SimCSE: Simple Contrastive Learning of Sentence Embeddings

Mohamad Arvin Fadriansyah
Faculty of Computer Science
Universitas Indonesia
Depok, Indonesia
Email: mohamad.arvin@ui.ac.id

***Abstract*—Penelitian ini mereproduksi eksperimen SimCSE: Simple Contrastive Learning of Sentence Embeddings untuk mengevaluasi efektivitas metode pembelajaran kontrastif dalam menghasilkan representasi kalimat berkualitas tinggi. Model supervised SimCSE berbasis BERT-base dilatih pada subset dataset publik dan dievaluasi menggunakan korelasi Spearman pada tujuh dataset Semantic Textual Similarity (STS). Konfigurasi default menghasilkan rata-rata korelasi 77.78. Modifikasi parameter pelatihan menunjukkan sensitivitas model terhadap pengaturan ini, dengan performa berkisar antara 72.20 hingga 77.38. Temuan ini menegaskan pentingnya optimasi parameter untuk generalisasi model, membuka peluang pengembangan representasi kalimat yang lebih baik.**

***Index Terms*—SimCSE, sentence embedding, cosine similarity, Spearman, Pearson, semantic textual similarity.**

I. Pendahuluan

Kesamaan semantik antar kalimat adalah komponen penting dalam berbagai aplikasi Pemrosesan Bahasa Alami (NLP), seperti sistem pencarian informasi, klasifikasi dokumen, dan sistem tanya jawab. Untuk mengukur kesamaan ini, pendekatan berbasis embedding seperti SimCSE menjadi pilihan populer karena kemampuannya menghasilkan representasi semantik yang akurat. SimCSE menggunakan metode pelatihan kontrastif dengan supervisi (*supervised*) atau tanpa supervisi (*unsupervised*) untuk mengoptimalkan embedding kalimat.

Dalam penelitian sebelumnya "*SimCSE: Simple Contrastive Learning of Sentence Embeddings*" oleh T. Gao et al. [1], SimCSE menunjukkan performa tinggi, khususnya dalam versi supervised, dengan rata-rata korelasi Spearman mencapai 81.57% pada berbagai dataset STS. Model ini unggul dibandingkan baseline seperti SBERTbase-whitening (77.00%) dan CT-SBERTbase (79.39%). Namun, potensi pengaruh parameter pelatihan dan dataset alternatif terhadap kualitas embedding masih perlu diteliti lebih lanjut.

Penelitian ini bertujuan mengevaluasi ulang model SimCSE dengan pelatihan pada subset dataset untuk memahami bagaimana variasi parameter memengaruhi performa. Kami membandingkan hasil eksperimen ini dengan penelitian acuan, yang menggunakan SimCSE pada arsitektur BERT. Hasil eksperimen kami menunjukkan bahwa meskipun model default mendekati performa baseline (77.78% vs. 81.57%), ada penurunan performa signifikan pada beberapa konfigurasi parameter dan subset dataset STS, seperti Eksperimen 3 dengan rata-rata hanya 72.20%.

Kontribusi utama penelitian ini adalah memberikan wawasan tentang sensitivitas model SimCSE terhadap parameter pelatihan dan dataset, serta membandingkan hasilnya dengan baseline dan model penelitian sebelumnya. Sistematika penulisan meliputi: Bab 1 (Pendahuluan); Bab 2 (cara kerja program), Bab 3 (hasil uji coba dan analisis), dan Bab 4 (kesimpulan dan saran).

II. Cara Kerja Program

A. Alur Kerja Program

Alur kerja program dengan menggunakan model default

```
from simcse import SimCSE
model = SimCSE("princeton-nlp/sup-simcse-bert-base-uncased")
13.3s
```

Fig. 1. Inisialisasi model SimCSE dengan pre-trained weights dari princeton-nlp/sup-simcse-bert-base-uncased.

Gambar 1 di atas menggunakan pustaka SimCSE, sebuah implementasi pre-trained model berbasis transformer untuk mengetahui kemiripan suatu teks berdarsarkan semantiknya. Dengan memuat model bernama "princeton-nlp/sup-simcse-bert-base-uncased", kode ini menginisialisasi model SimCSE yang telah dilatih menggunakan pendekatan supervised. Model ini dirancang untuk mengubah teks menjadi vektor representasi di ruang laten, sehingga memungkinkan perbandingan semantik antara teks. Versi bert-base-uncased mengacu pada model BERT ukuran sedang yang mengabaikan huruf besar/kecil (*case insensitive*). Model ini sering digunakan untuk tugas seperti pencarian semantik, deteksi duplikasi teks, atau klasterisasi dokumen berdasarkan kesamaan makna.

Gambar 2 menggambarkan proses *encoding* teks menggunakan model SimCSE. Perintah `model.encode("A woman is reading.")` mengambil input berupa sebuah kalimat dan mengubahnya menjadi representasi vektor dalam ruang laten. Representasi ini mencerminkan makna semantik dari teks dan dapat digunakan untuk berbagai tugas seperti pengukuran kesamaan antar kalimat, pencarian informasi, atau klasterisasi teks berdasarkan konteks semantik. Encoding ini sangat berguna dalam pemrosesan bahasa alami untuk menangkap hubungan semantik antar kalimat.

```
1  embeddings = model.encode("A woman is reading.")
✓ 0.8s
100%|██████████| 1/1 [00:00<00:00,  1.29it/s]
```

Fig. 2. Proses encoding teks menggunakan model SimCSE untuk menghasilkan representasi vektor.

```
1  sentences_a = ['A woman is reading.', 'A man is playing a guitar.']
2  sentences_b = ['He plays guitar.', 'A woman is making a photo.']
3  similarities = model.similarity(sentences_a, sentences_b)
4  similarities
✓ 0.1s
100%|██████████| 1/1 [00:00<00:00,  8.48it/s]
100%|██████████| 1/1 [00:00<00:00, 24.03it/s]

array([[0.01262088, 0.34469506],
       [0.89384246, 0.04842845]], dtype=float32)
```

Fig. 3. Hasil perhitungan kesamaan semantik antar kalimat menggunakan model SimCSE. Matriks kesamaan menunjukkan skor kesamaan antara pasangan kalimat dari dua daftar input.

Gambar 3 di atas menunjukkan matriks kesamaan yang dihasilkan menunjukkan hubungan semantik antar kalimat pada dua daftar input, `sentences_a` dan `sentences_b`. Setiap elemen matriks pada baris i dan kolom j merepresentasikan skor kesamaan antara kalimat ke-i di `sentences_a` dan kalimat ke-j di `sentences_b`. Sebagai contoh, kalimat *"A man is playing a guitar."* memiliki skor kesamaan yang tinggi dengan *"He plays guitar."* (0.89), sementara *"A woman is reading."* memiliki kesamaan yang lebih rendah dengan *"A woman is making a photo."* (0.34).

```
1  sentences = ['A woman is reading.', 'A man is playing a guitar.']
2  model.build_index(sentences)
3  results = model.search("She is reading.")
4  results
✓ 0.5s
11/26/2024 10:26:14 - INFO - faiss.loader -   Loading faiss with AVX2 support.
11/26/2024 10:26:15 - INFO - faiss.loader -   Successfully loaded faiss with AVX2 support.
11/26/2024 10:26:15 - INFO - simcse.tool -   Encoding embeddings for sentences...
100%|██████████| 1/1 [00:00<00:00, 19.29it/s]
11/26/2024 10:26:15 - INFO - simcse.tool -   Building index...
11/26/2024 10:26:15 - INFO - simcse.tool -   Use CPU-version faiss
11/26/2024 10:26:15 - INFO - simcse.tool -   Finished
100%|██████████| 1/1 [00:00<00:00, 13.56it/s]

[('A woman is reading.', np.float32(0.9345436))]
```

Fig. 4. Hasil pencarian semantik menggunakan model SimCSE. Query *"She is reading."* menemukan kalimat *"A woman is reading."* dengan skor kesamaan tertinggi, yaitu 0.9345436.

Gambar 4 menunjukkan bahwa model SimCSE digunakan untuk melakukan pencarian semantik berdasarkan kesamaan makna. Pada kode di atas, dua kalimat `sentences` (*"A woman is reading."* dan *"A man is playing a guitar."*) di encode menjadi representasi vektor , lalu dibuatkan indeks pencarian menggunakan pustaka FAISS (Fast Approximate Nearest Neighbors). Kemudian, kalimat *"She is reading."* digunakan sebagai query untuk mencari kalimat yang paling mirip dalam indeks.

Hasil pencarian menunjukkan bahwa kalimat *"A woman is reading."* memiliki kesamaan tertinggi dengan query tersebut, dengan skor `0.9345436`, menandakan hubungan semantik yang kuat.

```
1  sentences = ['A woman is reading.', 'A man is playing a guitar.']
2  model.build_index(sentences)
3  results = model.search("he play guitar.")
4  results

11/26/2024 11:24:34 - INFO - simcse.tool -   Encoding embeddings for sentences...
100%|██████████| 1/1 [00:00<00:00, 14.68it/s]
11/26/2024 11:24:34 - INFO - simcse.tool -   Building index...
11/26/2024 11:24:34 - INFO - simcse.tool -   Use CPU-version faiss
11/26/2024 11:24:34 - INFO - simcse.tool -   Finished
100%|██████████| 1/1 [00:00<00:00, 23.39it/s]

[('A man is playing a guitar.', np.float32(0.8915181))]
```

Fig. 5. Hasil pencarian semantik menggunakan model SimCSE. Query *"he play guitar."* menemukan kalimat *"A man is playing a guitar."* dengan skor kesamaan tertinggi, yaitu 0.8915181.

pada gambar 5 Model SimCSE digunakan untuk melakukan pencarian semantik dengan query *"he play guitar."*. Hasilnya menunjukkan bahwa kalimat *"A man is playing a guitar."* memiliki kesamaan tertinggi dengan query tersebut, dengan skor kesamaan `0.8915181`. Ini menunjukkan kemampuan model untuk memahami hubungan semantik antara query dan data yang diindeks, berguna untuk tugas-tugas seperti pencarian informasi berbasis teks.

Alur kerja program dengan menggunakan model default dengan menggunakan Huggingface

```
1  import torch
2  from scipy.spatial.distance import cosine
3  from transformers import AutoModel, AutoTokenizer
4
5
✓ 0.0s
```

Fig. 6. Proses Import Library.

Berikut adalah penjelasan gambar 6

- `torch`: library PyTorch digunakan untuk operasi tensor dan implementasi model deep learning.
- `scipy.spatial.distance.cosine`: Digunakan untuk menghitung jarak kosinus antara dua vektor, yang sering digunakan untuk mengukur kesamaan semantik antara representasi teks.
- `transformers`: Library dari Hugging Face untuk memuat model *transformers* pre-trained, seperti BERT, RoBERTa, dan lain-lain. Dalam kode ini, dua kelas diimpor:
  - `AutoModel`: Untuk memuat model *transformer* pre-trained secara otomatis berdasarkan nama model.
  - `AutoTokenizer`: Untuk memuat tokenizer yang sesuai dengan model pre-trained, digunakan untuk memproses teks menjadi token yang dimengerti oleh model.

Fig. 7. Tokenizer model default.

Berikut adalah penjelasan dari gambar 7 `AutoTokenizer.from_pretrained` digunakan untuk memuat tokenizer pre-trained yang sesuai dengan model `"princeton-nlp/sup-simcse-bert-base-uncased"` Berikut adalah detail outputnya,

- **Nama Tokenizer:** `BertTokenizerFast`
  Ini merupakan fast tokenizer yang kompatibel dengan model BERT.
- **Ukuran Kosakata:** 30522
  Tokenizer memiliki kosakata dengan total 30522 token unik.
- **Panjang Maksimum Model:** 512
  Tokenizer mendukung panjang maksimum teks sebanyak 512 token.
- **Padding dan Truncation:**
  - Padding dilakukan di sisi kanan (`padding_side='right'`).
  - Pemotongan (*truncation*) juga dilakukan di sisi kanan (`truncation_side='right'`).
- **Token Khusus:**
  - `[UNK]`: Token untuk kata yang tidak dikenal.
  - `[SEP]`: Token pemisah untuk teks atau segmen.
  - `[PAD]`: Token padding untuk menyesuaikan panjang teks.
  - `[CLS]`: Token awal untuk klasifikasi teks.
  - `[MASK]`: Token untuk pemrosesan tugas *masked language modeling*.
- **Fitur Tambahan:**
  - `clean_up_tokenization_spaces`: Diatur ke `True`, artinya tokenizer akan menghapus spasi ekstra selama proses tokenisasi.

Tokenizer ini memungkinkan pemrosesan teks menjadi token yang sesuai dengan model BERT yang digunakan, mendukung berbagai tugas NLP seperti klasifikasi, pencarian semantik, dan analisis teks.

Inisialiasi dan deskripsi model pada gambar 8 dan 9 `AutoModel.from_pretrained("princeton-nlp/sup-simcse-bert-base-uncased")` digunakan untuk memuat arsitektur pre-trained **BERT** yang dimodifikasi untuk tugas SimCSE (Semantic Text Similarity). Berikut adalah komponen utama dari model:

- **Embeddings**:
  - `word_embeddings`: Matriks embedding dengan ukuran kosakata sebesar 30522 dan dimensi embedding 768.
  - `position_embeddings`: Matriks embedding posisi dengan panjang maksimal 512.
  - `token_type_embeddings`: Matriks embedding tipe token dengan dua kategori (misalnya, segmen teks A dan B).

```
model = AutoModel.from_pretrained("princeton-nlp/sup-simcse-bert-base-uncased")
model
```

0.8s

```
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(30522, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-11): 12 x BertLayer(
        (attention): BertAttention(
          (self): BertSdpaSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
```

Fig. 8. Inisialisasi Model beserta deskripsinya.

```
        (attention): BertAttention(
          (self): BertSdpaSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
  )
)
```

Fig. 9. Lanjutan Deskripsi.

  - `LayerNorm`: Normalisasi lapisan dengan parameter `eps=1e-12`.
  - `dropout`: Dropout dengan probabilitas 0.1 untuk regularisasi.
- **Encoder**:
  - Terdiri dari 12 lapisan `BertLayer`.
  - Setiap `BertLayer` memiliki komponen:
    * `attention`: Modul perhatian (self-attention) dengan mekanisme `query`, `key`, dan `value`, masing-masing direpresentasikan oleh layer linear

berdimensi 768.
    * `intermediate`: Lapisan non-linear dengan fungsi aktivasi `GELU` dan dimensi sementara sebesar 3072.
    * `output`: Lapisan penghubung kembali ke dimensi 768 dengan normalisasi lapisan (`LayerNorm`) dan dropout.
- **Pooler**:
  - Lapisan pooling terakhir yang memproyeksikan representasi CLS token ke dimensi 768 menggunakan fungsi aktivasi `Tanh`.

```
# Tokenize input texts
texts = [
    "There's a kid on a skateboard.",
    "A kid is skateboarding.",
    "A kid is inside the house."
]
tokenized_words = [tokenizer.tokenize(text) for text in texts]

# Print the tokenized words
for i, tokens in enumerate(tokenized_words):
    print(f"Original text: {texts[i]}")
    print(f"Tokenized words: {tokens}")
    print()
```

0.0s

```
Original text: There's a kid on a skateboard.
Tokenized words: ['there', "'", 's', 'a', 'kid', 'on', 'a', 'skate', '##board', '.']

Original text: A kid is skateboarding.
Tokenized words: ['a', 'kid', 'is', 'skate', '##boarding', '.']

Original text: A kid is inside the house.
Tokenized words: ['a', 'kid', 'is', 'inside', 'the', 'house', '.']
```

Fig. 10. Tokenisasi tiap kalimat.

Tokenizer memproses teks menjadi token yang sesuai dengan kosakata model BERT, termasuk penanganan kata kompleks menggunakan subtokens dengan awalan `##`. Seperti pada gambar 10, teks `"There's a kid on a skateboard."` dipecah menjadi token seperti `'there'`, `"'"`, `'s'`, `'a'`, `'kid'`, `'on'`, `'a'`, `'skate'`, `'##board'`, dan `'.'`, dengan kata `'skateboard'` dipisah menjadi `'skate'` dan `'##board'`. Teks `"A kid is skateboarding."` menghasilkan token serupa, di mana `'skateboarding'` dipecah menjadi `'skate'` dan `'##boarding'`. Sementara itu, teks `"A kid is inside the house."` dipecah sepenuhnya ke dalam token individu tanpa penggunaan subtokens. Proses ini memastikan semua kata, baik yang sederhana maupun kompleks, dapat dipetakan ke dalam kosakata model untuk pemrosesan lebih lanjut.

```
inputs = tokenizer(texts, padding=True, truncation=True, return_tensors="pt")
inputs
```

0.0s

```
{'input_ids': tensor([[  101,  2045,  1005,  1055,  1037,  4845,  2006,  1037, 17260,  6277,
          1012,   102],
        [  101,  1037,  4845,  2003, 17260, 21172,  1012,   102,     0,     0,
             0,     0],
        [  101,  1037,  4845,  2003,  2503,  1996,  2160,  1012,   102,     0,
             0,     0]]), 'token_type_ids': tensor([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
        [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0],
        [1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]])}
```

Fig. 11. Pemanggilan fungsi tokenizer pada model default.

Hasil dari pemanggilan fungsi seperti pada gambar 11 yakni `tokenizer(texts, padding=True, truncation=True, return_tensors="pt")` adalah sebuah dictionary yang berisi tiga tensor utama: `input_ids`, `token_type_ids`, dan `attention_mask`.

- **input_ids**: Merupakan ID token yang sesuai dengan kata-kata dalam kalimat input. Setiap kata dipetakan ke ID numerik sesuai dengan kosakata tokenizer. Sebagai contoh:
  - Kalimat 1: "There's a kid on a skateboard."
    * Tokenisasi: `['there', "'", 's', 'a', 'kid', 'on', 'a', 'skate', '##board', '.']`
    * ID Token: `[101, 2045, 1005, 1055, 1037, 4845, 2006, 1037, 17260, 6277, 1012, 102]`
  - Kalimat 2: "A kid is skateboarding."
    * Tokenisasi: `['a', 'kid', 'is', 'skate', '##boarding', '.']`
    * ID Token: `[101, 1037, 4845, 2003, 17260, 21172, 1012, 102, 0, 0, 0, 0]` (dengan token padding `0` di akhir)
  - Kalimat 3: "A kid is inside the house."
    * Tokenisasi: `['a', 'kid', 'is', 'inside', 'the', 'house', '.']`
    * ID Token: `[101, 1037, 4845, 2003, 2503, 1996, 2160, 1012, 102, 0, 0, 0]` (dengan padding token)
- **token_type_ids**: Menunjukkan token mana yang termasuk dalam kalimat pertama atau kalimat kedua, jika bekerja dengan pasangan kalimat. Dalam kasus ini, semua token diberi label `0`, yang berarti input yang diberikan adalah kalimat tunggal (bukan pasangan kalimat).
  `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
  `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
  `[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]`
- **attention_mask**: Menunjukkan token mana yang perlu diperhatikan oleh model. Token yang sesuai dengan kata-kata nyata diberi label `1`, sementara token padding diberi label `0`. Sebagai contoh:
  - Kalimat 1: Tidak ada padding, sehingga semua token diberi `1`:

    `[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]`
  - Kalimat 2: Ada token padding setelah kata terakhir, sehingga posisi tersebut diberi `0`:

    `[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]`
  - Kalimat 3: Mirip dengan Kalimat 2, dengan padding setelah kata terakhir:

    `[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]`

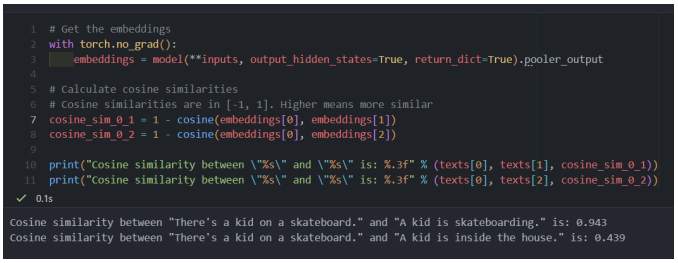

Fig. 12. Cosine similarity dengan menggunakan model default.

Tensor ini kemudian digunakan untuk memberi input pada model BERT untuk menghasilkan representasi yang lebih lanjut dari teks seperti pada gambar 12.

- **Menghitung Embedding**: Embedding dihitung menggunakan model `BERT` dengan memanfaatkan metode `torch.no_grad()` untuk menonaktifkan gradien selama perhitungan karena tidak ada proses pembaruan model. Model menghasilkan `pooler_output`, yang merupakan representasi embedding dari teks input.
- **Menghitung Kemiripan Kosinus**: Kemiripan kosinus dihitung antara pasangan teks dengan formula:

  $$\text{Kemiripan Kosinus} = 1 - \text{cosine}(\text{embedding}_1, \text{embedding}_2)$$

  Nilai ini berada dalam rentang $[-1, 1]$, di mana nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar.
- **Hasil Perhitungan**:
  - Kemiripan antara teks `"There's a kid on a skateboard."` dan `"A kid is skateboarding."` adalah 0.943, menunjukkan bahwa kedua teks memiliki tingkat kesamaan yang sangat tinggi.
  - Kemiripan antara teks `"There's a kid on a skateboard."` dan `"A kid is inside the house."` adalah 0.439, menunjukkan bahwa kedua teks hanya memiliki sedikit kesamaan.

Alur kerja program dengan menggunakan model lokal dengan parameter default menggunakan Huggingface

Berikut adalah penjelasan gambar 6

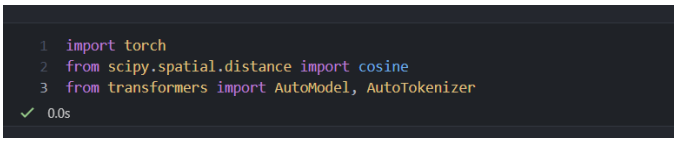

Fig. 13. Proses Import Library.

Berikut adalah penjelasan gambar 13

- `torch`: library PyTorch digunakan untuk operasi tensor dan implementasi model deep learning.
- `scipy.spatial.distance.cosine`: Digunakan untuk menghitung jarak kosinus antara dua vektor, yang sering digunakan untuk mengukur kesamaan semantik antara representasi teks.
- `transformers`: Library dari Hugging Face untuk memuat model *transformers* pre-trained, seperti BERT, RoBERTa, dan lain-lain. Dalam kode ini, dua kelas diimpor:
  - `AutoModel`: Untuk memuat model *transformer* pre-trained secara otomatis berdasarkan nama model.
  - `AutoTokenizer`: Untuk memuat tokenizer yang sesuai dengan model pre-trained, digunakan untuk memproses teks menjadi token yang dimengerti oleh model.

Fig. 14. Tokenizer model lokal.

Berikut adalah penjelasan dari gambar 14 `AutoTokenizer.from_pretrained` digunakan untuk memuat tokenizer pre-trained yang sesuai dengan model `"result/my-sup-simcse-bert-base-uncased"`. Berikut adalah detail outputnya,

- **Nama Tokenizer:** `BertTokenizerFast`
  Ini merupakan fast tokenizer yang kompatibel dengan model BERT.
- **Ukuran Kosakata:** 30522
  Tokenizer memiliki kosakata dengan total 30522 token unik.
- **Panjang Maksimum Model:** 512
  Tokenizer mendukung panjang maksimum teks sebanyak 512 token.
- **Padding dan Truncation:**
  - Padding dilakukan di sisi kanan (`padding_side='right'`).
  - Pemotongan (*truncation*) juga dilakukan di sisi kanan (`truncation_side='right'`).
- **Token Khusus:**
  - `[UNK]`: Token untuk kata yang tidak dikenal.
  - `[SEP]`: Token pemisah untuk teks atau segmen.
  - `[PAD]`: Token padding untuk menyesuaikan panjang teks.
  - `[CLS]`: Token awal untuk klasifikasi teks.
  - `[MASK]`: Token untuk pemrosesan tugas *masked language modeling*.
- **Fitur Tambahan:**
  - `clean_up_tokenization_spaces`: Diatur ke `True`, artinya tokenizer akan menghapus spasi ekstra selama proses tokenisasi.

Tokenizer ini memungkinkan pemrosesan teks menjadi token yang sesuai dengan model BERT yang digunakan, mendukung berbagai tugas NLP seperti klasifikasi, pencarian semantik, dan analisis teks.
Inisialiasi dan deskripsi model pada gambar 15 dan 16 `AutoModel.from_pretrained("result/my-sup-simcse-bert-base-uncased")` digunakan untuk memuat arsitektur pre-trained **BERT** yang dimodifikasi untuk

Fig. 15. Inisialisasi model lokal beserta deksripsinya.

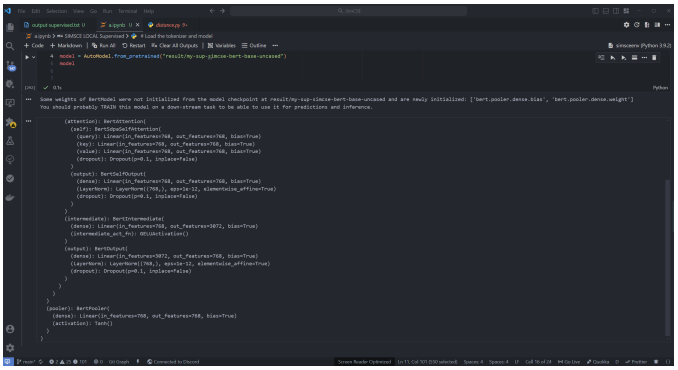

Fig. 16. lanjutann deskripsi model lokal.

tugas SimCSE (Semantic Text Similarity). Berikut adalah komponen utama dari model:

- **Embeddings**:
  - `word_embeddings`: Matriks embedding dengan ukuran kosakata sebesar 30522 dan dimensi embedding 768.
  - `position_embeddings`: Matriks embedding posisi dengan panjang maksimal 512.
  - `token_type_embeddings`: Matriks embedding tipe token dengan dua kategori (misalnya, segmen teks A dan B).
  - `LayerNorm`: Normalisasi lapisan dengan parameter `eps=1e-12`.
  - `dropout`: Dropout dengan probabilitas 0.1 untuk regularisasi.
- **Encoder**:
  - Terdiri dari 12 lapisan `BertLayer`.
  - Setiap `BertLayer` memiliki komponen:
    * `attention`: Modul perhatian (self-attention) dengan mekanisme `query`, `key`, dan `value`, masing-masing direpresentasikan oleh layer linear berdimensi 768.
    * `intermediate`: Lapisan non-linear dengan fungsi aktivasi `GELU` dan dimensi sementara sebesar 3072.
    * `output`: Lapisan penghubung kembali ke dimensi 768 dengan normalisasi lapisan (`LayerNorm`) dan dropout.
- **Pooler**:
  - Lapisan pooling terakhir yang memproyeksikan representasi CLS token ke dimensi 768 menggunakan fungsi aktivasi `Tanh`.

Fig. 17. Tokenisasi tiap kalimat.

Tokenizer memproses teks menjadi token yang sesuai dengan kosakata model BERT, termasuk penanganan kata kompleks menggunakan subtokens dengan awalan `##`. Seperti pada gambar 17, teks `"There's a kid on a skateboard."` dipecah menjadi token seperti `'there'`, `"'"`, `'s'`, `'a'`, `'kid'`, `'on'`, `'a'`, `'skate'`, `'##board'`, dan `'.'`, dengan kata `'skateboard'` dipisah menjadi `'skate'` dan `'##board'`. Teks `"A kid is skateboarding."` menghasilkan token serupa, di mana `'skateboarding'` dipecah menjadi `'skate'` dan `'##boarding'`. Sementara itu, teks `"A kid is inside the house."` dipecah sepenuhnya ke dalam token individu tanpa penggunaan subtokens. Proses ini memastikan semua kata, baik yang sederhana maupun kompleks, dapat dipetakan ke dalam kosakata model untuk pemrosesan lebih lanjut.

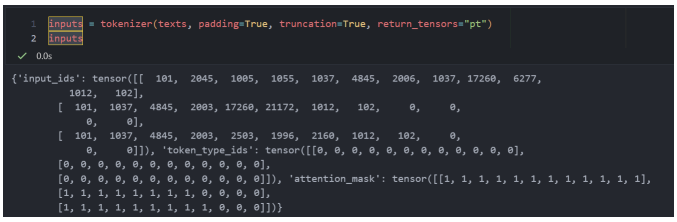

Fig. 18. Pemanggilan fungsi tokenizer pada model lokal.

Hasil dari pemanggilan fungsi seperti pada gambar 18 yakni `tokenizer(texts, padding=True, truncation=True, return_tensors="pt")` adalah sebuah dictionary yang berisi tiga tensor utama: `input_ids`, `token_type_ids`, dan `attention_mask`.

- **input_ids**: Merupakan ID token yang sesuai dengan kata-kata dalam kalimat input. Setiap kata dipetakan ke

ID numerik sesuai dengan kosakata tokenizer. Sebagai contoh:

- Kalimat 1: ”There’s a kid on a skateboard.”
  - Tokenisasi: `['there', "'", 's', 'a', 'kid', 'on', 'a', 'skate', '##board', '.']`
  - ID Token: `[101, 2045, 1005, 1055, 1037, 4845, 2006, 1037, 17260, 6277, 1012, 102]`
- Kalimat 2: ”A kid is skateboarding.”
  - Tokenisasi: `['a', 'kid', 'is', 'skate', '##boarding', '.']`
  - ID Token: `[101, 1037, 4845, 2003, 17260, 21172, 1012, 102, 0, 0, 0, 0]` (dengan token padding `0` di akhir)
- Kalimat 3: ”A kid is inside the house.”
  - Tokenisasi: `['a', 'kid', 'is', 'inside', 'the', 'house', '.']`
  - ID Token: `[101, 1037, 4845, 2003, 2503, 1996, 2160, 1012, 102, 0, 0, 0]` (dengan padding token)

- **token_type_ids**: Menunjukkan token mana yang termasuk dalam kalimat pertama atau kalimat kedua, jika bekerja dengan pasangan kalimat. Dalam kasus ini, semua token diberi label `0`, yang berarti input yang diberikan adalah kalimat tunggal (bukan pasangan kalimat).

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- **attention_mask**: Menunjukkan token mana yang perlu diperhatikan oleh model. Token yang sesuai dengan kata-kata nyata diberi label `1`, sementara token padding diberi label `0`. Sebagai contoh:
  - Kalimat 1: Tidak ada padding, sehingga semua token diberi `1`:

    `[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]`
  - Kalimat 2: Ada token padding setelah kata terakhir, sehingga posisi tersebut diberi `0`:

    `[1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0]`
  - Kalimat 3: Mirip dengan Kalimat 2, dengan padding setelah kata terakhir:

    `[1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0]`

```python
# Get the embeddings
with torch.no_grad():
    embeddings = model(**inputs, output_hidden_states=True, return_dict=True).pooler_output

# Calculate cosine similarities
# Cosine similarities are in [-1, 1]. Higher means more similar
cosine_sim_0_1 = 1 - cosine(embeddings[0], embeddings[1])
cosine_sim_0_2 = 1 - cosine(embeddings[0], embeddings[2])

print("Cosine similarity between \"%s\" and \"%s\" is: %.3f" % (texts[0], texts[1], cosine_sim_0_1))
print("Cosine similarity between \"%s\" and \"%s\" is: %.3f" % (texts[0], texts[2], cosine_sim_0_2))
```

```
✓ 0.1s
Cosine similarity between "There's a kid on a skateboard." and "A kid is skateboarding." is: 0.951
Cosine similarity between "There's a kid on a skateboard." and "A kid is inside the house." is: 0.660
```

Fig. 19. Cosine similarity dengan menggunakan model lokal.

Tensor ini kemudian digunakan untuk memberi input pada model BERT untuk menghasilkan representasi yang lebih lanjut dari teks seperti pada gambar 19.

- **Menghitung Embedding**: Embedding dihitung menggunakan model `BERT` dengan memanfaatkan metode `torch.no_grad()` untuk menonaktifkan gradien selama perhitungan karena tidak ada proses pembaruan model. Model menghasilkan `pooler_output`, yang merupakan representasi embedding dari teks input.
- **Menghitung Kemiripan Kosinus**: Kemiripan kosinus dihitung antara pasangan teks dengan formula:

  $$\text{Kemiripan Kosinus} = 1 - \text{cosine}(\text{embedding}_1, \text{embedding}_2)$$

  Nilai ini berada dalam rentang $[-1, 1]$, di mana nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar.
- **Hasil Perhitungan**:
  - Kemiripan antara teks `"There’s a kid on a skateboard."` dan `"A kid is skateboarding."` adalah 0.951, menunjukkan bahwa kedua teks memiliki tingkat kesamaan yang sangat tinggi.
  - Kemiripan antara teks `"There’s a kid on a skateboard."` dan `"A kid is inside the house."` adalah 0.660, menunjukkan bahwa kedua teks hanya memiliki lebih sedikit kesamaan.

B. Proses Pelatihan

Penjelasan Parameter Pelatihan

Fig. 20. Inisialasi parameter untuk proses pelatihan.

Berikut adalah penjelasan gambar 20,

```
(myenv) C:\Users\moham\OneDrive\Desktop\NLP\
    SimCSE>python train.py --
    model_name_or_path bert-base-uncased --
    train_file data/nli_for_simcse.csv --
    output_dir result/my-sup-simcse-bert-base-
    uncased --num_train_epochs 3 --
    per_device_train_batch_size 128 --
    learning_rate 5e-5 --max_seq_length 32 --
    evaluation_strategy steps --
    metric_for_best_model stsb_spearman --
    load_best_model_at_end --eval_steps 125 --
    pooler_type cls --overwrite_output_dir --
    temp 0.05 --do_train --do_eval --fp16
```

- `--model_name_or_path bert-base-uncased`
  Model yang digunakan sebagai backbone, dalam hal ini adalah `bert-base-uncased`, model BERT yang tidak case-sensitive.
- `--train_file data/nli_for_simcse.csv`
  Path ke file dataset yang akan digunakan untuk pelatihan. Dalam contoh ini, dataset berada di `data/nli_for_simcse.csv`.
- `--output_dir result/my-sup-simcse-bert-base-uncased`
  Direktori tempat model hasil pelatihan akan disimpan. Di sini, output akan disimpan di `result/my-sup-simcse-bert-base-uncased`.
- `--num_train_epochs 3`
  Jumlah epoch pelatihan, yaitu sebanyak 3 kali iterasi penuh terhadap dataset.
- `--per_device_train_batch_size 128`
  Ukuran batch untuk pelatihan per perangkat (misalnya per GPU/CPU). Ukuran batch ditetapkan menjadi 128.
- `--learning_rate 5e-5`
  Laju pembelajaran (*learning rate*) yang digunakan oleh optimizer selama pelatihan, dalam hal ini adalah 0.00005.
- `--max_seq_length 32`
  Panjang maksimum tokenisasi untuk setiap urutan teks. Teks yang lebih panjang dari 32 token akan dipotong.
- `--evaluation_strategy steps`
  Strategi evaluasi, dalam hal ini evaluasi dilakukan berdasarkan langkah (*steps*) tertentu.
- `--metric_for_best_model stsb_spearman`
  Metode evaluasi yang digunakan untuk menentukan model terbaik, yaitu *Spearman correlation* pada STS-B dataset.
- `--load_best_model_at_end`
  Instruksi untuk memuat model terbaik di akhir pelatihan berdasarkan metrik evaluasi.
- `--eval_steps 125`
  Interval evaluasi, yaitu setiap 125 langkah pelatihan akan dilakukan evaluasi.
- `--pooler_type cls`
  Tipe pooling yang digunakan untuk representasi akhir embedding, yaitu `cls` (menggunakan token [CLS]).
- `--overwrite_output_dir`
  Mengizinkan penimpaan (*overwrite*) direktori output jika sudah ada data sebelumnya.
- `--temp 0.05`
  Temperatur yang digunakan dalam loss fungsi kontrasif untuk SimCSE.
- `--do_train`
  Menjalankan proses pelatihan (*training*).
- `--do_eval`
  Menjalankan proses evaluasi (*evaluation*).
- `--fp16`
  Menggunakan *mixed precision training* untuk mempercepat pelatihan dan mengurangi penggunaan memori dengan representasi 16-bit (*float16*).

Arsitektur Model

```
[INFO|configuration_utils.py:481] 2024-11-18
    20:04:18,040 >> Model config BertConfig {
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "layer_norm_eps": 1e-12,
  "max_position_embeddings": 512,
  "model_type": "bert",
  "num_attention_heads": 12,
  "num_hidden_layers": 12,
  "pad_token_id": 0,
  "position_embedding_type": "absolute",
  "transformers_version": "4.2.1",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}
```

Berikut adalah penjelasan tiap parameter pada konfigurasi:

- **architectures**: Jenis model yang digunakan adalah `BertForMaskedLM`, yang dirancang untuk tugas Masked Language Modeling.
- **attention_probs_dropout_prob**: Probabilitas dropout pada mekanisme perhatian (*attention*) adalah 0.1.
- **hidden_act**: Fungsi aktivasi yang digunakan pada lapisan tersembunyi adalah `gelu`.
- **hidden_size**: Dimensi representasi tersembunyi (*hidden states*) adalah 768.
- **num_attention_heads**: Model memiliki 12 kepala perhatian (*attention heads*) pada setiap lapisan.
- **num_hidden_layers**: Terdapat 12 lapisan tersembunyi pada arsitektur BERT.
- **intermediate_size**: Ukuran lapisan *intermediate* adalah 3072.
- **max_position_embeddings**: Jumlah maksimum token yang dapat diproses adalah 512.
- **vocab_size**: Ukuran kosakata yang digunakan adalah 30,522 token.

Model ini merupakan implementasi dari arsitektur BERT dengan parameter-parameter standar untuk ukuran *base model*. Konfigurasi ini diatur menggunakan pustaka `transformers` versi 4.2.1.

Bukti bahwa pelatihan telah selesai dapat dilihat pada gambar 21.

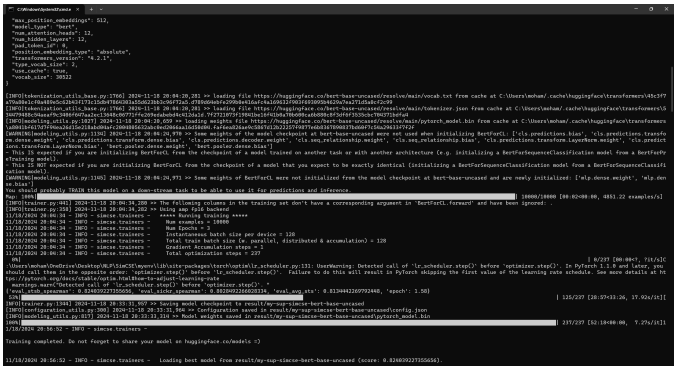

Fig. 21. Bukti bahwa pelatihan telah selesai.

Evaluasi Model Training

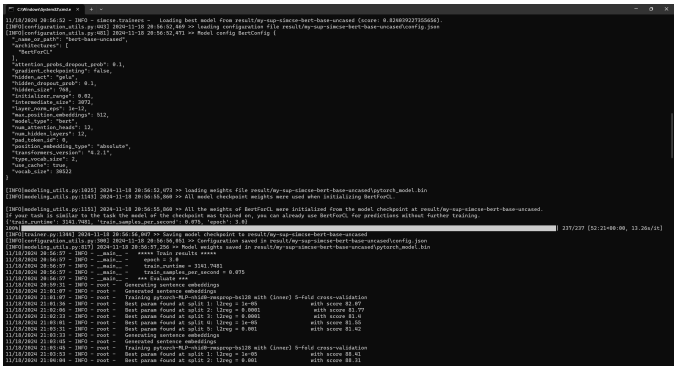

Fig. 22. Proses `--do_eval` pada proses training.

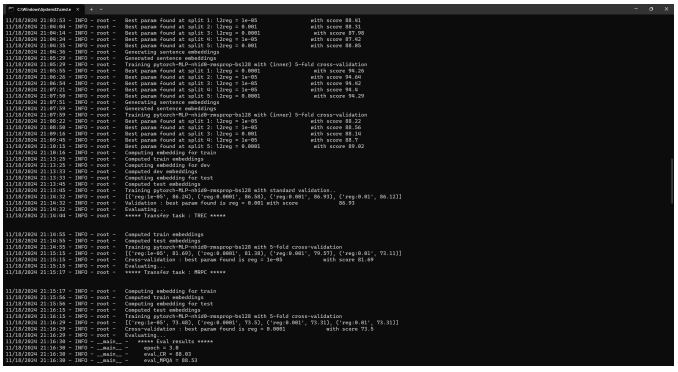

Fig. 23. Lanjutan proses `--do_eval` pada proses training.

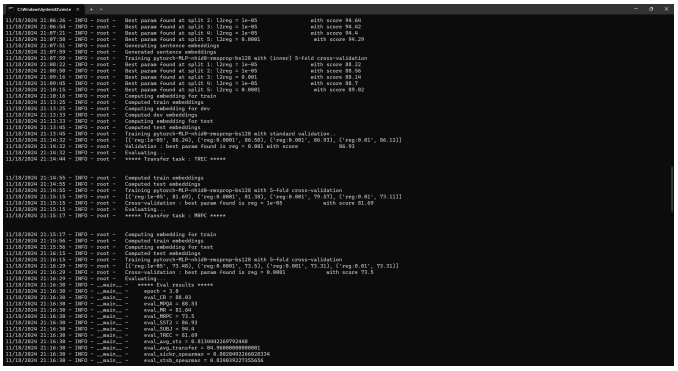

Fig. 24. Hasil dari proses `--do_eval`.

Berikut adalah penjelasan hasil akhir dari proses 22, 23, dan 24,

```
***** Eval results *****
   epoch = 3.0
   eval_CR = 88.03
   eval_MPQA = 88.53
   eval_MR = 81.64
   eval_MRPC = 73.5
   eval_SST2 = 86.93
   eval_SUBJ = 94.4
   eval_TREC = 81.69
   eval_avg_sts = 0.8134442269792448
   eval_avg_transfer = 84.96000000000001
   eval_sickr_spearman = 0.8028492266028334
   eval_stsb_spearman = 0.824039227355656
}
```

Berikut adalah interpretasi dari hasil evaluasi berdasarkan metrik-metrik yang digunakan:

- **Epoch (3.0)**: Model dievaluasi setelah 3 epoch, yang menunjukkan bahwa model telah melalui tiga kali proses pelatihan penuh terhadap seluruh dataset.
- **Eval_CR (Customer Reviews) = 88.03%**: Angka ini menunjukkan akurasi model dalam melakukan klasifikasi sentimen pada dataset ulasan pelanggan. Performa ini cukup tinggi, yang berarti model mampu menangkap konteks sentimen dalam teks pelanggan dengan baik.
- **Eval_MPQA (Multi-Perspective Question Answering) = 88.53%**: Angka ini merepresentasikan akurasi model dalam memahami polaritas opini (positif, negatif, netral) dalam teks dari berbagai perspektif. Performa ini menunjukkan model memiliki kemampuan yang baik untuk memahami teks opini dengan tingkat subjektivitas yang tinggi.
- **Eval_MR (Movie Reviews) = 81.64%**: Akurasi ini mengukur kemampuan model dalam menentukan sentimen (positif/negatif) pada ulasan film. Meskipun performanya lebih rendah dibanding dataset lainnya, angka ini tetap menunjukkan kinerja yang solid untuk tugas ini.
- **Eval_MRPC (Microsoft Research Paraphrase Corpus) = 73.5%**: Metrik ini mengevaluasi akurasi model dalam menentukan apakah dua kalimat merupakan parafrase (memiliki makna yang sama). Skor ini relatif rendah dibandingkan metrik lain, yang mengindikasikan model

sedikit kesulitan menangkap kesamaan semantik antar kalimat.

- **Eval_SST2 (Stanford Sentiment Treebank 2) = 86.93%**: Akurasi ini mengukur performa model dalam klasifikasi sentimen biner (positif/negatif) pada dataset SST2. Skor ini mengindikasikan bahwa model cukup efektif dalam memahami sentimen sederhana dalam teks.
- **Eval_SUBJ (Subjectivity Dataset) = 94.4%**: Skor ini menunjukkan akurasi model dalam membedakan kalimat subjektif (opini, emosi) dan objektif (fakta). Performa yang sangat tinggi ini menunjukkan bahwa model sangat andal dalam membedakan kedua jenis teks.
- **Eval_TREC (Text REtrieval Conference) = 81.69%**: Angka ini merepresentasikan akurasi model dalam klasifikasi pertanyaan ke dalam kategori tertentu. Performa ini menunjukkan model cukup baik dalam memahami konteks dan tipe pertanyaan.
- **Eval_Avg_STS (Semantic Textual Similarity) = 0.8134**: Skor rata-rata ini mencerminkan kemampuan model dalam menentukan kesamaan semantik antara dua teks menggunakan korelasi Spearman. Angka ini menunjukkan bahwa model mampu menangkap makna semantik dengan akurasi yang tinggi.
- **Eval_Avg_Transfer = 84.96%**: Skor rata-rata ini menunjukkan akurasi model dalam tugas transfer learning pada berbagai dataset. Nilai ini mengindikasikan bahwa representasi yang dihasilkan model bersifat general dan dapat diterapkan pada berbagai tugas.
- **Eval_SICKR_Spearman = 0.8028**: Korelasi Spearman ini mengukur kesamaan semantik pada dataset SICK-R (Semantic Relatedness). Skor ini menunjukkan bahwa model mampu memahami hubungan semantik antar kalimat dengan cukup baik.
- **Eval_STSB_Spearman = 0.8240**: Korelasi Spearman ini menunjukkan performa model dalam dataset STS-B (Semantic Textual Similarity Benchmark). Skor ini lebih tinggi dibanding SICK-R, yang berarti model lebih akurat dalam memahami kesamaan semantik pada dataset ini.

Secara keseluruhan, model menunjukkan performa yang baik pada sebagian besar metrik, terutama dalam tugas klasifikasi sentimen, subjektivitas, dan kesamaan semantik. Namun, performa pada tugas parafrase (MRPC) masih relatif lebih rendah dibandingkan tugas lainnya, yang bisa menjadi fokus untuk peningkatan di masa mendatang.

III. Hasil Uji Coba dan Analisis

A. Metriks Evaluasi

Spearman's Rank Correlation Coefficient

Spearman's rank correlation coefficient, yang disimbolkan dengan ρ atau r_s, digunakan untuk mengukur kekuatan dan arah hubungan monoton antara dua variabel. Berbeda dengan Pearson, Spearman tidak memerlukan asumsi hubungan linier antara variabel. Sebaliknya, ia mengukur apakah dua variabel bergerak dalam arah yang sama atau berlawanan (monotonik), terlepas dari apakah hubungan mereka linier atau tidak.

Spearman's ρ dihitung dengan cara mengonversi data mentah menjadi peringkat (ranking) dan kemudian menghitung korelasi antara peringkat tersebut. Rumus untuk Spearman's rank correlation coefficient adalah:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

Dimana:

- d_i adalah selisih antara peringkat masing-masing pasangan nilai.
- n adalah jumlah data.

Korelasi Spearman memberikan nilai antara -1 dan 1, di mana:

- $\rho = 1$ menunjukkan korelasi positif sempurna (dua variabel bergerak dalam arah yang sama secara monotonik).
- $\rho = -1$ menunjukkan korelasi negatif sempurna (dua variabel bergerak dalam arah yang berlawanan secara monotonik).
- $\rho = 0$ menunjukkan tidak ada korelasi monotonik.

Pearson's Correlation Coefficient

Pearson's correlation coefficient, yang disimbolkan dengan r, digunakan untuk mengukur kekuatan dan arah hubungan linier antara dua variabel. Ini adalah ukuran yang paling umum digunakan untuk korelasi, dan mengasumsikan bahwa hubungan antara dua variabel adalah linier serta data mengikuti distribusi normal.

Rumus untuk Pearson's correlation coefficient adalah:

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

Dimana:

- x_i dan y_i adalah nilai individual dari dua variabel yang dihitung.
- $\bar{x}$ dan $\bar{y}$ adalah rata-rata dari variabel x dan y.

Nilai r berkisar antara -1 hingga 1, dengan interpretasi berikut:

- $r = 1$ menunjukkan hubungan linier positif sempurna.
- $r = -1$ menunjukkan hubungan linier negatif sempurna.
- $r = 0$ menunjukkan tidak ada hubungan linier.

Pearson sangat berguna ketika hubungan antara variabel adalah linier, tetapi tidak cocok jika hubungan yang ada bersifat non-linier atau ada outlier yang memengaruhi hasil.

Cosine Similarity

Ketika kita menghitung *cosine similarity* antara dua embedding atau vektor, kita dapat menggunakan rumus berikut untuk mendeskripsikannya:

$$\text{Cosine Similarity} = 1 - \text{Cosine Distance}$$

Dimana, *cosine distance* dihitung menggunakan formula dasar *cosine similarity*:

$$\text{Cosine Similarity} = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\| \|\mathbf{v}\|}$$

Untuk tiga embedding atau vektor $\mathbf{e_0}$, $\mathbf{e_1}$, dan $\mathbf{e_2}$, kita dapat menuliskan *cosine similarity* sebagai berikut:

$$\text{Cosine Similarity}_{0,1} = 1 - \cos(\theta_{0,1}) = 1 - \frac{\mathbf{e_0} \cdot \mathbf{e_1}}{\|\mathbf{e_0}\|\|\mathbf{e_1}\|}$$

$$\text{Cosine Similarity}_{0,2} = 1 - \cos(\theta_{0,2}) = 1 - \frac{\mathbf{e_0} \cdot \mathbf{e_2}}{\|\mathbf{e_0}\|\|\mathbf{e_2}\|}$$

Dimana:

- $\mathbf{e_0}, \mathbf{e_1}, \mathbf{e_2}$ adalah embedding atau vektor representasi.
- $\mathbf{e_0} \cdot \mathbf{e_1}$ adalah *dot product* antara $\mathbf{e_0}$ dan $\mathbf{e_1}$:

$$\mathbf{e_0} \cdot \mathbf{e_1} = \sum_{i=1}^{n} e_{0i} e_{1i}$$

- $\|\mathbf{e_0}\|$ adalah panjang (*norma*) dari $\mathbf{e_0}$:

$$\|\mathbf{e_0}\| = \sqrt{\sum_{i=1}^{n} e_{0i}^2}$$

Hasil dari $\text{Cosine Similarity}_{0,1}$ dan $\text{Cosine Similarity}_{0,2}$ memberikan nilai kesamaan antara embedding $\mathbf{e_0}$ dengan $\mathbf{e_1}$, serta embedding $\mathbf{e_0}$ dengan $\mathbf{e_2}$. Semakin tinggi nilainya (mendekati 1), semakin mirip embedding tersebut.

B. Dataset

Berikut adalah dataset yang digunakan untuk Melatih model,

TABLE I
DATASET PELATIHAN

Nama Dataset	Deskripsi		
	Ukuran	*Bahasa*	*Referensi*
nli_for_simcse	10,000 pasang kalimat (subset)	Inggris	[Ref1]

[Ref1] https://huggingface.co/datasets/princeton-nlp/datasets-for-simcse/resolve/main/nli_for_simcse.csv

TABLE II
PENJELASAN KOLOM PADA DATASET PELATIHAN

Nama	Deskripsi
sent0	Kalimat pertama dalam pasangan yang digunakan untuk membangun hubungan semantik.
sent1	Merujuk ke kalimat kedua dalam pasangan yang digunakan, kalimat ini dibandingkan dengan sent0 untuk mengevaluasi kemiripan semantik.
hard_neg	Merujuk ke hard negative example, yaitu sebuah kalimat yang sengaja dipilih karena terlihat mirip secara sekilas dengan sent0 atau sent1, tetapi tidak memiliki kemiripan semantik sebenarnya

Berikut adalah dataset yang digunakan untuk Melakukan evaluasi pada model,

TABLE VII
SEMANTIC TEXTUAL SIMILARITY-2016

Nama Dataset	*Ukuran*	*Bahasa*	*Referensi*
answer-answer	1572 pasang kalimat	Inggris	
headlines	1498 pasang kalimat	Inggris	
plagiarism	1271 pasang kalimat	Inggris	
postediting	3287 pasang kalimat	Inggris	
question-question	1555 pasang kalimat	Inggris	

TABLE III
SEMANTIC TEXTUAL SIMILARITY-2012

Nama Dataset	Deskripsi		
	Ukuran	*Bahasa*	*Referensi*
MSR-Paraphrase	750 pasang kalimat	Inggris	[Ref1]
MSR-Video	750 pasang kalimat	Inggris	[Ref2]
SMTeuroparl	459 pasang kalimat	Inggris	[Ref3]
SMTnews	399 pasang kalimat	Inggris	[Ref4]
OnWN	750 pasang kalimat	Inggris	[Ref5]
Gabungan (ALL)	3,108 pasang kalimat	Inggris	-

[Ref1] http://research.microsoft.com/en-us/downloads/607d14d9-20cd-47e3-85bc-a2f65cd28042/
[Ref2] http://research.microsoft.com/en-us/downloads/38cf15fd-b8df-477e-a4e4-a4680caa75af/
[Ref3] http://www.statmt.org/wmt08/shared-evaluation-task.html
[Ref4] News conversation sentence pairs from WMT 399 pairs of sentences.
[Ref5] Pairs of sentences where the first comes from Ontonotes and the second from a WordNet definition.

TABLE IV
SEMANTIC TEXTUAL SIMILARITY-2013

Nama Dataset	*Ukuran*	*Bahasa*	*Referensi*
headlines	750 pasang kalimat	Inggris	[Ref1]
OnWN	561 pasang kalimat	Inggris	[Ref2]
FNWN	189 pasang kalimat	Inggris	[Ref3]
SMT	750 pasang kalimat	Inggris	[Ref4]

[Ref1] http://emm.newsexplorer.eu/NewsExplorer/home/en/latest.html
[Ref2] The sentences are sense definitions from WordNet and OntoNotes.
[Ref3] The sentences are sense definitions from WordNet and FrameNet. Note that some FrameNet definitions involve more than one sentence.
[Ref4] This SMT dataset comes from DARPA GALE HTER and HyTER, where one sentence is a MT output and the other is a reference translation where a reference is generated based on human post editing (provided by LDC) or an original human reference (provided by LDC) or a human generated reference based on FSM as described in (Dreyer and Marcu, NAACL 2012). The reference comes from post edited translations.

TABLE VIII
SEMANTIC TEXTUAL SIMILARITY-BENCHMARK

Dataset	*file*	*years*	*Train*	*Dev*	*Test*
news	MSRpar	2012	1000	250	250
news	headlines	2013-16	1999	250	250
news	deft-news	2014	300	0	0
captions	MSRvid	2012	1000	250	250
captions	images	2014-15	1000	250	250
captions	track5.en-en	2017	0	125	125
forum	deft-forum	2014	450	0	0
forum	answers-forums	2015	0	375	0
forum	750 answer-answer	2016	0	0	254

TABLE IX
SENTENCES INVOLVING COMPOSITIONAL KNOWLEDGE

Nama Dataset	*Ukuran*	*Bahasa*	*Referensi*
SICK_test_annotated	4927 pasang kalimat	Inggris	
SICK_train	4500 pasang kalimat	Inggris	
SICK_trial	500 pasang kalimat	Inggris	

TABLE V
SEMANTIC TEXTUAL SIMILARITY-2014

Nama Dataset	*Ukuran*	*Bahasa*	*Referensi*
image	750 pasang kalimat	Inggris	[Ref1]
OnWN	750 pasang kalimat	Inggris	[Ref2]
tweet-news	750 pasang kalimat	Inggris	[Ref3]
deft-news	300 pasang kalimat	Inggris	[Ref4]
deft-forum	450 pasang kalimat	Inggris	[Ref5]
headlines	750 pasang kalimat	Inggris	[Ref6]

[Ref1] The Image Descriptions data set is a subset of the PASCAL VOC-2008 data set (Rashtchian et al., 2010) . PASCAL VOC-2008 data set consists of 1,000 images and has been used by a number of image description systems. The image captions of the data set are released under a CreativeCommons Attribution-ShareAlike license, the descriptions itself are free.
[Ref2] The sentences are sense definitions from WordNet and OntoNotes. 5 pairs of sentences.
[Ref3] The tweet-news data set is a subset of the Linking-Tweets-to-News data set (Guo et al., 2013), which consists of 34,888 tweets and 12,704 news articles. The tweets are the comments on the news articles. The news sentences are the titles of news articles. one sentence.
[Ref4] A subset of news article data in the DEFT project.
[Ref5] A subset of discussion forum data in the DEFT project.
[Ref6] http://emm.newsexplorer.eu/NewsExplorer/home/en/latest.html

TABLE VI
SEMANTIC TEXTUAL SIMILARITY-2015

Nama Dataset	*Ukuran*	*Bahasa*	*Referensi*
answers-forums	2000 pasang kalimat	Inggris	
answers-students	1500 pasang kalimat	Inggris	
belief	2000 pasang kalimat	Inggris	
headlines	1500 pasang kalimat	Inggris	
images	1500 pasang kalimat	Inggris	

C. Skenario Eksperimen

Pada percobaan ini, kami menggunakan spesifikasi hardware dan software sebagai berikut untuk memastikan kelancaran pelatihan dan evaluasi model:

1) Spesifikasi Hardware:

- **CPU:** Intel Core i5-10300H
- **GPU:** NVIDIA GeForce RTX 2060 with Max-Q Design dengan 6GB VRAM
- **RAM:** 16 GB
- **Penyimpanan:** SSD dengan kapasitas 1TB

2) Spesifikasi Software dan library yang terinstall:

- **Sistem Operasi:** Windows 11
- **Bahasa Pemrograman:** Python 3.9.2
- **aiofiles**: 23.2.1
- **aiohappyeyeballs**: 2.4.3

TABLE X
DATASET PELATIHAN UNSUPERVISED LEARNING
(MY-UNSUP-SIMCSE-BERT-BASE-UNCASED)

Nama Dataset	Deskripsi		
	Ukuran	*Bahasa*	*Referensi*
wiki1m_for_simcse	100,000 pasang kalimat	Inggris	[Ref1]

[Ref1] https://huggingface.co/datasets/princeton-nlp/datasets-for-simcse/resolve/main/wiki1m_for_simcse.txt

- **aiohttp**: 3.10.10
- **aiosignal**: 1.3.1
- **annotated-types**: 0.7.0
- **anyio**: 4.6.2.post1
- **async-timeout**: 4.0.3
- **attrs**: 24.2.0
- **certifi**: 2024.8.30
- **charset-normalizer**: 3.4.0
- **click**: 8.1.7
- **colorama**: 0.4.6
- **contourpy**: 1.3.0
- **cycler**: 0.12.1
- **datasets**: 3.1.0
- **dill**: 0.3.8
- **exceptiongroup**: 1.2.2
- **fastapi**: 0.115.4
- **ffmpy**: 0.4.0
- **filelock**: 3.16.1
- **fonttools**: 4.54.1
- **frozenlist**: 1.5.0
- **fsspec**: 2024.9.0
- **gradio**: 4.44.1
- **gradio_client**: 1.3.0
- **h11**: 0.14.0
- **httpcore**: 1.0.6
- **httpx**: 0.27.2
- **huggingface-hub**: 0.26.2
- **idna**: 3.10
- **importlib_resources**: 6.4.5
- **Jinja2**: 3.1.4
- **joblib**: 1.4.2
- **kiwisolver**: 1.4.7
- **markdown-it-py**: 3.0.0
- **MarkupSafe**: 2.1.5
- **matplotlib**: 3.9.2
- **mdurl**: 0.1.2
- **mpmath**: 1.3.0
- **multidict**: 6.1.0
- **multiprocess**: 0.70.16
- **networkx**: 3.2.1
- **numpy**: 1.26.4
- **orjson**: 3.10.11
- **packaging**: 24.1
- **pandas**: 2.2.3
- **pillow**: 10.4.0
- **pip**: 24.2
- **prettytable**: 3.12.0
- **propcache**: 0.2.0
- **pyarrow**: 18.0.0
- **pydantic**: 2.9.2
- **pydantic_core**: 2.23.4
- **pydub**: 0.25.1
- **Pygments**: 2.18.0
- **pyparsing**: 3.2.0
- **python-dateutil**: 2.9.0.post0
- **python-multipart**: 0.0.17

- **pytz**: 2024.2
- **PyYAML**: 6.0.2
- **regex**: 2024.11.6
- **requests**: 2.32.3
- **rich**: 13.9.4
- **ruff**: 0.7.2
- **sacremoses**: 0.1.1
- **safetensors**: 0.4.5
- **scikit-learn**: 1.3.2
- **scipy**: 1.5.4
- **semantic-version**: 2.10.0
- **setuptools**: 75.1.0
- **shellingham**: 1.5.4
- **six**: 1.16.0
- **sniffio**: 1.3.1
- **starlette**: 0.41.2
- **sympy**: 1.13.1
- **threadpoolctl**: 3.5.0
- **tokenizers**: 0.9.4
- **tomlkit**: 0.12.0
- **torch**: 1.7.1+cu110
- **tqdm**: 4.67.0
- **transformers**: 4.2.1
- **typer**: 0.12.5
- **typing_extensions**: 4.12.2
- **tzdata**: 2024.2
- **urllib3**: 2.2.3
- **uvicorn**: 0.32.0
- **wcwidth**: 0.2.13
- **websockets**: 12.0
- **wheel**: 0.44.0
- **xxhash**: 3.5.0
- **yarl**: 1.17.1
- **zipp**: 3.20.2

3) Default Parameter: Parameter yang digunakan pada model `my-sup-simcse-bert-base-uncased`

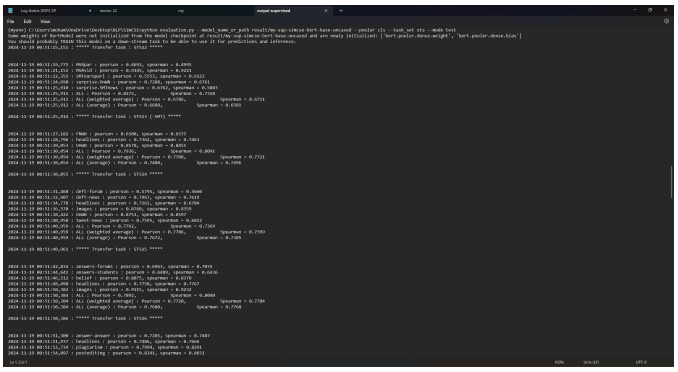

Fig. 25. Bukti evaluasi dengan default parameter.

Berikut adalah penjabaran dari gambar 25,

- `--model_name_or_path bert-base-uncased`
- `--train_file data/nli_for_simcse.csv`
- `--output_dir result/my-sup-simcse-bert-base-uncased`
- `--num_train_epochs 3`
- `--per_device_train_batch_size 128`
- `--learning_rate 5e-5`
- `--max_seq_length 32`
- `--evaluation_strategy steps`
- `--metric_for_best_model stsb_spearman`
- `--load_best_model_at_end`
- `--eval_steps 125`
- `--pooler_type cls`
- `--overwrite_output_dir`
- `--temp 0.05`
- `--do_train`
- `--do_eval`
- `--fp16`

4) Eksperimen 1: Mengubah Default hyperparameter dengan perubahan jumlah epoch training menjadi 1 dengan tujuan agar waktu eksekusi menjadi lebih cepat dan mengurangi resiko model mengalami overfitting

Fig. 26. Bukti evaluasi dengan eksperimen 1.

Berikut adalah penjabaran dari gambar 26,

- `--model_name_or_path bert-base-uncased`
- `--train_file data/nli_for_simcse.csv`
- `--output_dir result/my-sup-simcse-bert-base-uncased`
- `--num_train_epochs 1`
- `--per_device_train_batch_size 128`
- `--learning_rate 5e-5`
- `--max_seq_length 32`
- `--evaluation_strategy steps`
- `--metric_for_best_model stsb_spearman`
- `--load_best_model_at_end`
- `--eval_steps 125`
- `--pooler_type cls`
- `--overwrite_output_dir`
- `--temp 0.05`
- `--do_train`
- `--do_eval`
- `--fp16`

5) Eksperimen 2: Mengubah Default hyperparameter dengan perubahan jumlah epoch menjadi 1 dan train batch size menjadi 64 (dari yang awalnya 128) dengan tujuan agar waktu

eksekusi menjadi lebih cepat dan parameter akan diperbarui lebih sering dengan harapan bisa meningkatkan kecepatan konvergensi karena model mungkin akan lebih cepat mencapai global minima. Berikut adalah penjabaran dari gambar 27,

Fig. 27. Bukti evaluasi dengan eksperimen 2.

- `--model_name_or_path bert-base-uncased`
- `--train_file data/nli_for_simcse.csv`
- `--output_dir result/my-sup-simcse-bert-base-uncased`
- `--num_train_epochs 1`
- `--per_device_train_batch_size 64`
- `--learning_rate 5e-5`
- `--max_seq_length 32`
- `--evaluation_strategy steps`
- `--metric_for_best_model stsb_spearman`
- `--load_best_model_at_end`
- `--eval_steps 125`
- `--pooler_type cls`
- `--overwrite_output_dir`
- `--temp 0.05`
- `--do_train`
- `--do_eval`
- `--fp16`

6) Eksperimen 3: Mengubah Default hyperparameter dengan perubahan jumlah epoch menjadi 1 dan temperature menjadi 0.1 (dari yang awalnya 0.05) agar waktu eksekusi menjadi lebih cepat dan meningkatkan peforma model agar tidak mengalami overfitting, karena model tidak akan terlalu terpaku pada kelas tertentu dan akan memberikan lebih banyak bobot pada kelas-kelas yang memiliki probabilitas yang rendah sehingga meberikan hasil klasifikasi yang lebih beragam serta meningkatkan kinerja model pada data yang memiliki noise atau data yang lebih sulit untuk diprediksi dengan tegas.

Fig. 28. Bukti evaluasi dengan eksperimen 3.

Berikut adalah penjabaran dari gambar 28,

- `--model_name_or_path bert-base-uncased`
- `--train_file data/nli_for_simcse.csv`
- `--output_dir result/my-sup-simcse-bert-base-uncased`
- `--num_train_epochs 1`
- `--per_device_train_batch_size 128`
- `--learning_rate 5e-5`
- `--max_seq_length 32`
- `--evaluation_strategy steps`
- `--metric_for_best_model stsb_spearman`
- `--load_best_model_at_end`
- `--eval_steps 125`
- `--pooler_type cls`
- `--overwrite_output_dir`
- `--temp 0.1`
- `--do_train`
- `--do_eval`
- `--fp16`

D. Hasil Eksperimen

Spearman's Rank Correlation Coefficient dan Pearson's Correlation Coefficient

TABLE XI
HASIL EVALUASI MENGGUNAKAN METRIKS SPEARMAN'S RANK CORRELATION COEFFICIENT

Eksperimen	STS12	STS13	STS14	STS15	STS16	STSBenchmark	SICKRelatedness	Average
Default	73.28	80.91	73.69	80.40	78.47	79.53	78.15	**77.78**
Eksperimen 1	72.16	80.20	72.24	80.40	76.96	79.04	74.97	76.57
Eksperimen 2	72.98	80.82	72.85	79.76	78.48	79.05	77.70	77.38
Eksperimen 3	66.29	76.60	67.60	76.15	72.15	75.31	71.29	72.20

TABLE XII
HASIL EVALUASI MENGGUNAKAN METRIKS PEARSON'S CORRELATION COEFFICIENT

Eksperimen	STS12	STS13	STS14	STS15	STS16	STSBenchmark	SICKRelatedness	Average
Default	82.72	79.36	77.62	78.92	76.03	78.47	83.91	**79.58**
Eksperimen 1	81.78	79.12	77.44	79.16	74.93	78.61	82.58	79.09
Eksperimen 2	82.17	79.35	77.29	78.56	75.76	78.06	84.03	79.32
Eksperimen 3	77.67	76.34	74.70	75.16	69.59	76.70	80.49	75.81

- Seperti pada gambar 29 dan tabel XII, model dengan parameter default memberikan performa terbaik secara keseluruhan baik dengan metriks spearman atau pearson, dengan rata-rata skor tertinggi baik dengan menggunakan

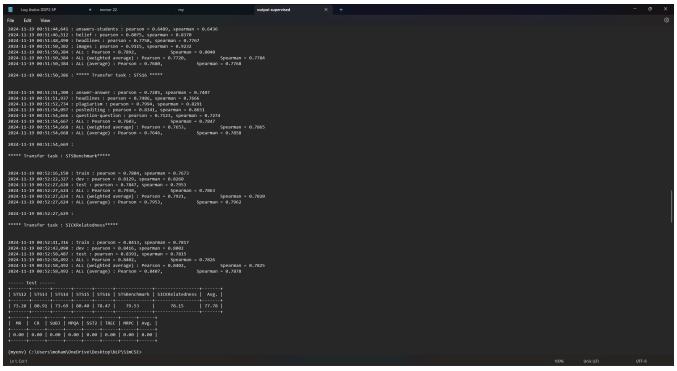

Fig. 29. Hasil evaluasi dengan parameter default.

spearman (rata-rata) sebesar 77.78 dan menggunakan pearson (rata-rata) sebesar 79.53 pada semua benchmark STS tasks.

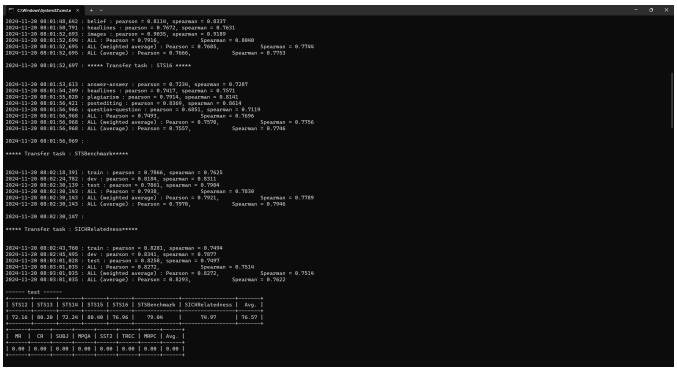

Fig. 30. Hasil evaluasi eksperimen 1.

- Seperti pada gambar 30 dan tabel XII, model pada eksperimen 1 memiliki skor spearman (rata-rata) 76.57 dan skor pearson (rata-rata) 79.09. Dengan pengurangan jumlah epoch menjadi 1, performa model sedikit menurun dibandingkan dengan default. Hal ini menunjukkan bahwa 3 epoch memberikan model lebih banyak kesempatan untuk mempelajari data, menghasilkan generalisasi yang lebih baik.

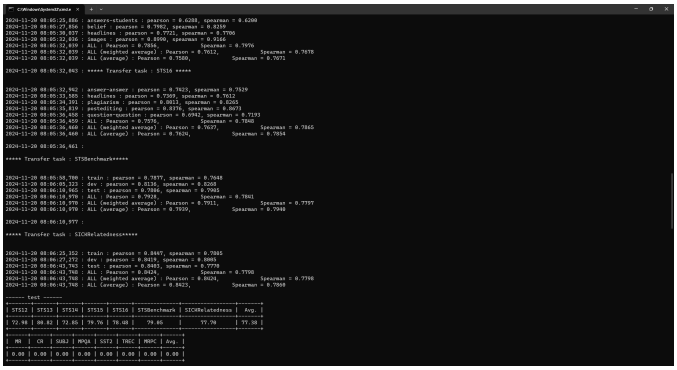

Fig. 31. Hasil evaluasi eksperimen 2.

- Seperti pada gambar 31 dan tabel XII, Model pada eksperimen 2 memiliki skor spearman (rata-rata) 77.38 dan skor pearson (rata-rata) 79.32. Mengurangi batch size menjadi 64 dengan 1 epoch hampir menyamai performa default. Hal ini menunjukkan bahwa batch size yang lebih kecil dapat mempercepat konvergensi ke global minima, meskipun membutuhkan waktu pelatihan yang lebih lama per epoch.

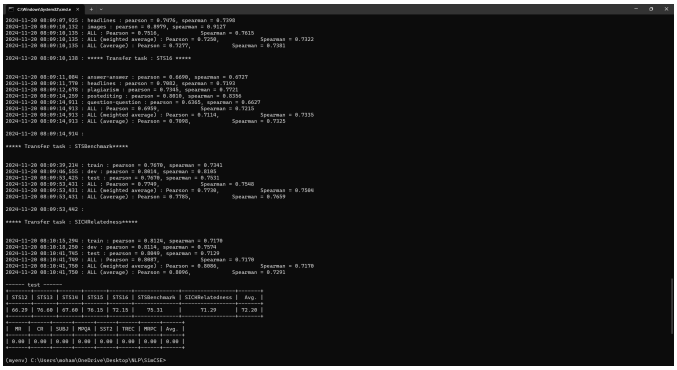

Fig. 32. Hasil evaluasi eksperimen 3.

- Seperti pada gambar 32 dan tabel XII, model pada eksperimen 3 memiliki skor spearman (rata-rata) 72.20 dan skor pearson (rata-rata) 75.81. Meningkatkan temperature menjadi 0.1 dengan 1 epoch secara signifikan menurunkan performa. Temperature yang lebih tinggi tampaknya membuat distribusi prediksi model menjadi terlalu seragam, sehingga kurang fokus pada pasangan kalimat yang relevan.

Cosine Similarity

TABLE XIII
HASIL EKSPERIMEN COSINE SIMILARITY

Model	Kalimat	Skor
princeton-nlp/sup-simcse-bert-base-uncased	[S0] dan [S1]	0.943
princeton-nlp/sup-simcse-bert-base-uncased	[S0] dan [S2]	0.439
my-sup-simcse-bert-base-uncased	[S0] dan [S1]	0.951
my-sup-simcse-bert-base-uncased	[S0] dan [S2]	0.660
exp1	[S0] dan [S1]	0.948
exp1	[S0] dan [S2]	0.682
exp2	[S0] dan [S1]	0.940
exp2	[S0] dan [S2]	0.603
exp3	[S0] dan [S1]	0.957
exp3	[S0] dan [S2]	0.461
my-unsup-simcse-bert-base-uncased	[S0] dan [S1]	0.898
my-unsup-simcse-bert-base-uncased	[S0] dan [S2]	0.717
MPNET	[S0] dan [S1]	0.804
MPNET	[S0] dan [S2]	0.797

S0 "There's a kid on a skateboard."
S1 "A kid is skateboarding."
S2 "A kid is inside the house."

- **princeton-nlp/sup-simcse-bert-base-uncased:** Model ini menunjukkan nilai cosine similarity yang sangat tinggi untuk pasangan kalimat $[S0]$ dan $[S1]$ (0.943), yang mengindikasikan bahwa model ini mampu mengenali

kedekatan semantik antara kalimat yang memiliki makna hampir identik dalam konteks skateboard. Namun, untuk pasangan kalimat $[S0]$ dan $[S2]$, nilai similarity jauh lebih rendah (0.439), menunjukkan bahwa model ini secara efektif mengenali perbedaan konteks antara kalimat yang menggambarkan situasi di luar ruangan dan di dalam rumah.

- **my-sup-simcse-bert-base-uncased:** Model ini menunjukkan hasil yang sangat mirip dengan model `princeton-nlp/sup-simcse-bert-base-uncased`, dengan nilai cosine similarity untuk pasangan kalimat $[S0]$ dan $[S1]$ sebesar 0.951, sedikit lebih tinggi, menunjukkan sensitivitas yang lebih baik terhadap kemiripan semantik. Namun, pada pasangan $[S0]$ dan $[S2]$, skor similarity yang lebih tinggi (0.660) mengindikasikan bahwa model ini sedikit kurang efektif dalam membedakan konteks yang berbeda dibandingkan model sebelumnya.
- **Eksperimen 1 (exp1):** Skor similarity untuk pasangan kalimat $[S0]$ dan $[S1]$ (0.948) hampir sama dengan `my-sup-simcse-bert-base-uncased`, menunjukkan kemampuan yang baik dalam mengenali kemiripan semantik. Namun, nilai cosine similarity untuk pasangan kalimat $[S0]$ dan $[S2]$ (0.682) lebih tinggi dibandingkan model supervised lainnya, yang menunjukkan bahwa model ini cenderung menggeneralisasi kemiripan antar kalimat meskipun konteksnya berbeda.
- **Eksperimen 2 (exp2):** Model ini menunjukkan skor cosine similarity yang sedikit lebih rendah untuk pasangan kalimat $[S0]$ dan $[S1]$ (0.940), namun tetap sangat baik. Untuk pasangan kalimat $[S0]$ dan $[S2]$, nilai similarity lebih rendah (0.603) dibandingkan dengan Eksperimen 1, yang mengindikasikan bahwa model ini lebih mampu membedakan konteks yang berbeda.
- **Eksperimen 3 (exp3):** Model ini menunjukkan performa terbaik untuk pasangan $[S0]$ dan $[S1]$, dengan skor similarity tertinggi (0.957). Namun, pada pasangan $[S0]$ dan $[S2]$, skor similarity turun drastis menjadi 0.461, yang menunjukkan kemampuan yang lebih baik dalam membedakan konteks yang berbeda. Hal ini mungkin disebabkan oleh penggunaan parameter temperatur lebih tinggi meningkatkan sensitivitas terhadap perbedaan semantik.
- **my-unsup-simcse-bert-base-uncased:** Model ini menunjukkan performa yang cukup baik meskipun tanpa pelatihan berbasis label. Untuk pasangan kalimat $[S0]$ dan $[S1]$, skor similarity adalah 0.898, lebih rendah dari model supervised tetapi masih menunjukkan sensitivitas terhadap kemiripan semantik. Namun, untuk pasangan kalimat $[S0]$ dan $[S2]$, skor similarity lebih tinggi (0.717), mengindikasikan bahwa model ini kurang optimal dalam membedakan konteks kalimat yang sangat berbeda, karena tidak dilatih dengan label eksplisit.
- **MPNET:** Model MPNET menunjukkan performa yang lebih rendah dibandingkan model berbasis SimCSE. Untuk pasangan kalimat $[S0]$ dan $[S1]$, skor cosine similarity adalah 0.804, yang cukup rendah dibandingkan dengan model lainnya. Untuk pasangan $[S0]$ dan $[S2]$, skor similarity adalah 0.797, hampir sama dengan pasangan sebelumnya. Performa yang hampir seragam ini menunjukkan bahwa model MPNET kurang sensitif terhadap perbedaan semantik antara kalimat dengan konteks yang sangat berbeda.

E. Diskusi

- Mengapa Default Parameter Memberikan Performa Terbaik?
  Jumlah Epoch: Dengan 3 epoch, model memiliki waktu lebih lama untuk mempelajari data. Ini memungkinkan pembelajaran yang lebih mendalam terhadap representasi semantik. Ukuran Batch size yang besar (128) memberikan stabilitas dalam pembaruan parameter, terutama pada model besar seperti bert-base-uncased. Temperature default (0.05) menghasilkan distribusi probabilitas yang cukup tajam, membantu model untuk menekankan pasangan kalimat yang relevan secara lebih efektif.
- Pengurangan Epoch (Eksperimen 1) membuat model memiliki iterasi pelatihan yang lebih sedikit, yang membatasi pembelajaran fitur secara mendalam. Hal ini mengindikasikan bahwa satu epoch tidak cukup untuk mengoptimalkan model dengan dataset ini.
- Pengurangan Batch Size (Eksperimen 2) menghasilkan pembaruan parameter yang lebih sering, tetapi juga meningkatkan volatilitas dalam pembelajaran. Hasil eksperimen menunjukkan bahwa pengurangan batch size tidak cukup untuk mengimbangi dampak pengurangan jumlah epoch
- Pengaruh Temperature yang Lebih Tinggi (Eksperimen 3) membuat model menjadi terlalu bebas dalam melakukan klasifikasi antar kalimat dengan tingkat kesamaan rendah. Ini menghasilkan distribusi prediksi yang kurang tajam, sehingga performa pada sebagian besar task STS menurun drastis.
- Performansi pada STS12 lebih rendah dibanding dataset lain di semua eksperimen ketikang meggunakan metriks spearman (Default: 73.28, Eksperimen 1: 72.16, Eksperimen 2: 72.98, Eksperimen 3: 66.29). Kemungkinan disebabkan dataset STS 12 mengandung lebih banyak pasangan teks yang sulit dibandingkan dataset lain, seperti pasangan yang membutuhkan reasoning atau generalisasi lebih dalam. salah satu solusi yang bisa diterapkan adalah melakukan augmentasi pada data untuk menambah variasi pasangan teks yang lebih kompleks.
- Ada risiko overfitting pada parameter default jika diterapkan pada dataset berbeda. Oleh karena itu, parameter ini mungkin tidak selalu optimal untuk dataset atau task lain. Hal ini terlihat dari jomplangnya skor spearman di dataset yang berbeda seperti STS 13 dengan STS 12.
- Kalimat yang panjang atau kalimat yang memiliki struktur yang kompleks cenderung lebih sulit dalam melakukan proses encoding. Contohnya terdapat pada dataset SICKrelatedness yang biasanya mengandung

struktur multi klausa, Hal ini menyebabkan skor spearman yang lebih rendah pada model yang memiliki konfigurasi hyperparameter yang lebih murah.
- Dataset STS 12 dan STS 14 memiliki variasi yang lebih beragam di kalimat-kalimat yang memiliki kemiripan, Hal ini membuat proses pengklasifikasian lebih sulit pada model yang memiliki konfigurasi hyperparameter yang kurang optimal.
- **Model Supervised:** Model supervised seperti `princeton-nlp/sup-simcse-bert-base-uncased` dan `my-sup-simcse-bert-base-uncased` menunjukkan performa yang sangat baik dalam mendeteksi kemiripan antara kalimat $[S0]$ dan $[S1]$. Skor similarity yang tinggi (lebih dari 0.94) menunjukkan bahwa model ini sangat sensitif terhadap kemiripan semantik antar kalimat yang memiliki konteks serupa. Namun, kedua model ini memiliki perbedaan pada pasangan $[S0]$ dan $[S2]$, di mana `my-sup-simcse-bert-base-uncased` menunjukkan skor similarity lebih tinggi (0.660) dibandingkan dengan `princeton-nlp/sup-simcse-bert-base-uncased` (0.439). Hal ini mengindikasikan bahwa model `my-sup-simcse-bert-base- uncased` cenderung lebih permisif dalam menilai kemiripan antar kalimat dengan konteks yang berbeda.
- **Eksperimen yang Dimodifikasi:** Eksperimen 1 (`exp1`), 2 (`exp2`), dan 3 (`exp3`) memberikan wawasan tambahan terkait pengaruh parameter terhadap performa model. Pada pasangan $[S0]$ dan $[S1]$, Eksperimen 3 menunjukkan skor similarity tertinggi (0.957), yang menandakan peningkatan kemampuan model untuk menangkap kemiripan semantik antar kalimat yang serupa. Di sisi lain, pada pasangan $[S0]$ dan $[S2]$, nilai similarity yang lebih rendah (0.461) menunjukkan bahwa model ini lebih baik dalam membedakan konteks yang berbeda. Hal ini menunjukkan bahwa parameter yang digunakan pada Eksperimen 3, seperti peningkatan temperatur, berhasil meningkatkan sensitivitas terhadap perbedaan semantik.
- Model `my-unsup-simcse-bert-base-uncased` menunjukkan performa yang cukup baik meskipun tidak dilatih dengan label eksplisit. Skor similarity pada pasangan $[S0]$ dan $[S1]$ sebesar 0.898 mengindikasikan bahwa model ini mampu mengenali kemiripan semantik dengan baik. Namun, skor similarity yang relatif tinggi pada pasangan $[S0]$ dan $[S2]$ (0.717) menunjukkan bahwa model ini memiliki keterbatasan dalam membedakan konteks kalimat yang sangat berbeda. Hal ini diakibatkan oleh kurangnya supervisi dalam proses pelatihan.
- **Performa Model MPNET:** Model MPNET menunjukkan hasil yang kurang optimal dibandingkan model berbasis SimCSE. Skor similarity untuk pasangan $[S0]$ dan $[S1]$ (0.804) cukup rendah, menunjukkan bahwa model ini kurang mampu menangkap hubungan semantik antar kalimat yang serupa. Selain itu, skor similarity untuk pasangan $[S0]$ dan $[S2]$ (0.797) hampir sama, mengindikasikan bahwa model ini kurang sensitif terhadap perbedaan konteks. Hal ini menunjukkan bahwa pendekatan generalisasi MPNET tidak seefektif metode yang lebih spesifik seperti SimCSE dalam tugas ini.
- Model berbasis SimCSE, terutama yang menggunakan pendekatan supervised dengan parameter yang disesuaikan, memiliki keunggulan dalam menangkap kemiripan semantik antar kalimat. Penyesuaian parameter seperti pada Eksperimen 3 dapat meningkatkan sensitivitas model terhadap perbedaan semantik, yang penting dalam aplikasi seperti pencarian informasi dan penyaringan relevansi.

IV. Kesimpulan dan Saran

Dalam penelitian sebelumnya "SimCSE: Simple Contrastive Learning of Sentence Embeddings" oleh T. Gao et al. [1], metode supervised SimCSE menunjukkan hasil yang unggul dengan rata-rata korelasi Spearman sebesar 81.57 untuk model berbasis BERT-base. Penelitian ini mereproduksi eksperimen mereka dengan beberapa variasi pengaturan pelatihan untuk mengevaluasi konsistensi dan efektivitas model terhadap berbagai dataset evaluasi.

Hasil evaluasi menunjukkan bahwa eksperimen default menghasilkan rata-rata korelasi Spearman sebesar 77.78, yang lebih rendah dibandingkan hasil yang dilaporkan oleh Gao et al. untuk SimCSE-BERT-base supervised (81.57). Eksperimen modifikasi lainnya, seperti Eksperimen 1 dan Eksperimen 2, menghasilkan rata-rata korelasi Spearman masing-masing sebesar 76.57 dan 77.38, dengan sedikit perbedaan dibandingkan baseline. Namun, Eksperimen 3 mengalami penurunan performa signifikan dengan rata-rata korelasi sebesar 72.20, yang menunjukkan pentingnya konfigurasi parameter pelatihan dalam menjaga performa model.

Perbedaan utama dari penelitian ini adalah dalam pengaruh variasi parameter terhadap generalisasi model di berbagai dataset evaluasi, seperti STSBenchmark dan SICK-Relatedness. Temuan ini memberikan wawasan tambahan tentang sensitivitas model terhadap konfigurasi pelatihan dan memberikan dasar untuk eksperimen lebih lanjut dalam penyempurnaan representasi embedding kalimat.

Untuk penelitian masa depan, direkomendasikan untuk mengeksplorasi pengaruh arsitektur model lainnya, seperti large pre-trained models (misalnya RoBERTa-large), serta integrasi dengan teknik regulasi tambahan guna mengatasi degradasi performa pada konfigurasi tertentu.

Acknowledgment

Laporan ini disusun sebagai bagian dari Proyek Akhir NLP di Universitas Indonesia. Penulis mengucapkan terima kasih kepada tim asisten dosen yang telah memberikan bimbingan selama proyek ini.

References

[1] T. Gao, X. Yao, and D. Chen, "SimCSE: Simple contrastive learning of sentence embeddings," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2021.