Maschinelle Übersetzung - Übung 4

Martina Stüssi 14-820-195 & Valentina Vogel 16-708-919

Bericht zu RNN Romanesco

Datenset

In diesem Spreadsheet haben wir das Datenset Happy Moments gefunden. Das uns inhaltlich einerseits interessiert hat und andererseits für unser Projekt tauglich erschien. Wir hatten auch die Hoffnung, dass besonders interessante neue Sätze generiert würden.
100'000 Happy moments, der letzten 24 Stunden: https://github.com/rit-public/HappyDB

Wir haben damit begonnen einfach einmal zu trainieren, um ein Gefühl für die Vorgänge zu bekommen. Zuerst noch mit dem winzigen Trumptext aus einer vorherigen Übung, dann mit unserem Happy Moments Korpus (zuerst sehr grobes Preprocessing und unverändertem romanesco Code)

10 Epochen, 10'000 Zeilen, Perplexity auf Trainingsdaten: 131.06, Perplexity auf Testdaten: 106.90 (vielleicht habe ich das falsch aufgeschrieben, bessere Resultate auf den Testdaten erscheint mir sehr unglaubwürdig)

20 Epochen, 10'000 Zeilen, P auf Trainingsdaten: 56.07, P auf Testdaten: 83.23 -> Overfitting
10 Epochen, 50'000 Zeilen, P auf Trainingsdaten: 44.02, P auf Testdaten: 56.09
10 Epochen auf ca. 100'000 Zeilen, P auf Trainingsdaten: 34.37, P auf Testdaten: 44.07

Dann haben wir versucht, ein Dropout Layer zu implementieren, um Overfitting zu verhindern. Training auf der Ganzen Datenbank (happy_moments_def.txt)

15 Epochen, Kein Dropout, P auf Trainingsdaten: 27.15, P auf Testdaten: 42.15
15 Epochen, 0.3 Dropout, P auf Trainingsdaten: 30.21, P auf Testdaten: 43.99

Wir waren sehr erstaunt darüber, dass die Unterschiede so gering sind. Zwar wurde durch das Dropoutlayer die Perplexität auf den Trainingsdaten deutlich weniger tief (anzunehmen, da weniger Neurone zur Verfügung stehen). Dies hat aber erstaunlicherweise nicht dazu geführt, dass die Resultate auf den Testdaten besser wären. Wir sind unsicher, ob wir das Dropoutlayer korrekt implementiert haben bzw. ob man es irgendwo wieder “entaktivieren” müsste (bswp. bei der Berechnung der Perplexität auf neuen Daten im score Code). Overfitting ist bei unserer grossen Datenmenge aber wohl nicht so ein Problem, wie wir zuerst erwartet haben.

15 Epochen, Kein Dropout, happy_quot.txt, P auf Trainingsdaten: 24.97, P auf Testdaten: 39.47

20 Epochen, Kein Dropout, happy_quot.txt, P auf Trainingsdaten: 19.74, P auf Testdaten: 39.44
20 Epochen, 0.3 Dropout, happy_quot.txt, P auf Trainingsdaten: 22.38, P auf Testdaten: 40.67

Insgesamt sind wir zur Überzeugung gekommen, dass es in unserem Fall sinnvoller ist, das Preprocessing zu optimieren, da wir mit TensorFlow noch nicht vertraut sind und auch die Hilfestellungen im Internet aus mangelndem Verständnis kaum nutzen konnten. Die rettende

Ausnahme ist dieser Artikel von Erik Hallström, der uns dazu befähigt hat, überhaupt etwas am Code zu verändern (https://medium.com/@erikhallstrm/using-the-dropout-api-in-tensorflow-2b2e6561dfeb). Wir sind nicht 100% sicher, ob wir das Dropout richtig implementiert haben, aber es sieht immerhin danach aus.

Im Nachhinein (bei genauerer Betrachtung des erstellten Vokabulars) sind wir zum Schluss gekommen, dass wir beim Preprocessing noch die Satzendpunkte besser hätten verarbeiten können, da diese im Vokabular immer noch an ihre ursprünglichen Wörter angehängt sind. Bei der ersten Übung haben wir aber bereits gelernt, dass “einfach abschneiden” in der Regel keine Lösung ist und hatten aus Zeitgründen leider keine Möglichkeit mehr, uns eine geeignete Umsetzung zu überlege.

Stolpersteine:
Code verstehen, den wir nicht selbst verfasst haben und der in TensorFlow geschrieben ist.
csv richtig einlesen (quotchar = "" beachten)

Unsere Vokabelänge umfasst 10001 Worte, wir haben erst beim Anschauen von vocab.json bemerkt, dass wir beim Preprocessing das Absplitten von Satzzeichen aussen vor gelassen haben was eine sehr einfache Massnahme im Preprocessing gewesen wäre.

Für das Training haben wir jeweils die unveränderten Hyperparameter des ursprünglichen Codes verwendet.

Einsichten aus dieser Übung.

Vor der git Nutzung etwas mehr nachdenken: es wäre angenehmer gewesen, wenn eine von uns romanesco geforked hätte und wir dann zwei Branches erstellt hätten. Ein besseres Zusammenarbeiten hätte resultieren können. Merge Probleme in git können verhindert werden indem immer zuerst gepullt und dann erst gepusht wird. Was ist Perplexität genau? Perplexität lässt sich irgendwie immer nur in einem direkten Vergleich von zwei Verfahren beurteilen daher ist es sinnvoll, immer nur ein Aspekt (Preprocessing, Datenmenge, Anzahl Epochen) zu ändern um zu sehen, ob das System besser wird. Tensorboard zu verstehen scheint eine eigene Wissenschaft. Glücklicherweise gibt es einige hilfreiche Blogposts.

Wie Prozess abbrechen, wenn mit nohup gestartet? -> nützliche Erkenntnis:
$ ps -ef # zeigt alle Prozesse an
$ kill 27404 # kann gebraucht werden um prozess abzubrechen