Fine-Tuning BERT project

Project: Test strategies to fine-tuning bert model for classification tasks (domain adaptation).

Author: matheusrdgsf@gmail.com

In [41]:
```
# Datasets
import os
import opendatasets as od

# Data visualisation
import altair as alt
alt.data_transformers.enable('default', max_rows=None)

# Text Mining
from nltk.tokenize import word_tokenize

# ML
import numpy as np
import pandas as pd
import torch
from sklearn.model_selection import train_test_split
from transformers import BertTokenizerFast, BertForSequenceClassification
from transformers import Trainer, TrainingArguments
from sklearn.metrics import accuracy_score, precision_score, f1_score
from sklearn import preprocessing
from sklearn.manifold import TSNE
import plotly.express as px
from sentence_transformers import SentenceTransformer
```

Load data - Kaggle Emotions in Text

In [2]:
```
DATAPATH = 'dataset/emotions-in-text'
```

In [3]:
```
if not os.path.isdir(DATAPATH):
    od.download('https://www.kaggle.com/datasets/ishantjuyal/emotions-in-text',
            data_dir='dataset/')
```

In [4]:
```
data = pd.read_csv(DATAPATH+'/Emotion_final.csv')
```

Data Understanding

Sample

In [5]:
```
data.shape
```

Out[5]:
```
(21459, 2)
```

In [6]:
```
data.head(5)
```

Out[6]:

	Text	Emotion
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger

3	i am ever feeling nostalgic about the fireplac...	love
4	i am feeling grouchy	anger

In [7]:
```
data.columns = list(map(lambda i: i.lower(), data.columns))
```

In [8]:
```
data.head()
```

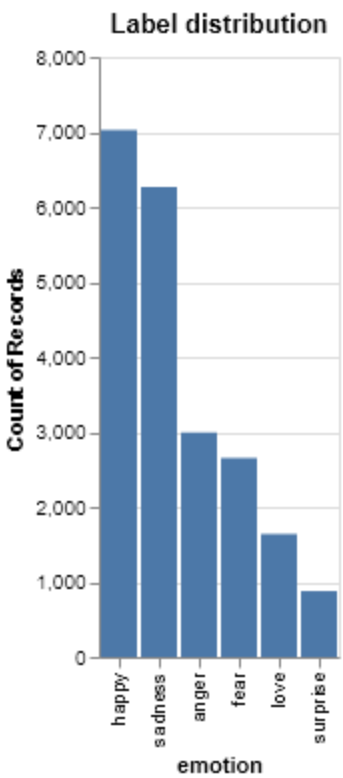
Out[8]:

	text	emotion
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	i am ever feeling nostalgic about the fireplac...	love
4	i am feeling grouchy	anger

Labels distribution

In [9]:
```
hist_labels = alt.Chart(data, title="Label distribution").mark_bar().encode(x = alt.X('e
hist_labels
```

Out[9]:



Tokens distribution - General

In [10]:
```
df_tokens = (data['text'].apply(lambda sentence: word_tokenize(sentence, language='engli
            .explode()
            .reset_index(drop=True)
            .to_frame())
df_tokens.columns = ['token']
```

In [11]:
```
df_tokens.head()
```

Out[11]:

	token

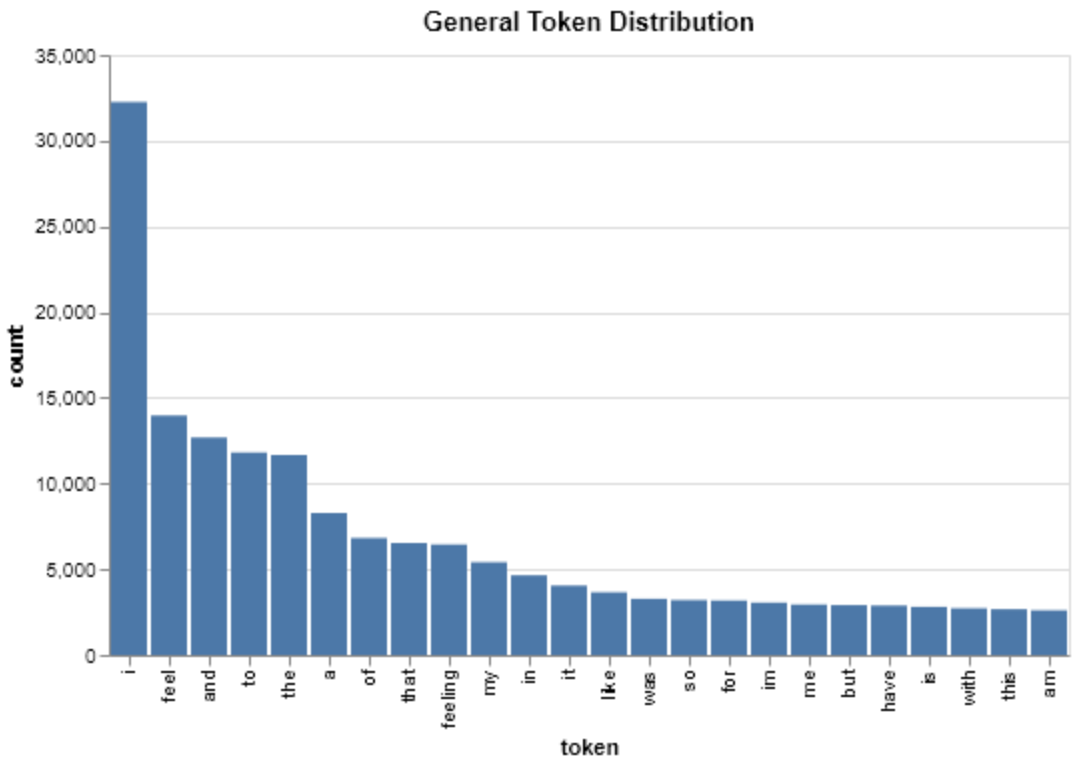
0	i
1	didnt
2	feel
3	humiliated
4	i

In [12]:
```
df_tokens.shape
```

Out[12]:
```
(409965, 1)
```

In [13]:
```
hist_tokens = (alt.Chart(df_tokens, title="General Token Distribution")
                .transform_aggregate(
                    count='count()',
                    groupby=['token'])
                .transform_window(
                    rank='rank(count)',
                    sort=[alt.SortField('count', order='descending')])
                .transform_filter(
                    alt.datum.rank < 25)
                .mark_bar().encode(
                    y='count:Q',
                    x = alt.X('token', sort='-y'),
                    tooltip=['count:Q']))
hist_tokens
```

Out[13]:



Distribution of tokens per label (with stopwords and with/without duplicates)

In [14]:
```
df_tokens_label = data.copy()
df_tokens_label['text'] = (df_tokens_label['text'].apply(lambda sentence: word_tokenize(
df_tokens_label = df_tokens_label.explode(['text']).reset_index(drop=True)
df_tokens_label.columns = ['token', 'label']
```

In [15]:
```
df_tokens_label.groupby('label').count().sort_values('token', ascending=False)
```

token

Out[15]:

label	
happy	136690
sadness	115719
anger	57049
fear	49794
love	33833
surprise	16880

In [16]:
```
df_tokens_label.drop_duplicates().groupby('label').count().sort_values('token', ascendin
```

Out[16]:

	token
label	
happy	10631
sadness	9220
anger	6457
fear	6044
love	4416
surprise	3102

Data Preparation

In this test we dont will pre-process anything because BERT has default pre-process steps. Before this steps, we will make the MLM pre-processing steps to train the model.

Modeling - Sequence Model

In [17]:
```
model_name = 'bert-base-uncased'
max_length = 512
```

In [18]:
```
tokenizer = BertTokenizerFast.from_pretrained(model_name, do_lower_case=True, use_cache=
```

Mapping token-index and index-token

In [19]:
```
token2idx = tokenizer.get_vocab()
idx2token = {value:key for key, value in token2idx.items()}
```

In [20]:
```
test_tensor = tokenizer.encode("i'm working in information retrieval course", return_ten
list(map(lambda i: idx2token[i], test_tensor[0].tolist()))
```

Out[20]:
```
['[CLS]',
 'i',
 "'",
 'm',
 'working',
 'in',
 'information',
 'retrieval',
```

```
 'course',
 '[SEP]']
```

Train and validation data split

In [21]:
```
data_batch = data.copy()
data_batch.columns = ['text', 'labels']
```

In [22]:
```
documents = data_batch[['text']]
labels = data_batch[['labels']]

# Encoding categorical labels
le = preprocessing.LabelEncoder()
label_encoder = le.fit(labels['labels'].tolist())
labels = label_encoder.transform(labels['labels'])

# Data split 80/20
train_data, valid_data, train_labels, valid_labels = train_test_split(documents, labels,
```

Tokenize data with truncation and padding

Truncate -> more tokens than max_lenght \ Padding -> less tokens than max_lenght (padding with special_token for padding set in model)

In [23]:
```
train_encodings = tokenizer(train_data['text'].tolist(), truncation=True, padding=True,
valid_encodings = tokenizer(valid_data['text'].tolist(), truncation=True, padding=True,
```

Dataset class definition

In [24]:
```
class EmotionDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {k: torch.tensor(v[idx]) for k, v in self.encodings.items()}
        item["labels"] = torch.tensor([self.labels[idx]]).type(torch.LongTensor) # Conve
                                                                                  # to pr
        return item

    def __len__(self):
        return len(self.labels)
```

In [25]:
```
train_dataset = EmotionDataset(train_encodings, train_labels)
valid_dataset = EmotionDataset(valid_encodings, valid_labels)
```

Load model

In [26]:
```
model = BertForSequenceClassification.from_pretrained(model_name, num_labels=len(label_e
```

```
Some weights of the model checkpoint at bert-base-uncased were not used when initializin
g BertForSequenceClassification: ['cls.predictions.transform.LayerNorm.bias', 'cls.predi
ctions.bias', 'cls.predictions.decoder.weight', 'cls.predictions.transform.dense.bias',
'cls.seq_relationship.bias', 'cls.predictions.transform.LayerNorm.weight', 'cls.seq_rela
tionship.weight', 'cls.predictions.transform.dense.weight']
- This IS expected if you are initializing BertForSequenceClassification from the checkp
oint of a model trained on another task or with another architecture (e.g. initializing
a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertForSequenceClassification from the ch
eckpoint of a model that you expect to be exactly identical (initializing a BertForSeque
```

```
nceClassification model from a BertForSequenceClassification model).
Some weights of BertForSequenceClassification were not initialized from the model checkp
oint at bert-base-uncased and are newly initialized: ['classifier.bias', 'classifier.wei
ght']
You should probably TRAIN this model on a down-stream task to be able to use it for pred
ictions and inference.
```

Training

In [27]:

```
def compute_metrics(pred):
    labels = pred.label_ids
    preds = pred.predictions.argmax(-1)
    # calculate accuracy using sklearn's function
    acc = accuracy_score(labels, preds)
    f1 = f1_score(labels, preds, average='macro')
    precision = precision_score(labels, preds, average='macro')
    return {
      'accuracy': acc,
      'f1_score': f1,
      'precision': precision
    }

training_args = TrainingArguments(
    output_dir='./models/checkpoint',# output directory
    num_train_epochs=3,              # total number of training epochs
    per_device_train_batch_size=16,  # batch size per device during training
    per_device_eval_batch_size=20,   # batch size for evaluation
    warmup_steps=500,                # number of warmup steps for learning rate schedule
    weight_decay=0.01,               # strength of weight decay
    logging_dir='./logs',            # directory for storing logs
    load_best_model_at_end=True,     # load the best model when finished training (defau
    # but you can specify `metric_for_best_model` argument to change to accuracy or othe
    logging_steps=500,               # log & save weights each logging_steps
    evaluation_strategy="steps",     # evaluate each `logging_steps`
    save_steps=1500,
    learning_rate=5e-5,
    seed=42,
)

trainer = Trainer(
    model=model,                         # the instantiated Transformers model to be tra
    args=training_args,                  # training arguments, defined above
    train_dataset=train_dataset,         # training dataset
    eval_dataset=valid_dataset,          # evaluation dataset
    compute_metrics=compute_metrics,     # the callback that computes metrics of interes
)
```

In [28]:

```
trainer.train()
```

```
C:\Users\Matheus\anaconda3\lib\site-packages\transformers\optimization.py:306: FutureWar
ning: This implementation of AdamW is deprecated and will be removed in a future versio
n. Use the PyTorch implementation torch.optim.AdamW instead, or set `no_deprecation_warn
ing=True` to disable this warning
  warnings.warn(
***** Running training *****
  Num examples = 17167
  Num Epochs = 3
  Instantaneous batch size per device = 16
  Total train batch size (w. parallel, distributed & accumulation) = 16
  Gradient Accumulation steps = 1
  Total optimization steps = 3219
```

[3219/3219 12:36, Epoch 3/3]

Step	Training Loss	Validation Loss	Accuracy	F1 Score	Precision

500	1.007400	0.367476	0.885834	0.852539	0.869502
1000	0.318700	0.231028	0.924511	0.885841	0.921565
1500	0.186800	0.172608	0.931267	0.902500	0.906342
2000	0.151200	0.156186	0.936626	0.905327	0.924921
2500	0.114900	0.145500	0.944548	0.923197	0.922014
3000	0.087800	0.152859	0.939422	0.908087	0.919948

```
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
Saving model checkpoint to ./models/checkpoint\checkpoint-1500
Configuration saved in ./models/checkpoint\checkpoint-1500\config.json
Model weights saved in ./models/checkpoint\checkpoint-1500\pytorch_model.bin
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
Saving model checkpoint to ./models/checkpoint\checkpoint-3000
Configuration saved in ./models/checkpoint\checkpoint-3000\config.json
Model weights saved in ./models/checkpoint\checkpoint-3000\pytorch_model.bin


Training completed. Do not forget to share your model on huggingface.co/models =)


Loading best model from ./models/checkpoint\checkpoint-3000 (score: 0.1528589129447937).
```

Out[28]:

```
TrainOutput(global_step=3219, training_loss=0.29593553451248933, metrics={'train_runtim
e': 757.9626, 'train_samples_per_second': 67.947, 'train_steps_per_second': 4.247, 'tota
l_flos': 2302606080827532.0, 'train_loss': 0.29593553451248933, 'epoch': 3.0})
```

In [29]:

```
trainer.evaluate()
```

```
***** Running Evaluation *****
  Num examples = 4292
  Batch size = 20
```

[215/215 00:13]

Out[29]:

```
{'eval_loss': 0.1528589129447937,
 'eval_accuracy': 0.9394221808014911,
 'eval_f1_score': 0.9080865946249088,
 'eval_precision': 0.9199482490527001,
 'eval_runtime': 13.6386,
 'eval_samples_per_second': 314.695,
 'eval_steps_per_second': 15.764,
 'epoch': 3.0}
```

Save Model

In [35]:

```
from datetime import datetime
now = datetime.now()
```

```
date_time = now.strftime("%m-%d-%Y_%H-%M-%S")

model_path = "models/emotion-bert-base-uncased_%s"%date_time
model.save_pretrained(model_path)
tokenizer.save_pretrained(model_path)
```

```
Configuration saved in models/emotion-bert-base-uncased_11-02-2022_19-45-07\config.json
Model weights saved in models/emotion-bert-base-uncased_11-02-2022_19-45-07\pytorch_mode
l.bin
tokenizer config file saved in models/emotion-bert-base-uncased_11-02-2022_19-45-07\toke
nizer_config.json
Special tokens file saved in models/emotion-bert-base-uncased_11-02-2022_19-45-07\specia
l_tokens_map.json
```

Out[35]:
```
('models/emotion-bert-base-uncased_11-02-2022_19-45-07\\tokenizer_config.json',
 'models/emotion-bert-base-uncased_11-02-2022_19-45-07\\special_tokens_map.json',
 'models/emotion-bert-base-uncased_11-02-2022_19-45-07\\vocab.txt',
 'models/emotion-bert-base-uncased_11-02-2022_19-45-07\\added_tokens.json',
 'models/emotion-bert-base-uncased_11-02-2022_19-45-07\\tokenizer.json')
```

Playground

In [36]:
```
def get_prediction(text):
    inputs = tokenizer(text, padding=True, truncation=True, max_length=max_length, retur
    outputs = model(**inputs)
    probs = outputs[0].softmax(1)
    return label_encoder.classes_[probs.argmax()]
```

In [37]:
```
get_prediction('i love the world. The people are beautiful.')
```

Out[37]:
```
'love'
```

In [38]:
```
get_prediction('the only purpouse of humans if destroy the earth.')
```

Out[38]:
```
'anger'
```

Getting embeddings from last hidden state of model

Get last hidden state of classification model

In [ ]:
```
my_text = "i'm so happy today."
_input = tokenizer(my_text, padding='max_length', truncation=True, max_length=max_length
_output = model(**_input, output_hidden_states=True)
```

In [ ]:
```
# 9 tokens total tokens -> including CLS and SEP.
tokenizer.tokenize(my_text)
```

In [ ]:
```
# Last Hidden State Values
lhs = _output.hidden_states[-1]
lhs.shape
```

In [ ]:
```
# Get padding indexes
attention_mask = _input['attention_mask']
```

In [ ]:
```
# unsqueeze mask
mask = attention_mask.unsqueeze(-1).expand(_output.hidden_states[0].shape).float()
```

In [ ]:
```
# Avoid masked values in embedding by zero this values with lhs*mask
masked_embeddings = lhs * mask
```

In [ ]:
```
masked_embeddings.shape
```

In [ ]:
```
# Pooling - SUM
summed = torch.sum(masked_embeddings, 1)
summed.shape
```

In [ ]:
```
summed.shape
```

In [ ]:
```
# Pooling - Count
cnts = torch.clamp(mask.sum(1), min=1e-8)
cnts.shape
```

In [ ]:
```
cnts[0][0].item()
```

In [ ]:
```
# Pooling - Mean
mean_pooled = summed/cnts
mean_pooled.shape
```

In [ ]:
```
mean_pooled
```

In [ ]:
```
from sentence_transformers import SentenceTransformer
model = SentenceTransformer('models/emotion-bert-base-uncased_11-02-2022_18-44-55')
```

In [ ]:
```
model.encode(my_text)
```

In [ ]:
```
mean_pooled_cpu = mean_pooled.cpu()
```

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

In [ ]:
```
anger_1 = data_batch[data_batch.labels == 'anger']['text'].iloc[0]
anger_2 = data_batch[data_batch.labels == 'anger']['text'].iloc[1]

happy_1 = data_batch[data_batch.labels == 'happy']['text'].iloc[0]
happy_2 = data_batch[data_batch.labels == 'happy']['text'].iloc[1]
```

In [ ]:
```
anger_1
```

In [ ]:
```
anger_2
```

In [ ]:
```
happy_1
```

In [ ]:
```
happy_2
```

In [ ]:
```
cosine_similarity([model.encode(happy_1)], [model.encode(anger_2)], dense_output=True)
```

Visualisation t-SNE Bert

In [42]:
```
model = SentenceTransformer(model_name)

data_vis = data_batch.head(5000).copy()

X = model.encode(data_vis['text'].tolist())
X_embedded = TSNE(n_components=2).fit_transform(X)

data_vis[['x', 'y']] = X_embedded
```

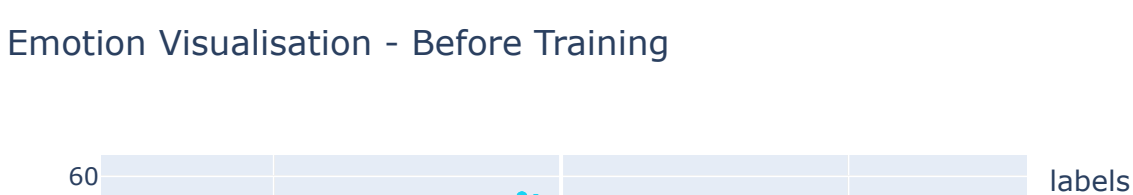
```
No sentence-transformers model found with name C:\Users\Matheus/.cache\torch\sentence_tr
```

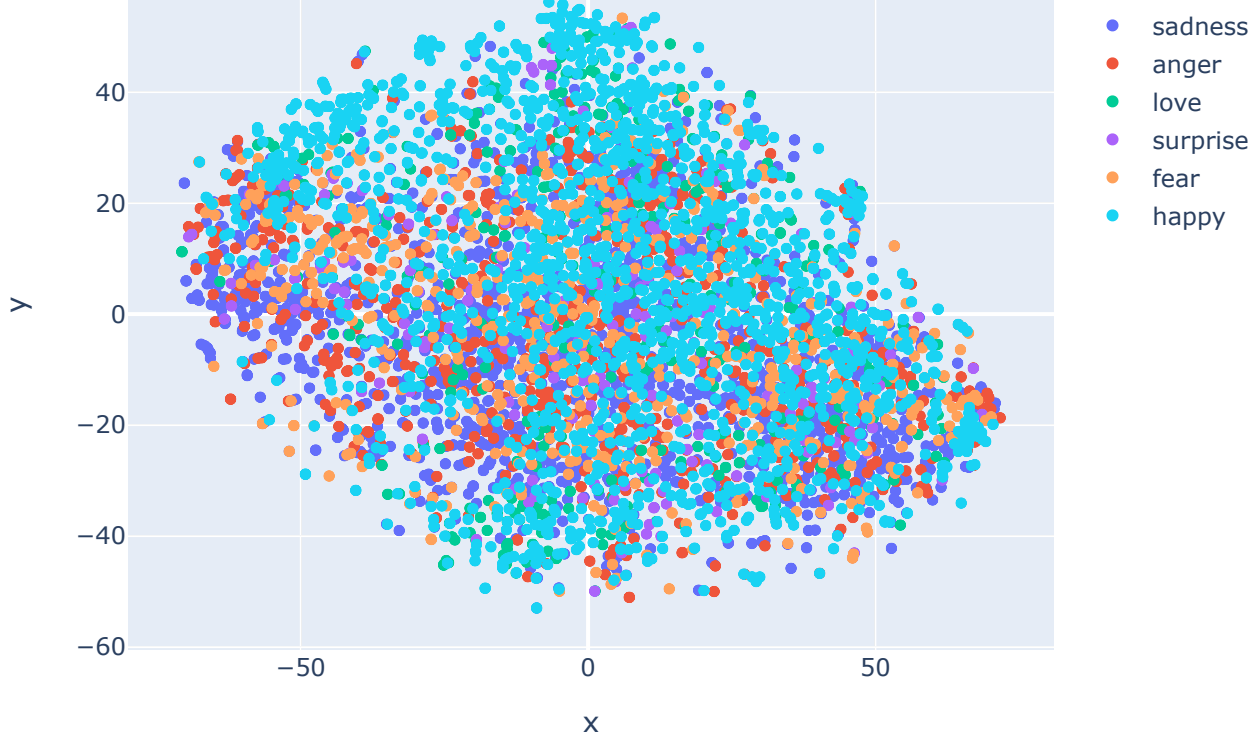
```
ansformers\bert-base-uncased. Creating a new one with MEAN pooling.
loading configuration file C:\Users\Matheus/.cache\torch\sentence_transformers\bert-base
-uncased\config.json
Model config BertConfig {
  "_name_or_path": "C:\\Users\\Matheus/.cache\\torch\\sentence_transformers\\bert-base-u
ncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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": "4.22.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}

loading weights file C:\Users\Matheus/.cache\torch\sentence_transformers\bert-base-uncas
ed\pytorch_model.bin
Some weights of the model checkpoint at C:\Users\Matheus/.cache\torch\sentence_transform
ers\bert-base-uncased were not used when initializing BertModel: ['cls.predictions.trans
form.LayerNorm.bias', 'cls.predictions.bias', 'cls.predictions.decoder.weight', 'cls.pre
dictions.transform.dense.bias', 'cls.seq_relationship.bias', 'cls.predictions.transform.
LayerNorm.weight', 'cls.seq_relationship.weight', 'cls.predictions.transform.dense.weigh
t']
- This IS expected if you are initializing BertModel from the checkpoint of a model trai
ned on another task or with another architecture (e.g. initializing a BertForSequenceCla
ssification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model
that you expect to be exactly identical (initializing a BertForSequenceClassification mo
del from a BertForSequenceClassification model).
All the weights of BertModel were initialized from the model checkpoint at C:\Users\Math
eus/.cache\torch\sentence_transformers\bert-base-uncased.
If your task is similar to the task the model of the checkpoint was trained on, you can
already use BertModel for predictions without further training.
loading configuration file C:\Users\Matheus/.cache\torch\sentence_transformers\bert-base
-uncased\config.json
Model config BertConfig {
  "_name_or_path": "C:\\Users\\Matheus/.cache\\torch\\sentence_transformers\\bert-base-u
ncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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": "4.22.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}

loading file vocab.txt
loading file tokenizer.json
loading file added_tokens.json
loading file special_tokens_map.json
loading file tokenizer_config.json
loading configuration file C:\Users\Matheus/.cache\torch\sentence_transformers\bert-base
-uncased\config.json
Model config BertConfig {
  "_name_or_path": "C:\\Users\\Matheus/.cache\\torch\\sentence_transformers\\bert-base-u
ncased",
  "architectures": [
    "BertForMaskedLM"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "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": "4.22.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}

C:\Users\Matheus\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:800: FutureWarni
ng: The default initialization in TSNE will change from 'random' to 'pca' in 1.2.
  warnings.warn(
C:\Users\Matheus\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:810: FutureWarni
ng: The default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
  warnings.warn(
```

In [43]:

```
fig = px.scatter(
    data_vis, x='x', y='y',
    color='labels', labels={'color': 'label'},
    hover_data=['text'], title = 'Emotion Visualisation - Before Training')
fig.show()
```





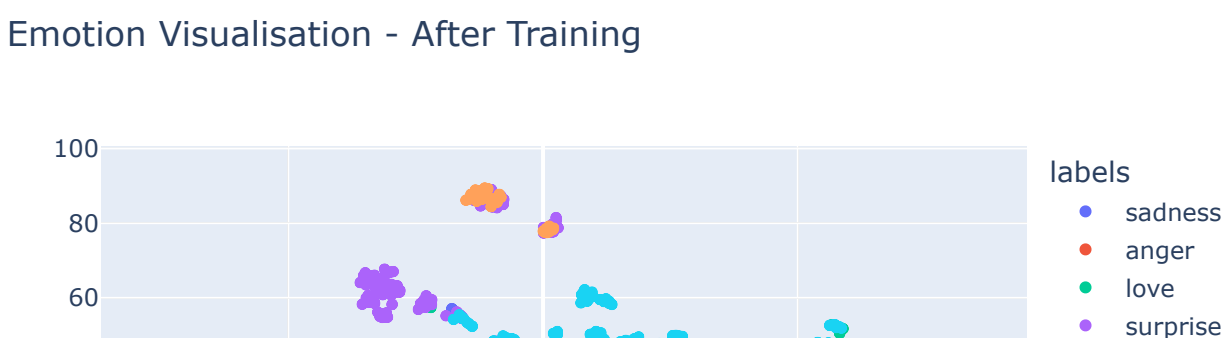
In [46]:
```
model = SentenceTransformer('models/emotion-bert-base-uncased_11-02-2022_19-45-07')

data_vis = data_batch.head(5000).copy()

X = model.encode(data_vis['text'].tolist())
X_embedded = TSNE(n_components=2).fit_transform(X)

data_vis[['x', 'y']] = X_embedded
```

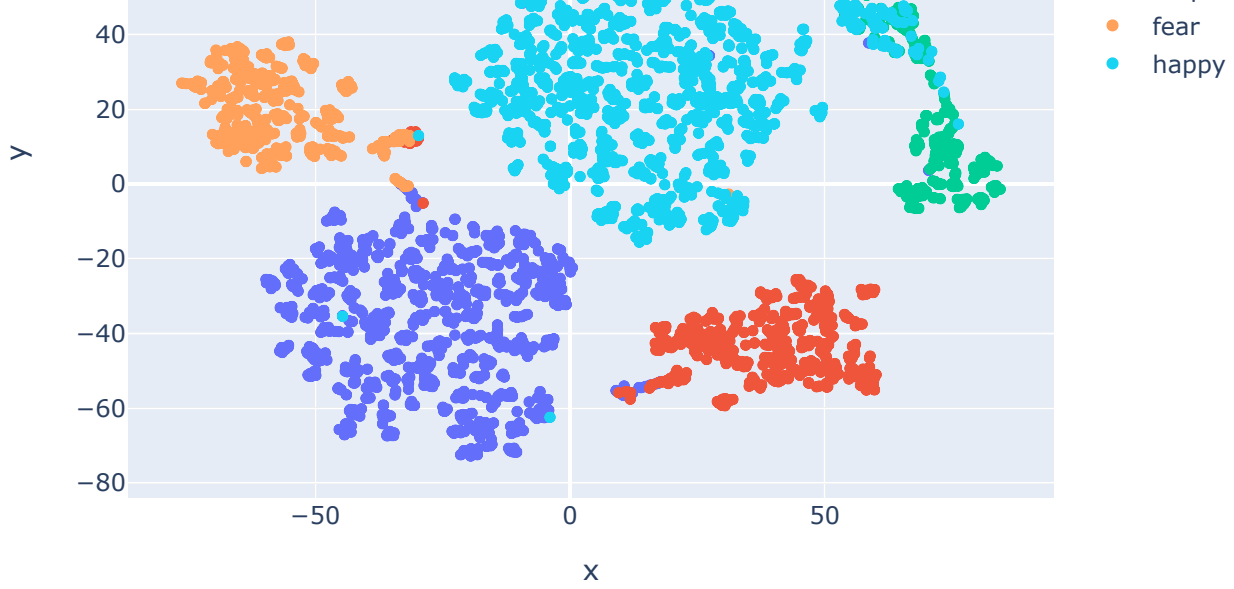
```
No sentence-transformers model found with name models/emotion-bert-base-uncased_11-02-20
22_19-45-07. Creating a new one with MEAN pooling.
loading configuration file models/emotion-bert-base-uncased_11-02-2022_19-45-07\config.j
son
Model config BertConfig {
  "_name_or_path": "models/emotion-bert-base-uncased_11-02-2022_19-45-07",
  "architectures": [
    "BertForSequenceClassification"
  ],
  "attention_probs_dropout_prob": 0.1,
  "classifier_dropout": null,
  "gradient_checkpointing": false,
  "hidden_act": "gelu",
  "hidden_dropout_prob": 0.1,
  "hidden_size": 768,
  "id2label": {
    "0": "LABEL_0",
    "1": "LABEL_1",
    "2": "LABEL_2",
    "3": "LABEL_3",
    "4": "LABEL_4",
    "5": "LABEL_5"
  },
  "initializer_range": 0.02,
  "intermediate_size": 3072,
  "label2id": {
    "LABEL_0": 0,
    "LABEL_1": 1,
    "LABEL_2": 2,
    "LABEL_3": 3,
```

```
    "LABEL_4": 4,
    "LABEL_5": 5
  },
  "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",
  "problem_type": "single_label_classification",
  "torch_dtype": "float32",
  "transformers_version": "4.22.0",
  "type_vocab_size": 2,
  "use_cache": true,
  "vocab_size": 30522
}

loading weights file models/emotion-bert-base-uncased_11-02-2022_19-45-07\pytorch_model.
bin
Some weights of the model checkpoint at models/emotion-bert-base-uncased_11-02-2022_19-4
5-07 were not used when initializing BertModel: ['classifier.bias', 'classifier.weight']
- This IS expected if you are initializing BertModel from the checkpoint of a model trai
ned on another task or with another architecture (e.g. initializing a BertForSequenceCla
ssification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model
that you expect to be exactly identical (initializing a BertForSequenceClassification mo
del from a BertForSequenceClassification model).
All the weights of BertModel were initialized from the model checkpoint at models/emotio
n-bert-base-uncased_11-02-2022_19-45-07.
If your task is similar to the task the model of the checkpoint was trained on, you can
already use BertModel for predictions without further training.
loading file vocab.txt
loading file tokenizer.json
loading file added_tokens.json
loading file special_tokens_map.json
loading file tokenizer_config.json
C:\Users\Matheus\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:800: FutureWarni
ng:

The default initialization in TSNE will change from 'random' to 'pca' in 1.2.

C:\Users\Matheus\anaconda3\lib\site-packages\sklearn\manifold\_t_sne.py:810: FutureWarni
ng:

The default learning rate in TSNE will change from 200.0 to 'auto' in 1.2.
```

In [47]:

```
fig = px.scatter(
    data_vis, x='x', y='y',
    color='labels', labels={'color': 'label'},
    hover_data=['text'], title = 'Emotion Visualisation - After Training')
fig.show()
```





Results PNG

alt text

alt text