Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Зимен семестър 2022/2023

Домашна задание №1

10.11.2022

Общ преглед

В това задание ще имплементираме вероятностен коректор на правописа, за да коригираме автоматично евентуалните правописни грешки в заявките. По-формално: Ако е дадена в изходен запис заявка r, която евентуално съдържа правописни грешки, целта ни е да намерим желаната заявка q, която максимизира вероятността $\Pr[q|r]$. Т.е. искаме да отгатнем заявката, която потребителят вероятно е искал да изпълни. Използвайки теоремата на Бейс, имаме, че:

$$\Pr[q|r] = \frac{\Pr[r|q]\ \Pr[q]}{\Pr[r]} \propto \Pr[r|q]\ \Pr[q].$$

Тъй като заявката ни е дадена и фиксирана, вероятността $\Pr[r]$ не е от значение. Трябва да намерим онази заявка, която максимизира $\Pr[r|q]\ \Pr[q]$. Имайки предвид горната формулировка, ще изградим вероятностен коректор на правописа, състоящ се от 4 части:

1. Езиков модел. Оценява вероятността за наблюдаването на заявката q на базата на биграмен езиков модел, което ни позволява да изчислим $\Pr[q]$.

2. Коригиращ модел. Ще оценим вероятността на елементарните правописни грешки, които могат да възникнат при изписването на заявката, което ни позволява да изчислим $\Pr[r|q]$. По-конкретно, в

тази част ще изчислим вероятността символите в дадена дума от заявката да бъдат изтрити по погрешка, вмъкнати, заместени, слети или разцепени.

3. Генератор на кандидати. По оригиналната заявка r, зададена от потребителя, се генерира множество от кандидати за дадената заявка.

4. Оценител. Комбинирайки 1, 2 и 3, ще намерим най-добрия кандидат за дадената заявка:

$$q = \arg\max_q \Pr[r|q]\ \Pr[q].$$

Забележка: За осигуряване на по-добра числова стабилност е желателно навсякъде вместо стойностите на вероятностите да се използват логаритъм от съответните вероятности. Тъй като логаритъмът е монотонно разстяща функция имаме, че $\arg\max p = \arg\max \log p$. Освен това $\log \prod_i p_i = \sum_i \log p_i$.

1 Разстояние на Левенщайн със сливания и разцепвания

Първата задача е да се имплементира функция, изчисляваща разстоянието на Левенщайн със сливания и разцепвания между две думи. Този вариант на Левенщайн разстоянието се използва често в системите за оптическо разпозваване на текстове (Optical Character Recognition), където се случва даден визуално по-широк символ да бъде объркан с два по-тесни (например ю $\to$ но, ж $\to$ хк, ф $\to$ ор и т.н.) или обратно – два символа да бъдат сбъркани с един по-широк. На първата лекция разгледахме дефиницията на Левенщайн разстояние и псевдокод на алгоритъм, който го изчислява. Разстоянието на Левенщайн със сливания и разцепвания е подобно на Левенщайн разстоянието, но като елементарни операции се допускат освен изтриване, вмъкване и субституции на символи, също и сливания на два съседни символа и разцепвания на символ в два символа. За да дефититпаве формално разстоянието на Левенщайн със сливания и разцепвания ще използваме следните означения: Ако $W \in \Sigma^*$ е низ:

- $|W| \in \mathbb{N}$ е дължината на низа,
- за $i = 1, 2, \ldots, |W|$ с $W_i \in \Sigma$ означаваме i-тия символ в низа,

- за $i, j = 1, 2, \dots, |W|$, където $i \leq j$ с $W_i^j \in \Sigma^*$ означаваме подниза $W_i W_{i+1} \dots W_j$.

Използвайки горните означения дефинираме разстоянието на Левенщайн със сливания и разцепвания между низовете $P, W \in \Sigma^*$ индуктивно:

$$d_L(P,W) = \min \begin{cases} 0 & \text{ако } |P| = |W| = 0 \\ d_L(P_1^{|P|-1}, W) + 1 & \text{ако } |P| > 0 \\ d_L(P, W_1^{|W|-1}) + 1 & \text{ако } |W| > 0 \\ d_L(P_1^{|P|-1}, W_1^{|W|-1}) + \delta_{P_{|P|} \neq W_{|W|}} & \text{ако } |P|, |W| > 0 \\ d_L(P_1^{|P|-2}, W_1^{|W|-1}) + 1 & \text{ако } |P| > 1, |W| > 0 \\ d_L(P_1^{|P|-1}, W_1^{|W|-2}) + 1 & \text{ако } |P| > 0, |W| > 1 \end{cases},$$

където $\delta_{a \neq b} = 1$, ако $a \neq b$, и $\delta_{a \neq b} = 0$ в противен случай.

Задачата е да попълните тялото на функцията `editDistance` в кода на програмата (2т.)

2 Коригиращ модел – тегло на редакция

Втората задача е свързана с намирането на теглото на редакцията за получаване от една дума на друга. Теглото на редакция съответства на минус логаритъм от вероятността т.е. $\omega(r, q) = -\log \Pr[r|q]$. В нашия модел ще предполагаме, че вероятността $\Pr[r|q]$ е произведение на вероятностите на елементарните (посимволови) редакции, които са необходими, за да получим от изписаната дума желаната. Формално дефинираме множеството от елементарни редакции Op над азбука от символи Σ като:

$$\mathrm{Op} = \mathrm{Id} \cup \mathrm{Ins} \cup \mathrm{Del} \cup \mathrm{Sub} \cup \mathrm{Merge} \cup \mathrm{Split},$$

където

$$\begin{aligned} \mathrm{Id} &= \{(\sigma, \sigma) | \sigma \in \Sigma\} \\ \mathrm{Ins} &= \{(\varepsilon, \sigma) | \sigma \in \Sigma\} \\ \mathrm{Del} &= \{(\sigma, \varepsilon) | \sigma \in \Sigma\} \\ \mathrm{Sub} &= \{(\sigma, \tau) | \sigma, \tau \in \Sigma, \sigma \neq \tau\} \\ \mathrm{Merge} &= \{(\sigma\tau, \eta) | \sigma, \tau, \eta \in \Sigma, \sigma \neq \eta, \tau \neq \eta\} \\ \mathrm{Split} &= \{(\eta, \sigma\tau) | \sigma, \tau, \eta \in \Sigma, \sigma \neq \eta, \tau \neq \eta\}. \end{aligned}$$

Ако $op = (x, y)$ то $op_1 = x, op_2 = y$. Ще предполагаме, че ни е дадена функция $\omega : \mathrm{Op} \to \mathbb{R}^+$, която на всяка елементарна операция ни съпоставя тегло, така че $\omega(op) = 0$ за $op \in \mathrm{Id}$. Ще предполагаме, че $\omega(op) = -\log \Pr[op]$. Последователността от елементарни операции $op^1, op^2, \ldots, op^k$ подравнява думата r с думата q, ако $r = op^1_1 op^2_1 \ldots op^k_1$ и $q = op^1_2 op^2_2 \ldots op^k_2$. Теглото на подравняването дефинираме като: $\omega(op^1, op^2, \ldots, op^k) = \sum_{i=1}^k \omega(op^i) = -\log \prod_{i=1}^k \Pr[op^i]$. В коригиращия модел ще моделираме вероятността за редакция при условие подравняването $op^1, op^2, \ldots, op^k$ като $\omega(op^1, op^2, \ldots, op^k) = -\log \Pr[r|q, op^1, op^2, \ldots, op^k] = -\log \prod_{i=1}^k \Pr[op^i]$

Дефинираме функцията Γ, която следва да връща минималното тегло на подравняване на думата P с думата W индуктивно:

$$\Gamma(P, W) = \min \begin{cases} 0 & \text{ако } |P| = |W| = 0 \\ \Gamma(P_1^{|P|-1}, W) + \omega(P_{|P|}, \varepsilon) & \text{ако } |P| > 0 \\ \Gamma(P, W_1^{|W|-1}) + \omega(\varepsilon, W_{|W|}) & \text{ако } |W| > 0 \\ \Gamma(P_1^{|P|-1}, W_1^{|W|-1}) + \omega(P_{|P|}, W_{|W|}) & \text{ако } |P|, |W| > 0 \\ \Gamma(P_1^{|P|-2}, W_1^{|W|-1}) + \omega(P_{|P|-1}^{|P|}, W_{|W|}) & \text{ако } |P| > 1, |W| > 0 \\ \Gamma(P_1^{|P|-1}, W_1^{|W|-2}) + \omega(P_{|P|}, W_{|W|-1}^{|W|}) & \text{ако } |P| > 0, |W| > 1 \end{cases}$$

Докажете, че $\Gamma(P, W)$ връща най-малкото тегло на подравняване на думата P с думата W (3 т.)

Попълнете тялото на функцията `editWeight` в кода на програмата, така че да имплементира функцията Γ (1 т.)

3 Генератор на кандидати

Ние ще търсим кандидатите за корекции на заявки на разстояние до 1 от r. За заявки r с нетривиална дължина, обаче, броят на кандидатите става огромен. Има различни подходи за ефективно генериране на кандидати, но предлагаме да се приложи следната идея: Започнете, като генерирате всички възможни редакции, които са на разстояние 1 от оригиналната заявка. Не забравяйте, че разглеждаме и тирета и интервали като символи. Това ще позволи да се разгледат някои сравнително често срещани грешки, например когато интервал е случайно вмъкнат в дума или две в заявката са залепени. Накрая ще изискваме всички думи в кандидат заявката да бъдат от речника.

Има, разбира се, други, много по-ефективни стратегии и много възможни разширения и вариации на стратегията, спомената тук.

Попълнете тялото на функцияте `generateEdits` в кода на програмата, така че по зададена оригинална заявка да се генерират всички заявки, които са на Левенщайн разстояние със сливания и разцепвания 1 до оригиналната и се състоят единствено от думи от речника на корпуса (2 т.)

4 Оценител

Работата на оценителя е да намери най-вероятната заявка q. Това се прави чрез комбиниране на вероятността от езиковия модел за $\Pr[q]$, и модела на вероятността за редактиране $\Pr[r|q]$. За получаването на кандидатите за q ще използваме генератора на кандидати. Формално, при дадена оригинална заявка r търсим:

$$q = \arg\max_{q_i} \Pr[q_i|r] = \arg\max_{q_i} \Pr[r|q_i]\ \Pr[q_i],$$

където максимумът е взет измежду всички кандидат заявки q_i произведени от кандидат генератора за оригиналната заявка r. Когато комбинираме вероятности от езиковия модел и модела за редактиране на вероятности, можем да използваме параметър μ за претегляне на двата модела по различен начин:

$$\Pr[q|r] \propto \Pr[r|q]\ \Pr[q]^{\mu}.$$

Може да експериментирате с различни стойности на μ за да видите кой ви дава най-добрата точност на корекция на правописа.

Попълнете тялото на функцията `correctSpelling` в кода на програмата, така че да имплементира функция за оценяване на кандидатите (2 т.)

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. Файл a1.py, съдържащ нанесените от вас промени
2. Доказателство на твърдението от точка 2 в условието. Доказателството може да е под форма на сканирано/снимано доказателство на хартия или pdf.