**NLP (*Natural Language Processing*)**

**Diskusi Pertemuan 3**

Nama : Hendro Gunawan

NIM : 200401072103

Kelas : IT701

***Hugging Face***

****

Gambar 1. *Hugging Face*

**3.1 *Hugging Face***

1. Selamat datang di *Hugging Face*: pelopor dalam *Natural Language Processing* (NLP).
2. Dibangun di atas *PyTorch*, ia menawarkan ekosistem yang kaya yang menyederhanakan bekerja dengan model *Transformer*.
3. Dengan *Hugging Face*, memanfaatkan kemampuan NLP yang canggih menjadi eksplorasi yang menarik daripada tugas yang rumit.

**3.2 Mengapa *Hugging Face*?**

* Jelajahi [*Hugging Face Library*](https://huggingface.co/docs/hub/index) untuk semua kebutuhan *Natural Language Processing* (NLP) Anda:
  1. Pengaturan Model: Mulai perjalanan AI Anda dengan cepat.
  2. Manajemen: Mengatur model Anda secara efisien.
  3. Penerapan: Luncurkan AI Anda dengan cepat.

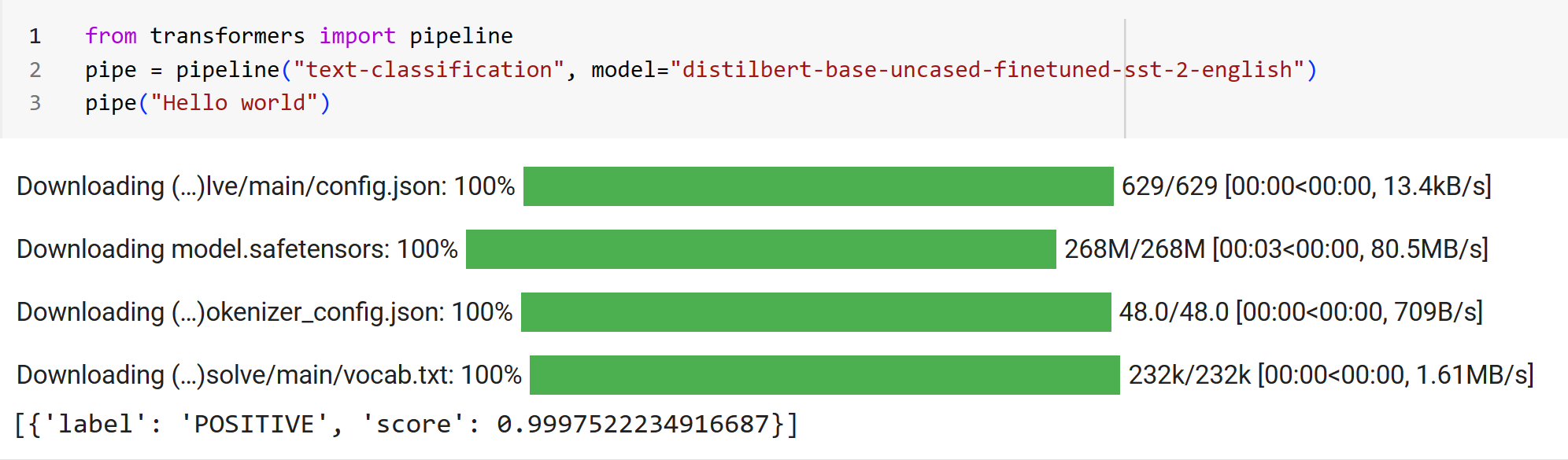
**3.3 Instalasi dan Konfigurasi**

* Memulai perjalanan NLP Anda dengan *Hugging Face Library* itu mudah! Jika Anda menggunakan *pip* atau *conda*, cukup buka terminal Anda dan ketik:

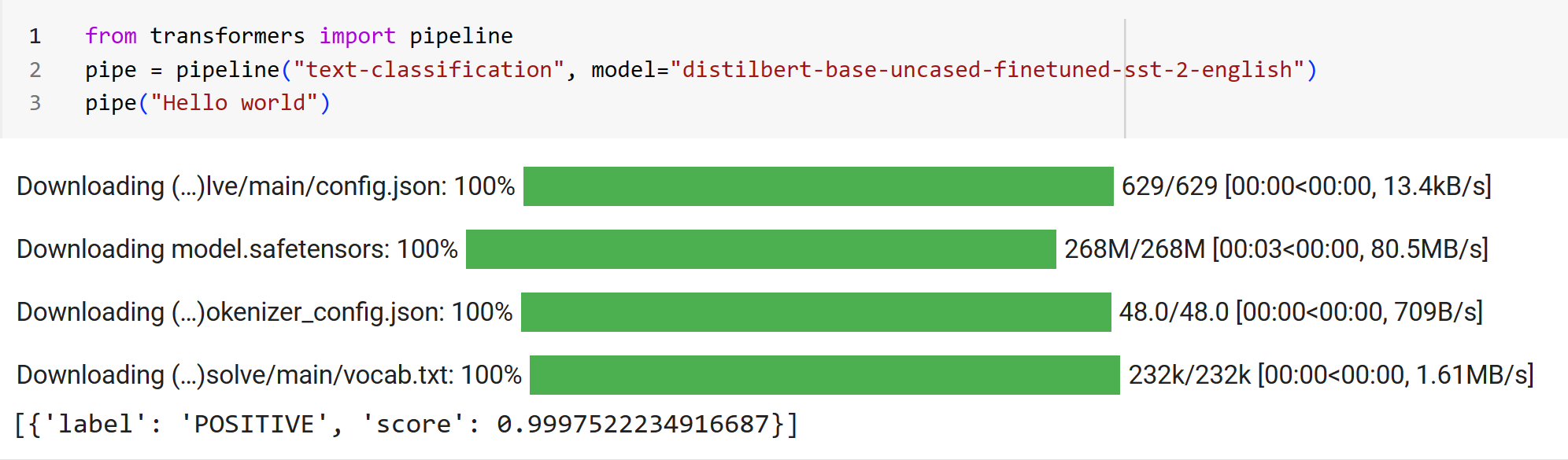


* Cuplikan kode menampilkan cara menggunakan *Hugging Face Library*:

Itu mengimpor *pipeline* dari *transformer* dan membangun klasifikasi teks *pipeline* dengan "*distilbert-base-uncased-finetuned-sst-2-english*" model. Kemudian mengklasifikasikan frasa "*Hello world*". Kode ringkas ini melakukan analisis sentimen menggunakan model *DistilBERT* yang telah dilatih sebelumnya!



*Output*:

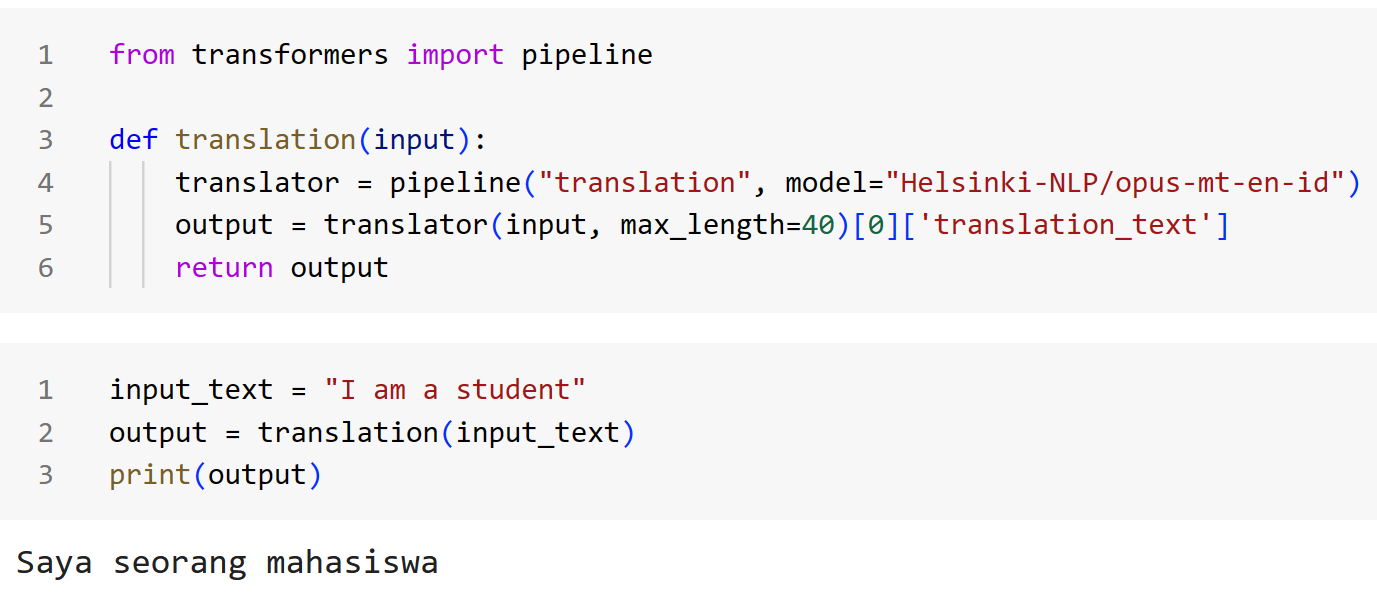


**3.4 *Transformers***

* Model ini, dari sebuah "*High-Level Look*", bertindak sebagai *'black box'* Dalam **aplikasi terjemahan mesin** dengan model *transformator*, mengubah kalimat dari satu bahasa menjadi terjemahannya di bahasa lain.



Gambar 2. Algoritma aplikasi terjemahan mesin.



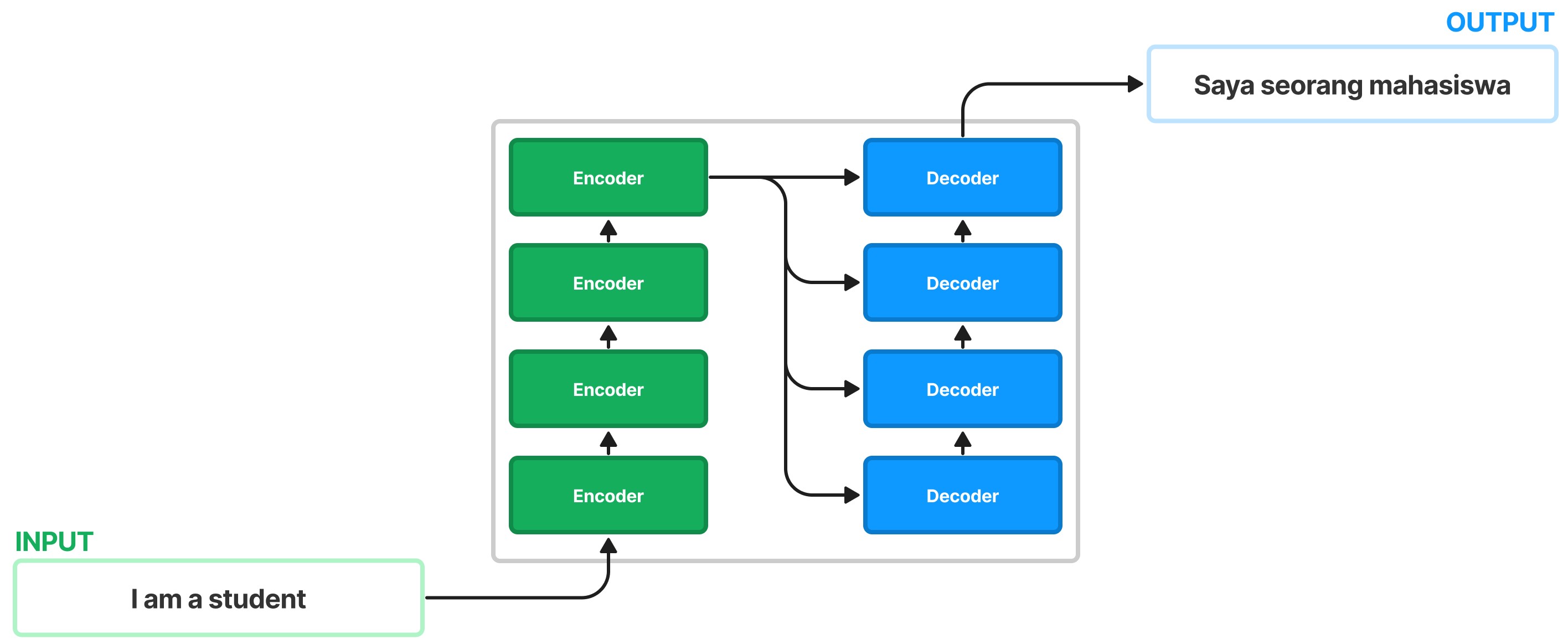
**3.5 Mari Kita Lihat Apa yang Ada di dalam *Transformer*?**

* Membuka *Transformer* itu, kita dapat mengamati komponen *encoding*, komponen *decoding*, dan koneksi rumit di antara mereka.



Gambar 3. Algoritma terjemahanmesin

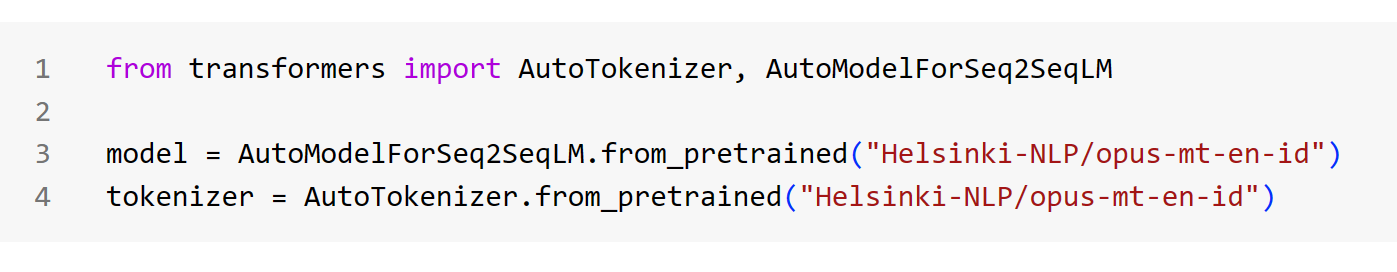
* Dalam model *Transformer*, bagian pengkodean terdiri dari tumpukan *encoder*, sedangkan bagian *decoding* memiliki tumpukan *decoder* yang sama.



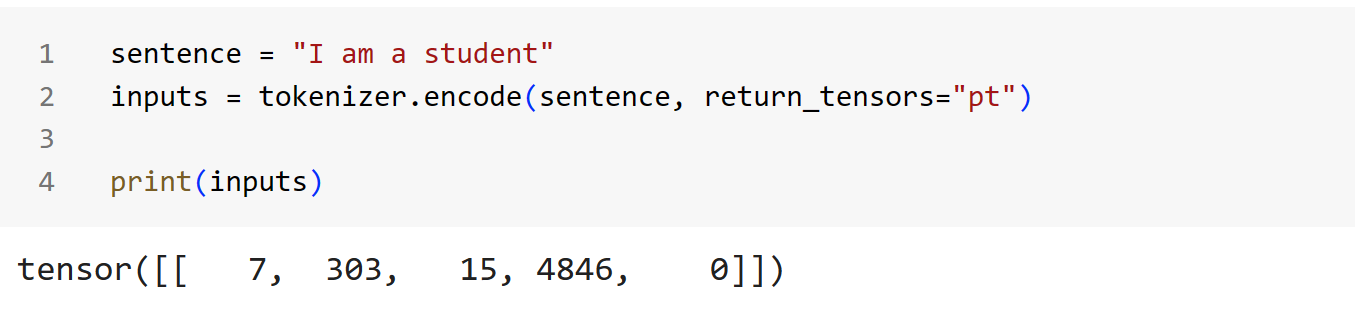
Gambar 4. Tumpukan *enkoder* dan *dekoder*

**3.6. Ikhtisar Kode Tentang Cara Kerja Model Enkoder dan *Transformer* Dekoder**

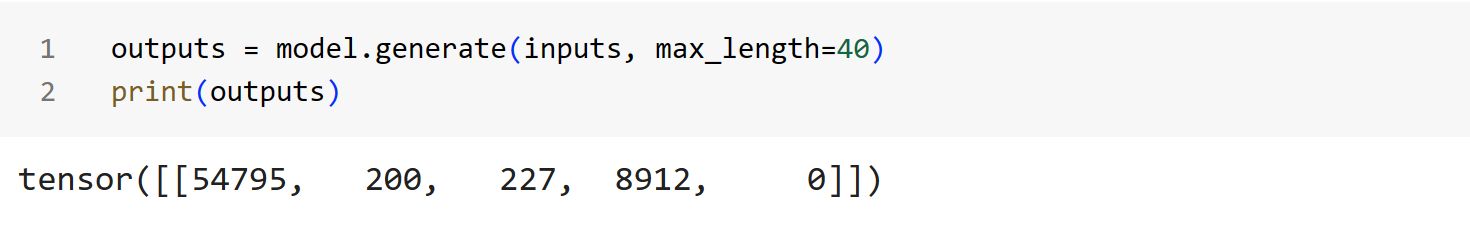
Pertama, impor model dan tokenizer dari transformator dengan Model terlatih "*Helsinki-NLP/opus-mt-en-id*":



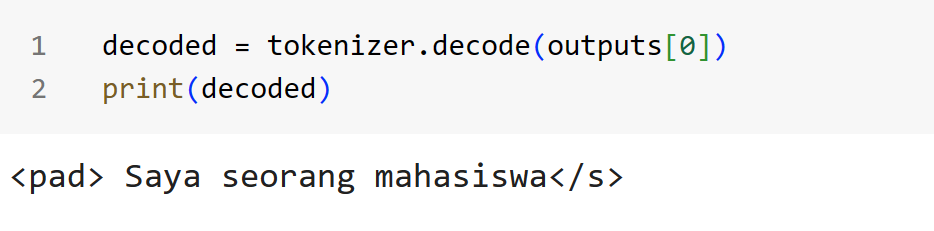
1. Di sini, model adalah model Seq2Seq *Transformer* terlatih (*Encoder* dan *Decoder*) dan tokenizer adalah cara untuk mengubah teks biasa menjadi token yang dapat dipahami model.
2. Selanjutnya, kita ambil kalimat, "Saya seorang siswa", dan mengubahnya menjadi token:



* **inputs** sekarang merupakan representasi token dari kalimat yang dapat diproses oleh model.
* Mengikuti alur kerja model Seq2Seq yang khas, pada langkah ini, *encoder* model Seq2Seq memproses input dan menghasilkan representasi internal:



* Output adalah representasi internal yang dibuat oleh bagian *encoder* dari model.
* Akhirnya, kami memecahkan kode *output* ini kembali ke teks yang dapat dibaca manusia:



* *Decoded* adalah *output* teks yang diterjemahkan oleh model.
* Pada tahap ini, *decoder* telah mengambil representasi internal dari *encoder* dan menghasilkan output teks yang diinginkan.

**3.7. Arus Informasi**

* Animasi di bawah ini menggambarkan bagaimana kita menerapkan *Transformer* ke terjemahan mesin.
* Klik di sini untuk melihat animasi

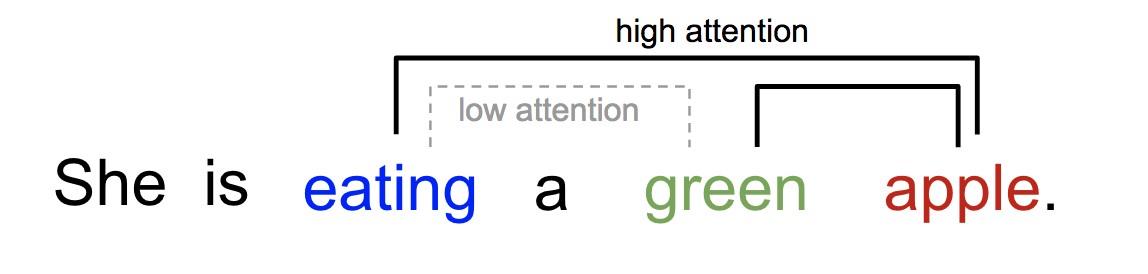


Gambar 5. Animasi Enkoding

**3.8. *Self-Attention***

* *Transformer* adalah model pembelajaran mendalam yang menggunakan *self attention* untuk memproses data input berurutan, seperti bahasa atau gambaralami.

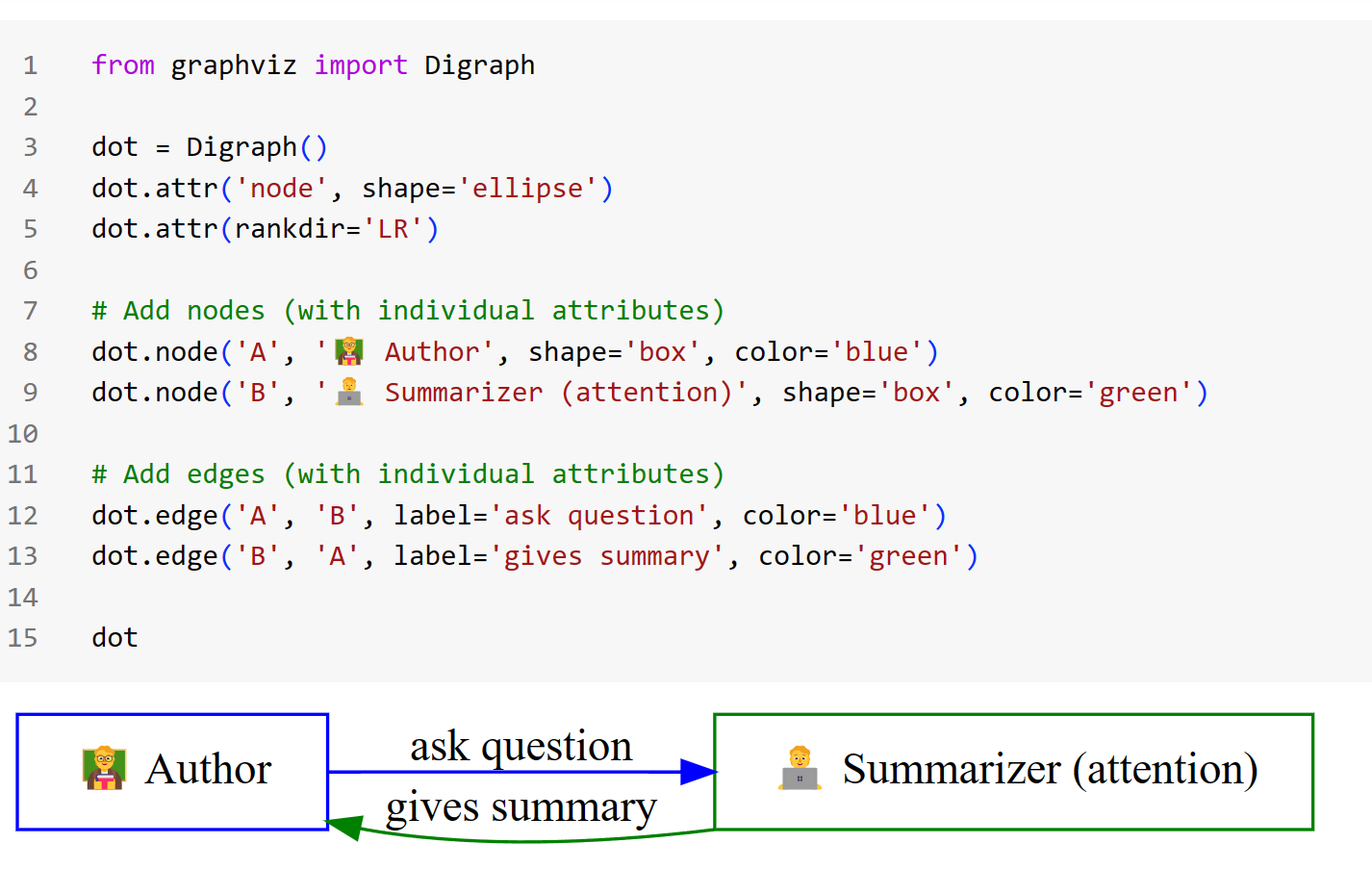
Sekarang, apa itu *Self-Attention*?



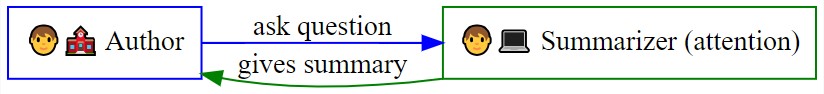
Gambar 6. *Self-Attention.*

* *Self-attention*, atau perhatian *transformator*, adalah komponen vital dalam model *Transformer* yang memungkinkan setiap token dalam kalimat untuk saling berhubungan dengan setiap token lainnya, terlepas dari jaraknya.
* Hal ini memungkinkan model untuk lebih memahami konteks dan nuansa linguistik.

**3.9. *Attention Basic***

******

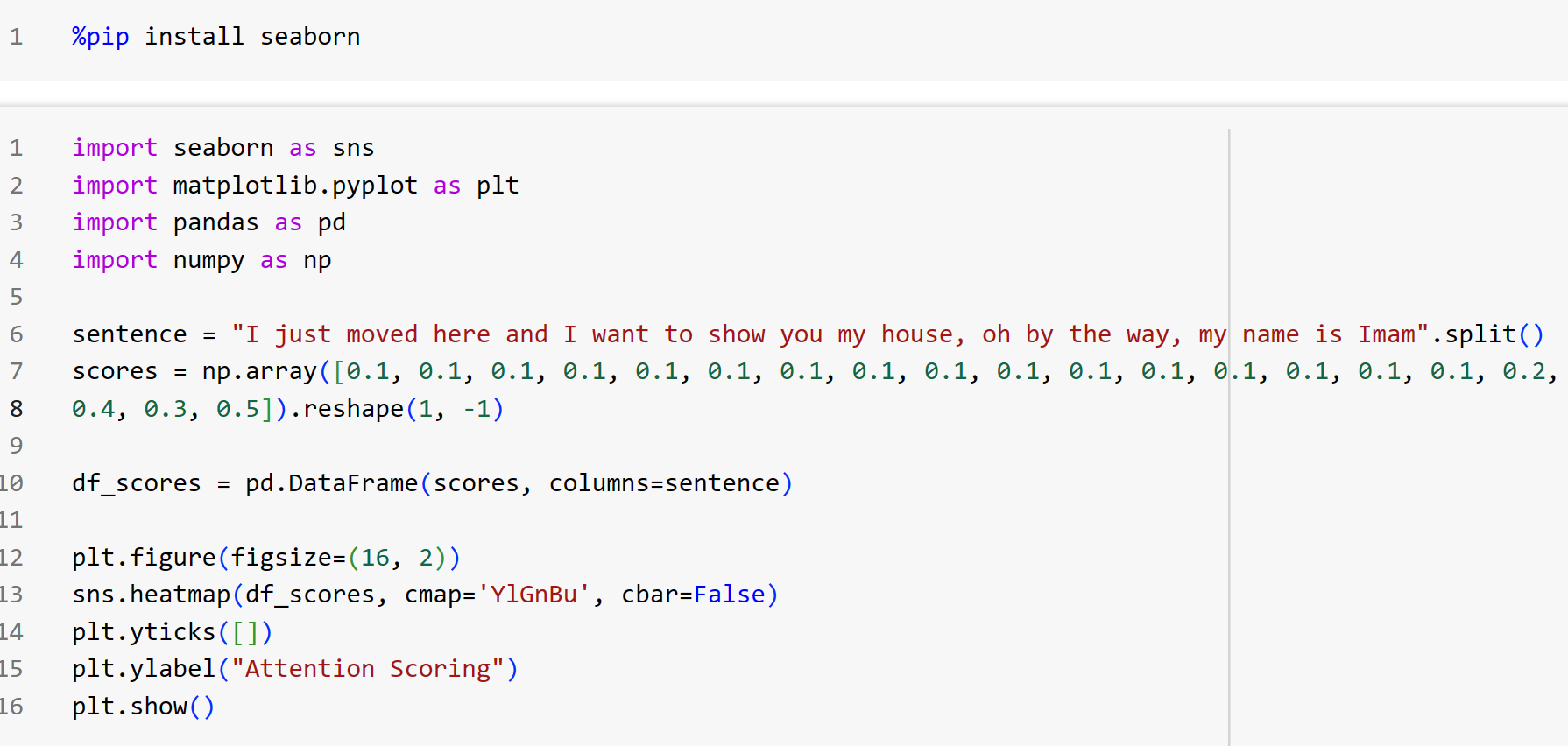
* Arsitektur "*Attention*" dalam pembelajaran mesin seperti seorang penulis yang menggunakan peringkas untuk menyoroti poin-poin penting.
* Ini seperti penulis mempertanyakan peringkas (mewakili mekanisme "*attention*"), yang menawarkan poin-poin penting yang diringkas alih-alih jawaban langsung.

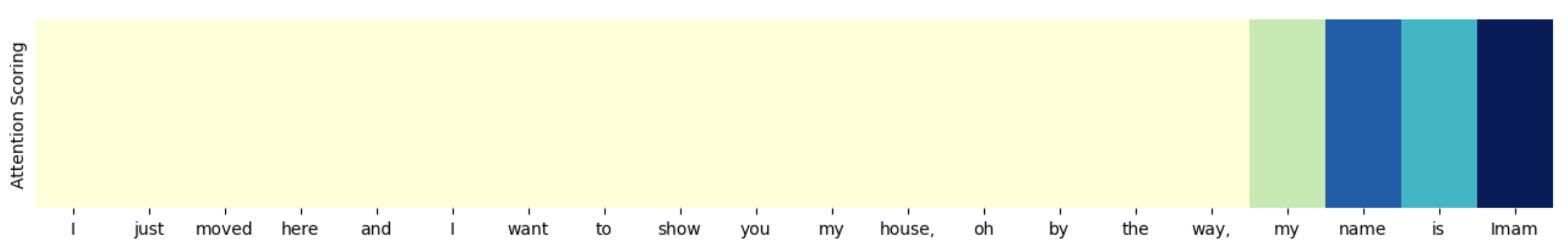


Gambar 7. Attention basic

**3.10. Contoh *Attention Basic***

1. Sebagai contoh: Untuk teks input, "Saya Jarwo, saya tinggal di Jakarta, saya *software engineer* di Unsia".
2. Ketika penulis bertanya, "Siapa yang diperkenalkan dalam kalimat ini?", Peringkas (*attention*) memberikan ringkasan: "Saya **Jarwo**, saya tinggal di **Jakarta**, saya **seorang insinyur perangkat lunak di Unsia**".
3. Contoh lain dengan visualisasi *seaborn*: untuk "Saya baru saja pindah ke sini dan saya ingin menunjukkan rumah saya, oh omong-omong, nama saya Imam", pertanyaannya adalah "Siapa nama saya?"





Gambar 8. *Attention scoring.*

* Visualisasi ini dapat digunakan untuk menunjukkan kata-kata mana dalam kalimat yang paling penting. Semakin gelap visualisasinya, semakin penting kata itu. Jika penulis membutuhkan informasi lebih lanjut, mereka dapat melihat kata kunci berikutnya yang lebih ringan.

**3.11. *Multi-Head Attention***

1. *Transformer* menggunakan mekanisme yang dikenal sebagai perhatian *multi-head*.
2. Tetapi mengapa ini penting? Perhatian *multi-head* memungkinkan model untuk fokus pada konteks posisi yang berbeda secara bersamaan.
3. Ini menyelesaikan tantangan untuk memahami berbagai implikasi kontekstual kata-kata dalam sebuah kalimat.
4. Setiap *'head'* dapat memperhatikan bagian input yang berbeda, sehingga memberikan pemahaman data yang lebih kaya.



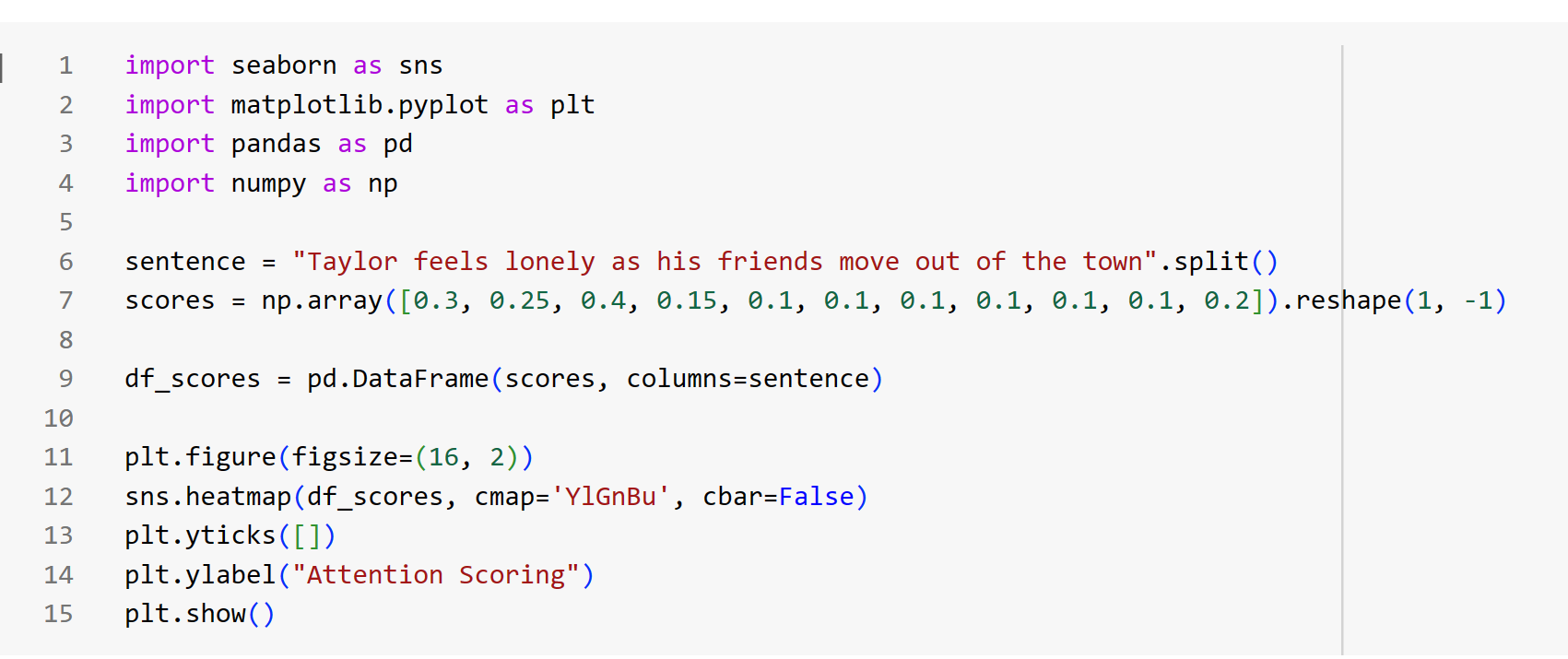


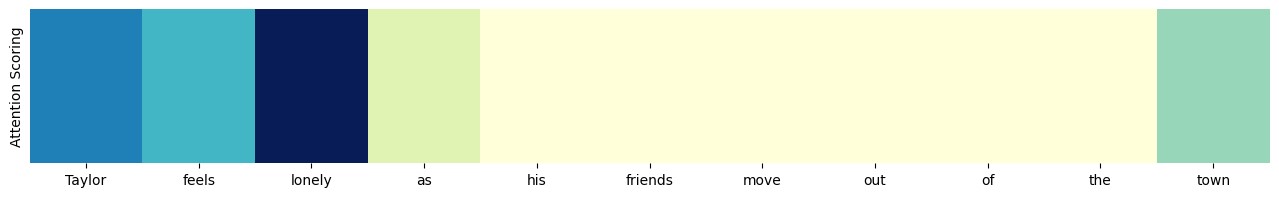


Gambar 9. *Multi-head attention*

**3.12. Contoh *Multi-Head Attention***

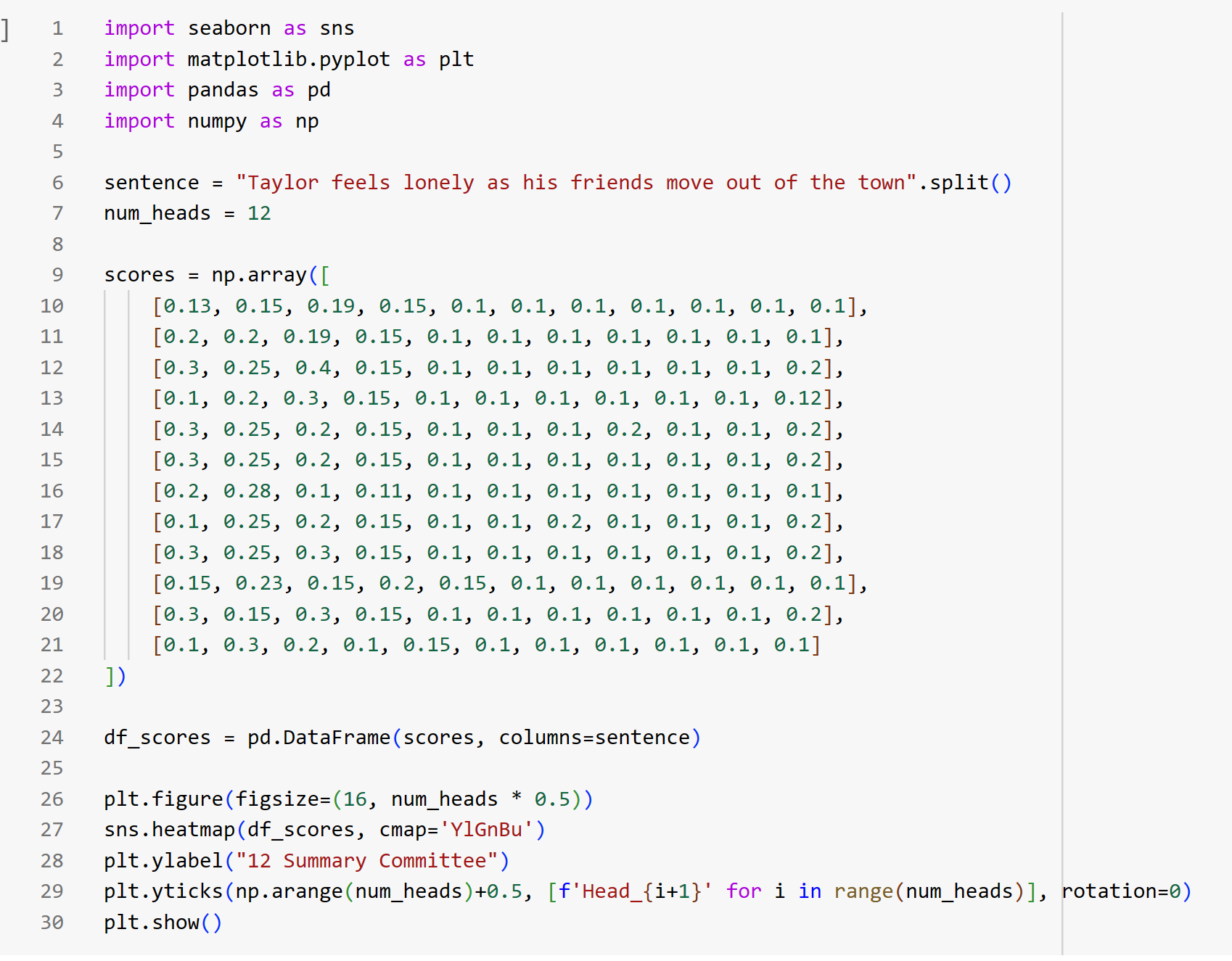
Contoh: Untuk teks input***, "Taylor feels lonely as his friends move out of the town".*** Saat penulis bertanya***, "What does Taylor feel?*"**

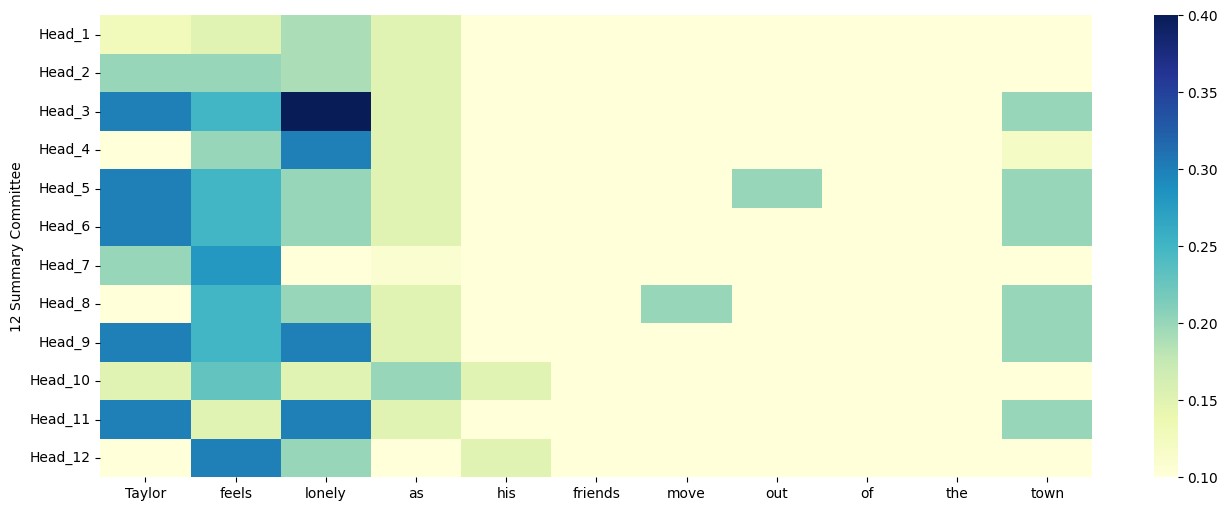
****



Gambar 10. Contoh *multi-head attention*

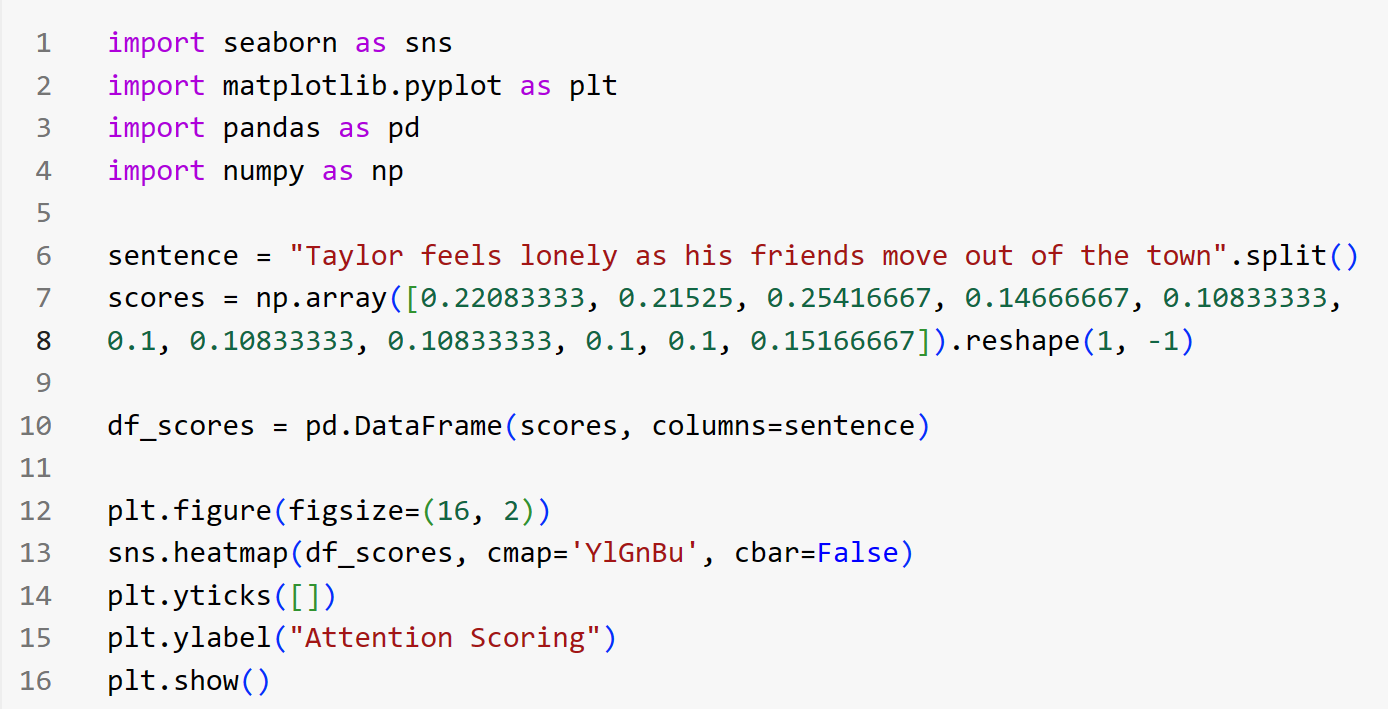
* Masing-masing dari mereka merangkum dengan gaya mereka sendiri

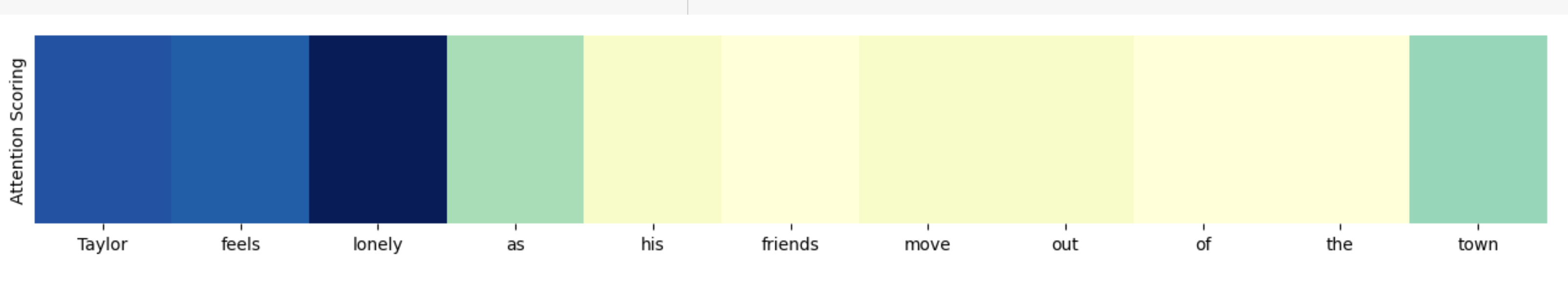
****



Gambar 11. Contoh *multi-head attention*

Suara mereka bersatu ...





Gambar 12. Contoh lain *multi-head attention*.

* Bisakah kita menjelaskan apa yang masing-masing kepala "pikirkan"? 🤔

Jawabannya adalah NO.

* Kesatuan suara kepala pada dasarnya adalah agregasi skor perhatian dari masing-masing kepala, menekankan bagian yang paling relevan dari input yang menjawab pertanyaan. Namun, tidak semua kepala setuju tentang pentingnya setiap kata. Misalnya, beberapa kepala mungkin menganggap kata "*move*" (pindah) relevan, karenanya warna cyan gelapnya, berpotensi karena mereka pikir itu adalah faktor penting yang menyebabkan kesepian (l*onely*) *Taylor*.
* Kami hanya bisa berspekulasi fokus masing-masing kepala. Mungkin satu berfokus pada struktur kalimat, yang lain pada konteks, dan sebagainya. Karena *inscrutability* jaringan saraf, kita tidak dapat memastikan proses berpikir yang tepat dari masing-masing kepala (kurangnya interpretabilitas).

**Kesimpulan:**

- Perhatian multi-kepala memungkinkan setiap kepala untuk fokus pada bagian input yang berbeda untuk meningkatkan pemahaman.

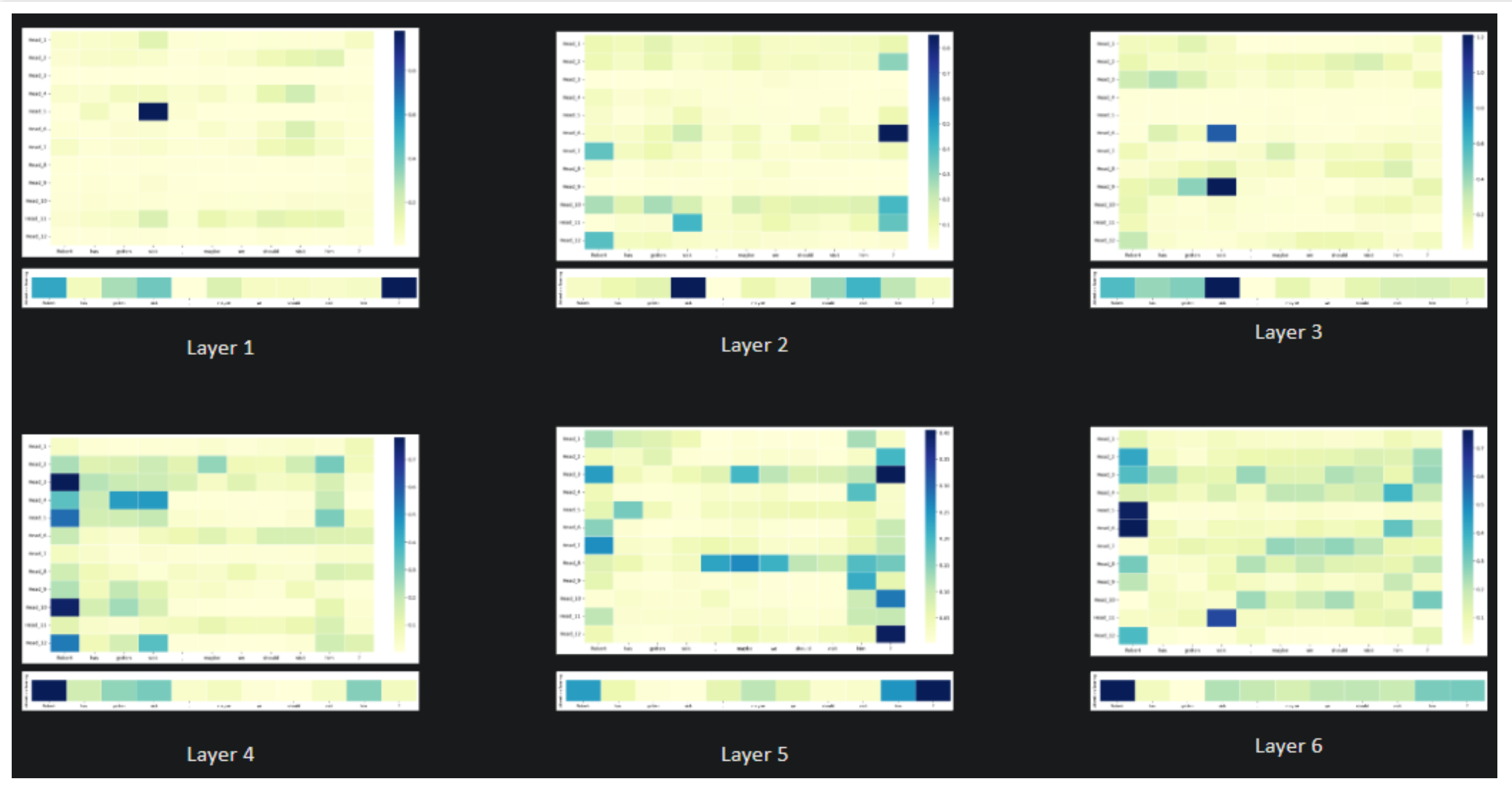
- Setiap kepala memberikan skor untuk setiap kata, menentukan relevansinya.

- Gabungan dari skor ini membentuk skor perhatian akhir.

- Jumlah kepala tergantung pada model.

**3.13. *Multi Layer Attention***

* *Transformer* juga menggabungkan perhatian multi-layer. Tapi apa artinya ini, dan mengapa itu perlu?
* Perhatian multi-layer memungkinkan model untuk secara iteratif memperbaiki pemahamannya tentang input pada beberapa lapisan.
* Setiap lapisan menawarkan model kesempatan lain untuk mempelajari asosiasi dalam data.
* Pendekatan berlapis ini membahas kebutuhan akan pemahaman yang lebih dalam dan hubungan yang lebih kompleks dalam data.

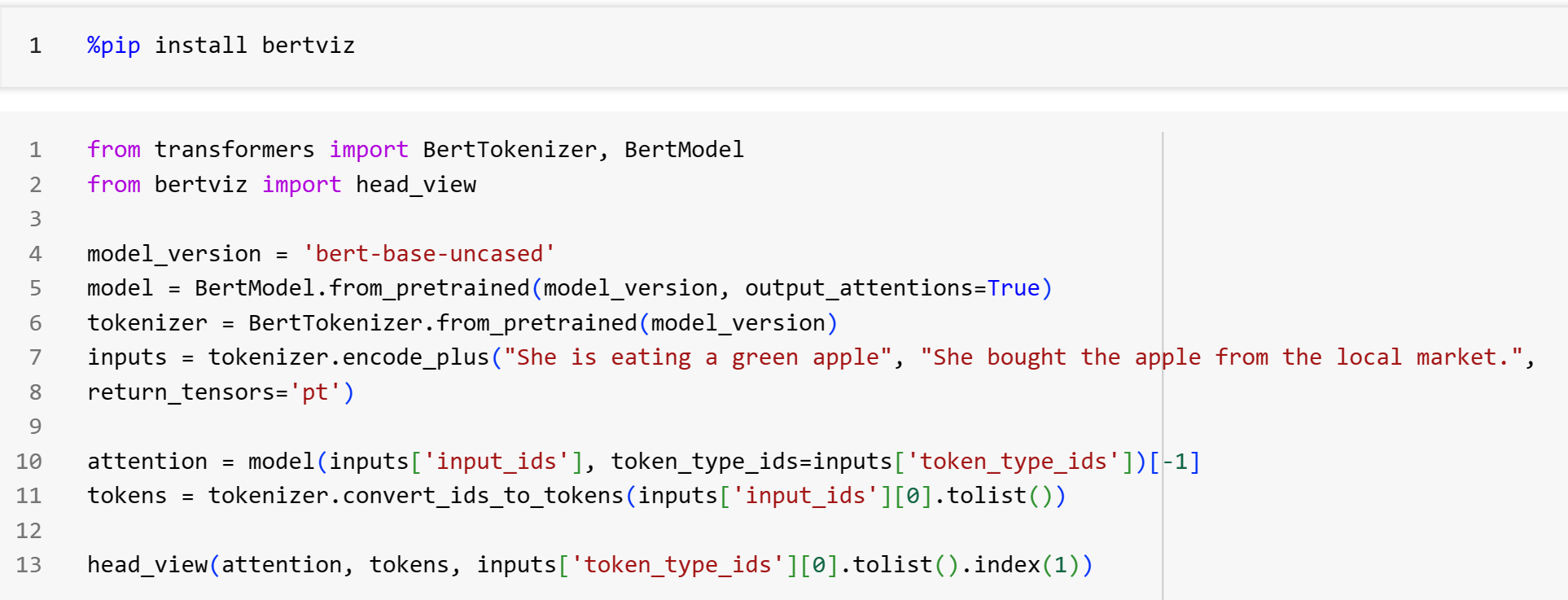


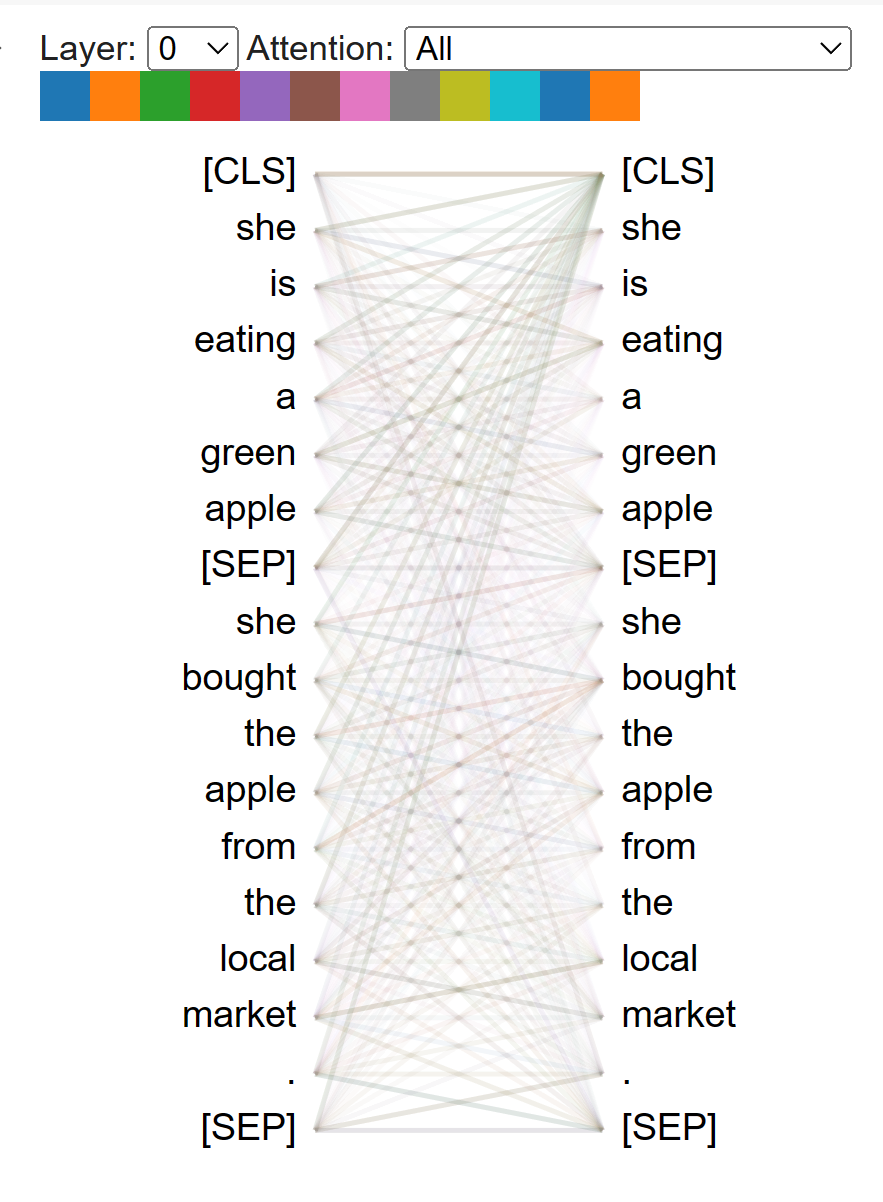
Gambar 13. *Multi layer attention*.

* Kepala diduplikasi di seluruh lapisan, masing-masing diminta untuk mempelajari kembali tugasnya.
* Jumlah lapisan bervariasi tergantung pada model.
* Lapisan terakhir menggabungkan kepala yang paling memahami tugas yang ada.
* Namun, semua kepala di semua lapisan dipertahankan.
* Dalam situasi yang jarang terjadi, kepala dari lapisan sebelumnya mungkin lebih efisien dalam menyelesaikan tugas daripada yang ada di lapisan selanjutnya.

**3.14. Perhatian Interaktif dengan *BertViz***

Sekarang, kita akan menggunakan *BertViz* untuk memvisualisasikan perhatian diri dalam model \*\*BERT yang telah dilatih sebelumnya\*\* dengan beberapa "*head*" yang menangkap hubungan kata yang berbeda. Model 12-*layer*, 12-*head* menghasilkan 144 mekanisme perhatian yang unik.





Gambar 14. *Layer* *Bertviz.*

**3.15. Aplikasi *Transformer***

*Transformer*s are adaptable and can learn from large data sets for tasks like:

1. *Translation*: Menterjemahkan bahasa satu ke bahasa yang lain.
2. *Summarization*: Mengubah teks panjang menjadi ringkasan singkat.
3. *Text Generation*: Membuat teks baru sendiri.
4. dan lain-lain.



**Referensi**

Syahid Abdullah, S. M. (2023, November 12). *Natural Language Processing*. Diambil kembali dari Edlink Universitas Siber Asia: https://edlink.id/panel/classes/563118

**Website**

Slide Share: <https://www.slideshare.net/HendroGunawan8/natural-language-processing-diskusi-3docx>

**Google Collab**

<https://colab.research.google.com/drive/1nBWcO-AkiMyb36BXKKTTKECLfWT0kECW#scrollTo=HvjAJVm_aOiu>