FRI
UNIVERSITY OF LJUBLJANA
Faculty of Computer and Information Science

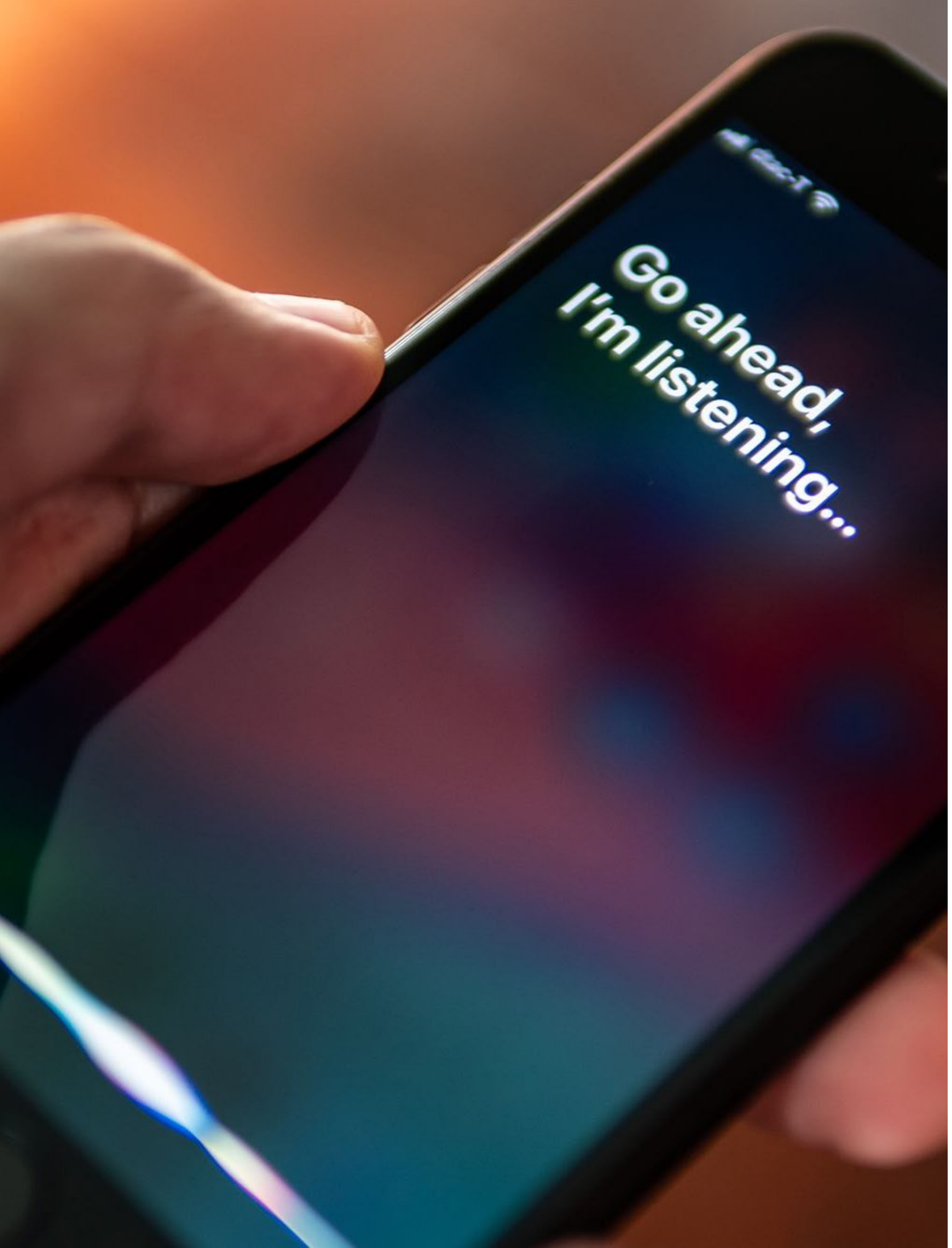
User Best Practice and Results on Vega

AI-Friendly EuroHPC Systems

Iztok Lebar Bajec

25. 9. 2024

Conversational AI



Slovene language?

- Approx 2.5M speakers
- Under/low resourced
- Often overlooked

Credit: wachiwit/shutterstock

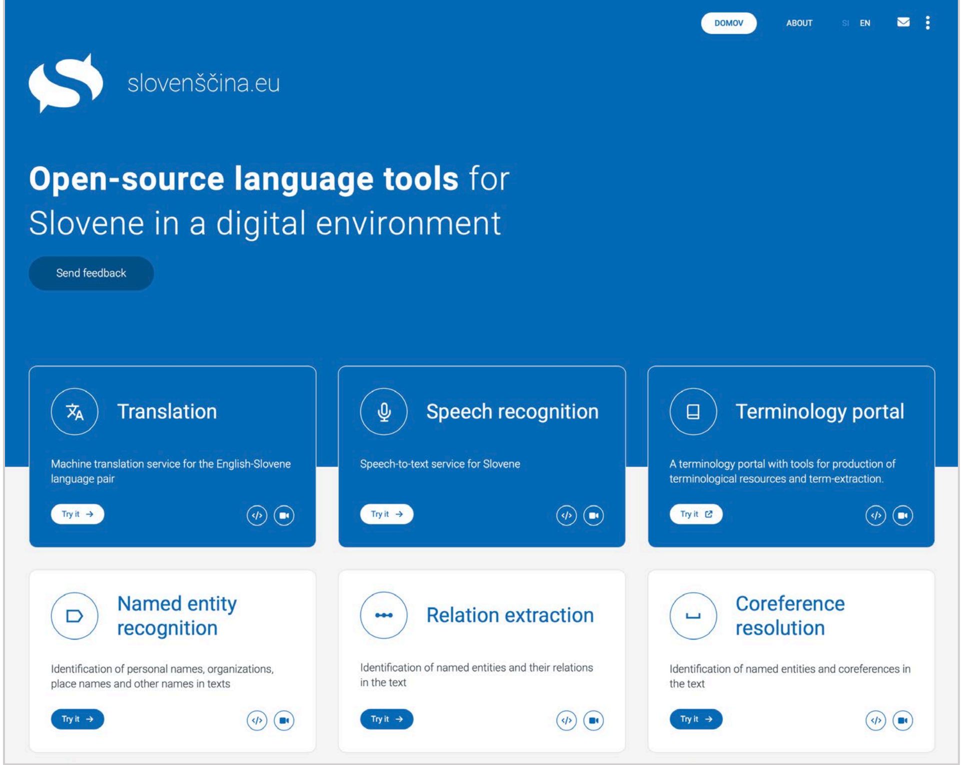
Projects & Results

Development of Slovene in a Digital Environment (RSDO)

Adaptive Natural Language Processing with Large Language Models (PoVeJMo)

Online Notes

Slobench



Open-source language tools for Slovene in a digital environment

Send feedback

Translation
Machine translation service for the English-Slovene language pair
Try it →

Speech recognition
Speech-to-text service for Slovene
Try it →

Terminology portal
A terminology portal with tools for production of terminological resources and term-extraction.
Try it

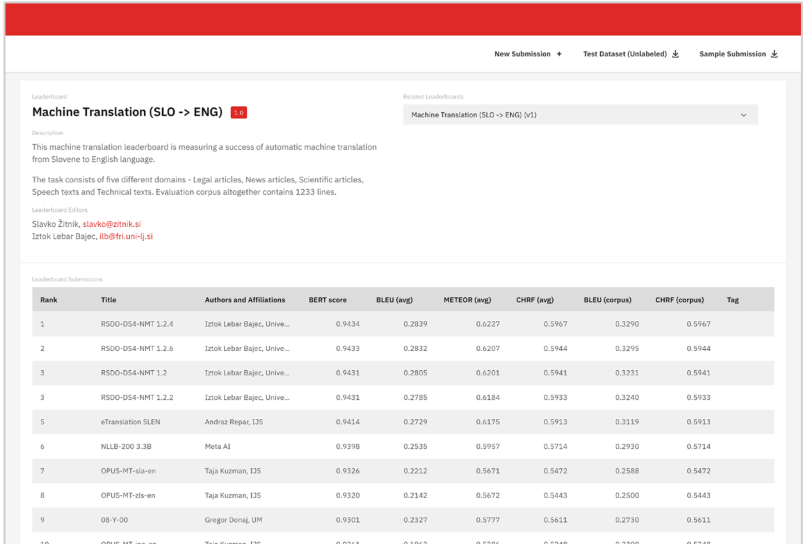
Named entity recognition
Identification of personal names, organizations, place names and other names in texts
Try it →

Relation extraction
Identification of named entities and their relations in the text
Try it →

Coreference resolution
Identification of named entities and coreferences in the text
Try it →

https://slovenscina.eu

https://slobench.cjvt.si



Leaderboard

Machine Translation (SLO -> ENG) 1.0

Related Leaderboards

Machine Translation (SLO -> ENG) (v1)

Description

This machine translation leaderboard is measuring a success of automatic machine translation from Slovene to English language.

The task consists of five different domains - Legal articles, News articles, Scientific articles, Speech texts and Technical texts. Evaluation corpus altogether contains 1233 lines.

Leaderboard Editors

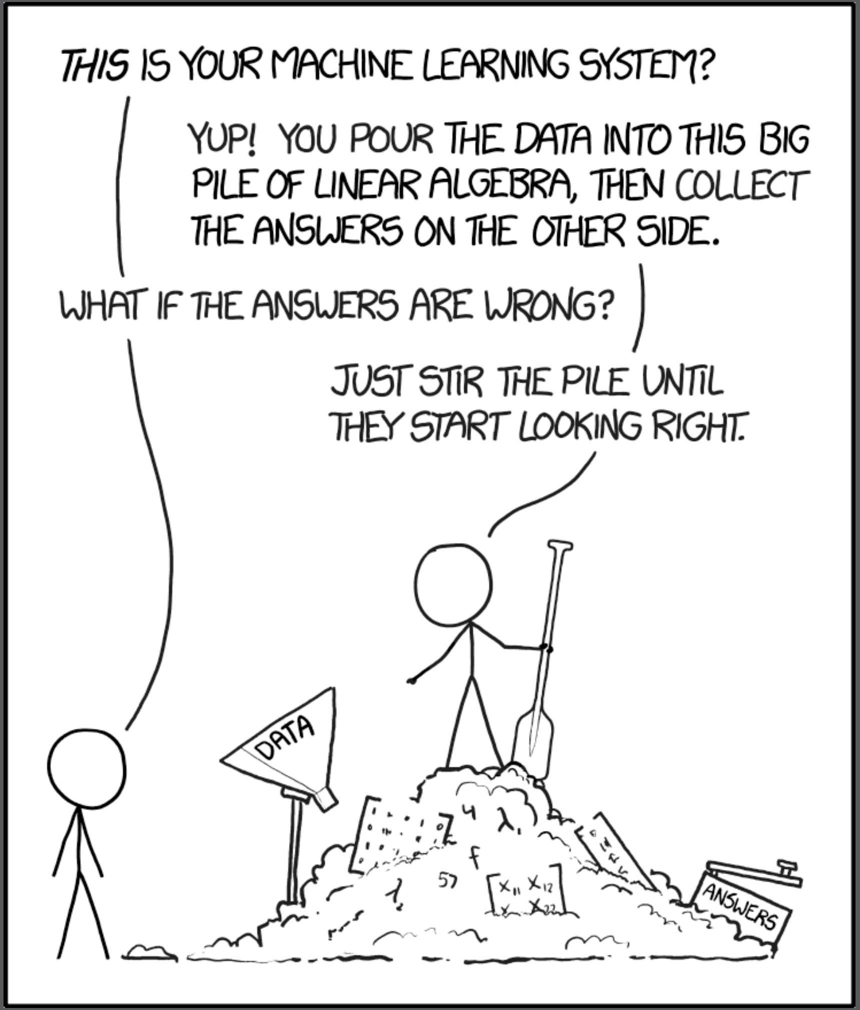
Slavko Žitnik, slavko@zitnik.si
Iztok Lebar Bajec, ilb@fri.uni-lj.si

Leaderboard Submissions

Rank	Title	Authors and Affiliations	BERT score	BLEU (avg)	METEOR (avg)	CHRF (avg)	BLEU (corpus)	CHRF (corpus)	Tag
1	RSDO-DS4-NMT 1.2.4	Iztok Lebar Bajec, Unive...	0.9434	0.2839	0.6227	0.5967	0.3290	0.5967	
2	RSDO-DS4-NMT 1.2.6	Iztok Lebar Bajec, Unive...	0.9433	0.2832	0.6207	0.5944	0.3295	0.5944	
3	RSDO-DS4-NMT 1.2	Iztok Lebar Bajec, Unive...	0.9431	0.2805	0.6201	0.5941	0.3231	0.5941	
3	RSDO-DS4-NMT 1.2.2	Iztok Lebar Bajec, Unive...	0.9431	0.2785	0.6184	0.5933	0.3240	0.5933	
5	eTranslation SLEN	Andraz Repar, IJS	0.9414	0.2729	0.6175	0.5913	0.3119	0.5913	
6	NLLB-200 3.3B	Meta AI	0.9398	0.2535	0.5957	0.5714	0.2930	0.5714	
7	OPUS-MT-sla-en	Taja Kuzman, IJS	0.9326	0.2212	0.5671	0.5472	0.2588	0.5472	
8	OPUS-MT-zls-en	Taja Kuzman, IJS	0.9320	0.2142	0.5672	0.5443	0.2500	0.5443	
9	08-Y-00	Gregor Donaj, UM	0.9301	0.2327	0.5777	0.5611	0.2730	0.5611	
10	OPUS-MT-ine-en	Taja Kuzman, IJS	0.9261	0.1963	0.5386	0.5248	0.2300	0.5248	

Online Notes

Best Practices...


THIS IS YOUR MACHINE LEARNING SYSTEM?
YUP! YOU POUR THE DATA INTO THIS BIG PILE OF LINEAR ALGEBRA, THEN COLLECT THE ANSWERS ON THE OTHER SIDE.
WHAT IF THE ANSWERS ARE WRONG?
JUST STIR THE PILE UNTIL THEY START LOOKING RIGHT.
DATA
ANSWERS

https://xkcd.com/1838

Process

Data acquisition & curation

Data preparation

Iterative training

Training and customization

Experiment management

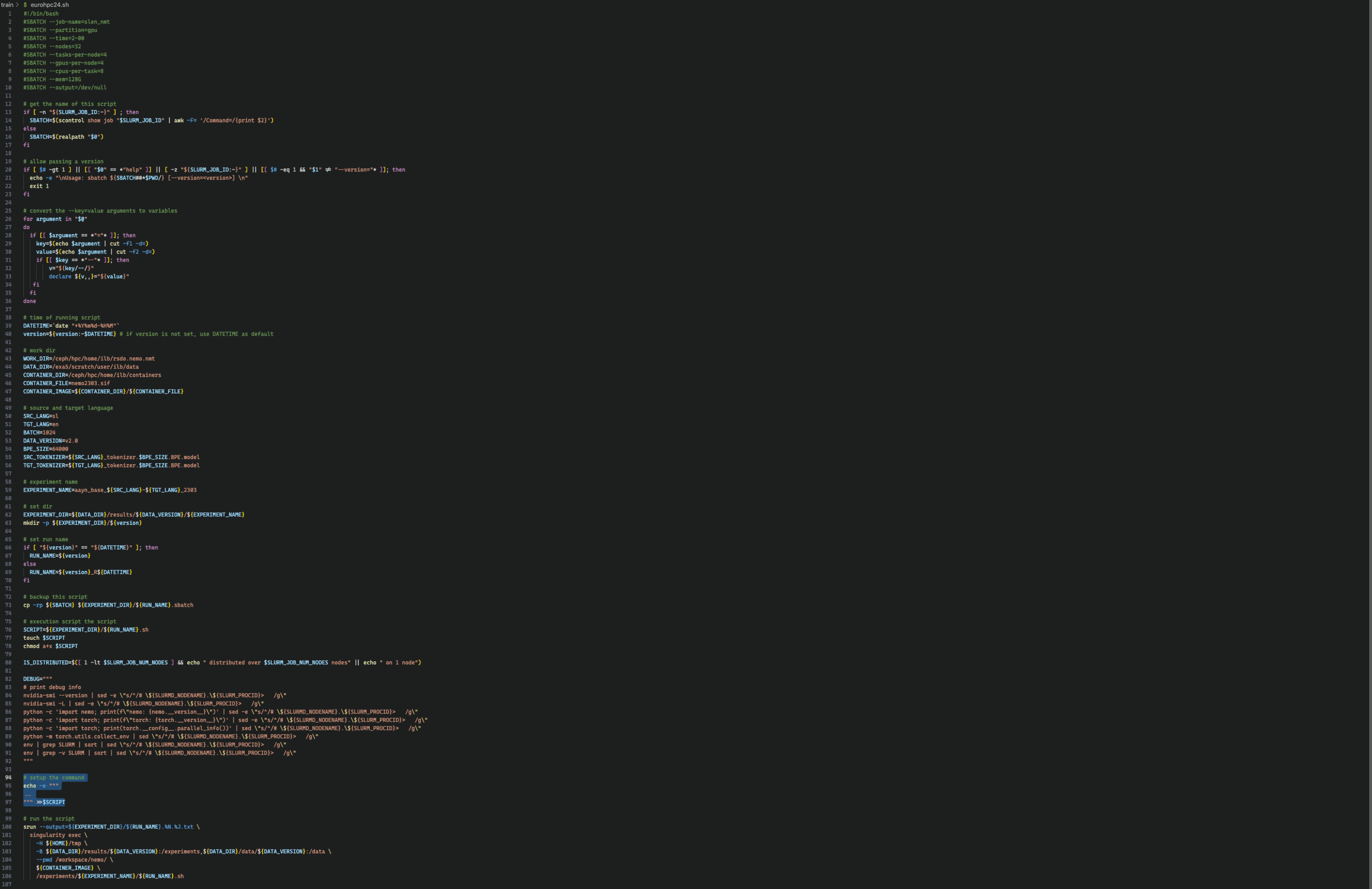
Training and customization

Artifact archival & use

Deployment

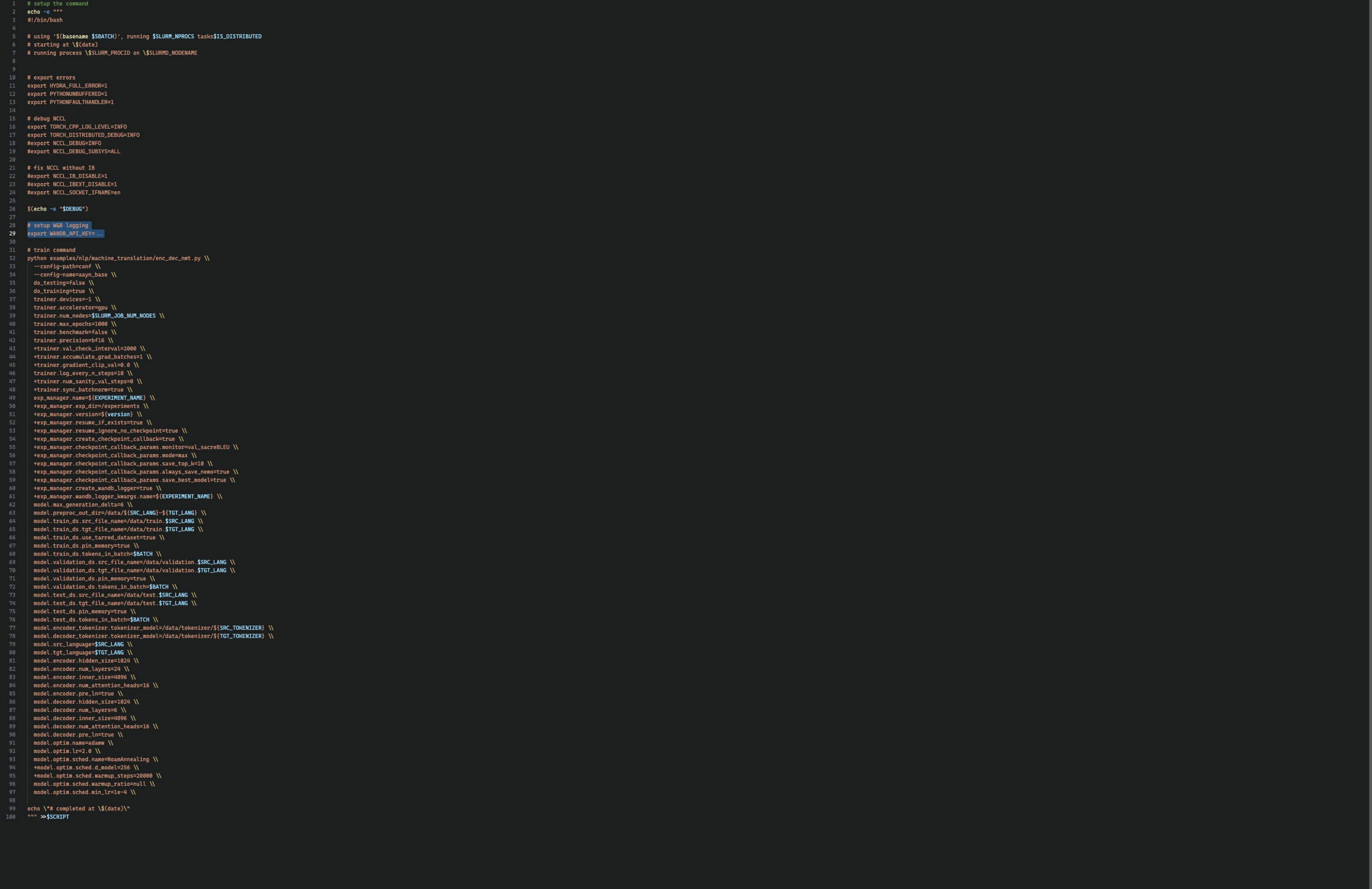
Toolkits and Frameworks

- **NVIDIA NeMo, github.io/nvidia/NeMo**
- Hydra, hydra.cc
- Lightning, lightning.ai
- Pytorch, pytorch.org
- NCCL
- …
- git, dvc, W&B, minio, …



train > $ eurohpc24.sh

```bash
#!/bin/bash
#SBATCH --job-name=slen_nmt
#SBATCH --partition=gpu
#SBATCH --time=2-00
#SBATCH --nodes=32
#SBATCH --tasks-per-node=4
#SBATCH --gpus-per-node=4
#SBATCH --cpus-per-task=8
#SBATCH --mem=128G
#SBATCH --output=/dev/null

# get the name of this script
if [ -n "${SLURM_JOB_ID:-}" ] ; then
  SBATCH=$(scontrol show job "$SLURM_JOB_ID" | awk -F= '/Command=/{print $2}')
else
  SBATCH=$(realpath "$0")
fi

# allow passing a version
if [ $# -gt 1 ] || [[ "$0" == *"help" ]] || [ -z "${SLURM_JOB_ID:-}" ] || [[ $# -eq 1 && "$1" != "--version="* ]]; then
  echo -e "\nUsage: sbatch ${SBATCH##*$PWD/} [--version=<version>] \n"
  exit 1
fi

# convert the --key=value arguments to variables
for argument in "$@"
do
  if [[ $argument == *"="* ]]; then
    key=$(echo $argument | cut -f1 -d=)
    value=$(echo $argument | cut -f2 -d=)
    if [[ $key == *"--"* ]]; then
        v="${key/--/}"
        declare ${v,,}="${value}"
    fi
  fi
done

# time of running script
DATETIME=`date "+%Y%m%d-%H%M"`
version=${version:-$DATETIME} # if version is not set, use DATETIME as default

# work dir
WORK_DIR=/ceph/hpc/home/ilb/rsdo.nemo.nmt
DATA_DIR=/exa5/scratch/user/ilb/data
CONTAINER_DIR=/ceph/hpc/home/ilb/containers
CONTAINER_FILE=nemo2303.sif
CONTAINER_IMAGE=${CONTAINER_DIR}/${CONTAINER_FILE}

# source and target language
SRC_LANG=sl
TGT_LANG=en
BATCH=1024
DATA_VERSION=v2.0
BPE_SIZE=64000
SRC_TOKENIZER=${SRC_LANG}_tokenizer.$BPE_SIZE.BPE.model
TGT_TOKENIZER=${TGT_LANG}_tokenizer.$BPE_SIZE.BPE.model

# experiment name
EXPERIMENT_NAME=aayn_base_${SRC_LANG}-${TGT_LANG}_2303

# set dir
EXPERIMENT_DIR=${DATA_DIR}/results/${DATA_VERSION}/${EXPERIMENT_NAME}
mkdir -p ${EXPERIMENT_DIR}/${version}

# set run name
if [ "${version}" == "${DATETIME}" ]; then
  RUN_NAME=${version}
else
  RUN_NAME=${version}_R${DATETIME}
fi

# backup this script
cp -rp ${SBATCH} ${EXPERIMENT_DIR}/${RUN_NAME}.sbatch

# execution script the script
SCRIPT=${EXPERIMENT_DIR}/${RUN_NAME}.sh
touch $SCRIPT
chmod a+x $SCRIPT

IS_DISTRIBUTED=$([ 1 -lt $SLURM_JOB_NUM_NODES ] && echo " distributed over $SLURM_JOB_NUM_NODES nodes" || echo " on 1 node")

DEBUG="""
# print debug info
nvidia-smi --version | sed -e \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
nvidia-smi -L | sed -e \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
python -c 'import nemo; print(f\"nemo: {nemo.__version__}\")' | sed -e \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
python -c 'import torch; print(f\"torch: {torch.__version__}\")' | sed -e \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
python -c 'import torch; print(torch.__config__.parallel_info())' | sed \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
python -m torch.utils.collect_env | sed \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
env | grep SLURM | sort | sed \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
env | grep -v SLURM | sort | sed \"s/^/# \${SLURMD_NODENAME}.\${SLURM_PROCID}>   /g\"
"""

# setup the command
echo -e """
...
""" >>$SCRIPT

# run the script
srun --output=${EXPERIMENT_DIR}/${RUN_NAME}.%N.%J.txt \
  singularity exec \
    -H ${HOME}/tmp \
    -B ${DATA_DIR}/results/${DATA_VERSION}:/experiments,${DATA_DIR}/data/${DATA_VERSION}:/data \
    --pwd /workspace/nemo/ \
    ${CONTAINER_IMAGE} \
    /experiments/${EXPERIMENT_NAME}/${RUN_NAME}.sh
```



```
# setup the command
echo -e """
#!/bin/bash

# using '$(basename $SBATCH)', running $SLURM_NPROCS tasks$IS_DISTRIBUTED
# starting at \$(date)
# running process \$SLURM_PROCID on \$SLURMD_NODENAME


# export errors
export HYDRA_FULL_ERROR=1
export PYTHONUNBUFFERED=1
export PYTHONFAULTHANDLER=1

# debug NCCL
export TORCH_CPP_LOG_LEVEL=INFO
export TORCH_DISTRIBUTED_DEBUG=INFO
#export NCCL_DEBUG=INFO
#export NCCL_DEBUG_SUBSYS=ALL

# fix NCCL without IB
#export NCCL_IB_DISABLE=1
#export NCCL_IBEXT_DISABLE=1
#export NCCL_SOCKET_IFNAME=en

$(echo -e "$DEBUG")

# setup W&B logging
export WANDB_API_KEY=...

# train command
python examples/nlp/machine_translation/enc_dec_nmt.py \\
  --config-path=conf \\
  --config-name=aayn_base \\
  do_testing=false \\
  do_training=true \\
  trainer.devices=-1 \\
  trainer.accelerator=gpu \\
  trainer.num_nodes=$SLURM_JOB_NUM_NODES \\
  trainer.max_epochs=1000 \\
  trainer.benchmark=false \\
  trainer.precision=bf16 \\
  +trainer.val_check_interval=2000 \\
  +trainer.accumulate_grad_batches=1 \\
  +trainer.gradient_clip_val=0.0 \\
  trainer.log_every_n_steps=10 \\
  +trainer.num_sanity_val_steps=0 \\
  +trainer.sync_batchnorm=true \\
  exp_manager.name=${EXPERIMENT_NAME} \\
  +exp_manager.exp_dir=/experiments \\
  +exp_manager.version=${version} \\
  +exp_manager.resume_if_exists=true \\
  +exp_manager.resume_ignore_no_checkpoint=true \\
  +exp_manager.create_checkpoint_callback=true \\
  +exp_manager.checkpoint_callback_params.monitor=val_sacreBLEU \\
  +exp_manager.checkpoint_callback_params.mode=max \\
  +exp_manager.checkpoint_callback_params.save_top_k=10 \\
  +exp_manager.checkpoint_callback_params.always_save_nemo=true \\
  +exp_manager.checkpoint_callback_params.save_best_model=true \\
  +exp_manager.create_wandb_logger=true \\
  +exp_manager.wandb_logger_kwargs.name=${EXPERIMENT_NAME} \\
  model.max_generation_delta=6 \\
  model.preproc_out_dir=/data/${SRC_LANG}-${TGT_LANG} \\
  model.train_ds.src_file_name=/data/train.$SRC_LANG \\
  model.train_ds.tgt_file_name=/data/train.$TGT_LANG \\
  model.train_ds.use_tarred_dataset=true \\
  model.train_ds.pin_memory=true \\
  model.train_ds.tokens_in_batch=$BATCH \\
  model.validation_ds.src_file_name=/data/validation.$SRC_LANG \\
  model.validation_ds.tgt_file_name=/data/validation.$TGT_LANG \\
  model.validation_ds.pin_memory=true \\
  model.validation_ds.tokens_in_batch=$BATCH \\
  model.test_ds.src_file_name=/data/test.$SRC_LANG \\
  model.test_ds.tgt_file_name=/data/test.$TGT_LANG \\
  model.test_ds.pin_memory=true \\
  model.test_ds.tokens_in_batch=$BATCH \\
  model.encoder_tokenizer.tokenizer_model=/data/tokenizer/${SRC_TOKENIZER} \\
  model.decoder_tokenizer.tokenizer_model=/data/tokenizer/${TGT_TOKENIZER} \\
  model.src_language=$SRC_LANG \\
  model.tgt_language=$TGT_LANG \\
  model.encoder.hidden_size=1024 \\
  model.encoder.num_layers=24 \\
  model.encoder.inner_size=4096 \\
  model.encoder.num_attention_heads=16 \\
  model.encoder.pre_ln=true \\
  model.decoder.hidden_size=1024 \\
  model.decoder.num_layers=6 \\
  model.decoder.inner_size=4096 \\
  model.decoder.num_attention_heads=16 \\
  model.decoder.pre_ln=true \\
  model.optim.name=adamw \\
  model.optim.lr=2.0 \\
  model.optim.sched.name=NoamAnnealing \\
  +model.optim.sched.d_model=256 \\
  +model.optim.sched.warmup_steps=20000 \\
  model.optim.sched.warmup_ratio=null \\
  model.optim.sched.min_lr=1e-4 \\

echo \"# completed at \$(date)\"
""" >>$SCRIPT
```

NeMo 2.0 Developer Experience

Unified Interface for Building Generative AI models across modalities

Pythonic APIs

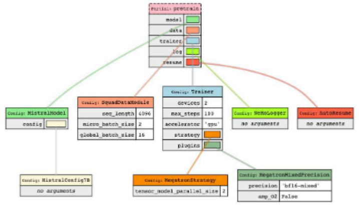
22X

Reduction in lines of code - From managing independent yaml files to Pythonic APIs

Manage Experiments Locally

Execute, manage and debug from local machine on any compute infrastructure – be it Slurm/K8 or any CSP – with a unified developer experience

Modular and Extensible



Partial: pretrain
model
data
trainer
log
resume

Config: MistralModel
config

Config: SquadDataModule
seq_length 4096
micro_batch_size 2
global_batch_size 16

Config: Trainer
devices 2
max_steps 100
accelerator 'gpu'
strategy
plugins

Config: NeMoLogger
no arguments

Config: AutoResume
no arguments

Config: MistralConfig7B
no arguments

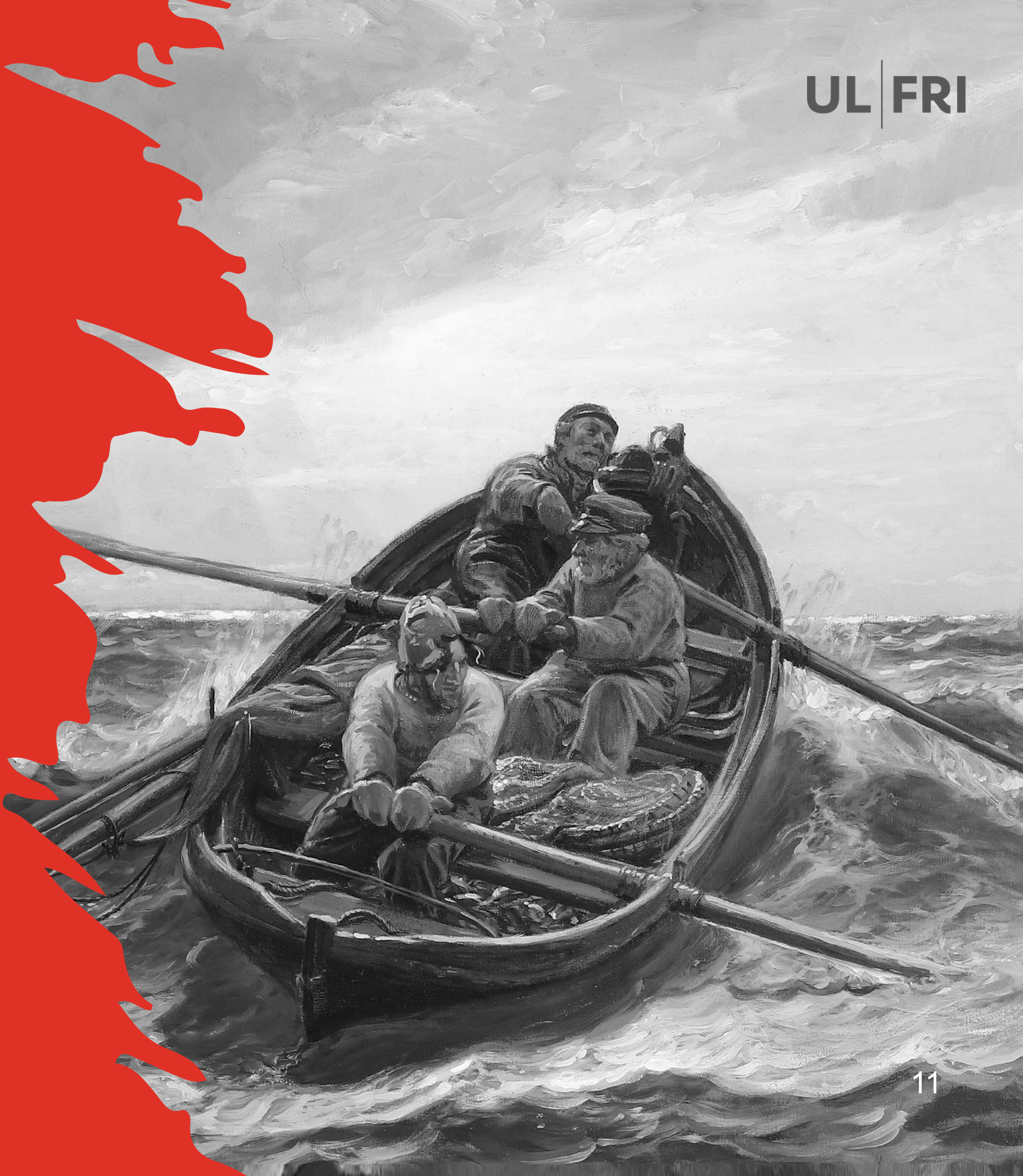
Config: MegatronStrategy
tensor_model_parallel_size 2

Config: MegatronMixedPrecision
precision 'bf16-mixed'
amp_O2 False

Easily change or modify components in a Pythonic environment

https://docs.nvidia.com/nemo-framework

War Stories?



Be wary of `--gpus-per-task=<no>`.*

https://github.com/NVIDIA/pyxis/issues/73, https://github.com/NVIDIA/nccl/issues/1066

Do not assume order of task startup.

Random crashes due to flaky nodes.

Be wary of library settings interdependency.

**https://github.com/numba/numba/issues/9387*

Know your toolkit(s)
Know your hardware
Be kind to sysadmins*

*and maintainers

ilb@fri.uni-lj.si