Dockerized edinmt decoder pipeline

Build the docker image

The source code comes with a simple command for building the docker image.

```
make docker-build
```

Translate

The Docker image comes with a translation function that translates input folders on the command line. For this, you need to mount volumes in the docker container as shown in the examples below.

Run the Docker image

To run the translate command pass the following environment variables to docker using the `-e` flag (or using an `--env-file` as desired):

```
DEVICES=0,1      #GPUs to run the servers on (default uses CPUs)
MODE=accurate    #"fast" or "accurate"
TYPE=text        #"text", or "audio" for model optimised for speech
NBEST=0          #output n-best sentences (N is pre-set in models)
NBEST_WORDS=0    #output n-best tokens in each sentence position
QUERY=0          #use query guided machine translation
                  (only available in kk->en and ka->en)
FMT=json         #output format of "json", "marian", "text"
```

NOTE: This translate command passes any additional unrecognized arguments directly to the marian decoder (see https://marian-nmt.github.io/docs/cmd/marian-decoder/), which override any pre-set environment variables (e.g. `--mini-batch-words 200`, etc.)

To use the docker container to translate a folder, you will need to use the `-v` flag to mount the input and output directory into the container. The directories need to be located where docker has read/write permissions.

The translate command can then be run as follows:

```
docker run --gpus all --rm \
     -v <input_dir>:/mt/input_dir \
     -v <output_dir>:/mt/output_dir \
     -e <the desired flags from above>
     --name edinmt \
     scriptsmt/systems:v22.1.2 \
     translate <src_lang> <tgt_lang> /mt/input_dir /mt/output_dir
```

For example:

```
docker run --gpus all --rm \
     -v ~/input_dir:/mt/input_dir \
     -v ~/output_dir:/mt/output_dir \
     -e DEVICES=0,1,2,3 -e NBEST_WORDS=1 \
     --name edinmt \
     scriptsmt/systems:v22.1.2 \
     translate fa en /mt/input_dir /mt/output_dir
```

NOTE: In case the system is unable to select the desired MT model for you, which may happen in case of specialty or fine-tuned systems, please try to use the `--system` flag, e.g. `--system faen_tweets`.

Input/Output Format

The input to the command is a directory (possibly with subdirectories) containing all files to be translated. The output is a new directory with the same subdirectories and the same file names, but containing translations. The default usage will format the translation files in json-lines format. For example, with NBEST_WORDS=1, the output would look like this:

```
{
  "id": 4,
  "nbest_words": [
    {"w11": score11, "w12": score12, ..., "w1N": score1N},
    {"w21": score21, "w22": score22, ..., "w2N": score2N},
                ...
    {"wT1": scoreT1, "wT2": scoreT2, ..., "wTN": scoreTN}
  ],
  "translation": "The sentence translation."
}
```

- **id** -- don't use this value.
- **nbest_words** are aligned along the time dimension for each subword token **wi**.
- Each token **wi** has N best candidates **wij.**
- **score-ij** is a negative log probability.
- **translation** is untokenized, natural language.

NOTE: if `NBEST=1` for n-best sentences, the json-lines format places each n-best sentence on its own line.

For those systems that support query-guided translation, in addition to setting the QUERY=1 option, queries should be incorporated in the input files as a second tab-separated field. The query-guided systems can still translate without queries (e.g. in case some sentences don't require or are missing queries), but performance may be slightly lower than using a non-query-guided system without queries.

Finetune

The Docker container can be used to fine-tune one of our pre-trained models on your own training and validation data. To use this functionality, you need to mount volumes in the docker container which contain your train/valid data, and provide the `finetune` command with the filepath arguments, e.g.:

```
docker run --gpus all --rm \
    -v ~/data:/mt/data \
```

```
    -v ~/finetuned_dir:/mt/finetuned_dir \
    --name edinmt scriptsmt/systems:v25.0.0 \
    finetune fa en /mt/finetuned_dir \
        --train /mt/data/train.fa /mt/data/train.en \
        --valid /mt/data/valid.fa /mt/data/valid.en
```

NOTE: We use an internal train config for this command, but we do pass any additional unrecognized arguments directly to the [marian](https://marian-nmt.github.io/docs/cmd/marian/) afterwards, which override any pre-set environment variables and the internal config.

Translate with a fine-tuned model

The fine-tuned model can then be used to translate folders (same command as above) by changing the SYSTEMS_DIR environment variable to point to the directory where the model was created. The directory will need to be re-mounted using the docker `-v` flag, e.g.

```
docker run --gpus all --rm \
    -v ~/finetuned_dir:/mt/finetuned_dir \
    -v ~/input_dir:/mt/input_dir \
    -v ~/output_dir:/mt/output_dir \
    -e SYSTEMS_DIR=/mt/finetuned_dir \
    --name edinmt scriptsmt/systems:v25.0.0 \
    translate fa en /mt/input_dir /mt/output_dir
```

System Requirements

- CPU
- RAM
- GPU
- GPU-RAM
- Target CUDA version and/or minimum NVIDIA driver version (current Scripts servers have CUDA: 11.2 and Driver: 460.32.03)

Approach

For fine-tuning the fa-en system on Twitter data we used the following approach:

finetune the fa->en system
forward translate the training data
finetune the en->fa system
back-translate the training data
concat all data
finetune final fa->en system

+ pre/post-processing at each previous steps