Математические модели в лингвистике

Исправление опечаток

Мати Пентус, Александр Пиперски, Алексей Сорокин

МГУ им. М. В. Ломоносова, межфакультетский курс,
осенний семестр 2015–2016 учебного года

Определение задачи

Исправление опечаток

Исправление опечаток (в широком смысле) — выявление ошибок правописания и поиск возможных вариантов их исправления.

Определение задачи

Исправление опечаток

Исправление опечаток (в широком смысле) — выявление ошибок правописания и поиск возможных вариантов их исправления.

- Опечатки и орфографические ошибки содержатся в любом достаточно длинном тексте.

Определение задачи

Исправление опечаток

Исправление опечаток (в широком смысле) — выявление ошибок правописания и поиск возможных вариантов их исправления.

- Опечатки и орфографические ошибки содержатся в любом достаточно длинном тексте.

Решено было не допустить ни одной ошибки. Держали двадцать корректур. И все равно на титульном листе было напечатано: Британская энциклопудия.

Илья Ильф, «Записные книжки»

Определение задачи

Исправление опечаток

Исправление опечаток (в широком смысле) — выявление ошибок правописания и поиск возможных вариантов их исправления.

- Опечатки и орфографические ошибки содержатся в любом достаточно длинном тексте.

Решено было не допустить ни одной ошибки. Держали двадцать корректур. И все равно на титульном листе было напечатано: Британская энциклопудия.

Илья Ильф, «Записные книжки»

- По запросу “сваего мнения” google.ru находит 689 результатов.

Применения исправления опечаток

- Информационный поиск (коррекция запросов)

Применения исправления опечаток

- Информационный поиск (коррекция запросов)
- Программы проверки правописания в текстовых редакторах

Применения исправления опечаток

- Информационный поиск (коррекция запросов)
- Программы проверки правописания в текстовых редакторах
- Нормализация текста (коррекция стилистических вариантов, актуальна для социальных медиа)

Применения исправления опечаток

- Информационный поиск (коррекция запросов)
- Программы проверки правописания в текстовых редакторах
- Нормализация текста (коррекция стилистических вариантов, актуальна для социальных медиа)

Применения исправления опечаток

- Информационный поиск (коррекция запросов)
- Программы проверки правописания в текстовых редакторах
- Нормализация текста (коррекция стилистических вариантов, актуальна для социальных медиа)
- Распознавание звучащей речи, изображений...

Применения исправления опечаток

- Информационный поиск (коррекция запросов)
- Программы проверки правописания в текстовых редакторах
- Нормализация текста (коррекция стилистических вариантов, актуальна для социальных медиа)
- 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).

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).
  - Региональная/стилистическая вариативность (“colour” $\leftrightarrow$ “color”, “dialogue” $\leftrightarrow$ “dialog”).

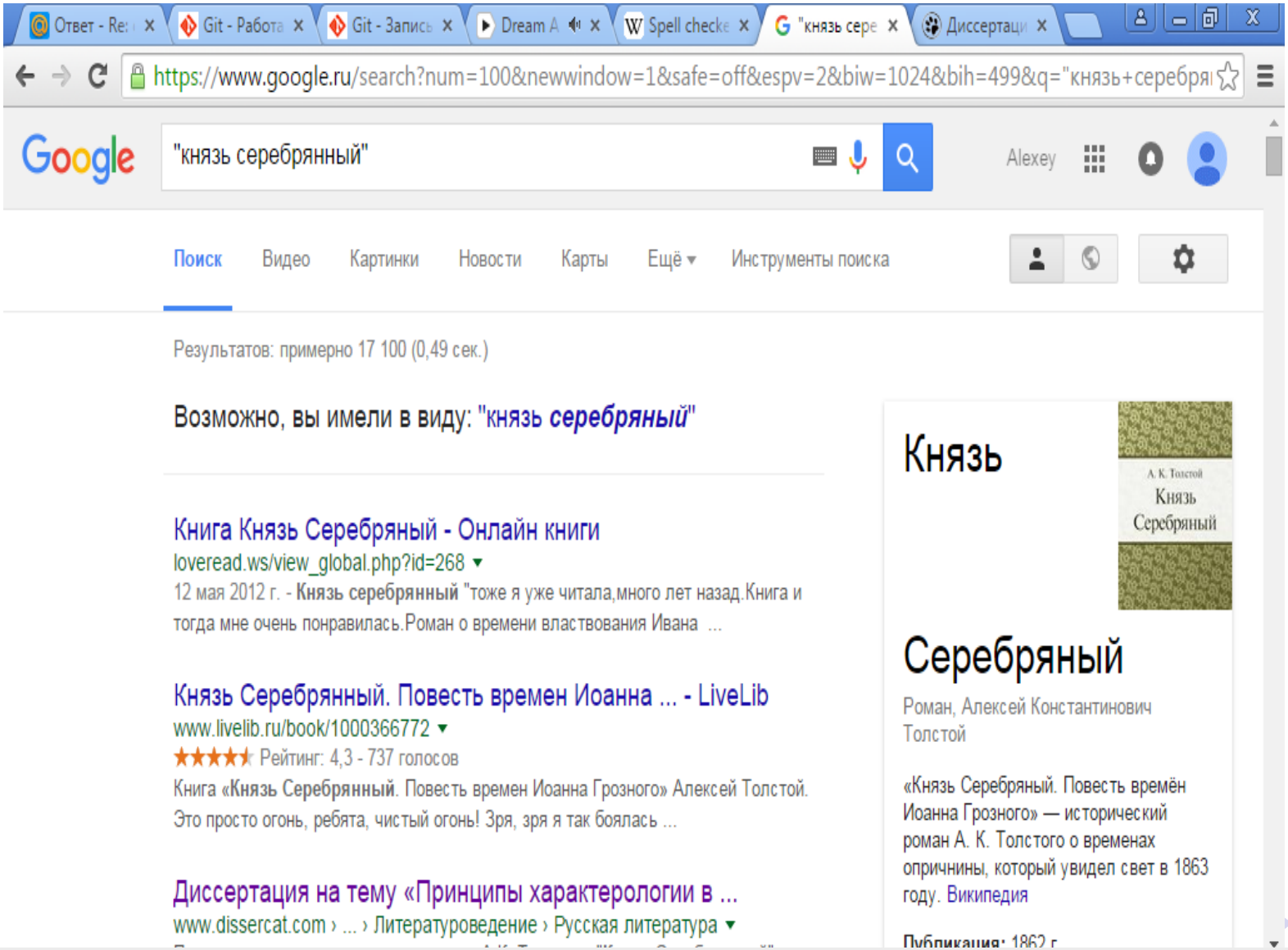
Причины опечаток

- 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).
  - Региональная/стилистическая вариативность (“colour” $\leftrightarrow$ “color”, “dialogue” $\leftrightarrow$ “dialog”).
- 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).
  - Региональная/стилистическая вариативность (“colour” $\leftrightarrow$ “color”, “dialogue” $\leftrightarrow$ “dialog”).
- Чаще всего ошибки локальны (затрагивают один-два символа)
- Однако может влиять и более широкий контекст (“ться” $\rightarrow$ *“цца”, “ян” $\rightarrow$ “янн/енн” в суффиксе прилагательного)

Пример


Ответ - Re
Git - Работа
Git - Запись
Dream A
Spell checke
"князь сере
Диссертаци
https://www.google.ru/search?num=100&newwindow=1&safe=off&espv=2&biw=1024&bih=499&q="князь+серебря
Google
"князь серебрянный"
Alexey
Поиск
Видео
Картинки
Новости
Карты
Ещё
Инструменты поиска
Результатов: примерно 17 100 (0,49 сек.)
Возможно, вы имели в виду: "князь серебряный"
Книга Князь Серебряный - Онлайн книги
loveread.ws/view_global.php?id=268
12 мая 2012 г. - Князь серебрянный "тоже я уже читала,много лет назад.Книга и тогда мне очень понравилась.Роман о времени властвования Ивана ...
Князь Серебрянный. Повесть времен Иоанна ... - LiveLib
www.livelib.ru/book/1000366772
Рейтинг: 4,3 - 737 голосов
Книга «Князь Серебрянный. Повесть времен Иоанна Грозного» Алексей Толстой. Это просто огонь, ребята, чистый огонь! Зря, зря я так боялась ...
Диссертация на тему «Принципы характерологии в ...
www.dissercat.com › ... › Литературоведение › Русская литература
Князь
А. К. Толстой
Князь Серебряный
Серебряный
Роман, Алексей Константинович Толстой
«Князь Серебряный. Повесть времён Иоанна Грозного» — исторический роман А. К. Толстого о временах опричнины, который увидел свет в 1863 году. Википедия
Публикация: 1862 г.

Методы исправления опечаток

- Как исправлять опечатки?

Методы исправления опечаток

- Как исправлять опечатки?
- Можно искать близкие слова в словаре.

Методы исправления опечаток

- Как исправлять опечатки?
- Можно искать близкие слова в словаре.
- Что значит “близкие”?
- Требуется задать функцию расстояния на множестве слов.

Методы исправления опечаток

- Как исправлять опечатки?
- Можно искать близкие слова в словаре.
- Что значит “близкие”?
- Требуется задать функцию расстояния на множестве слов.
- Кроме того хочется, чтобы эту функцию было легко вычислять или хотя бы оценивать.

Модель близости слов

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.

Модель близости слов

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.

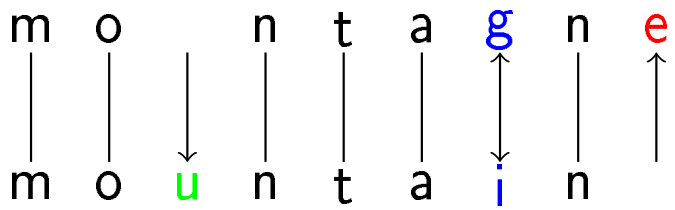
Появилась в работах В. И. Левенштейна (1965) и F. Damerau (1964).

Модель близости слов

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.

Появилась в работах В. И. Левенштейна (1965) и F. Damerau (1964).

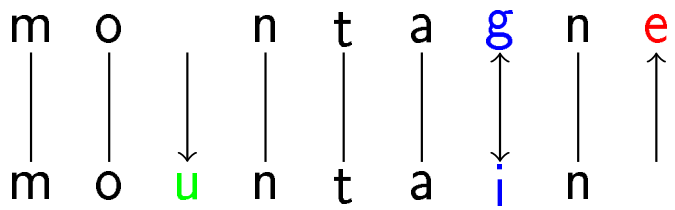


Модель близости слов

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.

Появилась в работах В. И. Левенштейна (1965) и F. Damerau (1964).



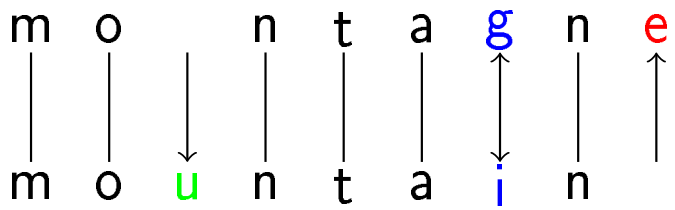
$$d(montagne, mountain) = 3$$

Модель близости слов

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.

Появилась в работах В. И. Левенштейна (1965) и F. Damerau (1964).



$$d(montagne, mountain) = 3$$

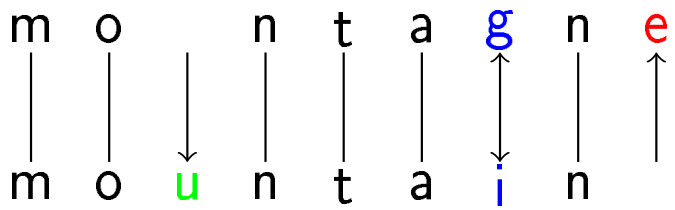
- Можно добавить перестановку соседних символов (с некоторыми ограничениями).

Модель близости слов

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.

Появилась в работах В. И. Левенштейна (1965) и F. Damerau (1964).



$$d(montagne, mountain) = 3$$

- Можно добавить перестановку соседних символов (с некоторыми ограничениями).
- Можно по-разному “штрафовать” за разные изменения.

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

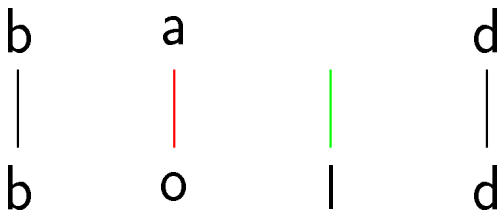
$$
\begin{aligned}
\rho(u[,i],v[,0]) &= i,\\
\rho(u[,0],v[,j]) &= j,\\
\rho(u[,i],v[,j]) &= \min\big(\rho(u[,i-1],v[,j-1]) + [\![u[i-1] \neq v[j-1]]\!],\\
&\qquad \rho(u[,i],v[,j-1]) + 1,\\
&\qquad \rho(u[,i-1],v[,j]) + 1\big)
\end{aligned}
$$

Рекурсивная формула

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

Возможные случаи:

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

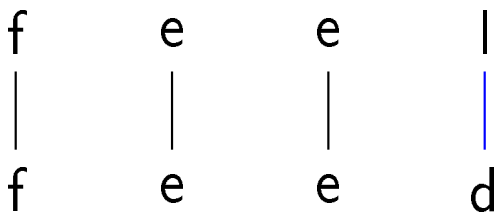


Рекурсивная формула

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

Возможные случаи:

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



Рекурсивная формула

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

Возможные случаи:

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

s	∅	l	i	d	e
|	|	|	|	|	|
s	o	l	i	d	∅

Рекурсивная формула

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

Возможные случаи:

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

Окончательная рекуррентная формула

Лемма

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

Окончательная рекуррентная формула

Окончательная формула для расстояния Левенштейна:

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

Окончательная рекуррентная формула

Окончательная формула для расстояния Левенштейна:

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

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] + 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]
```

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1						
т	2	2						
о	3	3						
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1						
т	2	2						
о	3	3						
р	4	4						
а	5	5						

$$T_{11} = \min(T_{00}, T_{01}, T_{10}) + 1 = 1 \text{ (т.к. с} \neq \text{ш)}.$$

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1					
т	2	2						
о	3	3						
р	4	4						
а	5	5						

$$T_{11} = \min(T_{00}, T_{01}, T_{10}) + 1 = 1 \text{ (т.к. с} \neq \text{ш)}.$$

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1					
т	2	2						
о	3	3						
р	4	4						
а	5	5						

Поскольку “ш” не входит в слово “строка”, получаем:

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2						
о	3	3						
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2						
о	3	3						
р	4	4						
а	5	5						

$$T_{21} = \min\left(T_{20}, T_{10}, T_{11}\right) + 1 = 2$$

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2					
о	3	3						
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2					
о	3	3						
р	4	4						
а	5	5						

$T_{22} = T_{11} = 1$, поскольку $u[1] = v[1] = \text{т}$

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1				
о	3	3						
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1				
о	3	3						
р	4	4						
а	5	5						

Дальше заполняем строку по алгоритму:

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3						
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3						
р	4	4						
а	5	5						

Заполняем третью строку:

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2			
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2			
р	4	4						
а	5	5						

Поскольку $u[2] = v[3] = \text{о}$, то $T_{34} = T_{23} = 2$

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2	2		
р	4	4						
а	5	5						

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2	2		
р	4	4						
а	5	5						

Действуя по алгоритму, заполняем таблицу до конца:

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2	2	3	4
р	4	4	4	3	2	3	3	4
а	5	5	5	4	3	3	4	3

Пример вычисления расстояния Левенштейна

$$d(\text{штора}, \text{строка}) = ?$$

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2	2	3	4
р	4	4	4	3	2	3	3	4
а	5	5	5	4	3	3	4	3

Таким образом, $d(\text{строка}, \text{штора}) = T_{65} = 3$

Восстановление оптимального выравнивания

По таблице можно найти оптимальное выравнивание:

Восстановление оптимального выравнивания

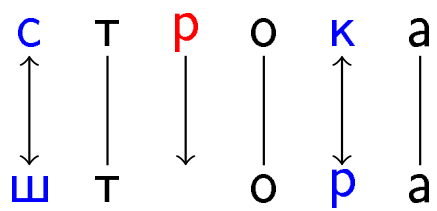
По таблице можно найти оптимальное выравнивание:

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2	2	3	4
р	4	4	4	3	2	3	3	4
а	5	5	5	4	3	3	4	3

Восстановление оптимального выравнивания

По таблице можно найти оптимальное выравнивание:

			с	т	р	о	к	а
		0	1	2	3	4	5	6
	0	0	1	2	3	4	5	6
ш	1	1	1	2	3	4	5	6
т	2	2	2	1	2	3	4	5
о	3	3	3	2	2	2	3	4
р	4	4	4	3	2	3	3	4
а	5	5	5	4	3	3	4	3

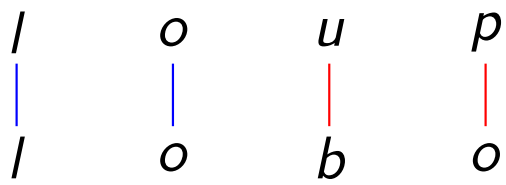


Взвешенное расстояние Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:

Взвешенное расстояние Левенштейна

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



Взвешенное расстояние Левенштейна

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

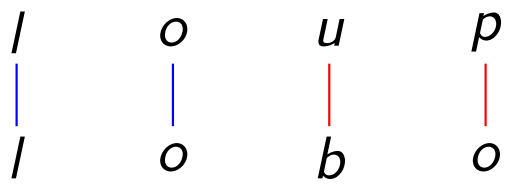
l	o	u	p
l	o	b	o

- Естественное выравнивание даёт $d = 3$:

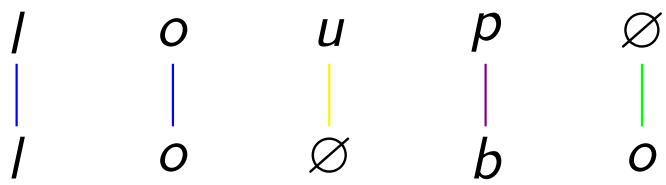
l	o	u	p	$\varnothing$
l	o	$\varnothing$	b	o

Взвешенное расстояние Левенштейна

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



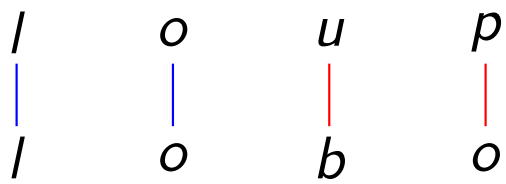
- Естественное выравнивание даёт $d = 3$:



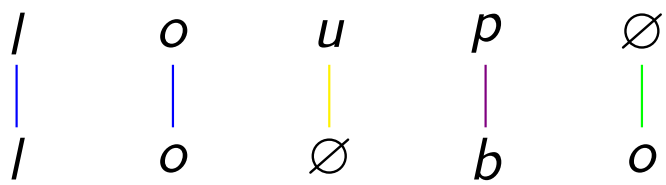
- Надо присвоить различным операциям веса, зависящие от заменяемых/удаляемых/вставляемых символов.

Взвешенное расстояние Левенштейна

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



- Естественное выравнивание даёт $d = 3$:



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

Применение к исправлению опечаток

- Как исправлять опечатки с помощью расстояния Левенштейна?

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:

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:
  - Словарь состоит из 119774 слов (стандартный словарь в /usr/share/dic в UNIX-системах),

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:
  - Словарь состоит из 119774 слов (стандартный словарь в /usr/share/dic в UNIX-системах),
  - Время расчёта (Intel Atom 1.67GHz):

Python3	Python3 (Levenshtein)	C++	C++ (-O3)
144.06	0.51	4.01	1.05

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:
  - Словарь состоит из 119774 слов (стандартный словарь в /usr/share/dic в UNIX-системах),
  - Время расчёта (Intel Atom 1.67GHz):

Python3	Python3 (Levenshtein)	C++	C++ (-O3)
144.06	0.51	4.01	1.05

- Менее наивный подход: породить все слова, расстояние до которых меньше порога d, найти их в словаре.

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:
  - Словарь состоит из 119774 слов (стандартный словарь в /usr/share/dic в UNIX-системах),
  - Время расчёта (Intel Atom 1.67GHz):

Python3	Python3 (Levenshtein)	C++	C++ (-O3)
144.06	0.51	4.01	1.05

- Менее наивный подход: породить все слова, расстояние до которых меньше порога d, найти их в словаре.
- Как порождать такие слова? С помощью конечных автоматов и преобразователей!

Применение к исправлению опечаток

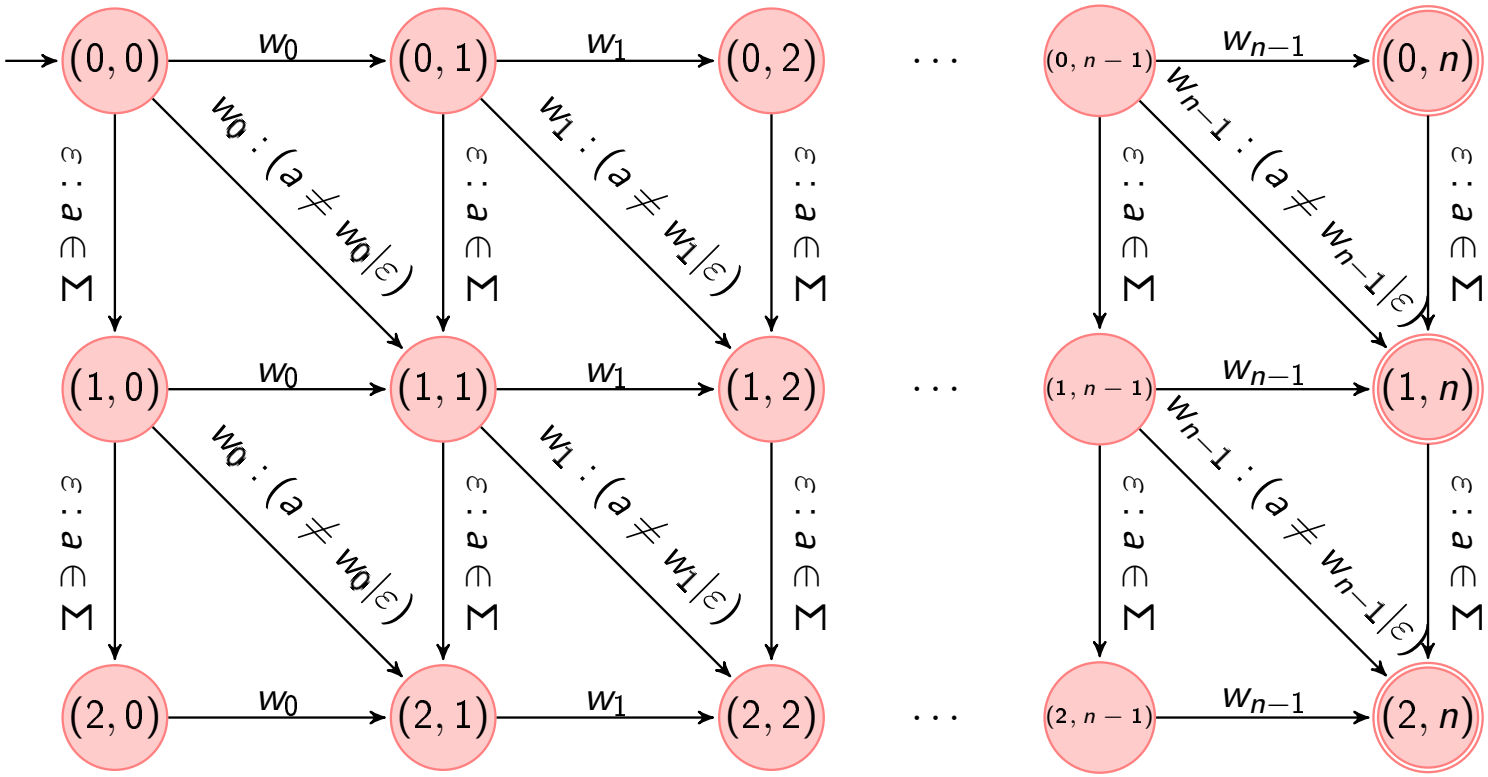
- Как исправлять опечатки с помощью расстояния Левенштейна?
- Наивный подход: пройти по словарю, подсчитать расстояние до каждого слова, выбрать ближайшее.
- Нельзя: словари большие (агглютинативные языки — сотни тысяч слов), а расстояние считается медленно:
  - Словарь состоит из 119774 слов (стандартный словарь в /usr/share/dic в UNIX-системах),
  - Время расчёта (Intel Atom 1.67GHz):

Python3	Python3 (Levenshtein)	C++	C++ (-O3)
144.06	0.51	4.01	1.05

- Менее наивный подход: породить все слова, расстояние до которых меньше порога d, найти их в словаре.
- Как порождать такие слова? С помощью конечных автоматов и преобразователей!
- Мы пока что не будем определять, что это такое.

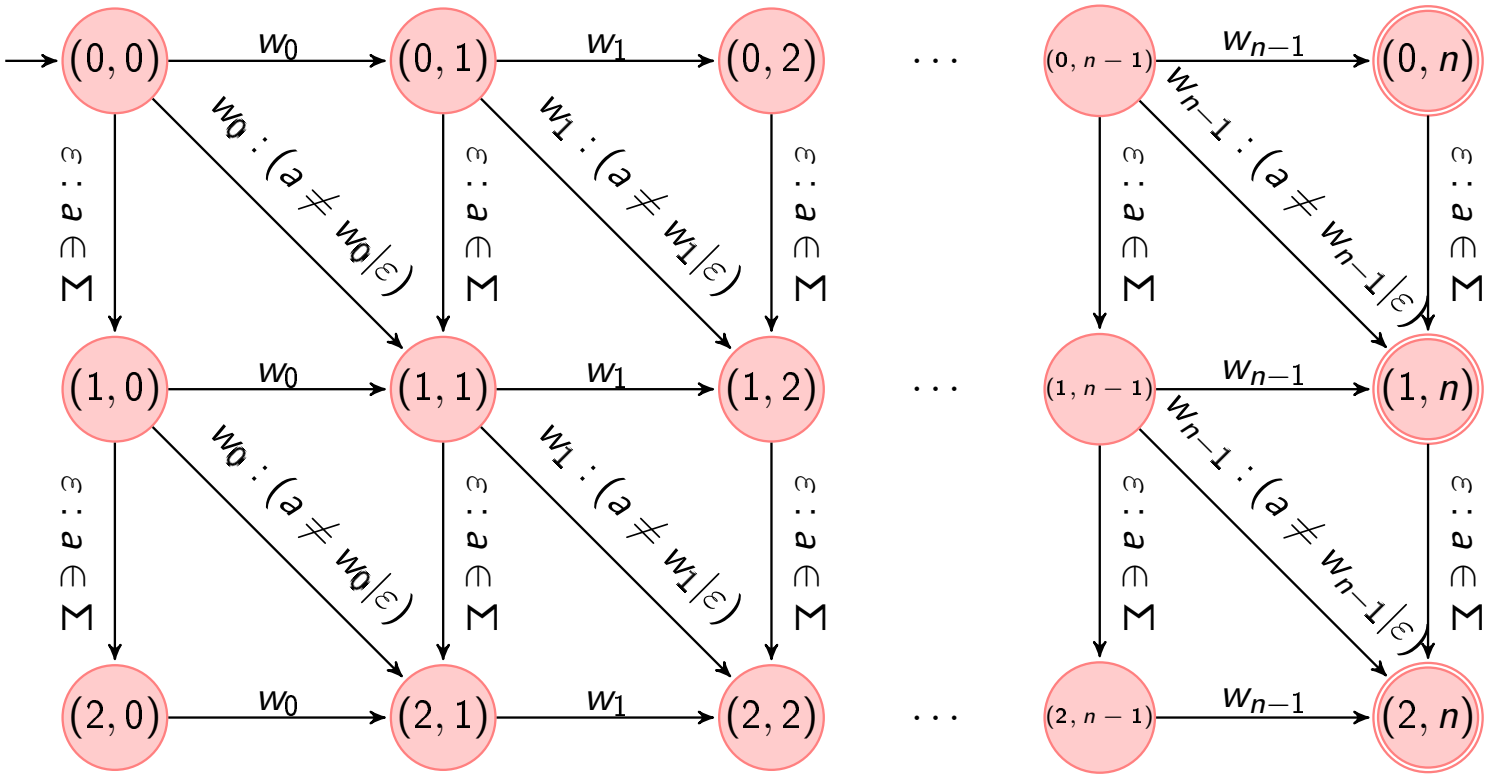
Автомат Левенштейна

Конечный преобразователь, порождающий все слова, находящиеся на расстоянии не больше d от слова $w = w_0 \dots w_{n-1}$.



Автомат Левенштейна

Конечный преобразователь, порождающий все слова, находящиеся на расстоянии не больше d от слова $w = w_0 \dots w_{n-1}$.



Состояние (i, j) принимает все слова, находящиеся на расстоянии i от префикса $w[, j]$

Применение автомата Левенштейна

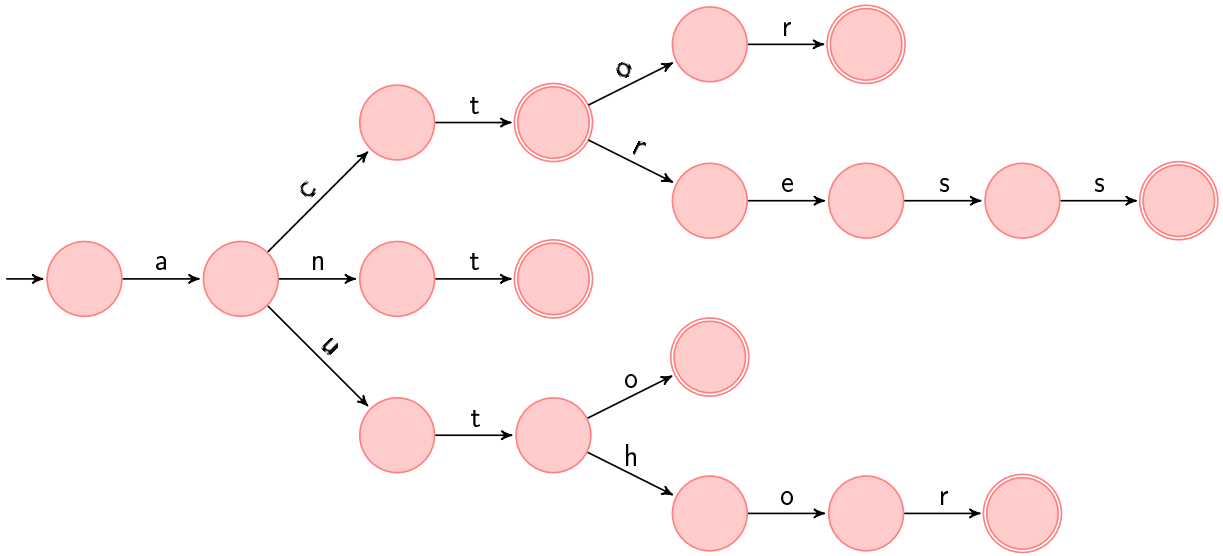
- Но считать расстояние Левенштейна можно и без автомата... Зачем он?

Применение автомата Левенштейна

- Но считать расстояние Левенштейна можно и без автомата... Зачем он?
- Экономия по памяти (при $d > 1$ список слов слишком длинный).

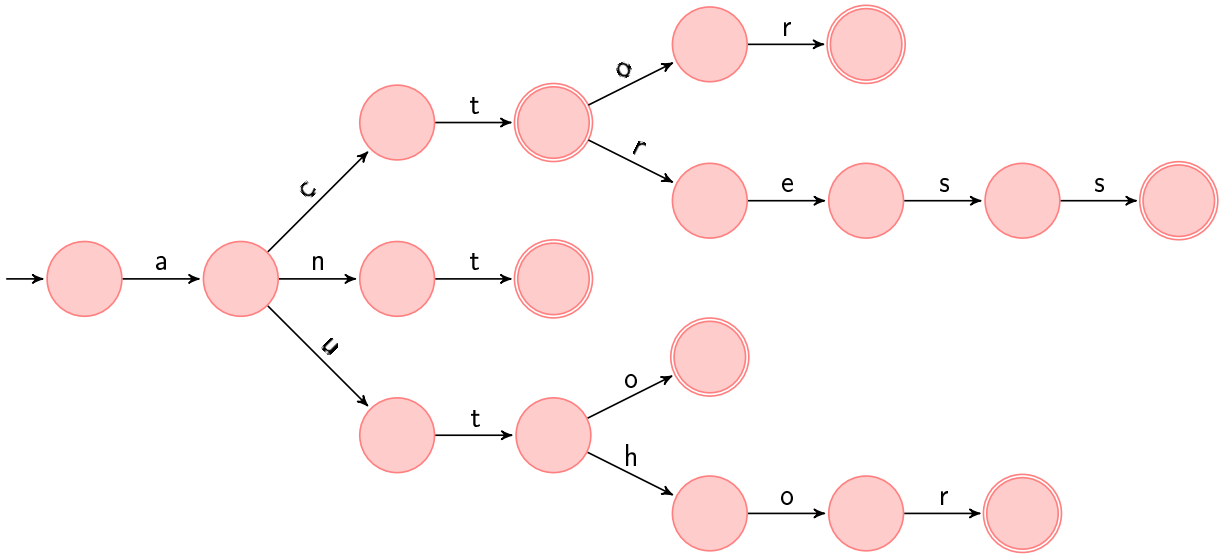
Применение автомата Левенштейна

- Но считать расстояние Левенштейна можно и без автомата... Зачем он?
- Экономия по памяти (при $d > 1$ список слов слишком длинный).
- Сам словарь тоже часто хранят в форме ациклического автомата (префиксный бор):



Применение автомата Левенштейна

- Но считать расстояние Левенштейна можно и без автомата... Зачем он?
- Экономия по памяти (при $d > 1$ список слов слишком длинный).
- Сам словарь тоже часто хранят в форме ациклического автомата (префиксный бор):



- Тогда можно параллельно идти по обоим автоматам...

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) автомат $A_{w,d}$ для текущего слова w и порога расстояния d.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря.
- Вернуть все или выбрать наиболее удачное (каким образом).

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря.
- Вернуть все или выбрать наиболее удачное (каким образом).
- Как инкорпорировать в эту модель веса операций?

Алгоритм поиска кандидатов

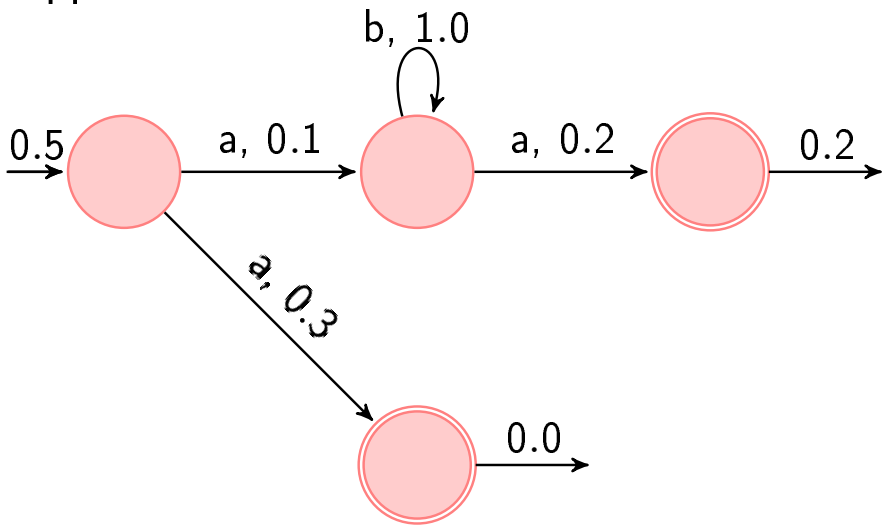
Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря.
- Вернуть все или выбрать наиболее удачное (каким образом).
- Как инкорпорировать в эту модель веса операций?
- Можно добавить веса на рёбра автомата...

Взвешенные конечные автоматы

Пример взвешенного автомата:

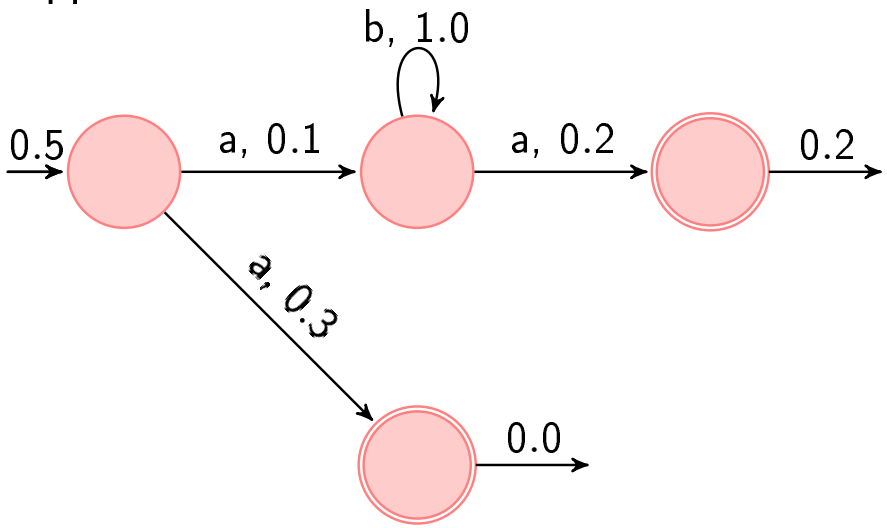
Данный автомат принимает слово a с весом 0.8, а также слова вида $ab^k a$ с весами $k + 1.0$.



Взвешенные конечные автоматы

Пример взвешенного автомата:

Данный автомат принимает слово a с весом 0.8, а также слова вида $ab^k a$ с весами $k + 1.0$.



Так же можно взвешивать и преобразователи. Но там много тонкостей... Нам важно, что при композиции (и пересечении) веса складываются.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) взвешенный автомат/преобразователь $A_{w,d}$ данного слова w и порога расстояния d.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) взвешенный автомат/преобразователь $A_{w,d}$ данного слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря вместе с их весами в автомате $A_{w,d}$.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) взвешенный автомат/преобразователь $A_{w,d}$ данного слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря вместе с их весами в автомате $A_{w,d}$.
- Выбрать кандидата (или несколько) наименьшего веса.

Алгоритм поиска кандидатов

Алгоритм поиска слов-кандидатов:

- Построить (можно неявно) взвешенный автомат/преобразователь $A_{w,d}$ данного слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$, где A_{dict} — автомат для словаря вместе с их весами в автомате $A_{w,d}$.
- Выбрать кандидата (или несколько) наименьшего веса.
- Откуда взять веса?

Вероятностная модель взвешивания

- Вспомним, что на рёбрах написаны операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.

Вероятностная модель взвешивания

- Вспомним, что на рёбрах написаны операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно как-то оценить.

Вероятностная модель взвешивания

- Вспомним, что на рёбрах написаны операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно как-то оценить.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.

Вероятностная модель взвешивания

- Вспомним, что на рёбрах написаны операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно как-то оценить.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.
- Маленьким вероятностям соответствуют большие веса.

Вероятностная модель взвешивания

- Вспомним, что на рёбрах написаны операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно как-то оценить.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.
- Маленьким вероятностям соответствуют большие веса.
- Положим входные и выходные веса равными 0.

Вероятностная модель взвешивания

- Вспомним, что на рёбрах написаны операции вида $\beta : \alpha$, соответствующие «опечаткам» вида $\alpha \to \beta$.
- У каждой такой операции есть вероятность, которую можно как-то оценить.
- Положим $w(\beta : \alpha) = -\log p(\alpha \to \beta)$.
- Маленьким вероятностям соответствуют большие веса.
- Положим входные и выходные веса равными 0.
- Какой в этом смысл?

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)...(a_r : b_r, w_r)$ — путь наименьшего веса в преобразователе M, «исправляющий» слово $u = a_1 \dots a_r$ на $v = b_1 \dots b_r$, а $W(a_1 \dots a_r : b_1 \dots b_r)$ — стоимость этого пути.

Вероятностная модель взвешивания: обоснование

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

Вероятностная модель взвешивания: обоснование

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

Вероятностная модель взвешивания: обоснование

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

Вероятностная модель взвешивания: обоснование

- Пусть $(a_1 : b_1, w_1)...(a_r : b_r, w_r)$ — путь наименьшего веса в преобразователе M, «исправляющий» слово $u = a_1 \dots a_r$ на $v = b_1 \dots b_r$, а $W(a_1 \dots a_r : b_1 \dots b_r)$ — стоимость этого пути.
- Тогда $W(a_1 \dots a_r : b_1 \dots b_r) = \sum_{i=1}^{r} w_i = -\sum_{i=1}^{r} \log p(b_i \to a_i) = -\log \prod_{i=1}^{r} p(b_i \to a_i) \approx -\log p(b_1 \dots b_r \to a_1 \dots a_r) \approx -\log p(v \to u)$.
- 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: $c = \operatorname{argmax}_v p(v|u)$.

Вероятностная модель исправления опечаток

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

Вероятностная модель исправления опечаток

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

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

Вероятностная модель исправления опечаток

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

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

- $p(u)$ не влияет на выбор кандидата, поэтому:

$$c = \operatorname*{argmax}_v p(u|v)p(v)$$

Вероятностная модель исправления опечаток

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

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

- $p(u)$ не влияет на выбор кандидата, поэтому:

$$c = \operatorname*{argmax}_v p(u|v)p(v)$$

- В модели весов это соответствует формуле $W(a_1 \dots a_r : b_1 \dots b_r) = -\sum_{i=1}^{r} \log p(b_i \to a_i) - \log p(v)$, где $v = b_1 \dots b_r$ — искомое словарное слово.

Вероятностная модель исправления опечаток

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

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

- $p(u)$ не влияет на выбор кандидата, поэтому:

$$c = \operatorname*{argmax}_v p(u|v)p(v)$$

- В модели весов это соответствует формуле $W(a_1 \dots a_r : b_1 \dots b_r) = -\sum_{i=1}^{r} \log p(b_i \to a_i) - \log p(v)$, где $v = b_1 \dots b_r$ — искомое словарное слово.
- Добавим $-\log p(v)$ в качестве выходного веса в автомат A_{dict} в вершину, соответствующую слову v.

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 $-\log p(c)$.
- Построить (можно неявно) взвешенный автомат $A_{w,d}$ для текущего слова w и порога расстояния d.

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 $-\log p(c)$.
- Построить (можно неявно) взвешенный автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$ с их весами.

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 $-\log p(c)$.
- Построить (можно неявно) взвешенный автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$ с их весами.
- Вернуть кандидата/несколько кандидатов наименьшего веса.

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 $-\log p(c)$.
- Построить (можно неявно) взвешенный автомат $A_{w,d}$ для текущего слова w и порога расстояния d.
- Вернуть список слов из языка $L(A_{dict} \cap A_{w,d})$ с их весами.
- Вернуть кандидата/несколько кандидатов наименьшего веса.
- Последняя проблема: как вычислять веса?

Вероятности опечаток

- Вероятности замен можно настроить по корпусу опечаток или с помощью эвристик.

Вероятности опечаток

- Вероятности замен можно настроить по корпусу опечаток или с помощью эвристик.
- Эвристики: что влияет на веса замен:
  - Опечатки: расположение символов на клавиатуре

Вероятности опечаток

- Вероятности замен можно настроить по корпусу опечаток или с помощью эвристик.
- Эвристики: что влияет на веса замен:
  - Опечатки: расположение символов на клавиатуре
  - Орфографические ошибки: фонетическая близость

Вероятности опечаток

- Вероятности замен можно настроить по корпусу опечаток или с помощью эвристик.
- Эвристики: что влияет на веса замен:
  - Опечатки: расположение символов на клавиатуре
  - Орфографические ошибки: фонетическая близость
  - Ошибки распознавания: графическая близость

Вероятности опечаток

- Вероятности замен можно настроить по корпусу опечаток или с помощью эвристик.
- Эвристики: что влияет на веса замен:
  - Опечатки: расположение символов на клавиатуре
  - Орфографические ошибки: фонетическая близость
  - Ошибки распознавания: графическая близость
- Веса можно настраивать и автоматически по обычному корпусу (*EM*-алгоритм), хотя это достаточно трудоёмко.

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`
- `http://web.stanford.edu/class/cs124/lec/spelling.pdf` — лекция Дэниэла Журафски по исправлению опечаток.