Исправление опечаток

Мати Рейнович Пентус, Алексей Андреевич Сорокин

МГУ им. М. В. Ломоносова

весенний семестр 2022/2023 учебного года
Межфакультетский курс
“Введение в компьютерную лингвистику”

Устранение опечаток

- Любой достаточно длинный текст содержит опечатки.

Устранение опечаток

- Любой достаточно длинный текст содержит опечатки.
- Их надо исправлять (автоматически):

Устранение опечаток

- Любой достаточно длинный текст содержит опечатки.
- Их надо исправлять (автоматически):
- Применения:
    - Информационный поиск (коррекция запросов)

Устранение опечаток

- Любой достаточно длинный текст содержит опечатки.
- Их надо исправлять (автоматически):
- Применения:
  - Информационный поиск (коррекция запросов)
  - Программы проверки правописания

Устранение опечаток

- Любой достаточно длинный текст содержит опечатки.
- Их надо исправлять (автоматически):
- Применения:
  - Информационный поиск (коррекция запросов)
  - Программы проверки правописания
  - Распознавание звучащей речи, изображений...

Устранение опечаток

- Любой достаточно длинный текст содержит опечатки.
- Их надо исправлять (автоматически):
- Применения:
  - Информационный поиск (коррекция запросов)
  - Программы проверки правописания
  - Распознавание звучащей речи, изображений...
  - Компаративистика («опечатки» — эволюционные изменения в словах языка или различия между родственными языками)

Причины опечаток

- Разновидности опечаток (в широком смысле):
  - Орфографические ошибки

Причины опечаток

- Разновидности опечаток (в широком смысле):
  - Орфографические ошибки
  - Типографские ошибки (опечатки в узком смысле/описки)

Причины опечаток

- Разновидности опечаток (в широком смысле):
  - Орфографические ошибки
  - Типографские ошибки (опечатки в узком смысле/описки)
  - Когнитивные ошибки (смешение понятий, “предать” $\leftrightarrow$ “придать”)

Причины опечаток

- Разновидности опечаток (в широком смысле):
  - Орфографические ошибки
  - Типографские ошибки (опечатки в узком смысле/описки)
  - Когнитивные ошибки (смешение понятий, “предать” $\leftrightarrow$ “придать”)
  - Ошибки при записи речи “на слух”.

Причины опечаток

- Разновидности опечаток (в широком смысле):
  - Орфографические ошибки
  - Типографские ошибки (опечатки в узком смысле/описки)
  - Когнитивные ошибки (смешение понятий, “предать” $\leftrightarrow$ “придать”)
  - Ошибки при записи речи “на слух”.
  - Транслитерационные ошибки (в иноязычных словах/именах собственных)
- Чаще всего ошибки локальны (затрагивают один-два символа)

Причины опечаток

- Разновидности опечаток (в широком смысле):
  - Орфографические ошибки
  - Типографские ошибки (опечатки в узком смысле/описки)
  - Когнитивные ошибки (смешение понятий, "предать" $\leftrightarrow$ "придать")
  - Ошибки при записи речи "на слух".
  - Транслитерационные ошибки (в иноязычных словах/именах собственных)
- Чаще всего ошибки локальны (затрагивают один-два символа)
- Однако может влиять и более широкий контекст ("ться" $\to$ *"цца", "ant" $\to$ "ent" в суффиксе прилагательного)

Модель близости слов

- Исправление опечаток требует поиска близких слов в словаре.

Модель близости слов

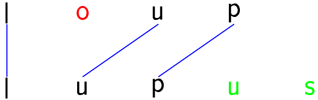
- Исправление опечаток требует поиска близких слов в словаре.
- Что значит “близких”, нужна функция расстояния.

Модель близости слов

- Исправление опечаток требует поиска близких слов в словаре.
- Что значит “близких”, нужна функция расстояния.

Расстояние Левенштейна

Расстояние Левенштейна $\rho_L(u, v)$ между словами u и v — минимальное число замен, вставок и удалений, необходимых, чтобы получить v из u.



$d(loup, lupus) = 3$

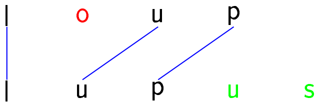
- Расстояние Левенштейна симметрично, неотрицательно, удовлетворяет неравенству: $\rho(u, v) \leqslant \rho(u, w) + \rho(w, v)$.

Модель близости слов

- Исправление опечаток требует поиска близких слов в словаре.
- Что значит “близких”, нужна функция расстояния.

Расстояние Левенштейна

Расстояние Левенштейна $\rho_L(u, v)$ между словами u и v — минимальное число замен, вставок и удалений, необходимых, чтобы получить v из u.



$d(loup, lupus) = 3$

- Расстояние Левенштейна симметрично, неотрицательно, удовлетворяет неравенству: $\rho(u, v) \leqslant \rho(u, w) + \rho(w, v)$.
- Иногда к допустимым операциям добавляют перестановку соседних символов (с некоторыми ограничениями).

Вычисление расстояния Левенштейна

Обозначения:

- $w = w_0 \dots w_{n-1}$ — слово, $|w| = n$ — длина слова.

Вычисление расстояния Левенштейна

Обозначения:

- $w = w_0 \dots w_{n-1}$ — слово, $|w| = n$ — длина слова.
- $w[i]$ — i-ый символ слова, $w[i, j]$ — подслово с i-ой по j-ую позицию (не включая j).

Вычисление расстояния Левенштейна

Обозначения:

- $w = w_0 \dots w_{n-1}$ — слово, $|w| = n$ — длина слова.
- $w[i]$ — i-ый символ слова, $w[i,j]$ — подслово с i-ой по j-ую позицию (не включая j).
- $w[,j]$ — префикс по j-ую позицию (не включая j).
- $w[i,]$ — суффикс с i-ой позиции (включая i).

Вычисление расстояния Левенштейна

Обозначения:

- $w = w_0 \dots w_{n-1}$ — слово, $|w| = n$ — длина слова.
- $w[i]$ — i-ый символ слова, $w[i, j]$ — подслово с i-ой по j-ую позицию (не включая j).
- $w[, j]$ — префикс по j-ую позицию (не включая j).
- $w[i,]$ — суффикс с i-ой позиции (включая i).

Идея алгоритма: будем вычислять $d_{ij} = \rho(u[, i], v[, j])$ рекурсивно через значения для меньших i, j. Если $|u| = m, |v| = n$, то ответом будет d_{mn}.

Рекурсивная формула

$$
\begin{aligned}
\rho(u[,i], v[,0]) &= i, \\
\rho(u[,0], v[,j]) &= j, \\
\rho(u[,i], v[,j]) &= \min\bigl(\rho(u[,i-1], v[,j-1]) + [\![u[i] \neq v[j]]\!], \\
&\qquad \rho(u[,i], v[,j-1]) + 1, \\
&\qquad \rho(u[,i-1], v[,j]) + 1\bigr)
\end{aligned}
$$

Рекурсивная формула

$$\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]) + [\![u[i] \neq v[j]]\!],\\
&\qquad \rho(u[,i],v[,j-1]) + 1,\\
&\qquad \rho(u[,i-1],v[,j]) + 1\big)
\end{aligned}$$

Возможные случаи:

- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1])$:
  $d(bad, bold) = d(ba, bol) = 2.$

b	a		d
|	|	|	|
b	o	l	d

Рекурсивная формула

$$
\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]) + [\![u[i] \neq v[j]]\!],\\
&\qquad\quad \rho(u[,i],v[,j-1]) + 1,\\
&\qquad\quad \rho(u[,i-1],v[,j]) + 1\big)
\end{aligned}
$$

Возможные случаи:

- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1])$:
  $d(bad, bold) = d(ba, bol) = 2$.
- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1]) + 1$:
  $\rho(feel, feed) = \rho(fee, fee) + 1 = 1$.

$$
\begin{array}{cccc}
\mathsf{f} & \mathsf{e} & \mathsf{e} & \mathsf{l}\\
| & | & | & |\\
\mathsf{f} & \mathsf{e} & \mathsf{e} & \mathsf{d}
\end{array}
$$

Рекурсивная формула

$$
\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]) + [\![u[i] \neq v[j]]\!],\\
&\qquad\quad \rho(u[,i],v[,j-1]) + 1,\\
&\qquad\quad \rho(u[,i-1],v[,j]) + 1\big)
\end{aligned}
$$

Возможные случаи:

- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1])$:
  $d(bad, bold) = d(ba, bol) = 2$.
- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1]) + 1$:
  $\rho(feel, feed) = \rho(fee, fee) + 1 = 1$.
- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j]) + 1$:
  $\rho(slide, solid) = \rho(slid, solid) + 1 = 2$.

$$
\begin{array}{cccccc}
\mathrm{s} & \varnothing & \mathrm{l} & \mathrm{i} & \mathrm{d} & \mathrm{e}\\
| & | & | & | & | & |\\
\mathrm{s} & \mathrm{o} & \mathrm{l} & \mathrm{i} & \mathrm{d} & \varnothing
\end{array}
$$

Рекурсивная формула

$$
\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]) + [\![u[i] \neq v[j]]\!],\\
&\qquad\quad \rho(u[,i],v[,j-1]) + 1,\\
&\qquad\quad \rho(u[,i-1],v[,j]) + 1\big)
\end{aligned}
$$

Возможные случаи:

- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1])$:
  $d(bad, bold) = d(ba, bol) = 2$.
- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j-1]) + 1$:
  $\rho(feel, feed) = \rho(fee, fee) + 1 = 1$.
- $\rho(u[,i],v[,j]) = \rho(u[,i-1],v[,j]) + 1$:
  $\rho(slide, solid) = \rho(slid, solid) + 1 = 2$.
- $\rho(u[,i],v[,j]) = \rho(u[,i],v[,j-1]) + 1$:
  $\rho(site, step) = \rho(site, ste) + 1 = 2$.

Свойства расстояния Левенштейна

Лемма

$\rho(ux, vx) = \rho(u, v)$

Свойства расстояния Левенштейна

Лемма

$\rho(ux, vx) = \rho(u, v)$

Доказательство

- Достаточно доказать для однобуквенного x, пусть $x = a$.

Свойства расстояния Левенштейна

Лемма

$\rho(ux, vx) = \rho(u, v)$

Доказательство

- Достаточно доказать для однобуквенного x, пусть $x = a$.
- Достаточно показать, что оптимальное выравнивание сопоставляет последние символы друг другу:

$$\begin{array}{cc} \overbrace{\alpha_1 \quad \cdots \quad \alpha_r}^{u} & \quad a \\ \underbrace{\beta_1 \quad \cdots \quad \beta_r}_{v} & \quad a \end{array}$$

Свойства расстояния Левенштейна

Лемма

$\rho(ux, vx) = \rho(u, v)$

Доказательство

- Достаточно доказать для однобуквенного x, пусть $x = a$.
- Достаточно показать, что оптимальное выравнивание сопоставляет последние символы друг другу:
- Пусть не так:

$$\begin{array}{cccc} \alpha_1 \quad \cdots \quad \alpha_s & \mathrm{a} & \varnothing & \varnothing \\ & | & | & | \\ \underbrace{\beta_1 \quad \cdots \quad \beta_s}_{v_1} & \beta & v_2 & a \end{array}$$

Свойства расстояния Левенштейна

Лемма

$\rho(ux, vx) = \rho(u, v)$

Доказательство

- Достаточно доказать для однобуквенного x, пусть $x = a$.
- Достаточно показать, что оптимальное выравнивание сопоставляет последние символы друг другу:
- Пусть не так:

$$\begin{array}{cccccc} \alpha_1 & \cdots & \alpha_s & \mathrm{a} & \varnothing & \varnothing \\ & & & | & | & | \\ \underbrace{\beta_1 \quad \cdots \quad \beta_s}_{v_1} & & & \beta & v_2 & a \end{array}$$

- Стоимость этого выравнивания $d_1(ua, va) = \rho(u, v_1) + \rho(a, \beta) + \rho(\varepsilon, v_2) + \rho(\varepsilon, a) \geqslant \rho(u, v_1) + \rho(\varepsilon, v_2) + 1$.

Свойства расстояния Левенштейна

Продолжение доказательства

- Перестроим выравнивание:

$$
\begin{array}{cccccc}
\alpha_1 & \cdots & \alpha_s & \varnothing & \varnothing & a \\
 & & & | & | & | \\
\beta_1 & \cdots & \beta_s & \beta & v_2 & a
\end{array}
$$

$$\underbrace{\beta_1 \quad \cdots \quad \beta_s}_{v_1}$$

Свойства расстояния Левенштейна

Продолжение доказательства

- Перестроим выравнивание:

$$\begin{array}{cccccc} \alpha_1 & \cdots & \alpha_s & \varnothing & \varnothing & a \\ & & & | & | & | \\ \underbrace{\beta_1 & \cdots & \beta_s}_{v_1} & \beta & v_2 & a \end{array}$$

- Стоимость выравнивания $d_2(x, y) = \rho(u, v_1) + \rho(\varepsilon, \beta v_2) + \rho(a, a) \leqslant \rho(u, v_1) + \rho(\varepsilon, \beta) + \rho(\varepsilon, v_2) + 0 \leqslant \rho(u, v_1) + 1 + \rho(\varepsilon, v_2)$.

Свойства расстояния Левенштейна

Продолжение доказательства

- Перестроим выравнивание:

$$\begin{array}{cccccc} \alpha_1 & \cdots & \alpha_s & \varnothing & \varnothing & a \\ & & & | & | & | \\ \beta_1 & \cdots & \beta_s & \beta & v_2 & a \end{array}$$

$$\underbrace{\beta_1 \ \cdots \ \beta_s}_{v_1}$$

- Стоимость выравнивания $d_2(x, y) = \rho(u, v_1) + \rho(\varepsilon, \beta v_2) + \rho(a, a) \leqslant \rho(u, v_1)+\rho(\varepsilon, \beta)+\rho(\varepsilon, v_2)+0 \leqslant \rho(u, v_1)+1+\rho(\varepsilon, v_2)$.
- Предыдущее выравнивание было не лучше оптимального. Лемма доказана.

Окончательная рекуррентная формула

$$\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \rho(u[,i-1],v[,j-1]),\ \text{если}\ u[i-1]=v[j-1]\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]),\\
&\qquad \rho(u[,i],v[,j-1]),\\
&\qquad \rho(u[,i-1],v[,j])\big)+1,\ \text{если}\ u[i-1]\neq v[j-1]
\end{aligned}$$

Окончательная рекуррентная формула

$$
\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \rho(u[,i-1],v[,j-1]),\ \text{если}\ u[i-1]=v[j-1]\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]),\\
&\qquad \rho(u[,i],v[,j-1]),\\
&\qquad \rho(u[,i-1],v[,j])\big)+1,\ \text{если}\ u[i-1]\neq v[j-1]
\end{aligned}
$$

Идея алгоритма: будем заполнять двумерную таблицу D размера $(m+1)\times(n+1)$, где в ячейке с номером (i,j) будет храниться $\rho(u[,i],v[,j])$. Заполняем рекурсивно по возрастанию i и j.

Псевдокод

Вход: Слова u, v длины m, n соответственно.
Выход: $d(u, v)$ — расстояние Левенштейна между u и v.

```
▷ D — таблица размера (m + 1) × (n + 1)
D[0, 0] = 0
for i = 1, ..., m do
    D[i, 0] = i
end for
for j = 1, ..., n do
    D[0, j] = j
end for
for i = 1, ..., m do
    for j = 1, ..., n do
        if u[i − 1] == v[j − 1] then
            D[i, j] = D[i − 1, j − 1]
        else
            d = min(D[i − 1, j − 1], D[i, j − 1], D[i − 1, j])
            D[i, j] = d + 1
        end if
    end for
end for
 return D[m, n]
```

Модификации алгоритма

Возможные модификации алгоритма:

- Различные веса операций. Нужно прибавлять не 0 или 1, а стоимости соответствующих операций с последними символами, и вычислять минимум.

Модификации алгоритма

Возможные модификации алгоритма:

- Различные веса операций. Нужно прибавлять не 0 или 1, а стоимости соответствующих операций с последними символами, и вычислять минимум.
- Перестановка соседних символов (один и тот же символ не может участвовать в нескольких перестановках):

$$d(\text{трата}, \text{тартан}) = 2$$

т р а т а

т а р т а н

Модификации алгоритма

Возможные модификации алгоритма:

- Различные веса операций. Нужно прибавлять не 0 или 1, а стоимости соответствующих операций с последними символами, и вычислять минимум.
- Перестановка соседних символов (один и тот же символ не может участвовать в нескольких перестановках):

$$d(\text{трата}, \text{тартан}) = 2$$

т р а т а

т а р т а н

- В рекурретной формуле добавляется вариант $d(u[, i{-}2], v[, j{-}2]) + 1$, если $u[i-2] = v[i-1]$ и $v[i-2] = u[i-1]$.

Применение к исправлению опечаток

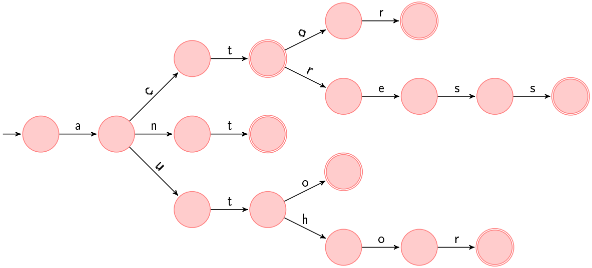
- Как исправлять опечатки с помощью расстояния Левенштейна?
- Наивный подход: пройти по словарю, подсчитать расстояние до каждого слова, выбрать ближайшее.

Применение к исправлению опечаток

- Как исправлять опечатки с помощью расстояния Левенштейна?
- Наивный подход: пройти по словарю, подсчитать расстояние до каждого слова, выбрать ближайшее.
- Нельзя: словари большие (агглютинативные языки — сотни тысяч слов), а расстояние считается медленно.
- Можно “на лету” отслеживать все префиксы слов из словаря, удалённые от прочитанного префикса на расстояние, не превышаю порога d.

Применение автомата Левенштейна

- Словарь часто хранят в форме ациклического автомата (префиксн бор):



Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить префиксный бор для словаря.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить префиксный бор для словаря.
- При точном поиске в префиксном боре разрешается идти только подряд по буквам слова.
- При приближённом поиске разрешается отклоняться от пути, но ограниченное количество раз.

Алгоритм поиска кандидатов

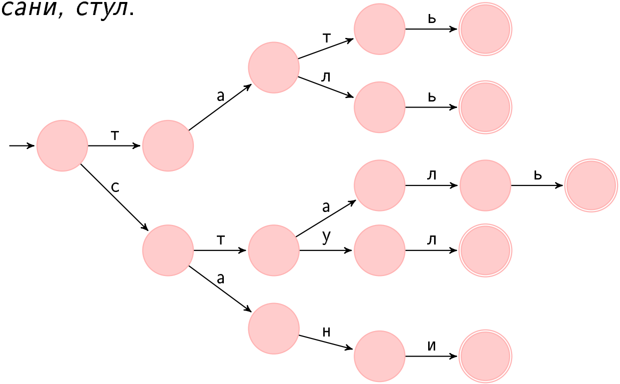
Алгоритм поиска слов-кандидатов:

- Построить префиксный бор для словаря.
- При точном поиске в префиксном боре разрешается идти только подряд по буквам слова.
- При приближённом поиске разрешается отклоняться от пути, но ограниченное количество раз.
- При этом нужно хранить несколько гипотез, в каждой из которых помнить число отклонений.

Приближённый поиск в префиксном боре

Пример.

Найти все слова на расстоянии Левенштейна не более 1 от слова “стать” в префиксном боре из слов *сталь*, *тать*, *таль*, *сани*, *стул*.



Приближённый поиск в префиксном боре

- Гипотеза — тройка (u, v, k), где u — прочитанный префикс слова, v — позиция в префиксном боре, k — уже допущенное число ошибок.
- $(\varepsilon, \varepsilon, 0)$ — начальная гипотеза.

Приближённый поиск в префиксном боре

- Гипотеза — тройка (u, v, k), где u — прочитанный префикс слова, v — позиция в префиксном боре, k — уже допущенное число ошибок.
- $(\varepsilon, \varepsilon, 0)$ — начальная гипотеза.
- Пытаемся расширить текущую гипотезу всеми возможными способами (сделать одну элементарную операцию).

Приближённый поиск в префиксном боре

- Гипотеза — тройка (u, v, k), где u — прочитанный префикс слова, v — позиция в префиксном боре, k — уже допущенное число ошибок.
- $(\varepsilon, \varepsilon, 0)$ — начальная гипотеза.
- Пытаемся расширить текущую гипотезу всеми возможными способами (сделать одну элементарную операцию).
- Новые гипотезы складываем в очередь, из которой на каждом шаге алгоритма извлекается и обрабатывается одна гипотеза.

Элементарные операции при поиске в префиксном боре

Возможные операции:

- Прочитать одинаковые буквы в слове и боре, сдвинуться на одну позицию в слове и боре, не меняя k.

После обработки нулевой гипотезы в очереди будут гипотезы:

	Гипотеза	Последняя операция	Предыдущая гипотеза
1	$\langle$с, с, $0\rangle$	с $\to$ с	$\langle\varepsilon, \varepsilon, 0\rangle$
2	$\langle$с, т, $1\rangle$	с $\to$ т	$\langle\varepsilon, \varepsilon, 0\rangle$
3	$\langle\varepsilon$, с, $1\rangle$	$\varepsilon \to$ с	$\langle\varepsilon, \varepsilon, 0\rangle$
4	$\langle\varepsilon$, т, $1\rangle$	$\varepsilon \to$ т	$\langle\varepsilon, \varepsilon, 0\rangle$
5	$\langle$с, $\varepsilon, 1\rangle$	с $\to \varepsilon$	$\langle\varepsilon, \varepsilon, 0\rangle$

Элементарные операции при поиске в префиксном боре

Возможные операции:

- Прочитать одинаковые буквы в слове и боре, сдвинуться на одну позицию в слове и боре, не меняя k.
- Прочитать разные буквы в слове и боре, сдвинуться и там, и там на одну позицию, увеличить k на 1.

После обработки нулевой гипотезы в очереди будут гипотезы:

	Гипотеза	Последняя операция	Предыдущая гипотеза
1	$\langle \text{с}, \text{с}, 0\rangle$	$\text{с} \to \text{с}$	$\langle \varepsilon, \varepsilon, 0\rangle$
2	$\langle \text{с}, \text{т}, 1\rangle$	$\text{с} \to \text{т}$	$\langle \varepsilon, \varepsilon, 0\rangle$
3	$\langle \varepsilon, \text{с}, 1\rangle$	$\varepsilon \to \text{с}$	$\langle \varepsilon, \varepsilon, 0\rangle$
4	$\langle \varepsilon, \text{т}, 1\rangle$	$\varepsilon \to \text{т}$	$\langle \varepsilon, \varepsilon, 0\rangle$
5	$\langle \text{с}, \varepsilon, 1\rangle$	$\text{с} \to \varepsilon$	$\langle \varepsilon, \varepsilon, 0\rangle$

Элементарные операции при поиске в префиксном боре

Возможные операции:

- Прочитать одинаковые буквы в слове и боре, сдвинуться на одну позицию в слове и боре, не меняя k.
- Прочитать разные буквы в слове и боре, сдвинуться и там, и там на одну позицию, увеличить k на 1.
- Прочитать в слове 1 букву, не двигаться в боре, увеличить k.

После обработки нулевой гипотезы в очереди будут гипотезы:

	Гипотеза	Последняя операция	Предыдущая гипотеза
1	$\langle \text{с}, \text{с}, 0 \rangle$	с $\to$ с	$\langle \varepsilon, \varepsilon, 0 \rangle$
2	$\langle \text{с}, \text{т}, 1 \rangle$	с $\to$ т	$\langle \varepsilon, \varepsilon, 0 \rangle$
3	$\langle \varepsilon, \text{с}, 1 \rangle$	$\varepsilon \to$ с	$\langle \varepsilon, \varepsilon, 0 \rangle$
4	$\langle \varepsilon, \text{т}, 1 \rangle$	$\varepsilon \to$ т	$\langle \varepsilon, \varepsilon, 0 \rangle$
5	$\langle \text{с}, \varepsilon, 1 \rangle$	с $\to \varepsilon$	$\langle \varepsilon, \varepsilon, 0 \rangle$

Элементарные операции при поиске в префиксном боре

Возможные операции:

- Прочитать одинаковые буквы в слове и боре, сдвинуться на одну позицию в слове и боре, не меняя k.
- Прочитать разные буквы в слове и боре, сдвинуться и там, и там на одну позицию, увеличить k на 1.
- Прочитать в слове 1 букву, не двигаться в боре, увеличить k.
- Сдвинуться в боре на 1 букву, не менять позицию в слове, увеличить k на 1.

После обработки нулевой гипотезы в очереди будут гипотезы:

	Гипотеза	Последняя операция	Предыдущая гипотеза
1	$\langle \text{с}, \text{с}, 0 \rangle$	$\text{с} \to \text{с}$	$\langle \varepsilon, \varepsilon, 0 \rangle$
2	$\langle \text{с}, \text{т}, 1 \rangle$	$\text{с} \to \text{т}$	$\langle \varepsilon, \varepsilon, 0 \rangle$
3	$\langle \varepsilon, \text{с}, 1 \rangle$	$\varepsilon \to \text{с}$	$\langle \varepsilon, \varepsilon, 0 \rangle$
4	$\langle \varepsilon, \text{т}, 1 \rangle$	$\varepsilon \to \text{т}$	$\langle \varepsilon, \varepsilon, 0 \rangle$
5	$\langle \text{с}, \varepsilon, 1 \rangle$	$\text{с} \to \varepsilon$	$\langle \varepsilon, \varepsilon, 0 \rangle$

Элементарные операции при поиске в префиксном боре

- Обрабатываемая гипотеза: $\langle \text{с}, \text{с}, 0 \rangle$.
- Возможные операции т $\to$ т, т $\to$ а, т $\to \varepsilon$, $\varepsilon \to$ т, $\varepsilon \to$ а.
- Получаются гипотезы:

6	$\langle \text{ст}, \text{ст}, 0 \rangle$	т $\to$ т	$\langle \text{с}, \text{с}, 0 \rangle$
7	$\langle \text{ст}, \text{са}, 1 \rangle$	т $\to$ а	$\langle \text{с}, \text{с}, 0 \rangle$
8	$\langle \text{ст}, \text{с}, 1 \rangle$	т $\to \varepsilon$	$\langle \text{с}, \text{с}, 0 \rangle$
9	$\langle \text{с}, \text{ст}, 1 \rangle$	$\varepsilon \to$ т	$\langle \text{с}, \text{с}, 0 \rangle$
10	$\langle \text{с}, \text{са}, 1 \rangle$	$\varepsilon \to$ а	$\langle \text{с}, \text{с}, 0 \rangle$

Элементарные операции при поиске в префиксном боре

- Обрабатываемая гипотеза: $\langle \text{с}, \text{т}, 1 \rangle$.
- Поскольку уже штраф 1, то разрешена только тождественная операция.

Элементарные операции при поиске в префиксном боре

- Обрабатываемая гипотеза: $\langle \text{с}, \text{т}, 1 \rangle$.
- Поскольку уже штраф 1, то разрешена только тождественная операция.
- Её нельзя осуществить, на этом шаге ничего не добавляем.

Элементарные операции при поиске в префиксном боре

- Обрабатываемая гипотеза: $\langle \text{с}, \text{т}, 1 \rangle$.
- Поскольку уже штраф 1, то разрешена только тождественная операция.
- Её нельзя осуществить, на этом шаге ничего не добавляем.
- Аналогично с гипотезами $\langle \varepsilon, \text{с}, 1 \rangle, \langle \varepsilon, \text{т}, 1 \rangle$.

Элементарные операции при поиске в префиксном боре

- Обрабатываемая гипотеза: $\langle \text{с}, \text{т}, 1 \rangle$.
- Поскольку уже штраф 1, то разрешена только тождественная операция.
- Её нельзя осуществить, на этом шаге ничего не добавляем.
- Аналогично с гипотезами $\langle \varepsilon, \text{с}, 1 \rangle, \langle \varepsilon, \text{т}, 1 \rangle$.
- Обрабатываемая гипотеза $\langle \text{с}, \varepsilon, 1 \rangle$.

Элементарные операции при поиске в префиксном боре

- Обрабатываемая гипотеза: $\langle \text{с}, \text{т}, 1 \rangle$.
- Поскольку уже штраф 1, то разрешена только тождественная операция.
- Её нельзя осуществить, на этом шаге ничего не добавляем.
- Аналогично с гипотезами $\langle \varepsilon, \text{с}, 1 \rangle$, $\langle \varepsilon, \text{т}, 1 \rangle$.
- Обрабатываемая гипотеза $\langle \text{с}, \varepsilon, 1 \rangle$.
- Возможна операция $\text{т} \to \text{т}$, получается гипотеза $\langle \text{ст}, \text{т}, 1 \rangle$.

Элементарные операции при поиске в префиксном боре

Продолжая, получаем

12	$\langle$ста, ста, $0\rangle$	а $\to$ а	$\langle$ст, ст, $0\rangle$
13	$\langle$ста, сту, $1\rangle$	а $\to$ у	$\langle$ст, ст, $0\rangle$
14	$\langle$ста, ст, $1\rangle$	а $\to \varepsilon$	$\langle$ст, ст, $0\rangle$
15	$\langle$ст, ста, $1\rangle$	$\varepsilon \to$ а	$\langle$ст, ст, $0\rangle$
16	$\langle$ст, сту, $1\rangle$	$\varepsilon \to$ у	$\langle$ст, ст, $0\rangle$
17	$\langle$ста, са, $1\rangle$	а $\to$ а	$\langle$ст, с, $1\rangle$
18	$\langle$ста, та, $1\rangle$	а $\to$ а	$\langle$ст, т, $1\rangle$

Элементарные операции при поиске в префиксном боре

Продолжая, получаем

19	$\langle$стат, стал, 1$\rangle$	т $\to$ л	$\langle$ста, ста, 0$\rangle$
20	$\langle$ста, стал, 1$\rangle$	$\varepsilon \to$ л	$\langle$ста, ста, 0$\rangle$
21	$\langle$стат, тат, 1$\rangle$	т $\to$ т	$\langle$ста, та, 1$\rangle$
22	$\langle$стать, сталь, 1$\rangle$	ь $\to$ ь	$\langle$стат, стал, 1$\rangle$
Получено слово из бора на расстоянии 1. Добавляем слово “сталь” в ответ			
23	$\langle$стать, тать, 1$\rangle$	ь $\to$ ь	$\langle$стат, тат, 1$\rangle$
Получено слово из бора на расстоянии 1. Добавляем слово “тать” в ответ			

Взвешенное расстояние Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:

Взвешенное расстояние Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:
- $d(loup, lobo) = 2$:

$$\begin{array}{cccc} l & o & u & p \\ | & | & | & | \\ l & o & b & o \end{array}$$

Взвешенное расстояние Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:
- $d(loup, lobo) = 2$:

$$\begin{array}{cccc} l & o & u & p \\ | & | & | & | \\ l & o & b & o \end{array}$$

- Естественное выравнивание даёт $d = 3$:

$$\begin{array}{ccccc} l & o & u & p & \varnothing \\ | & | & | & | & | \\ l & o & \varnothing & b & o \end{array}$$

Взвешенное расстояние Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:
- $d(loup, lobo) = 2$:

$$\begin{array}{cccc} l & o & u & p \\ | & | & | & | \\ l & o & b & o \end{array}$$

- Естественное выравнивание даёт $d = 3$:

$$\begin{array}{ccccc} l & o & u & p & \varnothing \\ | & | & | & | & | \\ l & o & \varnothing & b & o \end{array}$$

- Надо присвоить различным операциям веса, зависящие от заменяемых/удаляемых/вставляемых символов.

Взвешенное расстояние Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:
- $d(loup, lobo) = 2$:

$$\begin{array}{cccc} l & o & u & p \\ | & | & | & | \\ l & o & b & o \end{array}$$

- Естественное выравнивание даёт $d = 3$:

$$\begin{array}{ccccc} l & o & u & p & \varnothing \\ | & | & | & | & | \\ l & o & \varnothing & b & o \end{array}$$

- Надо присвоить различным операциям веса, зависящие от заменяемых/удаляемых/вставляемых символов.
- Алгоритм вычисления расстояния от этого не изменится (при естественных ограничениях на веса).
- Откуда взять веса?

Вероятностная модель взвешивания

- Элементарные операции вида $\beta : \alpha$, соответствующие «опечаткам вида $\alpha \to \beta$.

Вероятностная модель взвешивания

- Элементарные операции вида $\beta : \alpha$, соответствующие «опечаткам вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно посчитать по корпусу опечаток.

Вероятностная модель взвешивания

- Элементарные операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно посчитать по корпусу опечаток.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.

Вероятностная модель взвешивания

- Элементарные операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно посчитать по корпусу опечаток.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.
- Маленьким вероятностям соответствуют большие веса.

Вероятностная модель взвешивания

- Элементарные операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно посчитать по корпусу опечаток.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.
- Маленьким вероятностям соответствуют большие веса.

Вероятностная модель взвешивания

- Элементарные операции вида $\beta : \alpha$, соответствующие «опечаткам вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно посчитать по корпусу опечаток.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.
- Маленьким вероятностям соответствуют большие веса.
- Какой в этом смысл?

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)...(a_r : b_r, w_r)$ — путь, «исправляющий» слово $u = a_1 \dots a_r$ на $v = b_1 \dots b_r$.

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)\dots(a_r : b_r, w_r)$ — путь, «исправляющий» слово $u = a_1 \dots a_r$ на $v = b_1 \dots b_r$.
- Тогда $M(a_1 \dots a_r : b_1 \dots b_r) = \sum_{i=1}^{r} w_i = -\sum_{i=1}^{r} \log p(b_i \to a_i) = -\log \prod_{i=1}^{r} p(b_i \to a_i) \approx -\log p(b_1 \dots b_r \to a_1 \dots a_r) \approx -\log p(v \to u)$.

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)\ldots(a_r : b_r, w_r)$ — путь, «исправляющий» слово $u = a_1 \ldots a_r$ на $v = b_1 \ldots b_r$.
- Тогда $M(a_1 \ldots a_r : b_1 \ldots b_r) = \sum_{i=1}^{r} w_i = -\sum_{i=1}^{r} \log p(b_i \to a_i) = -\log \prod_{i=1}^{r} p(b_i \to a_i) \approx -\log p(b_1 \ldots b_r \to a_1 \ldots a_r) \approx -\log p(v \to u)$.
- Вес преобразования $u : v$ — (приблизительно) отрицательный логарифм вероятности получить v из u.

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)\ldots(a_r : b_r, w_r)$ — путь, «исправляющий» слово $u = a_1 \ldots a_r$ на $v = b_1 \ldots b_r$.
- Тогда $M(a_1 \ldots a_r : b_1 \ldots b_r) = \sum_{i=1}^{r} w_i = -\sum_{i=1}^{r} \log p(b_i \to a_i) = -\log \prod_{i=1}^{r} p(b_i \to a_i) \approx -\log p(b_1 \ldots b_r \to a_1 \ldots a_r) \approx -\log p(v \to u)$.
- Вес преобразования $u : v$ — (приблизительно) отрицательный логарифм вероятности получить v из u.
- Кажется весьма разумным...

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)...(a_r : b_r, w_r)$ — путь, «исправляющий» слово $u = a_1 \dots a_r$ на $v = b_1 \dots b_r$.
- Тогда $M(a_1 \dots a_r : b_1 \dots b_r) = \sum_{i=1}^{r} w_i = -\sum_{i=1}^{r} \log p(b_i \to a_i) = -\log \prod_{i=1}^{r} p(b_i \to a_i) \approx -\log p(b_1 \dots b_r \to a_1 \dots a_r) \approx -\log p(v \to u)$.
- Вес преобразования $u : v$ — (приблизительно) отрицательный логарифм вероятности получить v из u.
- Кажется весьма разумным...
- Но есть недостаток...

Вероятностная модель исправления опечаток

- Кажется естественным выбирать в качестве исправления c для слова u наиболее вероятное слово: $u = \operatorname{argmax} p(c|u)$.

Вероятностная модель исправления опечаток

- Кажется естественным выбирать в качестве исправления c для слова u наиболее вероятное слово: $u = \operatorname{argmax} p(c|u)$.
- У нас же $p(u|c)$,как перейти?

Вероятностная модель исправления опечаток

- Кажется естественным выбирать в качестве исправления c для слова u наиболее вероятное слово: $u = \operatorname{argmax} p(c|u)$.
- У нас же $p(u|c)$,как перейти?
- Формула Байеса:

$$p(c|u) = \frac{p(u|c)p(c)}{p(u)}$$

Вероятностная модель исправления опечаток

- Кажется естественным выбирать в качестве исправления c для слова u наиболее вероятное слово: $u = \operatorname{argmax} p(c|u)$.
- У нас же $p(u|c)$,как перейти?
- Формула Байеса:

$$p(c|u) = \frac{p(u|c)p(c)}{p(u)}$$

- $p(u)$ не влияет на выбор кандидата, поэтому: $c = \operatorname{argmax} p(u|c)p(c)$

Вероятностная модель исправления опечаток

- Кажется естественным выбирать в качестве исправления c для слова u наиболее вероятное слово: $u = \operatorname{argmax} p(c|u)$.
- У нас же $p(u|c)$,как перейти?
- Формула Байеса:

$$p(c|u) = \frac{p(u|c)p(c)}{p(u)}$$

- $p(u)$ не влияет на выбор кандидата, поэтому: $c = \operatorname{argmax} p(u|c)p(c)$
- В модели весов это соответствует формуле

$$M(a_1 \dots a_r : b_1 \dots b_r) = -\sum_{i=1}^{r} \log p(b_i \to a_i) - \log p(c)$$

Вероятностная модель исправления опечаток

- Кажется естественным выбирать в качестве исправления c для слова u наиболее вероятное слово: $u = \operatorname{argmax} p(c|u)$.
- У нас же $p(u|c)$,как перейти?
- Формула Байеса:

$$p(c|u) = \frac{p(u|c)p(c)}{p(u)}$$

- $p(u)$ не влияет на выбор кандидата, поэтому: $c = \operatorname{argmax} p(u|c)p(c)$
- В модели весов это соответствует формуле

$$M(a_1 \ldots a_r : b_1 \ldots b_r) = -\sum_{i=1}^{r} \log p(b_i \to a_i) - \log p(c)$$

- Добавим $-\log p(c)$ в качестве выходного штрафа.

Финальный алгоритм

Финальный алгоритм поиска слов-кандидатов:

- Взять словарь с проставленными вероятностями слов $p(c)$ (вероятности собрать из корпуса), преобразовать во взвешенный автомат с выходными весами $-\log p(c)$.

Финальный алгоритм

Финальный алгоритм поиска слов-кандидатов:

- Взять словарь с проставленными вероятностями слов $p(c)$ (вероятности собрать из корпуса), преобразовать во взвешенный автомат с выходными весами $-\log p(c)$.
- Найти все слова в данном автомате на расстоянии меньше d от текущего.

Финальный алгоритм

Финальный алгоритм поиска слов-кандидатов:

- Взять словарь с проставленными вероятностями слов $p(c)$ (вероятности собрать из корпуса), преобразовать во взвешенный автомат с выходными весами $-\log p(c)$.
- Найти все слова в данном автомате на расстоянии меньше d от текущего.
- Откуда взять вероятности? Можно настроить по обычному корпусу и словарю (без корпуса опечаток) или с помощью эвристик.

Финальный алгоритм

Финальный алгоритм поиска слов-кандидатов:

- Взять словарь с проставленными вероятностями слов $p(c)$ (вероятности собрать из корпуса), преобразовать во взвешенный автомат с выходными весами $-\log p(c)$.
- Найти все слова в данном автомате на расстоянии меньше d от текущего.
- Откуда взять вероятности? Можно настроить по обычному корпусу и словарю (без корпуса опечаток) или с помощью эвристик.
- Эвристики: что влияет на веса замен:
    - Опечатки: расположение символов на клавиатуре

Финальный алгоритм

Финальный алгоритм поиска слов-кандидатов:

- Взять словарь с проставленными вероятностями слов $p(c)$ (вероятности собрать из корпуса), преобразовать во взвешенный автомат с выходными весами $-\log p(c)$.
- Найти все слова в данном автомате на расстоянии меньше d от текущего.
- Откуда взять вероятности? Можно настроить по обычному корпусу и словарю (без корпуса опечаток) или с помощью эвристик.
- Эвристики: что влияет на веса замен:
  - Опечатки: расположение символов на клавиатуре
  - Орфографические ошибки: фонетическая близость

Финальный алгоритм

Финальный алгоритм поиска слов-кандидатов:

- Взять словарь с проставленными вероятностями слов $p(c)$ (вероятности собрать из корпуса), преобразовать во взвешенный автомат с выходными весами $-\log p(c)$.
- Найти все слова в данном автомате на расстоянии меньше d от текущего.
- Откуда взять вероятности? Можно настроить по обычному корпусу и словарю (без корпуса опечаток) или с помощью эвристик.
- Эвристики: что влияет на веса замен:
  - Опечатки: расположение символов на клавиатуре
  - Орфографические ошибки: фонетическая близость
  - Ошибки распознавания: графическая близость

Автоматическая настройка весов

Автоматическая настройка весов: неформальное описание алгоритма (EM-алгоритм):

- Задать вероятности замен случайным образом.

Автоматическая настройка весов

Автоматическая настройка весов: неформальное описание алгоритма (EM-алгоритм):

- Задать вероятности замен случайным образом.
- Для каждого слова u из тестового корпуса получить список кандидатов $c_{u,1}, \ldots, c_{u,n_u}$ с их вероятностями. Сохранить все пары вида $(u \rightsquigarrow c_{u,j}, p(c_{u,j}|u))$, где $(u \rightsquigarrow c_{u,j})$ — последовательность замен, переводящих $c_{u,j}$ в u.

Автоматическая настройка весов

Автоматическая настройка весов: неформальное описание алгоритма (EM-алгоритм):

- Задать вероятности замен случайным образом.
- Для каждого слова u из тестового корпуса получить список кандидатов $c_{u,1}, \ldots, c_{u,n_u}$ с их вероятностями. Сохранить все пары вида $(u \rightsquigarrow c_{u,j}, p(c_{u,j}|u))$, где $(u \rightsquigarrow c_{u,j})$ — последовательность замен, переводящих $c_{u,j}$ в u.
- Сложив для каждой замены вероятности пар, в которых она встретилась, получим частоты замен, а значит, их вероятности.

Автоматическая настройка весов

Автоматическая настройка весов: неформальное описание алгоритма (EM-алгоритм):

- Задать вероятности замен случайным образом.
- Для каждого слова u из тестового корпуса получить список кандидатов $c_{u,1}, \ldots, c_{u,n_u}$ с их вероятностями. Сохранить все пары вида $(u \rightsquigarrow c_{u,j}, p(c_{u,j}|u))$, где $(u \rightsquigarrow c_{u,j})$ — последовательность замен, переводящих $c_{u,j}$ в u.
- Сложив для каждой замены вероятности пар, в которых она встретилась, получим частоты замен, а значит, их вероятности.
- Повторим эту процедуру, пока алгоритм не сойдётся.

Онлайн-материалы

- `http://norvig.com/spell-correct.html` — страница Питера Норвига. Простейшая работающая программа для исправления опечаток из 21 строчки (Python 2.5).

Онлайн-материалы

- `http://norvig.com/spell-correct.html` — страница Питера Норвига. Простейшая работающая программа для исправления опечаток из 21 строчки (Python 2.5).
- Ещё много материалов: `http://norvig.com`

Онлайн-материалы

- http://norvig.com/spell-correct.html — страница Питера Норвига. Простейшая работающая программа для исправления опечаток из 21 строчки (Python 2.5).
- Ещё много материалов: http://norvig.com
- https://web.stanford.edu/~jurafsky/slp3/B.pdf — глава из учебника Дэниэла Журафски про исправление опечаток.