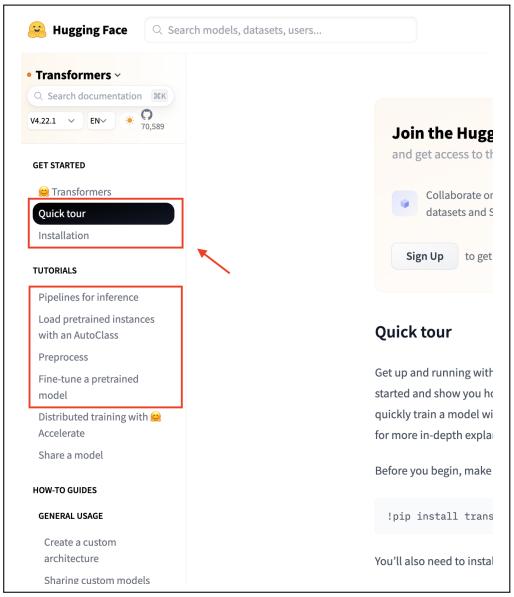
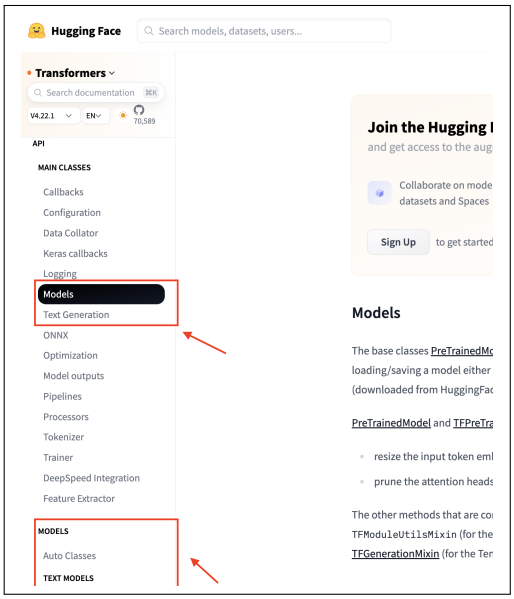
As part of this assignment, you will experience a taste of NLP leaderboard culture and participate in a text classification competition. The first step is to follow a tutorial on using HuggingFace to create a text classifier. As a next step, select one of the existing text classification tasks, a pre-trained model card, and replicate one of the state-of-the-art models. For the replication, you can use existing code written by authors that appear on the Papers-with-Code* leaderboard or use some basic Transformer models implemented in HuggingFace model libraries.†

Do not spend too much time replicating the exact code. Instead, you can run a code similar to the one used in the paper on your target dataset and compare your results to its. You will be considered to be cheating if you do not correctly cite any tools or papers you used.

Step 1: Getting used to HuggingFace library





Follow the basic instructions on inference, model loading, preprocessing, and fine-tuning in the HuggingFace tutorial: https://huggingface.co/docs/transformers/quicktour. It is highly recommended that you install the library and run the commands in the tutorial in your Google Colab ‡ or your local server using Jupyter Notebook§. In the tutorial document, you can find some default classes implemented by HuggingFace by scrolling down the left menu. You must first understand these abstract classes in order to run their models.

Step 2: Choose a Task and Dataset

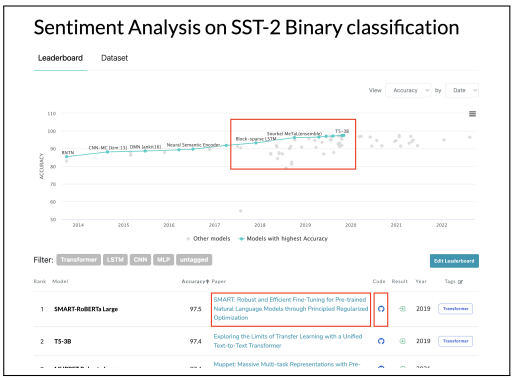
You can now select a task and dataset from the following list. Please contact DK a week before the deadline if you wish to choose another task and dataset. It is recommended that you read the original paper that describes the dataset first. After that, you can download the raw dataset or load it from the pre-formatted HuggingFace dataset. Below are links to the paper, raw dataset, and HuggingFace dataset.

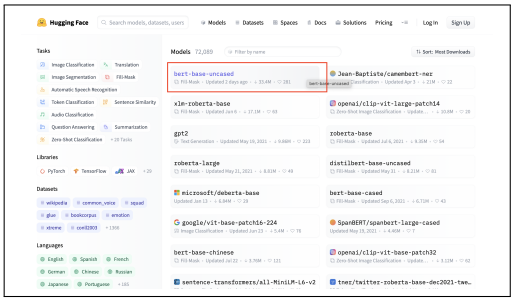
*https://paperswithcode.com/
†https://huggingface.co/models
‡urlhttps://colab.research.google.com/
§https://jupyter.org/

Table 1: List of classification tasks and dataset. The following list contains links to the PapersWithCode leaderboard, HuggingFace formatted dataset, and original paper. Question answering (QA) could be viewed as a classification task which predicts the appropriate start and end position of your answer span given a question.

Tasks	Datasets
Sentiment classification	SST2 (leaderboard, HF dataset, paper) DynaSent (leaderboard, HF dataset, paper)
Politeness classification	StanfordPoliteness (dataset, paper)
Social classification	Social Bias Inference (SBIC) (leaderboard, HF dataset, paper) Hate Speech Detection (HSD) leaderboard, HF dataset, paper)
Natural Language Inference	SNLI (leaderboard, HF dataset, paper) MNLI (leaderboard, HF dataset, paper) MRPC (leaderboard, HF dataset, paper)
Commonsense Reasoning	Winograd Challenge (leaderboard, HF dataset, paper) CommonsenseQA (leaderboard, HF dataset, paper)
(Visual) Question Answering	HotpotQA (leaderboard, HF dataset, paper) SQuAD 2.0 (leaderboard, HF dataset, paper GQA (leaderboard, HF dataset, paper) VQA 2.0 (leaderboard, HF dataset, paper)
Semantic Evaluation (SemEval)	SemEval (2020), SemEval (2021), SemEval (2022), Other SemEval tasks in HF dataset

Step 3: Choose a Model





The next step is to choose a model to replicate. We can start by looking at what models are ranked in the PapersWithCode leaderboards of each dataset and choosing one of the top models. Instead, you can choose one of HuggingFace's pre-trained models, such as BERT, GPT2, or RoBERTa, and fine-tune it. For both cases, you need to train the code provided by authors on the dataset or fine-tune the pre-trained model on your dataset (see Step 4).

Step 4: Replication

The final step is to train your text classifier on your dataset. You might consider choosing one of HuggingFace's pre-trained language models and fine-tuning it if the original authors' code is too

complicated to train. A tutorial on fine-tuning HuggingFace's pre-trained model can be found here tutorial. However, you are not allowed to use the default `Trainer` function in HuggingFace like below.

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=tokenized_imdb["train"],
    eval_dataset=tokenized_imdb["test"],
    tokenizer=tokenizer,
    data_collator=data_collator,
)

trainer.train()
```

Instead, you need to implement your own `Trainer` like `MyTrainer` and then inherit the default `Trainer` except for `_inner_training_loop` function. You can check how the default `_inner_training_loop` is implemented. In your customized `_inner_training_loop` function, you can just copy the code in the default `_inner_training_loop` function, but please understand how your training process is implemented as discussed in class, such as multiple epochs of training, forward and backward propagation, gradient update methods, gradient clipping, and parameter updating. As part of your assignment or class project, you may have to change some parts of this training function or take outputs from forward propagation. Note that your submitted code should include this customized `MyTrainer` with the copied (or modified) version of `_inner_training_loop` function.

Deliverable

Please upload your code and report to Canvas by Oct 6, 11:59pm.

Code: You should provide a zipped file containing your training/inference scripts or a link to your github repository.

Report: Maximum four pages PDF. Your report needs to include the following content:

- description of the task and models with references to the original papers and model cards/repository.
- learning curve graphs of your training losses from forward propagation
- evaluation metrics used in your experiment
- Test set performance and comparison with score reported in original paper or leaderboard. A justification is needed if it differs from the reported scores.
- training and inference time
- hyperparameters used in your experiment (e.g., number of epochs, learning parameter, dropout rate, hidden size of your model) and other details.
- A minimum of ten incorrectly predicted test samples and their ground-truth labels
- potential modeling or representation ideas to improve the errors