MT Übung 4

Thema: RNNs

Abgabetermin: Montag, 07. Mai 2018, 18 Uhr

Hinweise zur Abgabe:

- **Abgabeformat: Email**
- Für diese Übung werdet ihr per Email einen Link zu eurem GitHub Repository an chantal.amrhein@uzh.ch schicken. Auf OLAT müsst ihr nichts abgeben.
- Bitte gebt pünktlich ab. Emails, die nach Montag, 18 Uhr eintreffen, werden nicht berücksichtigt. Ebenso werden Git Commits, die nach 18 Uhr erfolgt sind, nicht berücksichtigt.

1 GitHub Account aufsetzen und Repository forken

Für diese Übung müsst ihr mit einem Repository auf GitHub arbeiten und dieses auch verändern. Dazu benötigt ihr einen GitHub Account. Wer noch keinen besitzt, kann hier einen erstellen.

Als Nächstes müsst ihr das `romanesco` Repository, welches ihr schon in der Vorlesung kennengelernt habt, zu euch forken. Öffnet dazu das Repository hier und klickt oben rechts auf `Fork`. Wenn ihr nun auf euer eigenes Profil geht, sollte das geforkte Repository unter `Repositories` erscheinen.

Jetzt sollt ihr euch auf dem Google Cloud Server anmelden, was ihr schon in der Vorlesung kennen gelernt habt. Clont euer geforktes Repository in euren home-Ordner. Verwendet dazu die folgenden Befehle (achtet euch darauf, dass die URL mit eurem Repository stimmt):

```
git clone https://github.com/your_username/romanesco
```

```
cd romanesco
```

```
pip install --user -e .
```

Für diese Übung müsst ihr eigentlich nur wissen, wie man das lokale Repository aktualisieren kann:

```
git pull origin master
```

Und wie man lokale Veränderungen in das remote Repository kopieren kann. Für Letzteres müsst ihr zuerst überprüfen, welche Dateien ihr verändert habt:

```
git status
```

Danach müsst ihr jede Datei, die ihr kopieren wollt, hinzufügen mit:

```
git add file_name
```

Jetzt müsst ihr die Veränderungen bestätigen und eine kurze Nachricht hinzufügen:

```
git commit -m "fix some error"
```

Zum Schluss kopieren wir jetzt die bestätigten Veränderungen in das remote Repository:

```
git push -u origin master
```

Falls ihr das zuerst ein wenig üben möchtet, gibt es hier ein gutes Tutorial dazu. Wir werden im Tutorat am Montag eine kurze Einführung zur Arbeit mit Git machen. Falls ihr davor schon Fragen habt, die Google nicht beantworten kann, postet bitte einen Beitrag ins OLAT-Forum.

gpustat

2 Tensorboard

In der Vorlesung habt ihr auch schon kurz angeschaut, was es mit Tensorboard auf sich hat. Für euer eigenes Training, welches ihr in dieser Übung durchführen werdet, sollt ihr Tensorboard auch benutzen. Dafür müsst ihr beim Einloggen auf dem Server einen bestimmten Port weiterleiten. Loggt euch folgendermassen auf dem Google Cloud Server ein:

```
ssh -L 16006:127.0.0.1:6006 your_username@35.196.130.179
```

Dabei wird nun Port 6006 auf dem Server auf euren lokalen Port 16006 umgeleitet. Nun könnt ihr Tensorboard unter `http://localhost:16006/` aufrufen und euer Training überwachen.

Text, Tect, Text.

Welches format?

Opus am einfachsten ein Satz pro Zeile

Sentence splitter.

Datenset gut anschauen

Weiteres Preprocessing. Mischen

3 Datenset

Gutenberg Wortebene, vs Zeichenebene

In dieser Übung werdet ihr selbst ein RNN auf Sprachdaten trainieren. Eure Aufgabe ist es, ein interessantes Datenset zu finden und damit zu experimentieren. Sucht nach geeigneten Datensets und wählt eines davon aus. Eine kleine Auswahl findet ihr in diesem Blog. Vielleicht findet ihr auch hier ein gutes Datenset, aber es sind längst nicht alle für diese Aufgabe geeignet. Die einzigen Vorgaben sind: Das Datenset muss aus Sprachdaten bestehen (keine Sequenzen von Zahlen) und sollte nicht kleiner als 1 MB Text sein. Überprüft, ob euer Datenset vielleicht noch Preprocessing benötigt (z.B. Transformation nach Plain Text).

Teilt nun euer Datenset in ein Trainings- und ein Dev-Set auf, wobei das Dev-Set etwa 10% der gesamten Daten ausmachen soll. Vergesst nicht das Datenset zuvor zu durchmischen. Das könnt ihr mit folgendem Befehl tun:

```
shuf some_file.txt > some_file.shuf
```

Überprüft, wie viele Zeilen eure Datei hat.

```
wc -l some_file.shuf
```

Berechnet dann, wie viel 10% sind, schneidet so viele Zeilen ab für ein Dev-Set und benutzt den Rest als Trainings-Set. Wenn 10% z.B. 7 Zeilen wären, führt die folgenden Befehle aus:

```
head -n 7 some_file.shuf > some_file.dev
```

```
sed "1,7d" some_file.shuf > some_file.train
```

4 Adaption

Für diese Übung sollt ihr den `romanesco` Code selbst weiterentwickeln und versuchen, ihn zu verbessern. Das kann eine ganz einfache Änderung im Preprocessing sein, aber auch ganz komplexe Anpassungen in der Architektur des RNNs sind erlaubt. Was und wie viel ihr verändert, ist euch überlassen. Wir möchten euch ermutigen, etwas Spannendes auszuprobieren, aber es gibt keinen Abzug, wenn ihr nur eine kleine Veränderung macht. Uns ist einfach wichtig, dass all eure Adaptionen motiviert sind (z.B. durch euer Datenset) und ihr sie dementsprechend begründen könnt. Falls ihr gar keine Idee habt, fragt am besten im Tutorat nach.

Wenn ihr mit der Implementation fertig seid, kopiert eure lokalen Veränderungen auf das remote Repository wie oben beschrieben.

Code ändern -> verbessern.

Preprocessing, Output

Zuerst Trainieren, danach nochmal?

5 Training

Nun sollt ihr mit dem von euch angepassten Code und euren Trainingsdaten ein Sprachmodell trainieren. Eine Anleitung dazu findet ihr in der `README.md` Datei im Repository. Fangt frühzeitig mit dem Training an, da kurz vor der Abgabefrist sicherlich viel mehr GPUs besetzt sein werden. Überprüft mit `gpustat` welche GPUs frei sind und startet euer Training dementsprechend auf einer von den acht GPUs.

6 Scoring

Wenn das Modell fertig trainiert ist, sollt ihr es dazu verwenden, um die Perplexität auf dem Dev-Set zu berechnen. Die Anleitung findet ihr wieder in der `README.md` Datei im Repository.

7 Sampling

Zum Schluss sollt ihr euer Modell noch dazu benutzen, neuen Text zu generieren. Speichert den erzeugten Text im remote Repository. Die Anleitung findet ihr wieder in der `README.md` Datei im Repository.

8 Modell-Schema

Zeichnen!

Visualisiert in einer Zeichnung, wie euer Sprachmodell in eurem Verständnis aussieht. Wenn euer Modell 1000 Neuronen pro Schicht habt, müsst ihr natürlich nicht alle aufzeichnen. Stellt schrittweise dar, wie eure Daten rein kommen, was im Preprocessing passiert und so weiter. Bitte beschriftet die einzelnen Teile klar, ansonsten sind eurer Kreativität keine Grenzen gesetzt. Auch hier gibt es keine Punktabzüge für krumme Linien oder Kaffeeflecken auf dem Papier ;)

Abgabe für Übung 4: Schickt den Link zu eurem remote Repository per Email. Das remote Repository soll Folgendes beinhalten:

- Den vollständigen Code (mit all euren Veränderungen), den ihr für das Training verwendet habt plus eventuelle externe Preprocessing-Skripte, die ihr selbst geschrieben habt. **(4 Punkte)**
- Einen kurzen Bericht (ca. 1 Seite), der beschreibt, welches Datenset ihr verwendet habt, wieso ihr euch dafür entschieden habt und woraus euer Preprocessing bestand. Weiterhin soll der Bericht eine genaue Auflistung beinhalten, welche Hyperparameter ihr für das Training benutzt habt (z.B. Vokabulargrösse) und jeweils eine Begründung dazu. Der Bericht soll auch im Detail beschreiben, was ihr euch bei euren Code-Veränderungen überlegt habt, wo ihr dabei Probleme hattet und wie ihr sie gelöst habt. Zum Schluss sollt ihr im Bericht noch die Perplexität angeben, die ihr beim Scoring auf dem Dev-Set berechnet habt. **(1 Punkte)**

Text
- Den erzeugten Text als .txt Datei, einen Screenshot von der Loss-Übersicht auf Tensorboard und euer gezeichnetes Schema als Bild-Datei. **(1 Punkte)**

-h zeigt alle gmöglichen Optionen an