takehomeassessment (3)

April 22, 2025

Task 0: Make Relevant Imports

[ ]:
```
!pip install transformers datasets sentence_transformers
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.11/dist-
packages (4.51.3)
Collecting datasets
  Downloading datasets-3.5.0-py3-none-any.whl.metadata (19 kB)
Requirement already satisfied: sentence_transformers in
/usr/local/lib/python3.11/dist-packages (3.4.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-
packages (from transformers) (3.18.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.30.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.30.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.11/dist-
packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.11/dist-
packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.11/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-
packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.21.1)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.11/dist-packages (from transformers) (0.5.3)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.11/dist-
packages (from transformers) (4.67.1)
Requirement already satisfied: pyarrow>=15.0.0 in
/usr/local/lib/python3.11/dist-packages (from datasets) (18.1.0)
Collecting dill<0.3.9,>=0.3.0 (from datasets)
  Downloading dill-0.3.8-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
(from datasets) (2.2.2)
Collecting xxhash (from datasets)
  Downloading
```

```
xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata
(12 kB)
Collecting multiprocess<0.70.17 (from datasets)
  Downloading multiprocess-0.70.16-py311-none-any.whl.metadata (7.2 kB)
Collecting fsspec<=2024.12.0,>=2023.1.0 (from
fsspec[http]<=2024.12.0,>=2023.1.0->datasets)
  Downloading fsspec-2024.12.0-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: aiohttp in /usr/local/lib/python3.11/dist-
packages (from datasets) (3.11.15)
Requirement already satisfied: torch>=1.11.0 in /usr/local/lib/python3.11/dist-
packages (from sentence_transformers) (2.6.0+cu124)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-
packages (from sentence_transformers) (1.6.1)
Requirement already satisfied: scipy in /usr/local/lib/python3.11/dist-packages
(from sentence_transformers) (1.14.1)
Requirement already satisfied: Pillow in /usr/local/lib/python3.11/dist-packages
(from sentence_transformers) (11.1.0)
Requirement already satisfied: aiohappyeyeballs>=2.3.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (2.6.1)
Requirement already satisfied: aiosignal>=1.1.2 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.3.2)
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.11/dist-
packages (from aiohttp->datasets) (25.3.0)
Requirement already satisfied: frozenlist>=1.1.1 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.5.0)
Requirement already satisfied: multidict<7.0,>=4.5 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (6.4.3)
Requirement already satisfied: propcache>=0.2.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (0.3.1)
Requirement already satisfied: yarl<2.0,>=1.17.0 in
/usr/local/lib/python3.11/dist-packages (from aiohttp->datasets) (1.19.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.11/dist-packages (from huggingface-
hub<1.0,>=0.30.0->transformers) (4.13.2)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers) (3.4.1)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-
packages (from requests->transformers) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers) (2.3.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->transformers)
(2025.1.31)
Requirement already satisfied: networkx in /usr/local/lib/python3.11/dist-
packages (from torch>=1.11.0->sentence_transformers) (3.4.2)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.11/dist-packages
(from torch>=1.11.0->sentence_transformers) (3.1.6)
Collecting nvidia-cuda-nvrtc-cu12==12.4.127 (from
```

```
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-runtime-cu12==12.4.127 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cuda-cupti-cu12==12.4.127 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cudnn-cu12==9.1.0.70 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cublas-cu12==12.4.5.8 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cufft-cu12==11.2.1.3 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-curand-cu12==10.3.5.147 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
Collecting nvidia-cusolver-cu12==11.6.1.9 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Collecting nvidia-cusparse-cu12==12.3.1.170 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl.metadata (1.6 kB)
Requirement already satisfied: nvidia-cusparselt-cu12==0.6.2 in
/usr/local/lib/python3.11/dist-packages (from
torch>=1.11.0->sentence_transformers) (0.6.2)
Requirement already satisfied: nvidia-nccl-cu12==2.21.5 in
/usr/local/lib/python3.11/dist-packages (from
torch>=1.11.0->sentence_transformers) (2.21.5)
Requirement already satisfied: nvidia-nvtx-cu12==12.4.127 in
/usr/local/lib/python3.11/dist-packages (from
torch>=1.11.0->sentence_transformers) (12.4.127)
Collecting nvidia-nvjitlink-cu12==12.4.127 (from
torch>=1.11.0->sentence_transformers)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl.metadata (1.5 kB)
```

```
Requirement already satisfied: triton==3.2.0 in /usr/local/lib/python3.11/dist-
packages (from torch>=1.11.0->sentence_transformers) (3.2.0)
Requirement already satisfied: sympy==1.13.1 in /usr/local/lib/python3.11/dist-
packages (from torch>=1.11.0->sentence_transformers) (1.13.1)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from
sympy==1.13.1->torch>=1.11.0->sentence_transformers) (1.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.11/dist-packages (from pandas->datasets) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
packages (from pandas->datasets) (2025.2)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-
packages (from scikit-learn->sentence_transformers) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-
learn->sentence_transformers) (3.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-
packages (from python-dateutil>=2.8.2->pandas->datasets) (1.17.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.11/dist-packages (from
jinja2->torch>=1.11.0->sentence_transformers) (3.0.2)
Downloading datasets-3.5.0-py3-none-any.whl (491 kB)
                         491.2/491.2 kB
10.5 MB/s eta 0:00:00
Downloading dill-0.3.8-py3-none-any.whl (116 kB)
                         116.3/116.3 kB
11.6 MB/s eta 0:00:00
Downloading fsspec-2024.12.0-py3-none-any.whl (183 kB)
                         183.9/183.9 kB
16.8 MB/s eta 0:00:00
Downloading multiprocess-0.70.16-py311-none-any.whl (143 kB)
                         143.5/143.5 kB
13.8 MB/s eta 0:00:00
Downloading nvidia_cublas_cu12-12.4.5.8-py3-none-manylinux2014_x86_64.whl
(363.4 MB)
                         363.4/363.4 MB
3.0 MB/s eta 0:00:00
Downloading nvidia_cuda_cupti_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (13.8 MB)
                         13.8/13.8 MB
125.2 MB/s eta 0:00:00
Downloading nvidia_cuda_nvrtc_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (24.6 MB)
                         24.6/24.6 MB
96.9 MB/s eta 0:00:00
Downloading nvidia_cuda_runtime_cu12-12.4.127-py3-none-
```

```
manylinux2014_x86_64.whl (883 kB)
                         883.7/883.7 kB
59.3 MB/s eta 0:00:00
Downloading nvidia_cudnn_cu12-9.1.0.70-py3-none-manylinux2014_x86_64.whl
(664.8 MB)
                         664.8/664.8 MB
1.7 MB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl
(211.5 MB)
                         211.5/211.5 MB
12.0 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-
manylinux2014_x86_64.whl (56.3 MB)
                         56.3/56.3 MB
44.6 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-
manylinux2014_x86_64.whl (127.9 MB)
                         127.9/127.9 MB
20.0 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-
manylinux2014_x86_64.whl (207.5 MB)
                         207.5/207.5 MB
3.8 MB/s eta 0:00:00
Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-
manylinux2014_x86_64.whl (21.1 MB)
                         21.1/21.1 MB
104.9 MB/s eta 0:00:00
Downloading
xxhash-3.5.0-cp311-cp311-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (194 kB)
                         194.8/194.8 kB
19.3 MB/s eta 0:00:00
Installing collected packages: xxhash, nvidia-nvjitlink-cu12, nvidia-
curand-cu12, nvidia-cufft-cu12, nvidia-cuda-runtime-cu12, nvidia-cuda-nvrtc-
cu12, nvidia-cuda-cupti-cu12, nvidia-cublas-cu12, fsspec, dill, nvidia-cusparse-
cu12, nvidia-cudnn-cu12, multiprocess, nvidia-cusolver-cu12, datasets
  Attempting uninstall: nvidia-nvjitlink-cu12
    Found existing installation: nvidia-nvjitlink-cu12 12.5.82
    Uninstalling nvidia-nvjitlink-cu12-12.5.82:
      Successfully uninstalled nvidia-nvjitlink-cu12-12.5.82
  Attempting uninstall: nvidia-curand-cu12
    Found existing installation: nvidia-curand-cu12 10.3.6.82
    Uninstalling nvidia-curand-cu12-10.3.6.82:
      Successfully uninstalled nvidia-curand-cu12-10.3.6.82
  Attempting uninstall: nvidia-cufft-cu12
    Found existing installation: nvidia-cufft-cu12 11.2.3.61
    Uninstalling nvidia-cufft-cu12-11.2.3.61:
      Successfully uninstalled nvidia-cufft-cu12-11.2.3.61
  Attempting uninstall: nvidia-cuda-runtime-cu12
```

```
    Found existing installation: nvidia-cuda-runtime-cu12 12.5.82
    Uninstalling nvidia-cuda-runtime-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-runtime-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-nvrtc-cu12
    Found existing installation: nvidia-cuda-nvrtc-cu12 12.5.82
    Uninstalling nvidia-cuda-nvrtc-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-nvrtc-cu12-12.5.82
  Attempting uninstall: nvidia-cuda-cupti-cu12
    Found existing installation: nvidia-cuda-cupti-cu12 12.5.82
    Uninstalling nvidia-cuda-cupti-cu12-12.5.82:
      Successfully uninstalled nvidia-cuda-cupti-cu12-12.5.82
  Attempting uninstall: nvidia-cublas-cu12
    Found existing installation: nvidia-cublas-cu12 12.5.3.2
    Uninstalling nvidia-cublas-cu12-12.5.3.2:
      Successfully uninstalled nvidia-cublas-cu12-12.5.3.2
  Attempting uninstall: fsspec
    Found existing installation: fsspec 2025.3.2
    Uninstalling fsspec-2025.3.2:
      Successfully uninstalled fsspec-2025.3.2
  Attempting uninstall: nvidia-cusparse-cu12
    Found existing installation: nvidia-cusparse-cu12 12.5.1.3
    Uninstalling nvidia-cusparse-cu12-12.5.1.3:
      Successfully uninstalled nvidia-cusparse-cu12-12.5.1.3
  Attempting uninstall: nvidia-cudnn-cu12
    Found existing installation: nvidia-cudnn-cu12 9.3.0.75
    Uninstalling nvidia-cudnn-cu12-9.3.0.75:
      Successfully uninstalled nvidia-cudnn-cu12-9.3.0.75
  Attempting uninstall: nvidia-cusolver-cu12
    Found existing installation: nvidia-cusolver-cu12 11.6.3.83
    Uninstalling nvidia-cusolver-cu12-11.6.3.83:
      Successfully uninstalled nvidia-cusolver-cu12-11.6.3.83
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
gcsfs 2025.3.2 requires fsspec==2025.3.2, but you have fsspec 2024.12.0 which is
incompatible.
Successfully installed datasets-3.5.0 dill-0.3.8 fsspec-2024.12.0
multiprocess-0.70.16 nvidia-cublas-cu12-12.4.5.8 nvidia-cuda-cupti-cu12-12.4.127
nvidia-cuda-nvrtc-cu12-12.4.127 nvidia-cuda-runtime-cu12-12.4.127 nvidia-cudnn-
cu12-9.1.0.70 nvidia-cufft-cu12-11.2.1.3 nvidia-curand-cu12-10.3.5.147 nvidia-
cusolver-cu12-11.6.1.9 nvidia-cusparse-cu12-12.3.1.170 nvidia-nvjitlink-
cu12-12.4.127 xxhash-3.5.0
```

```
[ ]: import kagglehub

# Download latest version
path = kagglehub.dataset_download("kashishparmar02/
 ↪social-media-sentiments-analysis-dataset")

print("Path to dataset files:", path)
```

```
Path to dataset files: /kaggle/input/social-media-sentiments-analysis-dataset
```

```
[ ]: import transformers
import sentence_transformers
import torch
import torch.nn as nn
import numpy as np
import datasets
```

```
[ ]: device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

```
[ ]: device(type='cuda')
```

Task 1: Sentence Transformer Implementation

Step 1: Load a standard SentenceTransformer model and generate embeddings

```
[ ]: generic_model = sentence_transformers.SentenceTransformer("all-MiniLM-L6-v2")
generic_model.to(device)
```

```
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94:
UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab
(https://huggingface.co/settings/tokens), set it as secret in your Google Colab
and restart your session.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access
public models or datasets.
  warnings.warn(

modules.json:   0%|          | 0.00/349 [00:00<?, ?B/s]

config_sentence_transformers.json:   0%|          | 0.00/116 [00:00<?, ?B/s]

README.md:   0%|          | 0.00/10.5k [00:00<?, ?B/s]

sentence_bert_config.json:   0%|          | 0.00/53.0 [00:00<?, ?B/s]

config.json:   0%|          | 0.00/612 [00:00<?, ?B/s]
```

```
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed.
Falling back to regular HTTP download. For better performance, install the
package with: `pip install huggingface_hub[hf_xet]` or `pip install hf_xet`
WARNING:huggingface_hub.file_download:Xet Storage is enabled for this repo, but
the 'hf_xet' package is not installed. Falling back to regular HTTP download.
For better performance, install the package with: `pip install
huggingface_hub[hf_xet]` or `pip install hf_xet`
```

```
model.safetensors:   0%|          | 0.00/90.9M [00:00<?, ?B/s]
```

```
tokenizer_config.json:   0%|          | 0.00/350 [00:00<?, ?B/s]
```

```
vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]
```

```
tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
```

```
special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]
```

```
config.json:   0%|          | 0.00/190 [00:00<?, ?B/s]
```

[ ]:
```
SentenceTransformer(
  (0): Transformer({'max_seq_length': 256, 'do_lower_case': False}) with
Transformer model: BertModel
  (1): Pooling({'word_embedding_dimension': 384, 'pooling_mode_cls_token':
False, 'pooling_mode_mean_tokens': True, 'pooling_mode_max_tokens': False,
'pooling_mode_mean_sqrt_len_tokens': False, 'pooling_mode_weightedmean_tokens':
False, 'pooling_mode_lasttoken': False, 'include_prompt': True})
  (2): Normalize()
)
```

[ ]:
```
# Some Input Sentences to Embed
input_sentences = ["I hate this app.",
                   "I love how it works omg",
                   "it's alright, but i wish it was blue",
                   "i never will download this app.",
                   "i might have something else.",
                   "This is the best thing I've ever worked with."]
# Labels representing sentiments attached to input sentences
input_labels = ["negative negative", "positive positive", "neutral positive",␣
 ↪"neutral neutral", "neutral negative", "positive positive"]
generic_embeddings = generic_model.encode(input_sentences)
generic_embeddings.shape
```

[ ]:
```
(6, 384)
```

[ ]:
```
generic_similarities = generic_model.similarity(generic_embeddings,␣
 ↪generic_embeddings)
generic_similarities.to(device)
# Direct Representation of Similarities
print(generic_similarities.shape)
```

```
generic_similarities
```

```
torch.Size([6, 6])
```

[ ]:
```
tensor([[ 1.0000,  0.1785,  0.1557,  0.7266,  0.1313,  0.2293],
        [ 0.1785,  1.0000,  0.0757,  0.1144, -0.0487,  0.4011],
        [ 0.1557,  0.0757,  1.0000,  0.0581,  0.2297,  0.1269],
        [ 0.7266,  0.1144,  0.0581,  1.0000,  0.0976,  0.2155],
        [ 0.1313, -0.0487,  0.2297,  0.0976,  1.0000,  0.1383],
        [ 0.2293,  0.4011,  0.1269,  0.2155,  0.1383,  1.0000]])
```

[ ]:
```
def display_similarities(similarities, input_sentences, input_labels,␣
↪sentence_indices, top_n):
    """
    Display the top-N most similar sentences (excluding self) for selected␣
↪input sentences.
    Parameters:
    -----------
    similarities : torch 2D array (n_sentences, n_sentences)
        A square matrix where similarities[i][j] represents the similarity␣
↪score between
        sentence i and sentence j.

    input_sentences : list[str]
        The list of input sentences corresponding to the rows/columns of the␣
↪similarity matrix.

    input_labels : list[str]
        Labels or categories associated with each input sentence (same order as␣
↪input_sentences).

    sentence_indices : list[int]
        Indices of the input sentences for which to display top-N most similar␣
↪sentences.

    top_n : int
        Number of most sentences to display for each.

    Returns: None
    """
    for i in sentence_indices:
        # Get indices sorted by similarity in descending order
        sorted_indices = np.argsort(np.asarray(similarities[i]))[::-1]

        # Exclude the sentence itself
        top_indices = [idx for idx in sorted_indices if idx != i][:top_n]
```

```
        print(f"Top {top_n} similar sentences to \n '{input_sentences[i]}'\n␣
↪label = {input_labels[i]}:")
        for idx in top_indices:
            print(f"  - '{input_sentences[idx]}', label = {input_labels[idx]}␣
↪(Similarity: {similarities[i][idx]:.4f})")

display_similarities(generic_similarities, input_sentences, input_labels,␣
↪[0,1,2], 4)
```

```
Top 4 similar sentences to
 'I hate this app.'
 label = negative negative:
  - 'i never will download this app.', label = neutral neutral (Similarity:
0.7266)
  - 'This is the best thing I've ever worked with.', label = positive positive
(Similarity: 0.2293)
  - 'I love how it works omg', label = positive positive (Similarity: 0.1785)
  - 'it's alright, but i wish it was blue', label = neutral positive
(Similarity: 0.1557)
Top 4 similar sentences to
 'I love how it works omg'
 label = positive positive:
  - 'This is the best thing I've ever worked with.', label = positive positive
(Similarity: 0.4011)
  - 'I hate this app.', label = negative negative (Similarity: 0.1785)
  - 'i never will download this app.', label = neutral neutral (Similarity:
0.1144)
  - 'it's alright, but i wish it was blue', label = neutral positive
(Similarity: 0.0757)
Top 4 similar sentences to
 'it's alright, but i wish it was blue'
 label = neutral positive:
  - 'i might have something else.', label = neutral negative (Similarity:
0.2297)
  - 'I hate this app.', label = negative negative (Similarity: 0.1557)
  - 'This is the best thing I've ever worked with.', label = positive positive
(Similarity: 0.1269)
  - 'I love how it works omg', label = positive positive (Similarity: 0.0757)
```

Step 2: Custom SentenceTransformer Model : pre-trained HuggingFace Model + weighted pooling

[ ]:
```
class SentenceTransformerModel_custom(nn.Module):
    """
        Custom model for generating sentence embeddings using a pretrained␣
↪transformer.

        Architecture:
        -------------
```

```python
    - Encoder: Pretrained transformer model (e.g., BERT) from Hugging Face.
    - Attention-Weighted Pooling: Softmax-weighted sum over token embeddings (excluding [CLS]).
    - Projection: Linear layer to reduce to output_dim.
    - Normalization: L2-normalizes the output embeddings.

    Parameters:
    -----------
    model_name : str
        Pretrained transformer model name or path.
    output_dim : int, optional (default=384)
        Output embedding dimension.

    Inputs:
    -------
    input_ids : torch.Tensor
        Tokenized input IDs (batch_size, seq_len).
    attention_mask : torch.Tensor
        Attention mask (batch_size, seq_len).

    Returns:
    --------
    torch.Tensor
        L2-normalized sentence embeddings (batch_size, output_dim).
"""

def __init__(self, model_name, output_dim=384):
    super(SentenceTransformerModel_custom, self).__init__()
    self.model = transformers.AutoModel.from_pretrained(model_name)
    hidden_size = self.model.config.hidden_size
    self.pooling = nn.Linear(hidden_size, output_dim)
    self.normalize = nn.functional.normalize
    self.weights = nn.Parameter(torch.randn(hidden_size))

def forward(self, input_ids, attention_mask):

    outputs = self.model(input_ids=input_ids, attention_mask=attention_mask)
    token_embeddings = outputs.last_hidden_state[:, 1:, :]
    softmaxed_embeddings = torch.nn.functional.softmax(token_embeddings, dim=1)
    weighted_sum = torch.sum(token_embeddings * softmaxed_embeddings, dim=1)
    pooled_embeddings = self.pooling(weighted_sum)
    normalized_embeddings = nn.functional.normalize(pooled_embeddings, p=2, dim=1)
    return normalized_embeddings
```

```
custom_model = SentenceTransformerModel_custom('nlptown/
↪bert-base-multilingual-uncased-sentiment')
custom_model.to(device)
tokenizer = transformers.AutoTokenizer.from_pretrained('nlptown/
↪bert-base-multilingual-uncased-sentiment')

#tokenizing the sample sentences such that they can be processed by the␣
↪transformer
inputs = tokenizer(input_sentences, padding=True, truncation=True,␣
↪return_tensors="pt")
inputs = inputs.to(device)
input_ids = inputs['input_ids']
attention_mask = inputs['attention_mask']

#encoding
with torch.no_grad():
    custom_embeddings = custom_model(input_ids, attention_mask)
    print(custom_embeddings)
    print(custom_embeddings.shape)
```

```
config.json:   0%|          | 0.00/953 [00:00<?, ?B/s]

model.safetensors:   0%|          | 0.00/669M [00:00<?, ?B/s]

tokenizer_config.json:   0%|          | 0.00/39.0 [00:00<?, ?B/s]

vocab.txt:   0%|          | 0.00/872k [00:00<?, ?B/s]

special_tokens_map.json:   0%|          | 0.00/112 [00:00<?, ?B/s]

tensor([[-0.0796,  0.0016,  0.0671,  …, -0.0840,  0.0042,  0.0487],
        [ 0.0800, -0.0603,  0.0246,  …, -0.0221,  0.0251, -0.0101],
        [ 0.0030, -0.0131, -0.0099,  …, -0.0367,  0.0039, -0.0144],
        [-0.0045, -0.0495,  0.0478,  …, -0.0581,  0.0097,  0.0125],
        [ 0.0193, -0.0705,  0.0302,  …, -0.0687,  0.0588,  0.0222],
        [ 0.0809, -0.0392,  0.0596,  …, -0.0239, -0.0016, -0.0056]],
       device='cuda:0')
torch.Size([6, 384])
```

Step 3 : Comparing Similarity Scores generated by Generic and Custom Models

[ ]:
```
from sklearn.metrics.pairwise import cosine_similarity

print(generic_embeddings.shape)
print(custom_embeddings.shape)

if generic_embeddings.device != 'cpu':
    generic_embeddings = generic_embeddings.cpu().numpy()

if custom_embeddings.device != 'cpu':
```

```
    custom_embeddings = custom_embeddings.cpu().numpy()

# Compute cosine similarity, comparing the "basic" model to the custom model
generic_similarity = cosine_similarity([generic_embeddings[1]],␣
 ↪[generic_embeddings[5]])
custom_similarity = cosine_similarity([custom_embeddings[1]],␣
 ↪[custom_embeddings[5]])

print(f"Generic Model Similarity for first sentence:␣
 ↪{generic_similarity[0][0]}")
print(f"Custom Model Similarity for first sentence: {custom_similarity[0][0]}")
```

```
(6, 384)
torch.Size([6, 384])
Generic Model Similarity for first sentence: 0.40108969807624817
Custom Model Similarity for first sentence: 0.7373437881469727
```

Explanation: While I've kept much of the transformer architecture the same, I decided that I would add a custom pooling method outside the transformer backbone such that more semantic information could be captured. For this task, I used a pre-trained model from HuggingFace, and kept the backbone the same. However, as I was attempting to capture sentiment within the embeddings, and I wanted the model to be sensitive towards these aspects, I implemented a normalized, attention-based (with the softmax), weighted sum that generated fixed-length embeddings.

Task 2: Multi-task Learning Expansion

I have implemented both Task A and Task B in one `MultiTaskLearning_ClassificationandSentiment(nn.Module)` class, so that the model is modular and can be used for both tasks.

[ ]:
```
class MultiTaskLearning_ClassificationandSentiment(nn.Module):
    """
    MTL model for joint classification and sentiment analysis.

    This model uses a shared SentenceTransformer encoder backbone to extract␣
 ↪sentence-level
    embeddings, with two task-specific heads:

    - A classification head for predicting categorical platform types
    - A sentiment analysis head for predicting sentiment categories

    Architecture:
    -------------
    - Shared Encoder: A pre-trained SentenceTransformer model ('nlptown/
 ↪bert-base-multilingual-uncased-sentiment')
                      used to compute fixed-size embeddings.
    - Classification Head: A two-layer feedforward neural network with ReLU␣
 ↪activation.
```

```
    - Sentiment Head: A two-layer feedforward neural network with ReLU␣
↪activation.

    Loss Computation:
    -----------------
    - Computes task-specific cross-entropy loss for both classification and␣
↪sentiment.
    - If only one task is active (i.e., its labels are provided), only that␣
↪loss is used.
    - If both tasks are active, the final loss is the average of the two.

    Parameters:
    -----------
    a_class_num : int
        Number of classes for the classification task.
    b_class_num : int
        Number of classes for the sentiment task.

    Inputs:
    -------
    input_ids : torch.Tensor
        Tokenized input IDs for the transformer encoder.
    attention_mask : torch.Tensor, optional
        Attention mask for the encoder input.
    sentiment_labels : torch.Tensor, optional
        Ground-truth sentiment labels for computing sentiment task loss.
    classification_labels : torch.Tensor, optional
        Ground-truth classification labels for computing classification task␣
↪loss.

    Returns:
    --------
    dict with keys:
        'classification' : torch.Tensor
            Logits for the classification task.
        'sentiment' : torch.Tensor
            Logits for the sentiment task.
        'loss' : torch.Tensor
            Combined or individual task loss, depending on available labels.
    """
    def __init__(self, a_class_num, b_class_num):
        super(MultiTaskLearning_ClassificationandSentiment, self).__init__()
        # shared base encoder for both tasks
        self.shared_encoder = SentenceTransformerModel_custom('nlptown/
↪bert-base-multilingual-uncased-sentiment')
        # output dim of the shared encoder
        hidden_size = self.shared_encoder.pooling.out_features
```

```python
        # head for the classification task
        self.classification_task = nn.Sequential(nn.Linear(hidden_size, hidden_size // 2),
            nn.ReLU(),
            nn.Linear(hidden_size // 2, a_class_num))

        # head for the sentiment task
        self.sentiment_task = nn.Sequential(nn.Linear(hidden_size, hidden_size // 2),
            nn.ReLU(),
            nn.Linear(hidden_size // 2, b_class_num))

    def forward(self, input_ids, attention_mask = None, sentiment_labels = None, classification_labels = None):
        shared_embeddings = self.shared_encoder(input_ids, attention_mask)

        classification_logits = self.classification_task(shared_embeddings)
        sentiment_logits = self.sentiment_task(shared_embeddings)

        loss = 0
        num_losses = 0

        # If either of the task specific heads are frozen (so the task_labels are None)
        # then train normally for only one of them -> adjust loss fn accordingly.

        if classification_labels is not None:
            classification_loss = nn.CrossEntropyLoss()(classification_logits, classification_labels)
            loss = loss + classification_loss
            num_losses += 1

        if sentiment_labels is not None:
            sentiment_loss = nn.CrossEntropyLoss()(sentiment_logits, sentiment_labels)
            loss = loss + sentiment_loss
            num_losses += 1

        # LOSS = 1/2(class_loss + sent_loss), adjusted appropriately if either is None
        # for task-specific training.
        if num_losses > 0:
            loss = loss / num_losses
```

```
        outputs = {
        'classification': classification_logits,
        'sentiment': sentiment_logits,
        'loss': loss}

        return outputs
```

Explanation: In order to modify the the model such that it can handle multi-task learning, I created another class `MultiTaskLearning_ClassificationandSentiment(nn.Module)` of Transformer model that took the original model as the "shared encoder" between the two tasks. Then, I added both a classification-specific head and a sentiment analysis-specific head so that the model could appropriately perform its downstream tasks.

I will train/fine-tune this model in Task 4 below.

Task 3: Training Considerations

The implications, advantages, and rationale for training the model based on the following criteria as it follows:

1. If the entire network is frozen, that means that no training can occur, and the network will default to the pre-trained model. This means that no learning can take place. Some advantages of this can include: fast deployment and preservation of pre-training – not having to train the model frees up resources and would work well if the task at hand is similar to the pre-training task. This configuration, however, is not ideal for multi-task learning, or any "new" tasks.
2. If only the transformer backbone is frozen, then not only can the model take advantage of its pre-training, but one can also add task-specific heads that can fine-tune its understanding for new tasks. This configuration is preferable as it balances its pre-trained knowledge and its ability to learn. It is used in transfer learning, where a pre-trained model is leveraged for new tasks.
3. If only one task-specific head is frozen, then this leads to an unbalanced learning, where one head can learn from the data, while the other does not learn. Hence, the model becomes very task-specific. This is ideal for when there is asymmetrical pre-training, where one task is not represented as well, and thus needs to be focused on in fine-tuning.

Transfer learning can be beneficial where we want to leverage the language understanding of a pre-trained model, and then fine-tune on a specific task. For example, one could use a pre-trained NLP model and fine-tune it to perform sentiment analysis on product reviews.

1. The choice of pre-trained model would be something trained on an immense amount of language data, so either BERT or RoBERTa. That way, we can leverage its robust language capabilites to understand the product reviews, but also fine-tune it such that we can extract the sentiment.
2. In this case, we would need to freeze the transformer backbone and un-freeze the classification head. This is so that we can allow the model adapt to the specific task and "learn" patterns within the task. This way, it is efficient in its resource while also "learning" how to do the new task of extracting sentiment from product reviews.
3. As stated above, freezing the backbone would allow for the preservation of pre-trained knowl-

edge and the ability to learn new tasks. This way, we preserve computational resources but also get task-specific adaptation.

Task 4: Training Loop Implementation

[ ]:
```
import pandas as pd

#importing data to fine-tune the model
df = pd.read_csv("/kaggle/input/social-media-sentiments-analysis-dataset/
↪sentimentdataset.csv")
df.drop(columns=['Unnamed: 0', 'Unnamed: 0.1'], inplace=True)
df["Platform"] = df["Platform"].str.strip()
df["Sentiment"] = df["Sentiment"].str.strip()
df.head()
```

[ ]:
```
                                               Text Sentiment  \
0   Enjoying a beautiful day at the park!         …  Positive
1   Traffic was terrible this morning.            …  Negative
2   Just finished an amazing workout!            …  Positive
3   Excited about the upcoming weekend getaway!  …  Positive
4   Trying out a new recipe for dinner tonight.  …   Neutral

             Timestamp            User   Platform  \
0  2023-01-15 12:30:00   User123          Twitter
1  2023-01-15 08:45:00   CommuterX        Twitter
2  2023-01-15 15:45:00   FitnessFan     Instagram
3  2023-01-15 18:20:00   AdventureX      Facebook
4  2023-01-15 19:55:00   ChefCook       Instagram

                                      Hashtags  Retweets  Likes       Country  \
0   #Nature #Park                                   15.0   30.0     USA
1   #Traffic #Morning                                5.0   10.0     Canada
2   #Fitness #Workout                               20.0   40.0   USA
3   #Travel #Adventure                               8.0   15.0     UK
4   #Cooking #Food                                  12.0   25.0    Australia

   Year  Month  Day  Hour
0  2023      1   15    12
1  2023      1   15     8
2  2023      1   15    15
3  2023      1   15    18
4  2023      1   15    19
```

[ ]:
```
from sklearn.model_selection import train_test_split
from transformers import AutoTokenizer, TrainingArguments, Trainer,␣
↪DataCollatorWithPadding
from datasets import Dataset, ClassLabel
import numpy as np
```

```python
def label_list_to_tensor(label_list : list[str]):
    """
    Converts a list of string labels into a numerical tensor suitable for our␣
↪PyTorch model.
    The function determines the unique label names, assigns an integer index to␣
↪each unique label,
    and converts the entire input list into a tensor of indices.

    Parameters:
    -----------
    label_list : list of str
        A list of string labels (e.g., ['Facebook', 'Instagram', 'Twitter', ...
↪]) to be converted.

    Returns:
    --------
    torch.Tensor
        A tensor of shape (len(label_list),) containing integer valued labels.
    """
    # get unique label names -> sort for consistency across runs (hashing is␣
↪randomized)
    unique_labels = list(sorted(set(label_list))) # Get unique labels
    # enumerate label names -> these are the mappings
    label_to_index = {label: index for index, label in enumerate(unique_labels)}
    indexed_labels = [label_to_index[label] for label in label_list]
    # convert to torch.tensor for model
    label_tensor = torch.tensor(indexed_labels)

    return label_tensor

def preprocess_data(df):
    """
    Preprocesses a DataFrame containing text reviews and their corresponding␣
↪sentiment and platform labels
    for multi-task learning.
    Parameters:
    -----------
    df : pandas.DataFrame containing three required columns:
        - 'Text': review text (strings)
        - 'Sentiment': sentiment labels (strings, e.g., 'positive', 'neutral',␣
↪'negative')
        - 'Platform': classification labels (strings, e.g., 'web', 'ios',␣
↪'android')
    Returns:
    --------
```

```
    dict
        A dictionary containing:
        - tokenized inputs (e.g., 'input_ids', 'attention_mask', etc.)
        - 'sentiment_labels': torch.Tensor of numerical sentiment class indices
        - 'classification_labels': torch.Tensor of numerical platform class␣
 ↪indices
    Notes:
    ------
    - The tokenizer used must be initialized globally before calling this␣
 ↪function.
    """
    reviews = df['Text']
    # convert labels to numerical tensor
    sentiments = label_list_to_tensor(df['Sentiment'])
    classifications = label_list_to_tensor(df["Platform"])
    # encode input sentences/reviews as embeddings
    encoding =tokenizer(
              reviews,
              padding='max_length',
              max_length=128,
              truncation=True,
              return_tensors=None)
    # add labels to embeddings for the model
    encoding['sentiment_labels'] = sentiments
    encoding['classification_labels'] = classifications
    return encoding

# Globally initialize tokenizer
tokenizer = AutoTokenizer.from_pretrained('nlptown/
 ↪bert-base-multilingual-uncased-sentiment')

# Split and Preprocess train/val data
full_dataset = Dataset.from_pandas(df)
train_dataset, val_dataset = full_dataset.train_test_split(test_size=0.2,␣
 ↪shuffle=True, seed=42).values()
train_dataset = train_dataset.map(preprocess_data, batched=True)
val_dataset = val_dataset.map(preprocess_data, batched=True)

# Compute number of sentiment and classification classes for initializing the␣
 ↪model
train_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask',
                                                'sentiment_labels',␣
 ↪'classification_labels'])
val_dataset.set_format(type='torch', columns=['input_ids', 'attention_mask',
                                              'sentiment_labels',␣
 ↪'classification_labels'])
```

```python
num_classification_classes = np.unique(df['Platform']).shape[0]
num_sentiment_classes = np.unique(df['Sentiment']).shape[0]
```

```
Map:   0%|          | 0/585 [00:00<?, ? examples/s]

Map:   0%|          | 0/147 [00:00<?, ? examples/s]
```

[ ]:
```python
# Initialize Model
model =␣
↪MultiTaskLearning_ClassificationandSentiment(num_classification_classes,␣
↪num_sentiment_classes)
model.to(device)
```

[ ]:
```
MultiTaskLearning_ClassificationandSentiment(
  (shared_encoder): SentenceTransformerModel_custom(
    (model): BertModel(
      (embeddings): BertEmbeddings(
        (word_embeddings): Embedding(105879, 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(
              (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()
        )
      )
      (pooling): Linear(in_features=768, out_features=384, bias=True)
    )
    (classification_task): Sequential(
      (0): Linear(in_features=384, out_features=192, bias=True)
      (1): ReLU()
      (2): Linear(in_features=192, out_features=3, bias=True)
    )
    (sentiment_task): Sequential(
      (0): Linear(in_features=384, out_features=192, bias=True)
      (1): ReLU()
      (2): Linear(in_features=192, out_features=191, bias=True)
    )
  )
```

[ ]:
```
from sklearn.metrics import accuracy_score, f1_score

import os
os.environ["WANDB_DISABLED"] = "true"

training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=10,
    per_device_train_batch_size=16,
    learning_rate=5e-4,
    weight_decay=0.01,
    optim='adamw_torch',
    lr_scheduler_type='linear',
    warmup_steps=500,
    eval_strategy='epoch',
    save_strategy='epoch',
    save_steps=1000,
    load_best_model_at_end=True
)

def compute_metrics(eval_pred):
    # Unpack predictions and labels
    logits, labels = eval_pred
```

```
    classification_logits, sentiment_logits = logits
    classification_labels, sentiment_labels = labels

    # For multi-task, logits and labels will be dictionaries
    classification_preds = np.argmax(classification_logits, axis = 1)
    sentiment_preds = np.argmax(sentiment_logits, axis = 1)
    print(classification_preds, sentiment_preds)
    # Calculate metrics for both tasks
    classification_acc = accuracy_score(classification_labels,␣
↪classification_preds)
    classification_f1 = f1_score(classification_labels, classification_preds,␣
↪average='weighted')

    sentiment_acc = accuracy_score(sentiment_labels, sentiment_preds)
    sentiment_f1 = f1_score(sentiment_labels, sentiment_preds,␣
↪average='weighted')

    # Calculate average metrics across tasks
    average_acc = (classification_acc + sentiment_acc) / 2
    average_f1 = (classification_f1 + sentiment_f1) / 2

    return {
        'classification_acc': classification_acc,
        'classification_f1': classification_f1,
        'sentiment_acc': sentiment_acc,
        'sentiment_f1': sentiment_f1,
        'average_acc': average_acc,
        'average_f1': average_f1
    }
```

```
Using the `WANDB_DISABLED` environment variable is deprecated and will be
removed in v5. Use the --report_to flag to control the integrations used for
logging result (for instance --report_to none).
```

[ ]:

```
from torch.utils.data import default_collate

def custom_collate(batch):
    return default_collate(batch)

# Initialize the trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics,
```

```
    data_collator=custom_collate
    )
```

[ ]:
```
trainer.train()
```

```
<IPython.core.display.HTML object>
```

```
[2 2 2 2 2 2 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 2 2 1 1 2 2 1 2 1 1 1 1 1 2 1 1
 2 1 1 2 1 1 1 1 1 2 1 1 2 2 1 1 1 1 1 2 2 1 1 2 1 2 1 2 1 1 1 2 1 1 2 1 1
 2 2 2 1 1 2 1 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 2 1 1 2 2 1 1 1 2 2 2 2 2 2
 1 2 2 2 2 2 2 2 2 1 2 2 1 1 2 1 1 1 1 1 2 2 2 2 1 2 2 1 2 1 1 2 2 1 2 2] [107
107 107 107 107  69 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107  69 107 107  69 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 147 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107]
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 0 2 2 2 2 2 2 2 2 0 2 2 1 0 2 1 0
 2 0 2 2 0 1 0 1 0 2 2 2 2 1 2 2 2 2 2 2 2 1 1 2 2 2 0 2 2 2 1 0 2 0 2 2 1
 2 2 2 0 1 0 2 1 2 1 2 2 2 0 2 0 2 0 0 1 2 2 2 2 2 0 0 2 2 1 2 2 2 2 2 2 2
 0 2 2 2 2 0 2 0 2 0 2 2 1 2 2 2 2 2 2 2 2 2 0 2 2 2 2 0 0 2 2 2 2 2 0 2] [147
107 107 107 107  69  69 107  69  69 107 107  69 107 107 107 107 147
 107 107 107 107 107 107 107 107 147 107  69  45 107 107 107 107 107 107
 107  69 107 107 107 107 107 107 107 107 107 107 107 107 107  45 107  69
 107 107 107 134 107 107 107  69 107 107 147 107  45 107 107 147 107 107
 107 107 107  69  69 107 107 107 147 107 107 107  45  45 107 107  69 107
 107 107 107 107 107 107 107 107 107 107 107 107  69 107 107 107 107 107
 107 107 107 107 107 107 147 107 107 107 107  69 107 147 107 107 107 107
  69 107 107  69 107 107 107 107 134  69 107 107 107 107 107  69 107 107
 107 107 134]
[2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 0 2 2 2 2 2 2 0 0 2 2 0
 2 0 0 2 2 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 1 2 2 0 2 0 1 1 2 2 1
 2 2 0 0 1 0 0 2 2 2 2 2 2 2 2 0 0 2 0 2 2 0 2 2 2 0 1 2 1 1 1 2 2 2 2 0 2
 2 2 2 2 2 0 2 0 2 2 2 0 2 2 2 2 1 2 2 2 2 2 1 2 2 0 2 0 2 2 0 2 2 2 0 2] [ 69
134 134 107 134 134  69 107  69  69 134  45 107 107 107  69  69 147
 134 107  69 134 107 107 107 107  69 134 134  69  69  69 107  45 134 107
  45  69  69  69 134  69  69  69  69  69 134 107  69 134 107 107  69  69
 107  69 107 134 134 107 107 107 107 107  69 134  45 107  45 147  69 107
 107 107 107  69  97  69 107  97  69 134 134 107 107  69 134  69 107  69
 107  69  69  69  45 107  69 107 107  69  69 134  69 107  69 107 134 134
 134 107 134  69 134 107  69 134 107 134  45 107 147  69  69 107 107 134
 134  69  69  69 107 134 134 107 134 134  69 134 107 107 107  69 134 107
 107  45 134]
[0 2 1 0 1 0 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 0 2 2 0 1 0 1 1 1 1 1 0
 0 0 0 2 0 1 1 1 1 0 1 1 0 1 2 1 1 0 1 1 2 1 1 1 0 1 1 0 1 0 1 0 1 1 1 0 1
```

```
 0 1 0 0 1 0 0 1 0 1 1 2 1 1 1 0 1 0 0 1 0 0 1 1 2 0 1 1 1 1 0 1 1 1 0 0 1
 0 1 1 0 2 0 0 0 1 1 0 0 1 1 2 1 0 1 1 0 1 2 0 2 2 0 0 1 1 1 1 1 1 0 0 0] [ 69
134 107 107 134  69  69 107  69  45 107 134  69 107 107  69  45 107
 134 107 107 134 107 107  69 107  69 134  69 107  45 107 107 107 134  69
 107  69 107  69 134 107 107 107 107 107 134 134  69 134 107 134 107 107
  45 107 107 134 107 107 107  69 107 107  69 107  45 107 107 107 107 107
  69 107 107  69  69 107 107  45  45 107  69 107 107  69 134 107 107 107
 107  45  45  69  45 107 134  69 134 107 107 134  69 107  45 107 134 134
 134 107 134  45 134 107  69 134 107 107  45  69  69  45  45 134 107 134
 107 107 107  69  69 107 134 107 134 134  45  69 107 107 107 107 107 107
  45  45 134]
[2 2 2 2 2 2 2 2 2 0 2 2 0 2 2 0 0 1 2 2 0 1 2 0 2 2 2 2 0 0 0 2 2 0 2 0 2
 0 0 0 2 2 2 1 2 0 2 2 2 2 2 2 2 2 0 0 2 2 2 2 2 2 2 2 2 1 0 2 0 1 2 2 2 2
 0 2 0 2 1 0 0 1 0 2 0 0 2 2 2 0 0 2 0 2 0 2 2 2 2 0 2 2 2 2 0 2 2 2 2 2 2
 0 2 1 2 2 0 2 0 0 2 2 0 2 2 2 2 2 2 2 2 2 2 0 2 2 2 2 2 1 2 2 2 2 0 0 2] [134
134 107 107 134 134 134 134 134  45 134 134  45 134 134  45 134 107
 134 134  69 134 134  69 134 134 134 134 134  45  69 134 134  45 134  45
 134 134  69  45 134 134 134 107 134  45 134 134 134 134 134 134 134 134
  45  45 134 134 134 134 107 134 134 134 134 107  45 134  45 134 134 134
 134 134  69 134  45 134 107  45  45 107  69 134  45  45 134 134 134 134
  45 134  69 134  45 134 134 134 134  69 134 134 134 134  69 134 134 134
 134 134 134 134 134 107 134 134  45 134  45  45 134 134  69 134 134 134
 134 134 134 134 134 107 134  45 134 134 134 134 134 107 134 134 134 134
  45  45 134]
[2 0 0 0 1 1 2 0 0 0 0 2 0 0 0 0 0 0 0 0 0 1 1 0 0 0 2 1 0 1 1 0 1 1 0 0 0
 0 1 0 0 0 0 1 1 0 0 2 0 0 1 0 1 0 0 0 0 0 1 1 0 1 0 2 0 0 0 0 0 1 0 1 0 2
 0 0 0 0 1 0 0 1 0 0 1 0 0 1 1 0 0 0 1 1 1 0 1 2 0 0 0 0 1 1 0 0 0 0 0 0 1
 0 1 1 2 1 0 0 0 0 2 0 0 2 0 0 1 0 1 1 1 0 0 0 1 0 0 0 0 1 2 1 1 0 0 0 1] [134
134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 107 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 107 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134]
[1 1 0 1 1 0 1 0 0 0 0 2 0 1 1 0 0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1
 0 1 0 0 1 1 1 1 0 0 1 1 0 1 1 0 0 1 0 1 1 1 1 1 2 0 1 1 1 0 1 0 1 0 1 1 1
 0 1 0 0 1 0 0 1 0 0 1 0 1 1 0 0 1 1 1 1 0 1 1 1 0 0 0 1 1 1 1 0 1 0 0 1 1
 0 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 1 1 1 1 0 1 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0] [134
134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
```

```
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134]
[2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2] [134
134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134]
[2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0 0 0 2 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 2 0 0 0 0 0 0 2 0 0 2 2 0 0 0 0 0 0 2 2 0 0 2 0 0 0 0 0 0 0 0 0 0 2 2
 0 0 0 0 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 2 0 2 0 2 2 0 0 2 0 0 0 0 0 0 2 2 2
 0 0 0 0 2 0 2 0 0 0 0 0 0 0 0 2 0 0 0 0 2 2 0 0 0 0 2 0 0 0 0 0 0 0 0 0] [107
107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107 107
 107 107 107]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1] [134
134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134 134
 134 134 134]
```

[ ]:
```
TrainOutput(global_step=370, training_loss=3.0135303394214525,
metrics={'train_runtime': 69.976, 'train_samples_per_second': 83.6,
'train_steps_per_second': 5.288, 'total_flos': 0.0, 'train_loss':
3.0135303394214525, 'epoch': 10.0})
```

[ ]:
```
from datasets import load_dataset

ds = load_dataset("stanfordnlp/imdb")
ds_train = ds["train"]
ds_test = ds["test"]
ds_train = ds_train.rename_column('label', 'sentiment_labels')
ds_test = ds_test.rename_column('label', 'sentiment_labels')
ds_train = ds_train.add_column('classification_labels', np.zeros(len(ds_train),␣
↪dtype=np.int64))
ds_test = ds_test.add_column('classification_labels', np.zeros(len(ds_test),␣
↪dtype=np.int64))


tokenizer = AutoTokenizer.from_pretrained('nlptown/
↪bert-base-multilingual-uncased-sentiment')
def tokenization(batch):
    return tokenizer(batch['text'], padding=True, truncation=True)

ds_train = ds_train.map(tokenization, batched=True, batch_size=None)
ds_test = ds_test.map(tokenization, batched=True, batch_size=None)
ds_train.set_format('torch', columns=['input_ids', 'attention_mask',␣
↪'sentiment_labels', 'classification_labels'])
ds_test.set_format('torch', columns=['input_ids', 'attention_mask',␣
↪'sentiment_labels', 'classification_labels'])

sent_nums = 2

model = MultiTaskLearning_ClassificationandSentiment(2, sent_nums)
model.to(device)

training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=3,
    per_device_train_batch_size=16,
    learning_rate=5e-4,
    weight_decay=0.01,
    optim='adamw_torch',
    lr_scheduler_type='linear',
    warmup_steps=500,
    eval_strategy='epoch',
    save_strategy='epoch',
    save_steps=1000,
    load_best_model_at_end=True
)

trainer = Trainer(
    model=model,
```

```
    args=training_args,
    train_dataset=ds_train,
    eval_dataset=ds_test,
    compute_metrics=compute_metrics,
    data_collator=custom_collate
    )


trainer.train()
```

```
README.md:   0%|          | 0.00/7.81k [00:00<?, ?B/s]

train-00000-of-00001.parquet:   0%|          | 0.00/21.0M [00:00<?, ?B/s]

test-00000-of-00001.parquet:   0%|          | 0.00/20.5M [00:00<?, ?B/s]

unsupervised-00000-of-00001.parquet:   0%|          | 0.00/42.0M [00:00<?, ?B/s]

Generating train split:   0%|          | 0/25000 [00:00<?, ? examples/s]

Generating test split:   0%|          | 0/25000 [00:00<?, ? examples/s]

Generating unsupervised split:   0%|          | 0/50000 [00:00<?, ? examples/s]

Map:   0%|          | 0/25000 [00:00<?, ? examples/s]

Map:   0%|          | 0/25000 [00:00<?, ? examples/s]

Using the `WANDB_DISABLED` environment variable is deprecated and will be
removed in v5. Use the --report_to flag to control the integrations used for
logging result (for instance --report_to none).

<IPython.core.display.HTML object>

[0 0 0 … 0 0 0] [1 1 1 … 1 1 1]
[0 0 0 … 0 0 0] [1 1 1 … 1 1 1]
[0 0 0 … 0 0 0] [0 0 0 … 0 0 0]
```

[ ]:
```
TrainOutput(global_step=4689, training_loss=0.3535468959177855,
metrics={'train_runtime': 1978.3621, 'train_samples_per_second': 37.91,
'train_steps_per_second': 2.37, 'total_flos': 0.0, 'train_loss':
0.3535468959177855, 'epoch': 3.0})
```

[ ]:
```
!pip freeze > requirements.txt
```

[ ]:
```
from google.colab import files
files.download('requirements.txt')
```