Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Зимен семестър 2021/2022

Домашно задание №3

16 януари 2022 г.

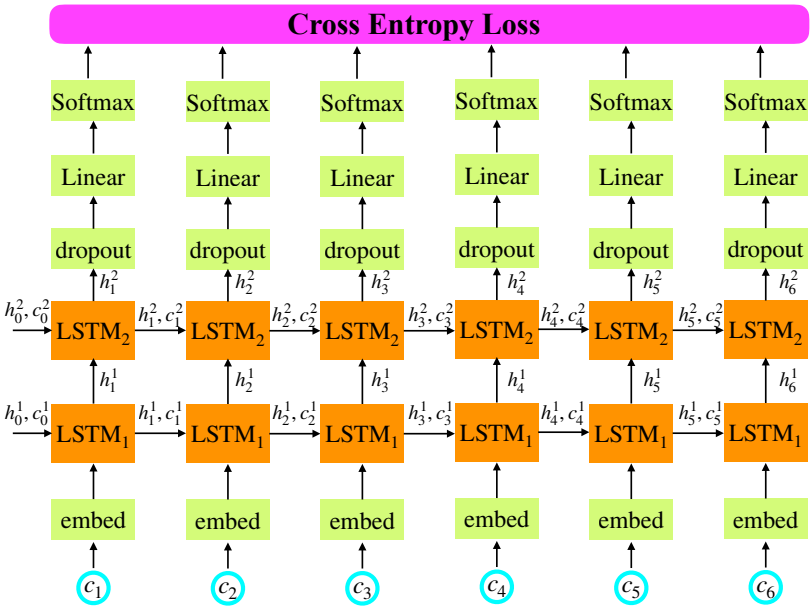
Общ преглед

В това задание ще имплементираме Дълбок невронен програмист, който автоматично ще генерира фрагменти програмен код на python[1]. За целта ще обучим програмиста използвайки 70000 програмни функции на python, с прилежащите им документиращи низове на английски език. Програмиста ще реализираме чрез използване на рекурентна невронна архитектура. По-конкретно, архитектурата е многослойна еднопосочна LSTM рекурентна невронна мрежа представена на фигура 1. Входът е последователност от влагания на символи, а след последното LSTM ниво, скрития вектор след dropout се проектира през линейна трансформация върху символите. Накрая, след softmax се получава разпределение за следващия символ в последователността.

Обучението ще извършим като минимизираме крос-ентропията на така съставения езиков модел на ниво символи.

Генерирането на нови програми се извършва по следната процедура: Започва се с произволен начален низ от символи, който следва да започва със символа за начало на последователност ‘ш’ (може да се състои и

[1]Генераторът на код, предмет на това задание, е сравнително примитивен. Генерираните от него програми нямат особен смисъл и вероятно дори не се изпълняват. В тази област има по-задълбочени подходи, които могат да генерират изпълним код със значително по-високо качество. Вижте например: Evaluating Large Language Models Trained on Code



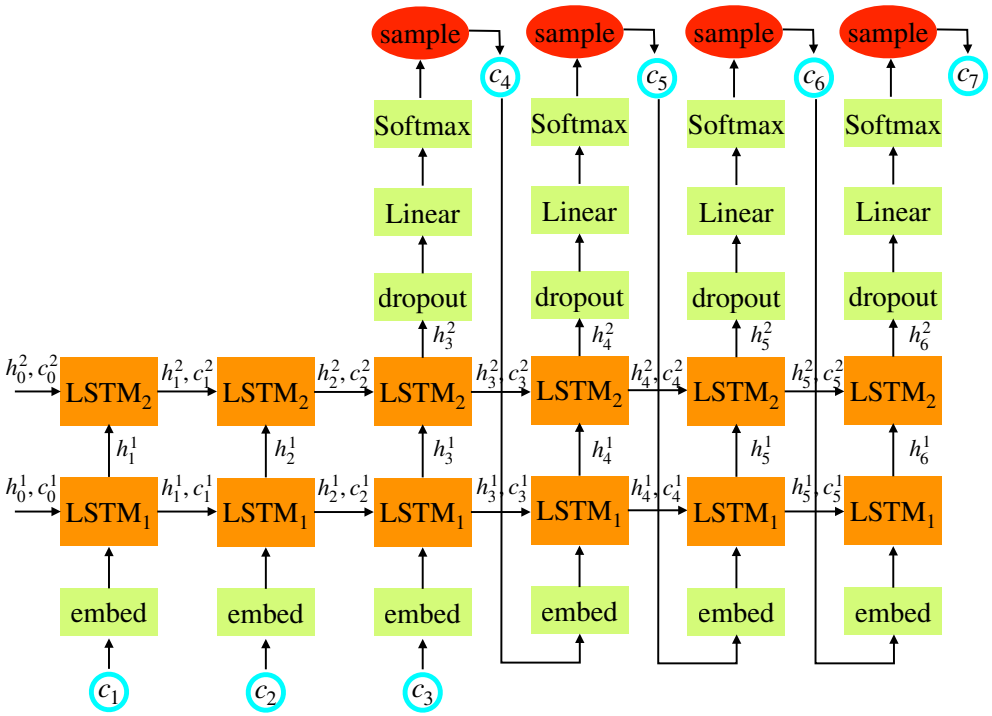
Фигура 1: Архитектура на рекурентната невронна мрежа, реализираща Дълбок невронен програмист.

само от този символ). С този начален низ се траверсира рекурентната невронна мрежа до получаването на съответната двойка от скрит вектор h и памет c. От h се получава разпределението p за следващия символ. Следващия символ се генерира случайно с разпределение p. Този символ се добавя към резултата и с него се захранва рекурентната невронна мрежа за получаване на следващата двойка от скрит вектор h и памет c. Тази процедура се повтаря до генерирането на символ за край на последователност или до достигане на даден лимит. На фигура 2 е представена схема на генератора на програми.[2]

Забележки

1. За това задание не се предоставя код за тестване. Препоръчва се вие сами да си направите тестови скриптове, с които да се уверите в коректността на програмите ви. Също така, в кода подсказките са силно ограничени и вие имате по-голяма свобода за структурирането на вашата програма.

[2]Подобни генератори, както и други такива с техни приложения можете да намерите в следния обзор: Machine Learning on Source Code.



Фигура 2: Схема на невронната мрежа за генериране на програми. В конкретния случай е представена двуслойна LSTM рекурентна невронна мрежа. На входа е зададен начален низ $c_1c_2c_3$. От скрития вектор h_3^2, след dropout, проектиране и softmax се получава разпределение, с което се семплира следващия символ c_4. Този символ се извежда и се подава като следващ символ на рекурентната невронна мрежа. Тази процедура се повтаря до извеждането на символ за край или достигането на зададен лимит.

2. Въпреки, че са дадени примерни стойности за метапараметрите като брой слоеве, размери на влагания, размер на скритите вектори, dropout, размер на партидата и т.н., вие имате пълната свобода да ги променяте, за да получите по-добри резултати. Това, което ще бъде оценено накрая, е доколко добре се справя вашата програма с поставената задача.

3. За обучението на модела може да ви бъде необходимо значително повече машинно време. Ако не разполагате с мощен компютър с графичен ускорител може да се възползвате от услугата Google Colab, където безплатно се предоставя изчислителна среда с инсталирани Python и Pytorch, която може да се конфигурира да използва графична карта (GPU).

Задача 1 (5 точки)

Задачата е да имплементирате езиковия модел на дълбокия невронен програмист, като допълните кода във файла `model.py`. За целта може да копирате части от кода от упражнение 13, който имплементира еднопосочен LSTM с пакетиране на партиди. Към вашия код е необходимо да добавите следните допълнения:

1. Възможност за задаване на повече нива на LSTM.

2. Допълнителен dropout слой след последния LSTM слой, преди линейната проекция.

След като реализирате езиковия модел следва да го обучите. В пакета със заданието е включен файла `corpusFunctions`, който съдържа 70000 програмни функции на python (заедно с документиращи низове). За произхода на този корпус вижте: A parallel corpus of Python functions and documentation strings for automated code documentation and code generation. Във файла `utils.py` са имплементирани функциите за подготовка на тренировачни данни. Вие първо трябва да подготвите данните като извикате:

```
python run.py prepare
```

След това тренирането става с извикването на програмата:

```
python run.py train
```

След завършване на обучението, програмата оценява перплексията на ниво символи на езиковия модел. Получената перплексия следва да е под 4.5, по възможност по-малка от 4. Желателно е да опитате да получите възможно най-добра стойност на перплексията, като променяте

стойностите на метапараметрите във файла `parameters.py`. Оценката ви ще зависи от получената перплексия.

Забележка: След успешна процедура по трениране, параметрите на модела се записват във файла `modelFileName = 'modelLSTM'`. Може да донатренирате вече запазен модел като извикате:

```
python run.py train modelLSTM
```

Перплексията на вече записан модел може отново да измерите с командата:

```
python run.py perplexity
```

Задача 2 (5 точки)

В тази задача трябва да реализирате процедура за генериране на програма. За целта трябва да попълните функцията `generateCode` във файла `generator.py`, като имплементирате процедурата, която е скицирана на фигура 2. Забележки:

1. По време на генерация за получаването на разпределението върху следващия символ ще добавим параметър "температура". Поконкретно вместо стандартното разпределението $p = \text{softmax}(\mathbf{z})$, където $\mathbf{z}$ е проекцията на последния скрит вектор върху символите, ще използваме разпределението $p_\tau = \text{softmax}(\mathbf{z}/\tau)$, където τ е параметър наречен температура. Варирането на този параметър в интервала $[0, 1]$ води до промяна на увереността на поета. По-високите стойности водят до по-разхвърляни (съдържащи повече синтактични грешки) програми. По-ниските стойности водят до по-регулярен код.

2. За семплирането на елемент с дадено разпределение може да използвате вградената в NumPy функция `numpy.random.choice`.

След като реализирате генератора може да го използвате за генериране на програми с командата:

```
python run.py generate
```

Към командата може да зададете начален низ и температура:

```
python run.py generate ш_Compute
python run.py generate ш  0.4
```

Инструкция за предаване на домашна работа

Изисква се в Moodle да бъде предаден архив FNXXX.zip (където XXX е вашият факултетен номер), който съдържа:

1. Файловете `model.py, parameters.py, generator.py` съдържащи нанесените от вас промени
2. Файлът `modelLSTM` получен след изпълнението на Задача 1
3. Текстът на някоя по-успешна програма (по ваш избор) на вашия програмист при празен начален низ.

Надяваме се, че ще се забавлявате.