# Dockerized umdnmt decoder pipeline

## Build the docker image

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

```
docker build -t umd-nmt:v8.2 -f Dockerfile .
```

## 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

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 \
     --name umdnmt \
     umd-nmt:v8.2 \
     translate <src_lang> <tgt_lang> <input-type> \
     <nbest-size> <gpu-ids> <jobs-per-gpu>
```

For example:

```
docker run --gpus all --rm \
     -v ~/input_dir:/mt/input_dir \
     -v ~/output_dir:/mt/output_dir \
     --name umdnmt \
     umd-nmt:v8.2 \
     translate fa en text 5 0 1
```

## 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 standard/stemmed translations and nbest-words in JSON format for each input file. Given an input file “file.txt”, it produces

- **file.txt:** plain translation file aligned with the input.
- **file.txt.nbest-words:** nbest-words lists for each line/translation. They are structured in json-lines format that looks 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.
- **file.txt.stem:** plain stemmed translation file aligned with the input. (only if tgt_lang=en)
- **file.txt.stem.nbest-words:** nbest-words lists for each stemmed translation. (only if tgt_lang=en)

## 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:

- Insert/replace placeholders for Twitter hashtags and handles as pre/post-processing
- finetune the fa->en system on Twitter data
- ensemble 4 models fine-tuned on Twitter data with different random seeds

# Dockerized umdsmt decoder pipeline

## Build the docker image

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

```
docker build -t umd-smt:v3.7.3 -f Dockerfile .
```

## 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

The translate command can then be run as follows:

```
docker run --rm \
     -v <input_dir>:/mt/input_dir \
     -v <output_dir>:/mt/output_dir \
     --name umdsmt \
     umd-smt:v3.7.3 \
     <src_lang> <tgt_lang> <num_threads>
```

For example:

```
docker run --rm \
     -v ~/input_dir:/mt/input_dir \
     -v ~/output_dir:/mt/output_dir \
     --name umdsmt \
     umd-smt:v3.7.3 \
     fa en text 1
```

## 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 standard/stemmed translations. Given an input file “file.txt”, it produces

- **file.txt:** plain translation file aligned with the input.
- **file.txt.input:** the preprocessed input.
- **file.txt.trans:** the translation file before post-processing.
- **file.txt.align:** word alignments between sentences in file.txt.input and file.txt.trans.
- **file.txt.stem:** plain stemmed translation file aligned with the input (only if tgt_lang=en).
- **file.txt.stem.input:** the preprocessed input for stemmed translation.
- **file.txt.stem.trans:** the stemmed translation before post-processing.
- **file.txt.stem.align:** word alignments between sentences in file.txt.stem.input and file.txt.stem.trans.

## System Requirements

- CPU
- RAM