Hochschule RheinMain
Prof. Dr. Adrian Ulges

22.05.2024

Künstliche Intelligenz (SoSe 2024)

Aufgabenblatt 5

zu bearbeiten bis: 05.06.2024

In dieser Übung werden wir für einen News Classifier auf Basis eines Transformers bauen. Wir verwenden den bekannten BERT-Transformer, und fine-tunen ihn auf unseren New York Times - Artikeln. Das resultierende Modell nennen wir *NYTimesBERT*.

Hinweise zur Orga: Am 29.05. hält Kollege Krechel KI, am 05.06. Kollege Ulges, dann ab dem 12.06. nur noch Kollege Krechel. Kollege Ulges nimmt dieses Blatt am 12.06. ab. Sollten Sie vorher Fragen haben, nutzen Sie gerne das Forum im read.mi.

Hinweise zur Technik: Sie können das Modell entweder auf dem eigenen Rechner fine-tunen (auf meinem Laptop benötigte ich ca. 15GB Hauptspeicher (was man ggfs. durch eine kleinere Batchsize reduzieren kann), und eine Trainingsepoche benötigte ca. 45 Minuten (3 Epochen sollten ausreichen)). Alternativ können Sie auf dem Server `megagpu(.local.cs.hs-rm.de)` *auf den GPUs 0 und 1 rechnen, wo eine Epoche mit GPU-Support 1 Minute benötigt. Eine Anleitung hierzu findet sich am Ende des Übungsblatts.*

Aufgabe 5.1 (Setup)
Laden Sie `nytimes-bert.zip` aus dem read.mi herunter. Sie finden dort ein richtiges kleines Projekt, das schon fast vollständig implementiert ist:

- `config.py`: Definition von Hyperparametern (immer sauber von der Implementierung trennen!).
- `dataset.py`: Hier werden Artikel gelesen und zu Batches von Tokens vorverarbeitet.
- `model.py`: Definition von Modell und Tokenizer.
- `train.py`: Code zum Training des Modells.
- `test.py`: Code zum Testen des Modells.

Das Projekt verwendet die Library pytorch-lightning, so dass viel Code (z.B. die Trainingsschleife) nicht mehr selbst geschrieben werden muss. Als Daten verwenden wir `train.json`+`test.json` (siehe Naive Bayes).

Setzen Sie zunächst Ihr Python-Environment auf (sollten Sie die Server-Umgebung verwenden, finden sich andere Installationsanweisungen am Ende des Übungsblatts):

```
python -m venv myenv              # lege ein sog. 'virtual environment' an.

source myenv/bin/activate         # aktiviere das Environment (in MacOS/Linux)
myenv\Scripts\activate            # aktiviere das Environment (in Windows)

pip install -r requirements.txt  # installiere die benoetigten Python-Pakete.
```

Listing 1: Projekt-Setup

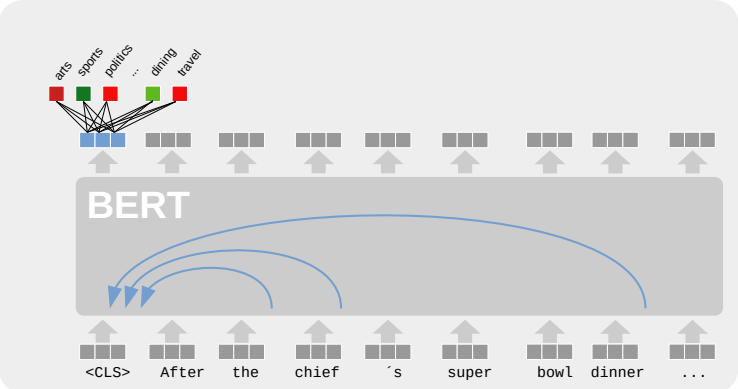
Führen Sie dann train.py aus.

```
python train.py data/train.json data/test.json
```

Listing 2: Projekt-Setup

Das Skript sollte Sie nun nach einem *W&B (weights&biases)-Account* fragen. Weights&biases ist ein Tool, welches Ihnen erlaubt, den Fortschritt Ihres Trainings unter *wandb.ai* zu tracken. Erstellen Sie sich dort einen Account.
Nachdem Sie den Account eingegeben haben, sollten Sie einen `RaiseNotImplementedError()` erhalten, da der Code noch nicht vollständig ist.

Aufgabe 5.2 (Implementierung)



Unser Ziel ist es, in `model.py` die obige Modell-Architektur aufzubauen: Wir verwenden den vortrainierten BERT-Transformer (`ID='bert-base-cased'`). Dieser liefert uns für eine Sequenz von n Tokens n Embeddings. Das vorderste Embedding (das sogenannte CLS-Embedding) füttern wir in eine voll verbundene Schicht ("linear layer") mit 8 Ausgabe-Neuronen, die für die 8 Klassen stehen.

- Implementieren Sie den fehlenden Code im Konstruktor und in `forward()`.
- Implementieren Sie dann die Methoden `training_step()` und `validation_step()`, die – gegebenenfalls einen Batch von Rezepten – den Forward Pass aufrufen und die 8 Aktivierungen der Ausgabe-Neuronen zusammen mit den Labels in einen sogenannten CrossEntropyLoss füttern.
- Sie sollten nun `train.py` aufrufen können. Beachten Sie, dass dies Ende jeder Epoche ein Modell unter `checkpoints/` speichert (jedes ist 1,3 GB groß). Führen Sie das Training für 3 Epochen aus, und beobachten Sie unter `wandb.ai` die Loss-Kurven. [Deliverable: Loss-Kurven].

Aufgabe 5.3 (Evaluation)
Ergänzen Sie zuletzt den fehlenden Code in `test.py` und rufen Sie die Evaluation auf:

```
python test.py checkpoints/2.cpt data/test.json
```

Listing 3: Projekt-Setup

Berechnen Sie do die Genauigkeit Ihres Classifiers: Funktioniert der Transformer besser als das Naive Bayes - Modell? [Deliverable] Messung der Genauigkeit.

Aufgabe 5.4 (Optional: Arbeiten auf dem GPU-Server)
Wenn Sie möchten, können Sie über ssh auf `megagpu(.local.cs.hs-rm.de)` zugreifen (von zu Hause aus müssen Sie unser Informatik-OpenVPN verwenden).
Sie können sich hier eine eigene Sandkiste zum Arbeiten anlegen:

```
/data/stud/IHRE_KENNUNG
```

Um Pip-Pakete zu installieren, erstellen Sie (siehe oben) Ihre eigene virtuelle Umgebung, in der Sie Pip ohne Root-Rechte aufrufen können:

```
python3 -m venv my_env; source my_env/bin/activate
```

Der Server bietet acht Grafikkarten mit jeweils 48 GB. Wir haben die **Karten Nr. 0 und 1** für diesen Kurs reserviert. Wählen Sie sie aus, wenn Sie Ihr Trainingsskript aufrufen, z.B.

```
CUDA_VISIBLE_DEVICES=0 python train.py ...
```

Bitte respektieren Sie die Rechenbedürfnisse anderer und verwenden Sie (außer für kleine kurzfristige Tests) nur diese beiden Karten. Um die Auslastung der GPUs zu überprüfen, verwenden Sie den Shell-Befehl:

```
nvidia-smi
```