# Project 2 - Emotion

In this project, we will apply the knowledge we have collected in the course to a natural language processing problem. You will need a python environment with the following packages installed: torch (PyTorch) and datasets (HuggingFace Datasets) – as well as matplotlib or seaborn for visualization.

# The Project

### The Dataset

We will be working with the Emotion dataset. Emotion is a dataset of English Twitter messages with six basic emotions: anger, fear, joy, love, sadness, and surprise. You can download the dataset through using the load_data function of the Python package datasets – the package can be installed via

```
pip install datasets
```

Then, the data can be loaded as follows:

```
from datasets import load_dataset
ds = load_dataset("dair-ai/emotion", "split")
```

The data comes with a pre-defined train/val/test split, which we will also use for this project. Each example consists of a short text and a classification label. The labels have the following possible values: sadness (0), joy (1), love (2), anger (3), fear (4), surprise (5).

### Task 1 - Data Preparation

Our main goal for preparing the data is to convert the texts into numerical representations such that we can later feed them into our model. On the way, we will also calculate basic dataset statistics.

Step 1: Extract the class labels from all splits into lists of integer values. Count the occurrences of each class and print out and/or visualize the class distribution for each split. Is the class distribution balanced? Is the class distribution of the training examples representative for the class distribution of the validation data? What is the chance accuracy level? What would be the accuracy of a classifier that only predicts the most common class seen in training?

Step 2: Extract the texts for all splits and split each text into tokens. For instance, you can split the text by whitespace. Apply the tokenizer to all examples in all splits. Each example should now be a list of tokens. Analyze the distribution of text lengths by providing its range, mean and standard deviation.

Step 3: Build a vocabulary (a mapping from string to integer) based on the train split of the data. Reserve a slot in the vocabulary for a special token for padding. Have in mind that you may need to deal with out-of-vocabulary words at test time – think of a strategy for how to deal with such cases & implement it.

Step 4: Encode all texts with this vocabulary, while making use of your out-of-vocabulary strategy. Each text should now be in the format of a list of integer values. Apply this procedure to all splits. Each split of the dataset should be a list of list of integers.

Step 4: In order to convert these lists of lists into a tensor, they all need to be the same length. Decide for a reasonable maximum length depending on the length distribution that you have analyzed in Step 2. Then, ensure that all your sequences of tokens are of the same length by adding the special padding token that you have defined in Step 3. Convert the data of each split into a Tensor and initialize loaders.

## Task 2 - Model 1

Design a model that is suitable for the task. Train the model, while keeping track of training loss, validation loss and validation accuracy. Visualize training loss and validation loss over the course of training. Tweak the model architecture and tune the hyperparameters to improve validation accuracy. Then, evaluate the final model on the test set. What accuracy does it achieve?

## Task 3 - Model 2

Design a second model that is suitable for the task, but differs in a crucial aspect from the first model (e.g., it has a different architecture). Again, visualize training and validation loss over the course of training. Tweak the hyper parameter and model architecture. Then, evaluate this second model on the final test set. What accuracy does it achieve? Which model is better? Give some possible reasons.

## Task 4 - Analysis

Perform a basic analysis to better understand *why* your model (choose one or repeat for both) makes a certain prediction. You can start by manually inspecting failure cases where the predicted class differs from the true class. Give some possible reasons.

To test your hypotheses, add a functionality for your model to take in text on-the-fly (e.g., from the standard input) and respond with the classification label. Then, take some of the failure cases and and change words until the classification is correct. Also, try to take a correct prediction and modify the text until it gets misclassified – with as minimal changes as possible. Discuss your findings. For instance: Does the model prediction match with your intuition? Are there any particular words that are indicative for a class? Does the model generalize to longer texts? List any other interesting observations here.

Hint: A function to convert a sequence of token indices to back to text may be helpful for easier inspection.

## Task 5 - Fine-tuning a pre-trained language model (Bonus)

Download a pre-trained language model (e.g., bert-base-uncased from HuggingFace model hub), and familiarize yourself with how to apply it. For instance, you may want to employ the model-specific tokenizer. Then, fine-tune the model on our dataset – again while observing training and validation loss. Tune the hyperparameters of the fine-tuning to improve validation accuracy. After tuning hyperparameters, start a final run and evaluate the model's test accuracy. Does this model achieve a higher accuracy than the models trained from scratch? Again perform some basic analysis like in Task 4, and discuss the differences to the models trained from scratch, if there are any.

## When to submit

You need to submit before Tuesday 2024-12-03 at 12:00 (noon) on itslearning.

## What to submit

The hand-in must include a short report. In the report you must include what you've done, the reasoning behind, your results, and a short explanation of your results. You are welcome to include screenshots of your networks training. You must also submit your code.

## Notes

You should form groups of 2-3 people to do this assignment – group size 1 is permitted in exceptional cases. As training your models can take a long time, remember to start early on this assignment.