Търсене и извличане на информация. Приложение на дълбоко машинно обучение

Зимен семестър 2024/2025

Домашна задание №1

20.11.2024

Общ преглед

В това задание ще имплементираме вероятностен коректор на правопис, 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 или транспонирани.

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:

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

Вашата имплементационна работа се състои в допълването на празните функции на програмата във файла a1.py, който е приложен в пакета. По-конкретни пояснения за отделните задачи са дадени в следващите раздели, както и в коментарите в кода на програмата. В пакета е приложена и програмата test.py, с която може да направите тестове на Вашата имплементация. Тези тестове са повърхностни и тяхното успешно преминаване е само предпоставка за приемането, но не означава задължително, че програмата Ви ще бъде приета.

Забележка: За осигуряване на по-добра числова стабилност е желателно вместо стойностите на вероятностите да се използва логаритъм от вероятностите. 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 - Дамерау

Първата задача е да се имплементира функция, изчисляваща разстоянието на Левенщайн - Дамерау между две думи. Този вариант на Левенщайн разстоянието се използва обикновено в системите за корекция на текстове набирани на клавиатура, където се случва сравнително често и транспозицията на два съседни символа. На първата лекция разгледахме дефиницията на Левенщайн разстояние и псевдокод на алгоритъм, който го изчислява. Разстоянието на Левенщайн - Дамерау е подобно на Левенщайн разстоянието, но като елементарни операции се допускат освен изтриване, вмъкване и субституции на символи, също и транспозиция на два съседни символа. За да дефинираме формално разстоянието на Левенщайн - Дамерау 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 $i, j \in \{1, 2, \ldots, |W|\}$, където $i \leq j$ с $W_i^j \in \Sigma^*$ означаваме подниза $W_i W_{i+1} \ldots W_j$.

Използвайки горните означения дефинираме разстоянието на Левенщайн - Дамерау между низовете $P, W \in \Sigma^*$ индуктивно:

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

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

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

2 Тегло на редакция

Втората задача е свързана с намирането на теглото на редакцията за получаване от една дума на друга. Формално дефинираме множеството от елементарни редакции Op над азбука от символи Σ като:

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

където

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

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 редакция op ни съпоставя тегло $\omega(op)$. Последователността от елементарни операции

$op^1, op^2, \ldots, op^k$ подравнява думата r с думата q, ако $r = op_1^1 op_1^2 \ldots op_1^k$ и $q = op_2^1 op_2^2 \ldots op_2^k$. Теглото на подравняването дефинираме като: $\omega(op^1, op^2, \ldots, op^k) = \sum_{i=1}^{k} \omega(op^i)$. Дефинираме функцията Γ, която следва да връща минималното тегло на подравняване на думата P с думата W индуктивно:

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

Забележка: Ако за $op \in \mathrm{Id}$ имаме, че $\omega(op) = 0$ и за всяка елементарна редакция $op \in \mathrm{Ins} \cup \mathrm{Del} \cup \mathrm{Sub} \cup \mathrm{Tr}$ е изпълнено $\omega(op) = 1$, то теглото на редакция съвпада с разстоянието на Левенщайн - Дамерау.

Докажете, че $\Gamma(P, W)$ връща най-малкото тегло на подравняване на думата P с думата W (2 т.)

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

3 Коригиращ модел

Коригиращият модел ще моделира вероятността $\Pr[r|q]$.
Нека с $\mathbf{op} = op^1, op^2, \ldots, op^k$ означаваме последователност от елементарни редакции, подравняваща r с q, $r = \mathbf{op}_1 = \mathbf{op}_1^1 \mathbf{op}_1^2 \ldots \mathbf{op}_1^k$ и $q = \mathbf{op}_2 = \mathbf{op}_2^1 \mathbf{op}_2^2 \ldots \mathbf{op}_2^k$. Тогава

$$\Pr[r|q] = \sum_{\mathbf{op} \in \mathrm{Op}^* : \mathbf{op}_1 = r \,\&\, \mathbf{op}_2 = q} \Pr[r, \mathbf{op}|q] = \sum_{\mathbf{op} \in \mathrm{Op}^* : \mathbf{op}_1 = r \,\&\, \mathbf{op}_2 = q} \Pr[\mathbf{op}|q].$$

Ще предположим, че елементарните редакции са независими вероятностни събития и не зависят от заявката q. Тогава получаваме:

$$\Pr[r, \mathbf{op}|q] = \prod_{i=1}^{|\mathbf{op}|} \Pr[\mathbf{op}^i|q] = \prod_{i=1}^{|\mathbf{op}|} \Pr[\mathbf{op}^i]$$

Нека дефинираме $\omega(op) = -\log \Pr[op]$. Тогава получаваме:

$$\omega(\mathbf{op}) = \sum_{i=1}^{|\mathbf{op}|} \omega(\mathbf{op}^i) = -\sum_{i=1}^{|\mathbf{op}|} \log \Pr[\mathbf{op}^i] = -\log \prod_{i=1}^{|\mathbf{op}|} \Pr[\mathbf{op}^i] = -\log \Pr[r, \mathbf{op}|q].$$

По този начин коригиращия модел моделира вероятността за дадена редакция при условие желаната заявка.

Остава да се научат теглата на елементарните редакции, които следва да са минус логаритъм от вероятността за дадената елементарна редакция. Съответните вероятности ще получим като използваме принципа за максимизиране на правдоподобието. Обучението ще се извърши на базата на даден корпус съдържащ двойки $(r, q) \in \Sigma^* \times \Sigma^*$, състоящи се от сгрешен низ r и коригиран низ q. За всяка от двойките, чрез функцията `bestAlignment` ще бъде намерено най-доброто подравняване на двата низа при използването на разстоянието Левенщайн - Дамерау. Резултатът от подравняването е най-добра последователност от елементарни редакции $op^1, op^2, \ldots, op^k$, които подравняват думата r с думата q. Честотна статистика за срещанията на дадена елементарна редакция ще получим, като преброим броя на срещанията на дадената елементарна редакция в (най-добрите) подравнявания на двойките низове от корпуса за обучение. Това е осъществено във функцията `trainWeights`.

Попълнете тялото на функцията `bestAlignment` в кода на програмата, така че да бъде намерено най-доброто подравняване. Прочетете упътването в коментара към кода на функцията (2 т.)

4 Генератор на кандидати

Тъй като почти всички правописни грешки се намират в рамките на разстояние за редактиране до 2 от желаната от потребителя заявка, ще търсим кандидатите за корекции на заявки на разстояние до 2 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: Ще започнем, като генерираме всички възможни редакции, които са на разстояние 1 от оригиналната заявка чрез помощната функция `generateEdits`. Не трябва да забравяме, че разглеждаме и тирета и интервалите като символи. Това ще позволи да се разгледат някои сравнително често срещани грешки, например когато интервал е случайно вмъкнат в дума или две думи в заявката са залепени. След това във функцията `generateCandidates`, за да получим кандидатите на разстояние 2, ще извикаме втори път функцията `generateEdits` върху низовете на разстояние 1. Накрая ще филтрираме кандидатите, като 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 функцията `generateEdits` в кода на програмата, така че по зададена оригинална заявка да се генерират всички низове, които са на Левенщайн - Дамерау разстояние до оригиналната. (1 т.)

5 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, 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.