivasi non-linear (seperti ReLU). Setiap posisi dalam urutan diproses secara independen oleh lapisan FFNN yang sama.
- **Add & Norm**
 - **Residual Connection**, Output dari sub-lapisan self-attention dan FFNN ditambahkan kembali ke input asli dari sub-lapisan tersebut (skip connection).
 - **Layer Normalization**, dilakukan setelah penambahan untuk menjaga stabilitas pelatihan

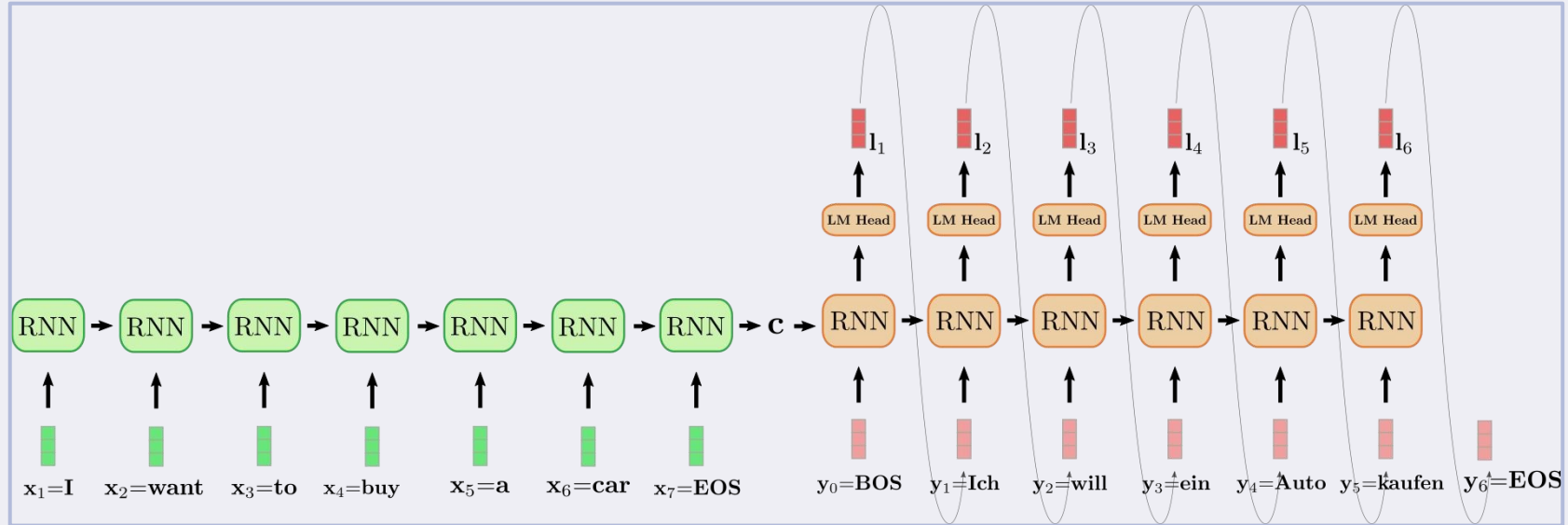
Arsitektur Transformers



Pada Lapisan Decoder:

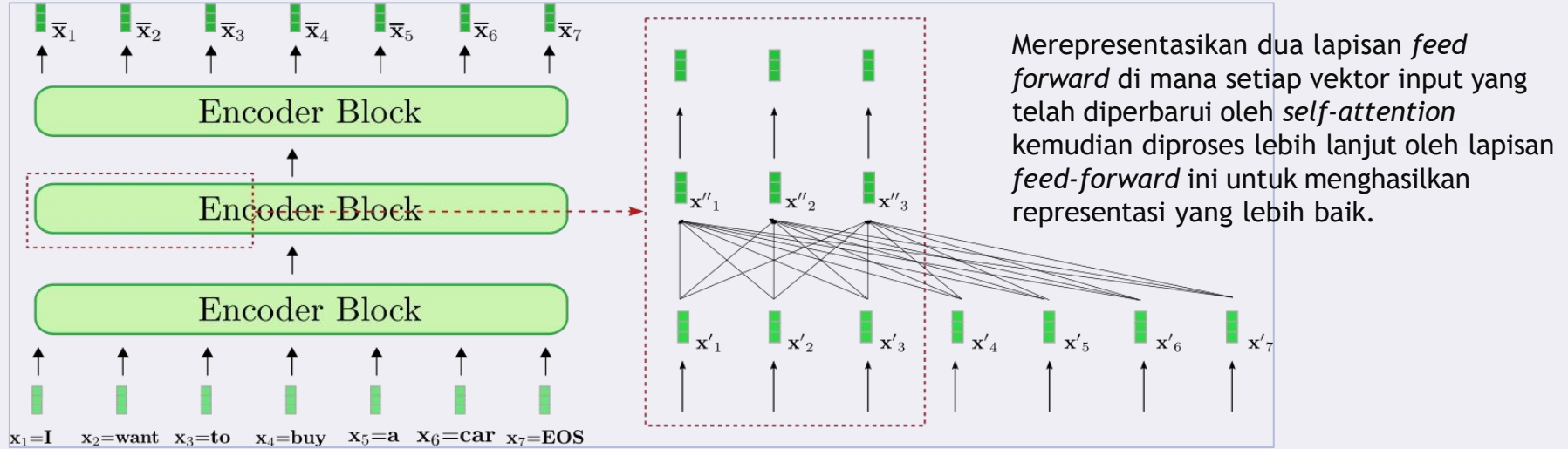
- **Masked Multi-Head Self Attention**
Sama seperti self-attention di encoder, tetapi dengan mekanisme masking untuk memastikan bahwa prediksi untuk suatu posisi hanya bergantung pada posisi sebelumnya dalam urutan (causal attention).
- **Multi-Head Self Attention**
Menggunakan output dari encoder untuk membantu dalam menghasilkan output yang lebih informatif dan kontekstual
- **Feed-Forward Neural Network (FFNN)**
Sama seperti pada encoder, setiap posisi dalam urutan diproses secara independen oleh lapisan FFNN yang sama.
- **Add & Norm**
Sama seperti pada encoder, residual connection dan layer normalization digunakan setelah setiap sub-lapisan.

Mekanisme Attention pada RNN



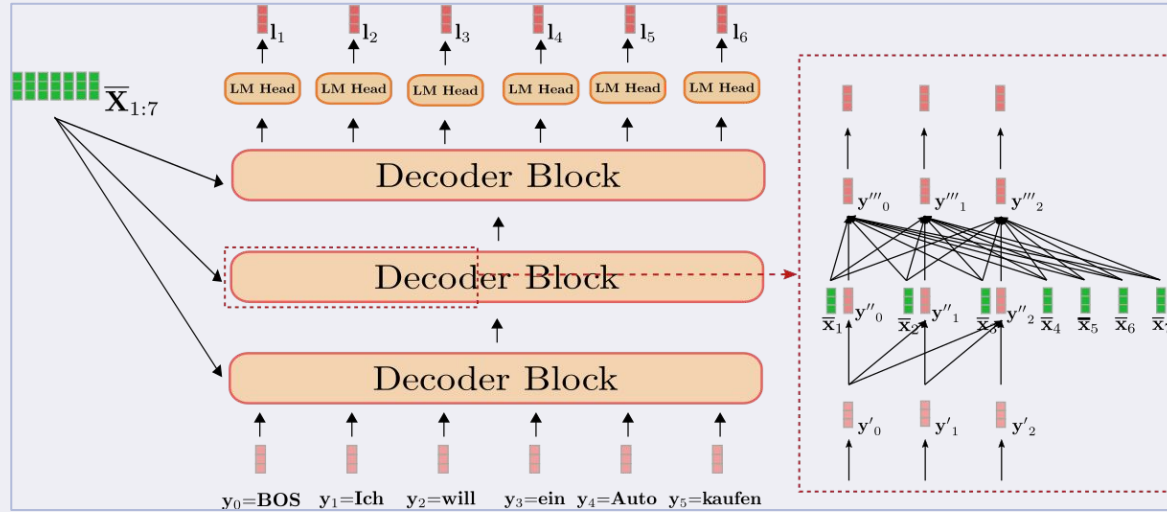
Pada arsitektur RNN, EOS (*End of Sentence*) berperan sebagai elemen terakhir vektor input yang mengakhiri deret yang akan di-encode. Sedangkan BOS (*Begin of Sentence*) berperan sebagai target pertama yang akan di-decode ke dalam arsitektur RNN. Karena sifatnya yang tidak paralel, maka waktu proses dengan RNN akan linier dengan panjang deret yang diproses.

Mekanisme Attention pada Transformers



Representasi *bidirectional self attention* di mana setiap node (representasi dari vektor input) terhubung dengan node lainnya, menunjukkan bahwa setiap vektor input memperhatikan (menerima informasi dari) vektor input lainnya.

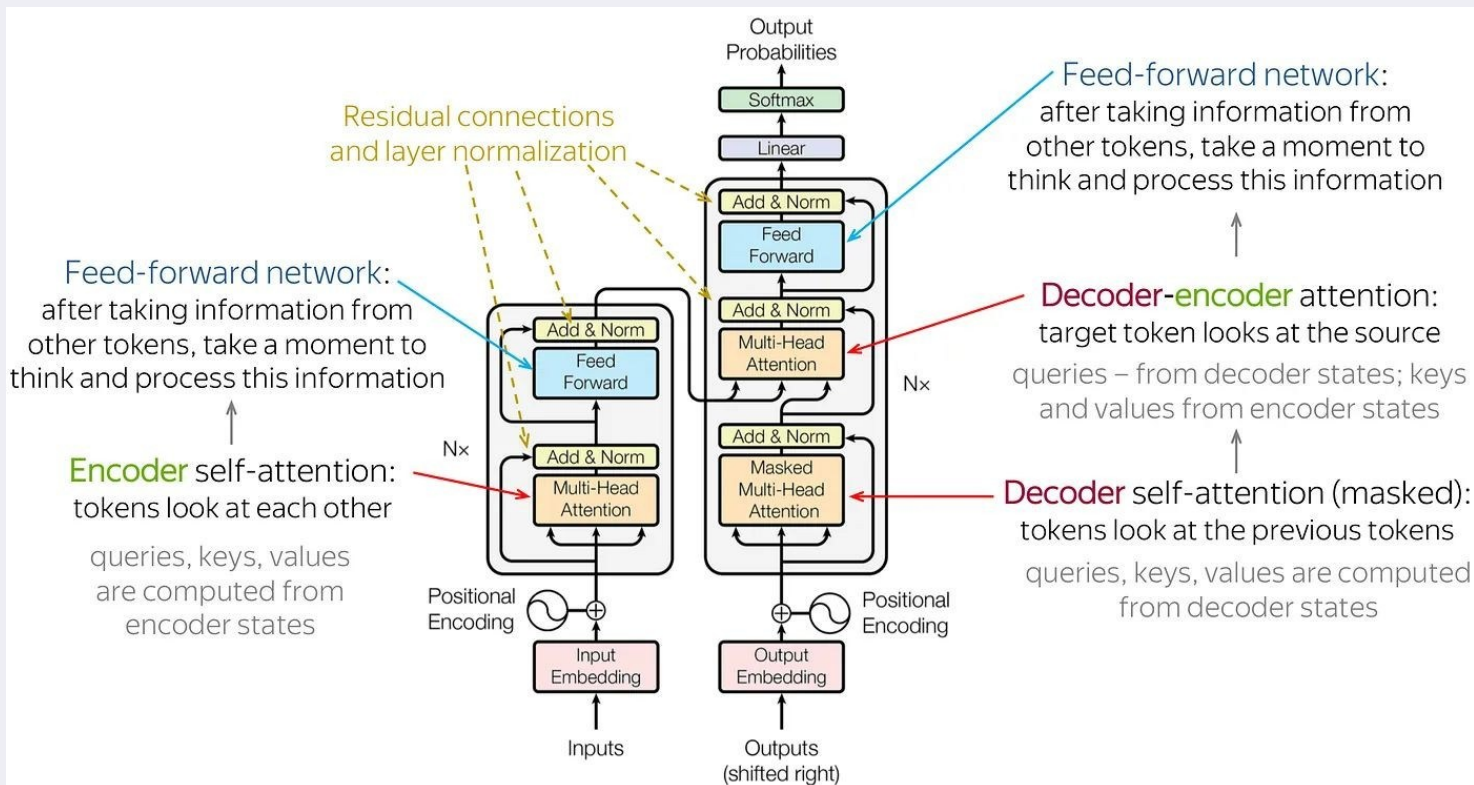
Mekanisme Attention pada Transformers

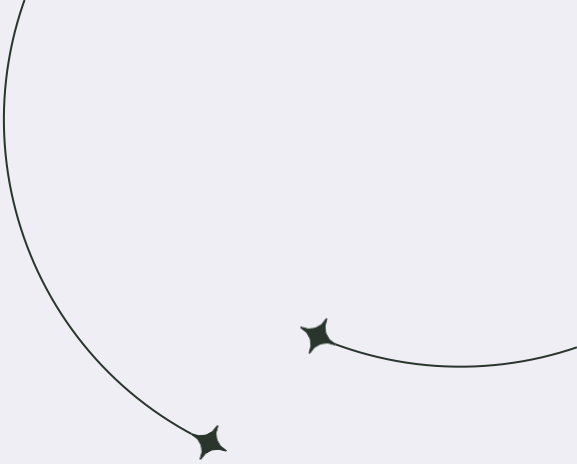


Pada arsitektur encoder, vektor outputnya dirancang untuk merepresentasikan vektor target berikutnya dan dikondisikan pada seluruh urutan encoding. Blok decoder terdiri dari lapisan self-attention searah yang memastikan hanya informasi dari posisi sebelumnya yang digunakan, lapisan cross-attention yang menghubungkan vektor input dengan vektor encoding, dan dua lapisan feed-forward untuk pemrosesan lebih lanjut.

Mekanisme ini memungkinkan decoder untuk memprediksi urutan target dengan mempertimbangkan konteks penuh dari input yang diberikan oleh encoder.

Encoding vs Decoding



A decorative graphic in the top right corner consisting of a thin, dark grey curved line that starts from the top edge and curves downwards and to the left, ending with a small, dark grey four-pointed star.

03

Implementasi dan Variasi Transformers

Implementasi Dasar



PyTorch

Menggunakan pendekatan pemrograman yang lebih dinamis sebagaimana Python bekerja, dan bisa dilakukan debugging. Selain itu, PyTorch memungkinkan modifikasi grafik komputasi secara dinamis pada runtime. PyTorch memiliki ekosistem kuat untuk penelitian AI dan sangat populer di kalangan peneliti akademik. Fleksibel dan API nya lebih sederhana dan intuitif untuk pemula.



Tensorflow

Menggunakan grafik komputasi yang statis, optimisasi komputasi lebih baik tetapi kurang fleksibel dibandingkan PyTorch dalam hal runtime. Biasa digunakan pada proses produksi dan deployment berskala besar dan mendukung deployment di berbagai platform (mobile, web, server), performa lebih tinggi, eksekusinya efisien, dan lebih komprehensif dalam menawarkan berbagai tools untuk workflow ML mulai dari training hingga deployment.



Transformers untuk NLP



Machine Translation

Untuk penerjemahan mesin, transformer biasanya terdiri dari dua bagian utama: encoder dan decoder. Encoder memproses teks input dalam bahasa sumber, sementara decoder menghasilkan teks dalam bahasa target. Contohnya adalah pengembangan **Google Translate** dan **DeepL Translator**.

Pemodelan Bahasa (LM)

Untuk pemodelan bahasa, transformer seperti GPT (Generative Pre-trained Transformer) digunakan untuk memprediksi kata berikutnya dalam sebuah urutan teks. Contohnya pengembangan **GPT-3** dan **BERT** yang digunakan untuk Chatbots dan Autocompletion

Summarization

Untuk summarization, model transformer seperti **BART** (Bidirectional and Auto-Regressive Transformers) dapat digunakan. Model ini belajar untuk memahami teks panjang dan menghasilkan ringkasan yang lebih pendek dan informatif. Selain itu ada model T5 (Text-To-Text Transfer Transformer)





04

Pelatihan dan Evaluasi

Persiapan Data



Pra-Pemrosesan

Proses tokenisasi memecah teks menjadi token yang dapat berupa kata, sub-kata, atau karakter.



Padding

Setelah tokenisasi, kita harus memastikan semua urutan memiliki panjang yang sama dengan menambahkan padding. Hal ini diperlukan agar batch data dapat diproses secara efisien.



Data Final

Terdiri dari:

- Input IDs: Urutan token yang telah di-tokenisasi dan di-padding.
- Attention Masks: Mask yang menunjukkan token mana yang merupakan bagian dari teks asli (1) dan mana yang merupakan padding (0).



Proses Pelatihan

Beberapa Hyperparameter Tuning:

1. **Learning Rate**
Kecepatan model dalam memperbarui bobot selama pelatihan.
2. **Batch Size**
Menentukan jumlah sampel yang diproses sebelum model memperbarui bobotnya.
3. **N_Layers**
Jumlah lapisan dalam model transformer.
4. **Hidden Size**
Dimensi dari representasi laten di setiap lapisan.
5. **Number of Attention Heads**
Jumlah head dalam mekanisme multi-head attention.
6. **Epochs**
Banyak iterasi training sampai konvergen.

Beberapa teknik optimasi: Adam (Adaptive Moment Estimation)

Adam mempertahankan momen pertama (mean) dan kedua (variance) dari gradien.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

AdamW

AdamW adalah varian dari Adam yang menambahkan regularisasi weight decay secara eksplisit.

$$\theta_t = \theta_{t-1} - \alpha \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_{t-1} \right)$$

Beberapa teknik regularisasi:

Weight Decay

Menambahkan penalti terhadap besarnya bobot dalam fungsi loss untuk mencegah bobot menjadi terlalu besar dengan menambah bobot pada optimizer.

Dropout

Secara acak mengabaikan (drop) unit-unit tertentu dalam jaringan selama pelatihan untuk mencegah model terlalu tergantung pada unit tertentu dengan menambahkan layer dropout.

Evaluasi dan Validasi



Metrik

Metrik yang digunakan bisa dengan Accuracy, Recall, Precision, F1 Score, AUC, dan lain-lain.



Cross-Validation

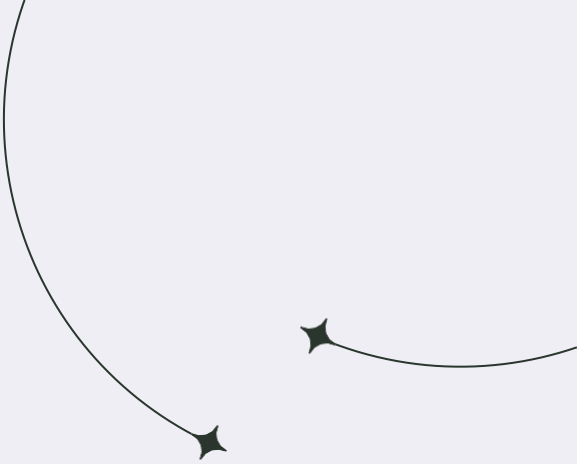
Beberapa teknik CV yang digunakan antara lain K-Fold, Stratified K-Fold, dan Leave-One-Out (LOOCV).



Pengembangan

Untuk pengembangan model dan evaluasi lanjutan, dapat dengan melakukan *ensembling*, *stacking*, atau *bootstrapping*.





05

Aplikasi dan Tantangan Masa Depan

Aplikasi Nyata



Chatbot dan Diagnosa Medis

Penggunaan transformer dalam mengembangkan chatbot kesehatan yang dapat memberikan saran medis dasar, menjawab pertanyaan pasien, dan mengingatkan mereka untuk minum obat.



Analisis Sentimen Pasar

Model transformer dapat menganalisis berita keuangan dan media sosial untuk mengukur sentimen pasar.



Pembuatan Konten

Model transformer seperti GPT-3 digunakan untuk menghasilkan konten otomatis seperti artikel, blog, dan konten media sosial, membantu penulis dan pemasaran dalam menghasilkan ide dan konten yang berkualitas.

Tantangan dan Batasan



Computational Cost dan Efisiensi

Tantangan ini mencakup kebutuhan akan GPU atau TPU dengan memori besar dan kemampuan komputasi tinggi, dapat diatasi dengan distilasi model (DistilBERT) untuk mengurangi kebutuhan komputasi.



Interpretabilitas dan Bias dalam Model NLP

Model transformer sering kali dianggap sebagai "black box" karena kompleksitas arsitekturnya, membuat interpretabilitas hasilnya menjadi sulit. Selain itu, model ini bisa mengandung bias yang berasal dari data pelatihan yang tidak seimbang atau mengandung bias. Pengembangan teknik interpretabilitas seperti LIME (Local Interpretable Model-agnostic Explanations) dan SHAP (SHapley Additive exPlanations) untuk menjelaskan keputusan model. Menggunakan pendekatan fairness-aware training dan melakukan evaluasi bias untuk mengurangi dampak bias dalam model.

Penelitian Terbaru



Transformer-XL

Model yang mengatasi keterbatasan panjang konteks dengan menggunakan mekanisme memori yang panjang untuk menangani teks yang lebih panjang.



T5 (Text-To-Text Transfer Transformer)

Model serba guna yang menggunakan format teks-ke-teks untuk berbagai tugas NLP, meningkatkan fleksibilitas dan performa model.

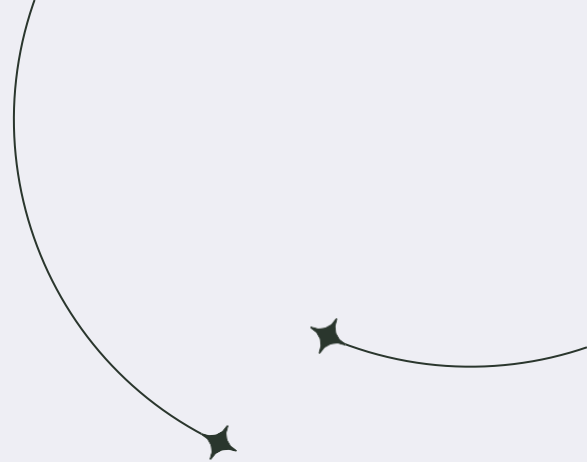


Switch Transformers

Menggunakan konsep mixture of experts untuk meningkatkan efisiensi dengan mengaktifkan hanya sebagian kecil parameter model selama inferensi.

06

BERT



Pengantar



Definisi

[BERT \(Bidirectional Encoder Representations from Transformers\)](#) adalah model bahasa pre-trained yang dikembangkan oleh Google pada tahun 2018. Model ini memanfaatkan arsitektur transformer dan dilatih secara unsupervised (tanpa label) pada jumlah besar teks dari web.



Fitur Utama

BERT unggul dengan kemampuannya untuk memahami konteks kata dalam sebuah kalimat. Hal ini dicapai melalui pendekatan "masked language modeling" (MLM), di mana model diajari untuk memprediksi kata yang di-mask pada kalimat yang diberikan. Selain itu, BERT juga menggunakan pendekatan "next sentence prediction" (NSP) di mana model diajari untuk memprediksi apakah sebuah kalimat adalah kalimat yang mengikuti kalimat sebelumnya dalam sebuah dokumen.



“Attention is All You Need.”

—Vaswani, et al (2017)



Referensi



Tunstall, L., Von Werra, L., & Wolf, T. (2022). *Natural language processing with transformers*. " O'Reilly Media, Inc.". Chicago.

Xiao, T., & Zhu, J. (2023). Introduction to Transformers: an NLP Perspective. *arXiv preprint* arXiv:2311.17633.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Gillioz, A., Casas, J., Mugellini, E., & Abou Khaled, O. (2020, September). Overview of the Transformer-based Models for NLP Tasks. *In 2020 15th Conference on Computer Science and Information Systems (FedCSIS)* (pp. 179-183). IEEE.

✦ ✦ Hasan, M. M. (2022). Transformers in natural language processing.

Jurafsky, D., & Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint* arXiv:1810.04805.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint* arXiv:1910.01108.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint* arXiv:1907.11692.



Colaboratory



Thanks!

CREDITS: This presentation template was created by [Slidesgo](#),
including icons by [Flaticon](#) and infographics & images by [Freepik](#)