Anleitung zur Verwendung des Programms

Die vorliegende Anleitung zur Verwendung des Programms ist auf einem `Ubuntu 18.04` System getestet.

Eine Liste der Argumente zu allen Skripten kann für jeden Skript mit dem Argument `--help` ausgegeben werden.

Skripte

Um die Ergebnisse aus dem Kapitel *Experimente* zu reproduzieren, sind im Verzeichnis `scripts` alle ausgeführten Trainingsskripte verfügbar.

Modelle

Die trainierten Modelle aus dem Kapitel *Experimente* befinden sich im Verzeichnis `results/trained_model`.

Umgebung einrichten

Um die Umgebung einzurichten, empfiehlt sich die Installation von einer virtuellen Umgebung für Python 3, zum Beispiel mit virtualenv:
`python3 -m venv env`

Umgebung aktivieren:
`source env/bin/activate`

Skript setup.sh starten, um die notwendigen Dependencies zu installieren: `bash setup.sh`

Datensätze vorverarbeiten

Der Europarl-Korpus wird von der Opus-Plattform heruntergeladen:
http://opus.nlpl.eu/Europarl.php

Insbesondere wird auf die Korpora aus der Matrix "Statistics and TMX/Moses Downloads" aus dem oberen rechten Dreieck (TMX-Dateien) zugegriffen.

Um den *Europarl*-Korpus herunterzuladen und vorzuverarbeiten, muss der Skript `preprocess.py` verwendet werden. Standardmäßig verarbeitet der Skript den Datensatz für die Sprachen *Englisch* und *Deutsch*.
Um eine andere Sprache auszuwählen, als Deutsch, kann der Parameter `--lang_code` verwendet werden, z. B. `python3 preprocess.py --lang_code it` (für Englisch/Italienisch). Die verfügbaren Sprachcodes können der Tabelle aus der OPUS-Website entnommen werden.

Rohe Dateien werden im Verzeichnis `data/raw/europarl/<lang_code>` gespeichert. Vorverarbeitete Dateien werden im Verzeichnis `data/preprocessed/europarl/<lang_code>/splits/30/` gespeichert. Im Verzeichnis `data/preprocessed/europarl/de/splits/30/` sind die Dateien für Deutsch bereits gespeichert. Ein Training ist auch ohne Vorverarbeitung möglich.

Modell trainieren

Um ein Modell zu trainieren, muss der Skript `train_model.py` verwendet werden. Der Skript akzeptiert zahlreiche Argumente, unter anderem folgende:

Training Preprocessing	**Training**	**Modellkonfiguration**
`--train, --val, --test`: Einstellung der Datenmenge für das Training, die Validierung und das Testen	`--lr`: Einstellung der Lernrate. Standardmäßig: 0,0002	`--hs`: Einstellung der RNN-Dimension (hidden size), Standardmäßig: 300
`--max_len`: Einstellung einer maximalen Länge, auf die die Sätze abgeschnitten werden. Standardmäßig: 30	--epochs: Epochen. Standardmäßig: 80	`--emb`: Einstellung der Embedding-Dimension. Standardmäßig: 300
`--data_dir`: Dateinverzeichnis, falls die Dateien nicht vom standardmäßigen Verzeichnis geladen werden sollen. Standardmäßig: `None (data/preprocessed/europarl/de/splits/30/)`	`--cuda`: Training auf GPU. Standardmäßig: `True`. Das Programm überprüft trotzdem nach der Verfügbarkeit von CUDA.	`--num_layers`: Einstellung der Schichten. Standardmäßig: 2 (Encoder bidirektional, d.h. Decoder erhält 4 Schichten)
`--tok: Infix der tokenisierten Dateien. tok weist auf eine standardmäßige Zerlegung (auf Wortebene). Standardmäßig: "tok"`	`--b`: Stellt die Batchgröße ein. Standardmäßig: `64`.	`--dp`: Dropout. Standardmäßig: `0,25`
`--lang_code`: Die zweite Sprache neben Englisch, auf dem das Modell trainiert wird. Standardmäßig: `"de"`. Achtung: Voraussetzung ist, dass die entsprechenden Dateien bereits tokenisiert wurden!	`--norm`: Stellt einen Schwellenwert für das Klippen der Gradienten. Standardmäßig: `-1.0`, d.h. Die Gradientennorm wird auf den Schwellenwert 1.0 geklippt, keine Ausgabe am Bildschirm	`--attn`: Stellt Attention ein. Mögliche Werte: "dot" und "none". Standardmäßig: "dot".
`--v`: Vocabulary-Größe.	`--reverse`: Wenn True,	--tied: Stellt Weight Tying

Standardmäßig: 30000	trainiert das Modell in der Sprachkombination `<lang_code> → en`. Wenn False, trainiert das Modell in der Sprachkombination `en → <lang_code>`. Standardmäßig: `True` (Training de > en)	ein. Standardmäßig: True
`--min`: Minimale Häufigkeit. Standardmäßig: 5		--beam: Stellt die Beam-Tiefe ein. Standardmäßig: 5
`--corpus`: Korpus, auf dem das Modell trainiert werden muss. Standardmäßig: “europarl”. Wenn die leere Zeichenkette übergeben wird, dann wird das Modell auf dem IWSLT-Datensatz aus Torchtext trainiert.		`--reverse_input`: Wenn True, trainiert das Modell mit umgekehrten Eingaben. Standardmäßig: `False`, da Encoder standardmäßig bidirektional
		`--bi`: Bidirektional. Wenn True, dann trainiert das Modell mit bidirektionalem Encoder. Standardmäßig: `True`

Beispielaufruf für die besten Attention basierten Modelle:

```
GRU:
python3 train_model.py --hs 300 --emb 300 --num_layers 2 --dp 0.25 --reverse_input
False --bi True --reverse True --epochs 80 --v 30000 --b 64 --train 170000 --val
1020 --test 1190  --lr 0.0002 --tok tok --tied True --rnn gru --beam 5 --attn dot

LSTM:
python3 train_model.py --hs 300 --emb 300 --num_layers 2 --dp 0.25 --reverse_input
False --bi True --reverse True --epochs 80 --v 30000 --b 64 --train 170000 --val
1020 --test 1190  --lr 0.0002 --tok tok --tied True --rnn lstm --beam 5--attn dot
```

Übersetzen

Um zu **übersetzen**, kann der Skript `translate.py` verwendet werden. Dazu muss der Parameter `--path` übergeben werden mit dem Pfad zu einem trainierten Modell. Beispielaufruf aus dem Root-Verzeichnis:

```
python translate.py --path
results/trained_models/de_en/04_attention/80/lstm/2/bi/2019-08-11-
10-46-58
```

Während der Übersetzung kann die **Beam-Tiefe verändert** werden: `#10` wechselt zur Beam-Tiefe 10.

Wenn **aus einer Datei** übersetzt werden soll, dann muss dazu noch der Parameter `--file` übergeben werden mit dem Pfad zu einer Datei:

```
python translate.py --path
results/trained_models/de_en/04_attention/80/lstm/2/bi/2019-08-11-
10-46-58 --file
data/preprocessed/europarl/de/splits/30/translation.tok.de
```

Die standardmäßige **Beam-Tiefe** ist 5. Eine unterschiedliche Beam-Tiefe kann mit dem Argument `--beam` zu Beginn eingestellt werden. Bei der Übersetzung aus einer Datei bleibt die Größe fest.