Исправление опечаток

Алексей Сорокин

МГУ им. М. В. Ломоносова, филологический факультет,
отделение теоретической и прикладной лингвистики
весенний семестр 2015–2016 учебного года

Исправление опечаток

- Задача исправления опечаток: восстановление правильного написания в случае случайного или намеренного искажения.

Исправление опечаток

- Задача исправления опечаток: восстановление правильного написания в случае случайного или намеренного искажения.
- Случайное искажение: *клрова* (корова), *мнея* (меня).

Исправление опечаток

- Задача исправления опечаток: восстановление правильного написания в случае случайного или намеренного искажения.
- Случайное искажение: *клрова* (корова), *мнея* (меня).
- Намеренные искажения: орфографические ошибки, когнитивные ошибки.

Исправление опечаток

- Задача исправления опечаток: восстановление правильного написания в случае случайного или намеренного искажения.
- Случайное искажение: *клрова* (корова), *мнея* (меня).
- Намеренные искажения: орфографические ошибки, когнитивные ошибки.
- В 80% процентов случаев исправление находится на расстоянии 1 по Левенштейну.

Исправление опечаток

- Задача исправления опечаток: восстановление правильного написания в случае случайного или намеренного искажения.
- Случайное искажение: *клрова* (корова), *мнея* (меня).
- Намеренные искажения: орфографические ошибки, когнитивные ошибки.
- В 80% процентов случаев исправление находится на расстоянии 1 по Левенштейну.
- Остальные случаи: омофония и неоднозначная транслитерация (*ph-f, th-s, ться-цца*).

Проблемы с исправлением опечаток

- Слов-кандидатов на расстоянии 1 по Левенштейну может быть много:

$$\textit{свией} \mapsto \textit{сваей}, \textit{своей}, \textit{свиней}, \ldots$$

Проблемы с исправлением опечаток

- Слов-кандидатов на расстоянии 1 по Левенштейну может быть много:

$$\textit{свией} \mapsto \textit{сваей, своей, свиней, } \ldots$$

- Как выбрать наилучшего кандидата?
- А кстати, как найти всех кандидатов?

Проблемы с исправлением опечаток

- Слов-кандидатов на расстоянии 1 по Левенштейну может быть много:

$$\textit{свией} \mapsto \textit{сваей, своей, свиней, } \ldots$$

- Как выбрать наилучшего кандидата?
- А кстати, как найти всех кандидатов?
- На вычисление расстояния между словами длины m и n нужно Cmn операций.

Проблемы с исправлением опечаток

- Слов-кандидатов на расстоянии 1 по Левенштейну может быть много:

$$\textit{свией} \mapsto \textit{сваей, своей, свиней, } \ldots$$

- Как выбрать наилучшего кандидата?
- А кстати, как найти всех кандидатов?
- На вычисление расстояния между словами длины m и n нужно Cmn операций.
- Значит, расстояние до всех словарных слов вычисляется за CmW, где W — суммарная длина всех слов словаря.

Проблемы с исправлением опечаток

- Слов-кандидатов на расстоянии 1 по Левенштейну может быть много:

$$\textit{свией} \mapsto \textit{сваей, своей, свиней, } \ldots$$

- Как выбрать наилучшего кандидата?
- А кстати, как найти всех кандидатов?
- На вычисление расстояния между словами длины m и n нужно Cmn операций.
- Значит, расстояние до всех словарных слов вычисляется за CmW, где W — суммарная длина всех слов словаря.
- Оценим величины: $m \approx 8, W \approx 8|D| \approx 8000000$, $C \sim 10$, всего ≈ 600000000 операций на слово.

Проблемы с исправлением опечаток

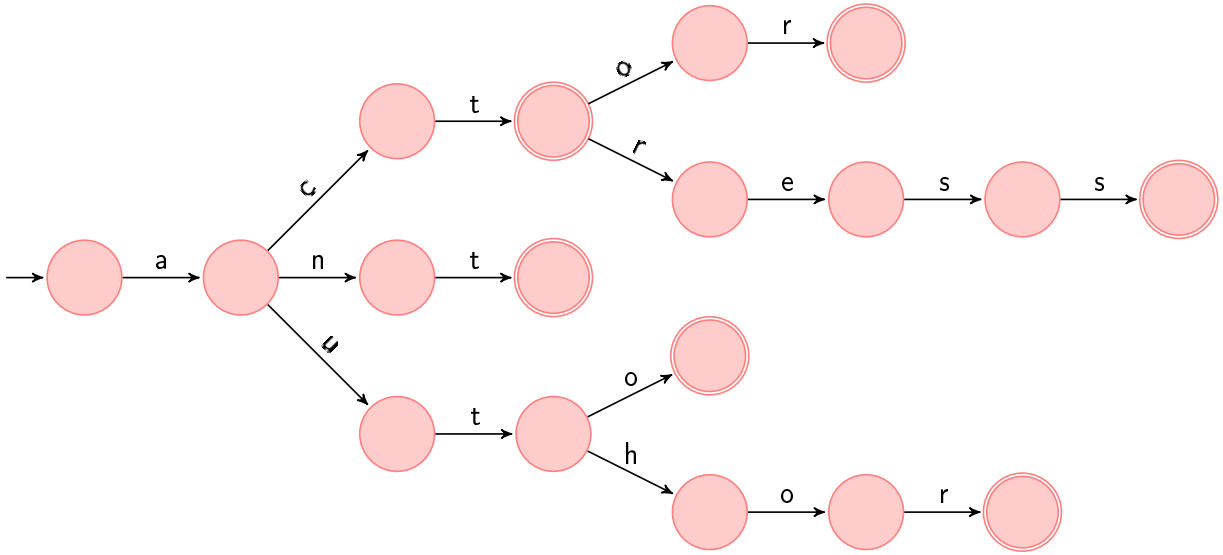
- Слов-кандидатов на расстоянии 1 по Левенштейну может быть много:

$$\textit{свией} \mapsto \textit{сваей, своей, свиней, } \ldots$$

- Как выбрать наилучшего кандидата?
- А кстати, как найти всех кандидатов?
- На вычисление расстояния между словами длины m и n нужно Cmn операций.
- Значит, расстояние до всех словарных слов вычисляется за CmW, где W — суммарная длина всех слов словаря.
- Оценим величины: $m \approx 8, W \approx 8|D| \approx 8000000$, $C \sim 10$, всего ≈ 600000000 операций на слово.
- Это неприемлемо!

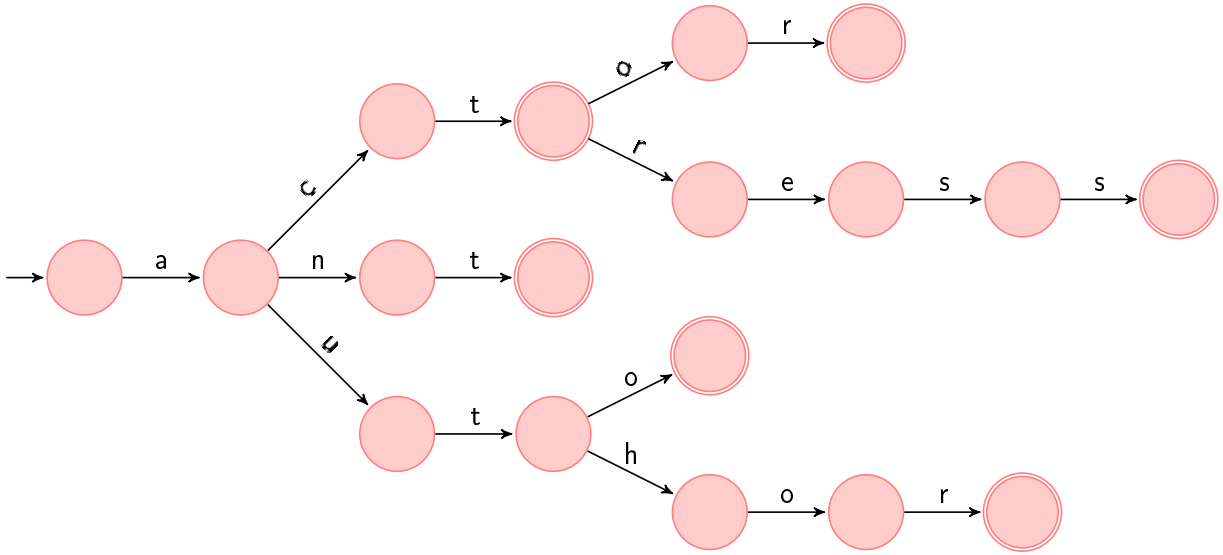
Хранение словаря

- Словарь хранится в виде префиксного бора



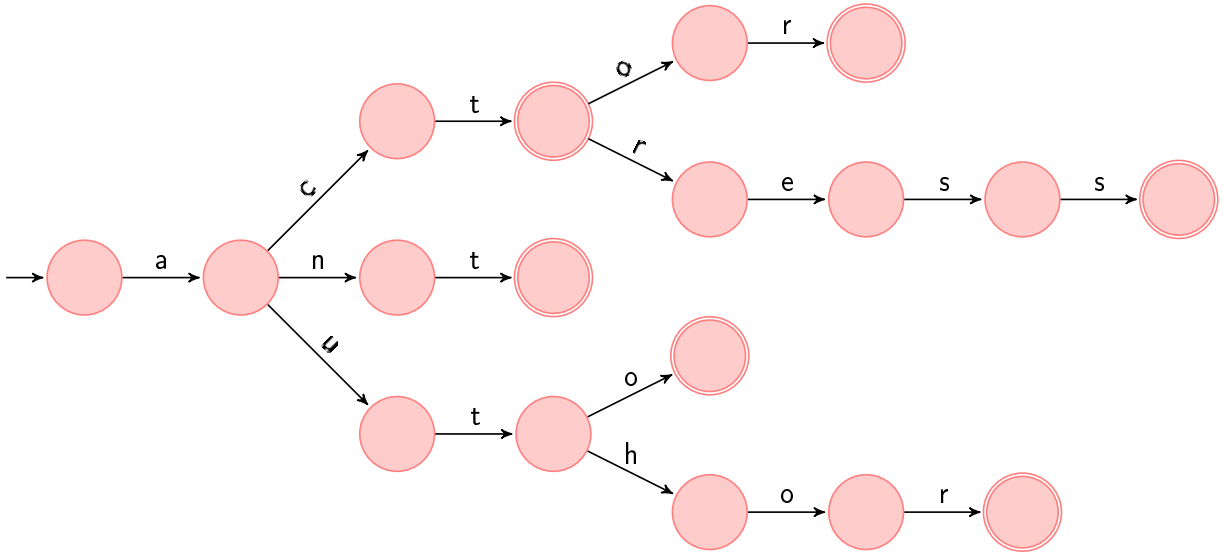
Хранение словаря

- Словарь хранится в виде префиксного бора



Хранение словаря

- Словарь хранится в виде префиксного бора



- Найти слово в таком дереве легко.
- Но нам нужно искать приближённо...

Алгоритм приближённого поиска в префиксном боре

Вход: w, T — префиксный бор со словарём, d — порог расстояния.
Выход: $L = \{(u, d(w,u)) \in T \mid d(w,u) < d\}$. ▷ список исправлений вместе с их стоимостью

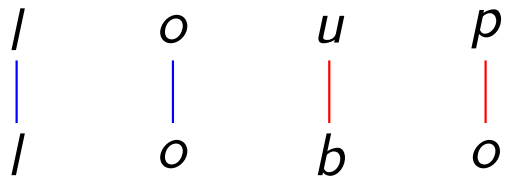
```
L = ∅                                                  ▷ текущий список исправлений
hyp = (0, ε, 0)                                        ▷ начальная гипотеза
▷ очередь с приоритетом для гипотез вида (позиция в w, префикс-кандидат, текущая стоимость)
H = [hyp]
while not H.isempty() do
    pos, u, cost = H.pop()                             ▷ извлекаем из H текущую гипотезу
    if u.isfinal() then                                ▷ текущая вершина — принимающая
        L.add((u, cost))
    end if
    ▷ Пробуем продвинуться в слове и боре на одну элементарную операцию
    if pos < |w| then
        for a ∈ Σ do                                                          ▷ замены
            if ua ∈ T  &  cost + c(a, w[pos]) ⩽ d then   ▷ есть ребро из q по a и стоимость мала
                H.add((pos + 1, ua, cost + c(w[pos], a))))
            end if
        end for
        if cost + c(w[pos], ε) ⩽ d then                                       ▷ вставка
            H.add((pos + 1, u, cost + c(ε, w[pos])))
        end if
    end if
    for a ∈ Σ do                                                              ▷ вставки
        if ua ∈ T  &  cost + c(ε, a) ⩽ d then ▷ есть ребро из q по a и стоимость не превышает порог
            H.add((pos, ua, cost + c(a, ε)))
        end if
    end for
end while
    return L
```

Недостатки расстояния Левенштейна

- Не всегда расстояние Левенштейна приводит к оптимальному выравниванию:

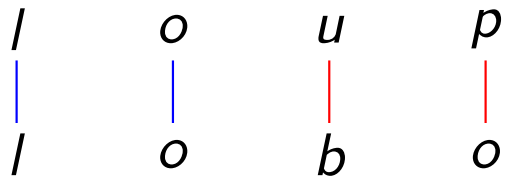
Недостатки расстояния Левенштейна

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

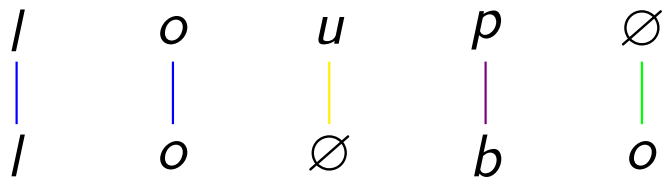


Недостатки расстояния Левенштейна

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

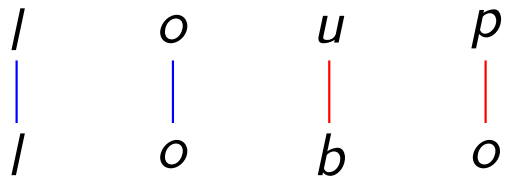


- Естественное выравнивание даёт $d = 3$:

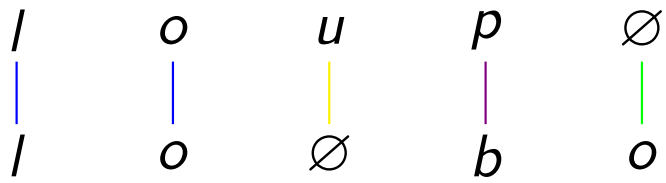


Недостатки расстояния Левенштейна

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



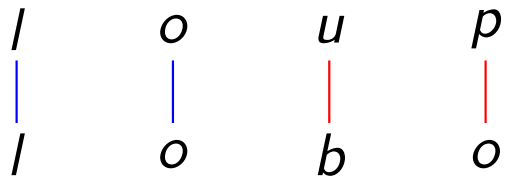
- Естественное выравнивание даёт $d = 3$:



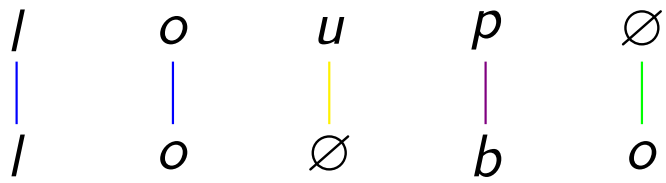
- Надо присвоить различным операциям веса, зависящие от заменяемых/удаляемых/вставляемых символов.

Недостатки расстояния Левенштейна

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



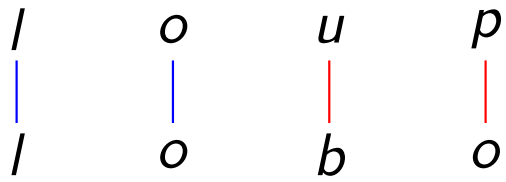
- Естественное выравнивание даёт $d = 3$:



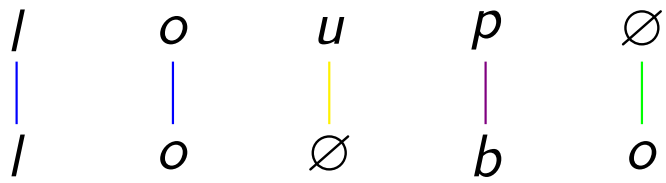
- Надо присвоить различным операциям веса, зависящие от заменяемых/удаляемых/вставляемых символов.
- Алгоритм вычисления расстояния от этого не изменится (при естественных ограничениях на веса).

Недостатки расстояния Левенштейна

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



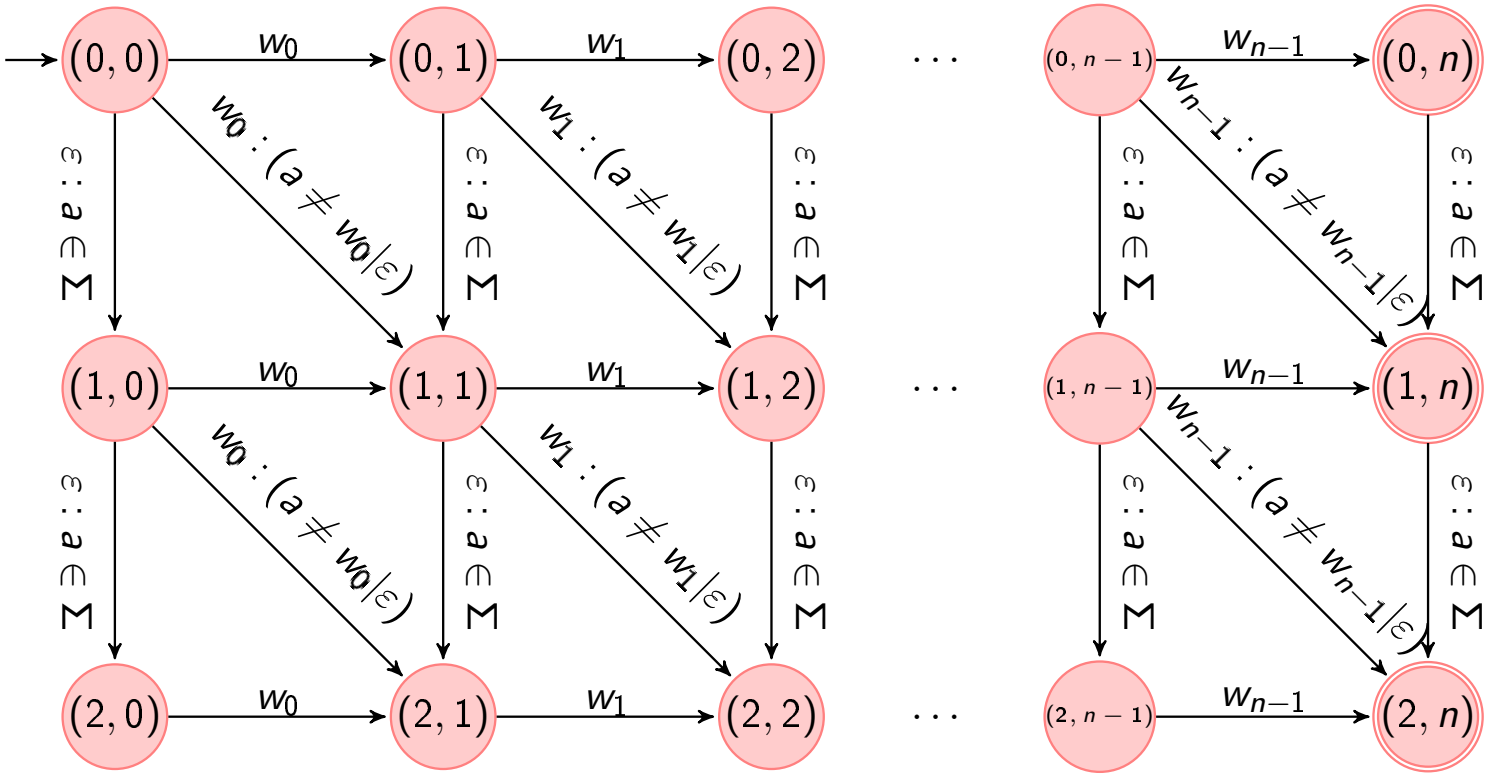
- Естественное выравнивание даёт $d = 3$:



- Надо присвоить различным операциям веса, зависящие от заменяемых/удаляемых/вставляемых символов.
- Алгоритм вычисления расстояния от этого не изменится (при естественных ограничениях на веса).
- Можно добавить веса и для неэлементарных операций ('ться' $\rightarrow$ 'цца').

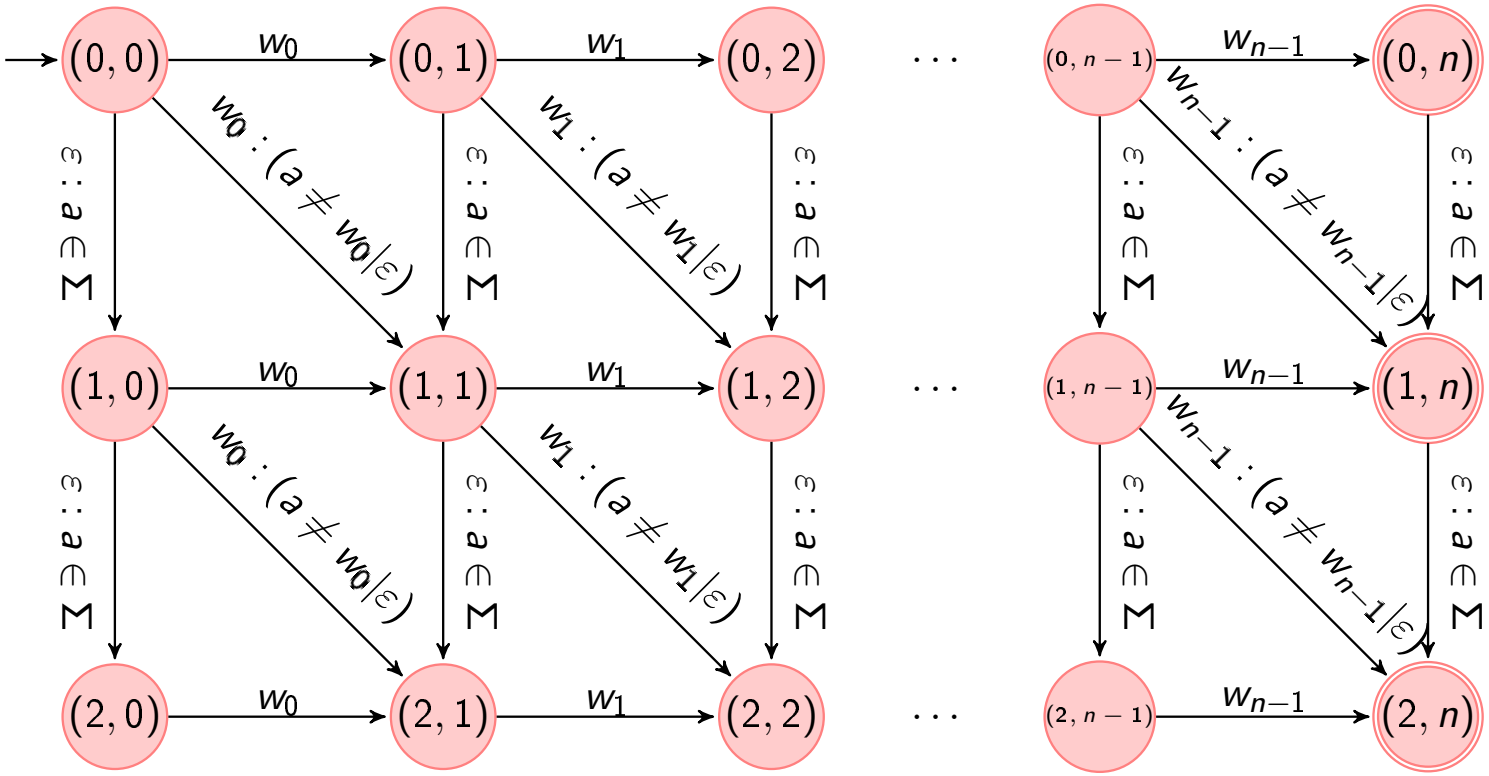
Автомат Левенштейна

Для порождения всех слов на расстоянии d и меньше от данного можно использовать конечный автомат.



Автомат Левенштейна

Для порождения всех слов на расстоянии d и меньше от данного можно использовать конечный автомат.



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

Поиск кандидата

- Префиксный бор — тоже конечный автомат. Тогда поиск слова-кандидата — это перечисление слов, одновременно принимаемых двумя конечными автоматами.

Поиск кандидата

- Префиксный бор — тоже конечный автомат. Тогда поиск слова-кандидата — это перечисление слов, одновременно принимаемых двумя конечными автоматами.
- Пересечение двух автоматных языков — тоже автоматный язык.

Поиск кандидата

- Префиксный бор — тоже конечный автомат. Тогда поиск слова-кандидата — это перечисление слов, одновременно принимаемых двумя конечными автоматами.
- Пересечение двух автоматных языков — тоже автоматный язык.
- Тогда алгоритм приближённого поиска, по сути, строит “на лету” автомат и перечисляет все принимаемые им слова.

Поиск кандидата

- Префиксный бор — тоже конечный автомат. Тогда поиск слова-кандидата — это перечисление слов, одновременно принимаемых двумя конечными автоматами.
- Пересечение двух автоматных языков — тоже автоматный язык.
- Тогда алгоритм приближённого поиска, по сути, строит “на лету” автомат и перечисляет все принимаемые им слова.
- При этом веса различных операций — это штрафы за проход по рёбрам автомата.

Взвешенные конечные автоматы

Определение

Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,

Взвешенные конечные автоматы

Определение

Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,
- $I, F \subseteq Q$ — множества начальных и конечных состояний,

Взвешенные конечные автоматы

Определение

Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,
- $I, F \subseteq Q$ — множества начальных и конечных состояний,
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{R} \times Q$ — множество переходов вида $\langle q_1, a \rangle \to \langle q_2, w \rangle$, где $w \in \mathbb{R}$ — вес перехода.

Взвешенные конечные автоматы

Определение

Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,
- $I, F \subseteq Q$ — множества начальных и конечных состояний,
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{R} \times Q$ — множество переходов вида $\langle q_1, a \rangle \to \langle q_2, w \rangle$, где $w \in \mathbb{R}$ — вес перехода.
- $\lambda \colon I \to \mathbb{R}$ — входные веса.

Взвешенные конечные автоматы

Определение

Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,
- $I, F \subseteq Q$ — множества начальных и конечных состояний,
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{R} \times Q$ — множество переходов вида $\langle q_1, a \rangle \to \langle q_2, w \rangle$, где $w \in \mathbb{R}$ — вес перехода.
- $\lambda \colon I \to \mathbb{R}$ — входные веса.
- $\rho \colon F \to \mathbb{R}$ — выходные веса.

Взвешенные конечные автоматы

Определение

Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,
- $I, F \subseteq Q$ — множества начальных и конечных состояний,
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{R} \times Q$ — множество переходов вида $\langle q_1, a \rangle \to \langle q_2, w \rangle$, где $w \in \mathbb{R}$ — вес перехода.
- $\lambda \colon I \to \mathbb{R}$ — входные веса.
- $\rho \colon F \to \mathbb{R}$ — выходные веса.

- Метка пути — конкатенация меток его рёбер. Вес пути $\pi = \langle e_1, \ldots, e_r \rangle$ из q_1 в q_2: $w(\pi) = \pi(q_1) + \sum_i^r w(e_i) + \rho(q_2)$.

Взвешенные конечные автоматы

Определение

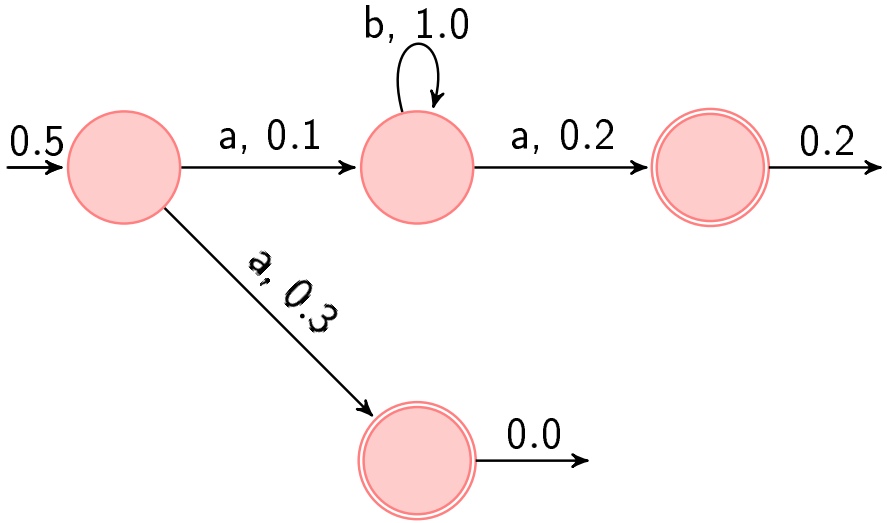
Взвешённый конечный автомат: кортеж $M = \langle Q, \Sigma, \Delta, I, F, \lambda, \rho \rangle$, где

- Q — множество состояний, Σ — конечный алфавит,
- $I, F \subseteq Q$ — множества начальных и конечных состояний,
- $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{R} \times Q$ — множество переходов вида $\langle q_1, a \rangle \to \langle q_2, w \rangle$, где $w \in \mathbb{R}$ — вес перехода.
- $\lambda \colon I \to \mathbb{R}$ — входные веса.
- $\rho \colon F \to \mathbb{R}$ — выходные веса.

- Метка пути — конкатенация меток его рёбер. Вес пути $\pi = \langle e_1, \ldots, e_r \rangle$ из q_1 в q_2: $w(\pi) = \pi(q_1) + \sum_i^r w(e_i) + \rho(q_2)$.
- $\Pi(u)$ — множество принимающих путей для слова u, тогда $M(u) = \min\limits_{\pi \in \Pi(u)} w(\pi)$ — вес слова u.

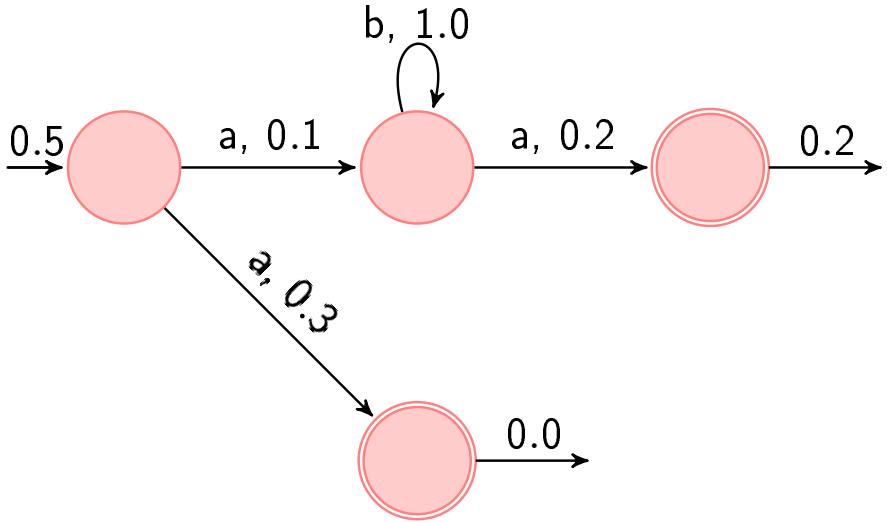
Пример взвешенного автомата

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



Пример взвешенного автомата

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

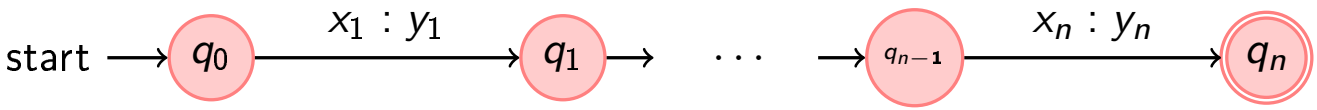


Так же можно взвешивать и преобразователи. Но там много тонкостей...

Вероятностная интерпретация

- Штраф за преобразование $x_1 \dots x_n$ в $y_1 \dots y_n$ при фиксированном разбиении:

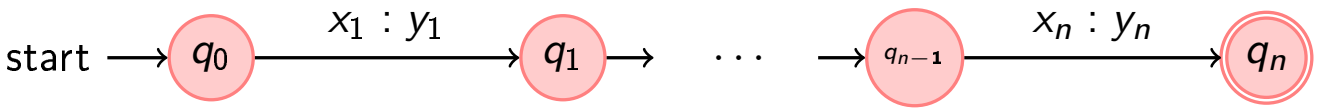
$$C(x_1 \dots x_n \to y_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \sum_{i=1}^{n} w(q_{i-1}, x_i, y_i)$$



Вероятностная интерпретация

- Штраф за преобразование $x_1 \dots x_n$ в $y_1 \dots y_n$ при фиксированном разбиении:

$$C(x_1 \dots x_n \to y_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \sum_{i=1}^{n} w(q_{i-1}, x_i, y_i)$$



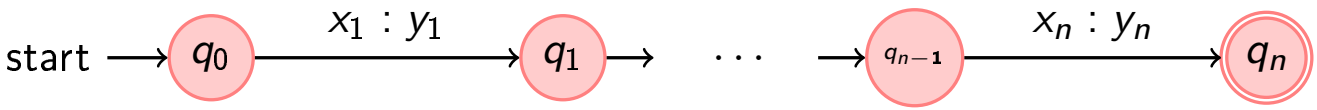
- Вероятность перехода $y_1 \dots x_n$ в $x_1 \dots y_n$ при фиксированном разбиении:

$$p(y_1 \dots x_n \to x_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \prod_{i=1}^{n} p(y_i \to x_i)$$

Вероятностная интерпретация

- Штраф за преобразование $x_1 \dots x_n$ в $y_1 \dots y_n$ при фиксированном разбиении:

$$C(x_1 \dots x_n \to y_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \sum_{i=1}^{n} w(q_{i-1}, x_i, y_i)$$



- Вероятность перехода $y_1 \dots x_n$ в $x_1 \dots y_n$ при фиксированном разбиении:

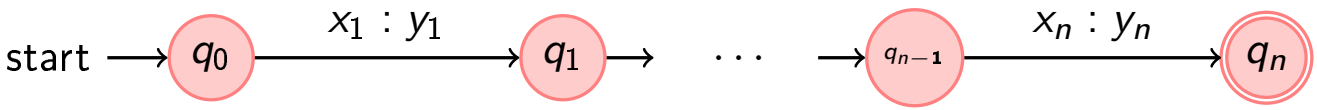
$$p(y_1 \dots x_n \to x_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \prod_{i=1}^{n} p(y_i \to x_i)$$

- Можно положить $w(q, x_i, y_i) = -\log p(y_i \to x_i)$.

Вероятностная интерпретация

- Штраф за преобразование $x_1 \dots x_n$ в $y_1 \dots y_n$ при фиксированном разбиении:

$$C(x_1 \dots x_n \to y_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \sum_{i=1}^{n} w(q_{i-1}, x_i, y_i)$$



- Вероятность перехода $y_1 \dots x_n$ в $x_1 \dots y_n$ при фиксированном разбиении:

$$p(y_1 \dots x_n \to x_1 \dots y_n | (x_1, y_1) \dots (x_n, y_n)) = \prod_{i=1}^{n} p(y_i \to x_i)$$

- Можно положить $w(q, x_i, y_i) = -\log p(y_i \to x_i)$.
- Как оценить вероятности $p(y_i \to x_i)$? Можно посчитать частоты исправлений...

Вычисление вероятностей замен

- Если есть корпус опечаток с исправлениями, то вероятности $p(y_i \to x_i)$ считаются естественным образом.
- Что делать если его нет, но есть большое число слов с опечатками?

Вычисление вероятностей замен

- Если есть корпус опечаток с исправлениями, то вероятности $p(y_i \to x_i)$ считаются естественным образом.
- Что делать если его нет, но есть большое число слов с опечатками?
- Эти вероятности можно посчитать с помощью аналога EM-алгоритма. Вначале вероятности всех нетождественных операций одинаковы или случайны.

Вычисление вероятностей замен

EM-алгоритм для вычисления вероятностей замен $p(y_i \to x_i)$. Вначале вероятности инициализированы произвольными значениями.

- Найти для слов $x_1, \ldots, x_n$ все возможные исправления с их вероятностями.

Вычисление вероятностей замен

EM-алгоритм для вычисления вероятностей замен $p(y_i \to x_i)$. Вначале вероятности инициализированы произвольными значениями.

- Найти для слов $x_1, \ldots, x_n$ все возможные исправления с их вероятностями.
- Получим список троек $\langle$слово, исправление, вероятность$\rangle$:

$$\begin{array}{ccc} w_1 & u_1 & p_1 \\ \ldots & \ldots & \ldots \\ w_n & u_n & p_n \end{array}$$

Вычисление вероятностей замен

EM-алгоритм для вычисления вероятностей замен $p(y_i \to x_i)$. Вначале вероятности инициализированы произвольными значениями.

- Найти для слов $x_1, \ldots, x_n$ все возможные исправления с их вероятностями.
- Получим список троек $\langle$слово, исправление, вероятность$\rangle$:

$$\begin{array}{ccc} w_1 & u_1 & p_1 \\ \ldots & \ldots & \ldots \\ w_n & u_n & p_n \end{array}$$

- Найти, сколько раз сегмент y переходит в сегменты $x_1, \ldots, x_r$ (при этом операции в преобразовании с вероятностью p считаются с весом p)

Вычисление вероятностей замен

EM-алгоритм для вычисления вероятностей замен $p(y_i \to x_i)$. Вначале вероятности инициализированы произвольными значениями.

- Найти для слов $x_1, \ldots, x_n$ все возможные исправления с их вероятностями.
- Получим список троек $\langle$слово, исправление, вероятность$\rangle$:

$$\begin{array}{ccc} w_1 & u_1 & p_1 \\ \ldots & \ldots & \ldots \\ w_n & u_n & p_n \end{array}$$

- Найти, сколько раз сегмент y переходит в сегменты $x_1, \ldots, x_r$ (при этом операции в преобразовании с вероятностью p считаются с весом p)
- По частотам $c(y \to x_j)$ вычислить вероятности $p(x_j \mid y)$.

Вычисление вероятностей замен

EM-алгоритм для вычисления вероятностей замен $p(y_i \to x_i)$. Вначале вероятности инициализированы произвольными значениями.

- Найти для слов $x_1, \ldots, x_n$ все возможные исправления с их вероятностями.
- Получим список троек $\langle$слово, исправление, вероятность$\rangle$:

$$\begin{array}{ccc} w_1 & u_1 & p_1 \\ \ldots & \ldots & \ldots \\ w_n & u_n & p_n \end{array}$$

- Найти, сколько раз сегмент y переходит в сегменты $x_1, \ldots, x_r$ (при этом операции в преобразовании с вероятностью p считаются с весом p)
- По частотам $c(y \to x_j)$ вычислить вероятности $p(x_j \mid y)$.
- Повторять, пока вероятности не перестанут меняться.

Вероятностная интерпретация

- Пусть мы ищем наиболее вероятное исправление u для w в контексте C:

$$u = \text{argmax}_{u \in D}\, p(u|w, C) = \text{argmax}_{u \in D}(p(w|u, C)p(u|C)$$

Вероятностная интерпретация

- Пусть мы ищем наиболее вероятное исправление u для w в контексте C:

$$u = \text{argmax}_{u \in D}\, p(u|w, C) = \text{argmax}_{u \in D}(p(w|u, C)p(u|C)$$

- Первое слагаемое — вероятность опечатки:

$$p(w|u, C) \approx \prod_{i=1}^{m} p(u_i \to w_i)$$

Здесь $(u_1, w_1), \ldots, (u_m, w_m)$ — оптимальное выравнивание между u и w.

Вероятностная интерпретация

- Пусть мы ищем наиболее вероятное исправление u для w в контексте C:

$$u = \text{argmax}_{u \in D}\, p(u|w, C) = \text{argmax}_{u \in D}(p(w|u, C)p(u|C)$$

- Первое слагаемое — вероятность опечатки:

$$p(w|u, C) \approx \prod_{i=1}^{m} p(u_i \to w_i)$$

Здесь $(u_1, w_1), \ldots, (u_m, w_m)$ — оптимальное выравнивание между u и w.

- $p(u|C)$ – вероятность встретить слово u в контексте C.

Вероятностная интерпретация

- Пусть мы ищем наиболее вероятное исправление u для w в контексте C:

$$u = \text{argmax}_{u \in D}\, p(u|w, C) = \text{argmax}_{u \in D}(p(w|u, C)p(u|C)$$

- Первое слагаемое — вероятность опечатки:

$$p(w|u, C) \approx \prod_{i=1}^{m} p(u_i \to w_i)$$

  Здесь $(u_1, w_1), \ldots, (u_m, w_m)$ — оптимальное выравнивание между u и w.
- $p(u|C)$ – вероятность встретить слово u в контексте C.
- Как её вычислять?

Подбор слова в зависимости от контекста

t : стадо свией бросилось со скалы в море и утонуло

s : стадо X бросилось со скалы в море и утонуло

X = *свиней, своей, сваей, ...*?

Подбор слова в зависимости от контекста

t : стадо свией бросилось со скалы в море и утонуло

s : стадо X бросилось со скалы в море и утонуло

X = *свиней, своей, сваей, ...*?

- Нужно подобрать слово X так, чтобы произведение $p(t|s)p(s)$ было максимальным

Подбор слова в зависимости от контекста

t : стадо свией бросилось со скалы в море и утонуло

s : стадо X бросилось со скалы в море и утонуло

X = *свиней, своей, сваей, ...*?

- Нужно подобрать слово X так, чтобы произведение $p(t|s)p(s)$ было максимальным
- Для простоты снова считаем, что построено выравнивание между предложениями $s = s_1 \dots s_m$ и $t_1, \dots, t_m$.

Подбор слова в зависимости от контекста

t : стадо свией бросилось со скалы в море и утонуло

s : стадо X бросилось со скалы в море и утонуло

X = *свиней, своей, сваей, ...*?

- Нужно подобрать слово X так, чтобы произведение $p(t|s)p(s)$ было максимальным
- Для простоты снова считаем, что построено выравнивание между предложениями $s = s_1 \dots s_m$ и $t_1, \dots, t_m$.
- Также считаем, что s_j, t_j — отдельные слова (нет ошибок вида вставка/удаление пробела).

Подбор слова в зависимости от контекста

t : стадо свией бросилось со скалы в море и утонуло

s : стадо X бросилось со скалы в море и утонуло

X = *свиней, своей, сваей, ...*?

- Нужно подобрать слово X так, чтобы произведение $p(t|s)p(s)$ было максимальным
- Для простоты снова считаем, что построено выравнивание между предложениями $s = s_1 \dots s_m$ и $t_1, \dots, t_m$.
- Также считаем, что s_j, t_j — отдельные слова (нет ошибок вида вставка/удаление пробела).
- Что-то очень знакомое...

Разложение модели ошибок

- Вероятность ошибки $P(t|s)$ считать легко:

$$P(t|s) = \prod_{i=1}^{n} P(t_i|s_i) = \prod_{i=1}^{n} p(s_i|t_i)$$

Разложение модели ошибок

- Вероятность ошибки $P(t|s)$ считать легко:

$$P(t|s) = \prod_{i=1}^{n} P(t_i|s_i) = \prod_{i=1}^{n} p(s_i|t_i)$$

- Вероятность $P(s)$ можно посчитать по энграммным моделям:

$$\begin{aligned} p(s) &= p(s_1 \dots s_m) \\ &= p(s_1)p(s_2|s_1)\dots p(s_m|s_1 \dots s_{m-1}) \\ &= (\text{каждое слово зависит от } k \text{ предыдущих}) \\ &= p(s_1)p(s_2|s_1)\dots p(s_{k+1}|s_1 \dots s_k)\dots p(s_m|s_{m-k} \dots s_{m-1}) \end{aligned}$$

Разложение модели ошибок

- Вероятность ошибки $P(t|s)$ считать легко:

$$P(t|s) = \prod_{i=1}^{n} P(t_i|s_i) = \prod_{i=1}^{n} p(s_i|t_i)$$

- Вероятность $P(s)$ можно посчитать по энграммным моделям:

$$\begin{aligned} p(s) &= p(s_1 \dots s_m) \\ &= p(s_1)p(s_2|s_1)\dots p(s_m|s_1 \dots s_{m-1}) \\ &= (\text{каждое слово зависит от } k \text{ предыдущих}) \\ &= p(s_1)p(s_2|s_1)\dots p(s_{k+1}|s_1 \dots s_k)\dots p(s_m|s_{m-k} \dots s_{m-1}) \end{aligned}$$

- В итоге получаем

$$\begin{aligned} p(s|t) &= \prod_{i=1}^{n} p(s_i \to t_i) + \prod_{i=1}^{n} p(s_i|s_{\max(i-k,1)} \dots s_{i-1}) \\ \log p(s|t) &= \sum_{i=1}^{n} \log p(s_i \to t_i) + \prod_{i=1}^{n} \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}) \\ \log p(s|t) &= \sum_{i=1}^{n} (\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))) \end{aligned}$$

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.
- Однако в реальных текстах это не так:

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.
- Однако в реальных текстах это не так:
  - Когнитивные ошибки (компания $\leftrightarrow$ кампания, piece $\leftrightarrow$ peace),

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.
- Однако в реальных текстах это не так:
  - Когнитивные ошибки (компания $\leftrightarrow$ кампания, piece $\leftrightarrow$ peace),
  - "Словарные" опечатки (real-word errors): *своей* $\rightarrow$ *свой*, *сваей*, *твоей*

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.
- Однако в реальных текстах это не так:
  - Когнитивные ошибки (компания $\leftrightarrow$ кампания, piece $\leftrightarrow$ peace),
  - "Словарные" опечатки (real-word errors): *своей* $\rightarrow$ *свой, сваей, твоей*
- Можно завести дополнительный классификатор, определяющий, есть ли в слове опечатка.

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.
- Однако в реальных текстах это не так:
  - Когнитивные ошибки (компания $\leftrightarrow$ кампания, piece $\leftrightarrow$ peace),
  - "Словарные" опечатки (real-word errors): *своей* $\rightarrow$ *свой, сваей, твоей*
- Можно завести дополнительный классификатор, определяющий, есть ли в слове опечатка.
- Однако он тоже может ошибаться (неологизмы, окказионализмы, имена собственные...)

Исправление опечаток в предложении

- Мы преполагали, что заранее известно, есть ли в слове опечатка.
- Однако в реальных текстах это не так:
  - Когнитивные ошибки (компания $\leftrightarrow$ кампания, piece $\leftrightarrow$ peace),
  - "Словарные" опечатки (real-word errors): *своей* $\rightarrow$ *свой, сваей, твоей*
- Можно завести дополнительный классификатор, определяющий, есть ли в слове опечатка.
- Однако он тоже может ошибаться (неологизмы, окказионализмы, имена собственные...)
- Проще предполагать возможность ошибки в каждом слове.

Подбор предложений-кандидатов

Стадо	свией	бросилось	со	скалы	в	море
стадо	свиней своей сваей	бросилось	со сто до то	скалы шкалы скулы	в во с	море мире горе торе

Подбор предложений-кандидатов

Стадо	свией	бросилось	со	скалы	в	море
стадо	свиней своей сваей	бросилось	со сто до то	скалы шкалы скулы	в во с	море мире горе торе

- Возможные предложения:

  стадо свиней бросилось со скалы в море

  стадо сваей бросилось со скалы в море

  стадо своей бросилось со скалы в море

  ...

  стадо своей бросилось со скулы в хоре

Подбор предложений-кандидатов

Стадо	свией	бросилось	со	скалы	в	море
стадо	свиней своей сваей	бросилось	со сто до то	скалы шкалы скулы	в во с	море мире горе торе

- Возможные предложения:

  стадо свиней бросилось со скалы в море

  стадо сваей бросилось со скалы в море

  стадо своей бросилось со скалы в море

  ...

  стадо своей бросилось со скулы в хоре
- Число предложений растёт экспоненциально с ростом длины предложения.

Подбор предложений-кандидатов

Стадо	свией	бросилось	со	скалы	в	море
стадо	свиней своей сваей	бросилось	со сто до то	скалы шкалы скулы	в во с	море мире горе торе

- Возможные предложения:

  стадо свиней бросилось со скалы в море

  стадо сваей бросилось со скалы в море

  стадо своей бросилось со скалы в море

  ...

  стадо своей бросилось со скулы в хоре
- Число предложений растёт экспоненциально с ростом длины предложения.
- Считать $p(s|t)$ для всех слишком долго.

Частичные гипотезы

- Формула логарифмической вероятности исправления:

$$p(s|t) = \sum_{i=1}^{n}(\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))$$

Частичные гипотезы

- Формула логарифмической вероятности исправления:

$$p(s|t) = \sum_{i=1}^{n}(\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))$$

- Заметим, что первые m слагаемых зависят только от первых m слов.

Частичные гипотезы

- Формула логарифмической вероятности исправления:

$$p(s|t) = \sum_{i=1}^{n}(\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))$$

- Заметим, что первые m слагаемых зависят только от первых m слов.
- При этом если у двух гипотез s и s' общее начало $s_1 \dots s_m$, то первые m слагаемых в сумме одинаковы.

Частичные гипотезы

- Формула логарифмической вероятности исправления:

$$p(s|t) = \sum_{i=1}^{n}(\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))$$

- Заметим, что первые m слагаемых зависят только от первых m слов.
- При этом если у двух гипотез s и s' общее начало $s_1 \dots s_m$, то первые m слагаемых в сумме одинаковы.
- Вычисления для них можно объединить.

Частичные гипотезы

- Формула логарифмической вероятности исправления:

$$p(s|t) = \sum_{i=1}^{n}(\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))$$

- Заметим, что первые m слагаемых зависят только от первых m слов.
- При этом если у двух гипотез s и s' общее начало $s_1 \dots s_m$, то первые m слагаемых в сумме одинаковы.
- Вычисления для них можно объединить.
- На m-ом шаге вычислений разумно хранить частичные гипотезы $s_1 \dots s_m$ вместе с их вероятностями.

Частичные гипотезы

- Формула логарифмической вероятности исправления:

$$p(s|t) = \sum_{i=1}^{n}(\log p(s_i \to t_i) + \log p(s_i|s_{\max(i-k,1)} \dots s_{i-1}))$$

- Заметим, что первые m слагаемых зависят только от первых m слов.
- При этом если у двух гипотез s и s' общее начало $s_1 \dots s_m$, то первые m слагаемых в сумме одинаковы.
- Вычисления для них можно объединить.
- На m-ом шаге вычислений разумно хранить частичные гипотезы $s_1 \dots s_m$ вместе с их вероятностями.
- При переходе к $(m+1)$-ому шагу ко всем гипотезам добавляет все возможные s_{m+1} и перевычисляем вероятности.

Отсечение частичных гипотез

- Даже при такой организации вычислений число гипотез растёт экспоненциально.

Отсечение частичных гипотез

- Даже при такой организации вычислений число гипотез растёт экспоненциально.
- Давайте дополнительно отсекать “плохие” частичные гипотезы.

Отсечение частичных гипотез

- Даже при такой организации вычислений число гипотез растёт экспоненциально.
- Давайте дополнительно отсекать “плохие” частичные гипотезы.
- Если $\log p(s'_1 \dots s'_m | t_1 \dots t_m) \gg \log p(s_1 \dots s_m | t_1 \dots t_m)$, то хранить $s'_1 \dots s'_m$ не имеет смысла.

Отсечение частичных гипотез

- Даже при такой организации вычислений число гипотез растёт экспоненциально.
- Давайте дополнительно отсекать “плохие” частичные гипотезы.
- Если $\log p(s'_1 \dots s'_m | t_1 \dots t_m) \gg \log p(s_1 \dots s_m | t_1 \dots t_m)$, то хранить $s'_1 \dots s'_m$ не имеет смысла.
- Можно на каждом шаге хранить только N лучших частичных гипотез (например, $N = 50$).

Отсечение частичных гипотез

- Даже при такой организации вычислений число гипотез растёт экспоненциально.
- Давайте дополнительно отсекать “плохие” частичные гипотезы.
- Если $\log p(s'_1 \dots s'_m | t_1 \dots t_m) \gg \log p(s_1 \dots s_m | t_1 \dots t_m)$, то хранить $s'_1 \dots s'_m$ не имеет смысла.
- Можно на каждом шаге хранить только N лучших частичных гипотез (например, $N = 50$).
- Можно дополнительно отсекать гипотезу h', если $p(h' | t_1 \dots t_m) \geqslant p(h | t_1 \dots t_m)$, где α — некоторая константа.

Отсечение частичных гипотез

- Даже при такой организации вычислений число гипотез растёт экспоненциально.
- Давайте дополнительно отсекать “плохие” частичные гипотезы.
- Если $\log p(s'_1 \dots s'_m | t_1 \dots t_m) \gg \log p(s_1 \dots s_m | t_1 \dots t_m)$, то хранить $s'_1 \dots s'_m$ не имеет смысла.
- Можно на каждом шаге хранить только N лучших частичных гипотез (например, $N = 50$).
- Можно дополнительно отсекать гипотезу h', если $p(h' | t_1 \dots t_m) \geqslant p(h | t_1 \dots t_m)$, где α — некоторая константа.
- Тогда объём вычислений существенно уменьшится.

Ещё несколько замечаний

- В реальной модели добавляют веса:

$$\hat{s} = \text{argmax}_s \, \alpha \log p(s \to t) + \beta \log p(s)$$

Ещё несколько замечаний

- В реальной модели добавляют веса:

$$\hat{s} = \text{argmax}_s \, \alpha \log p(s \to t) + \beta \log p(s)$$

- Веса α, β настраивают по обучающей выборке.

Ещё несколько замечаний

- В реальной модели добавляют веса:

$$\hat{s} = \text{argmax}_s \, \alpha \log p(s \to t) + \beta \log p(s)$$

- Веса α, β настраивают по обучающей выборке.
- Моделей $\log p(s)$ может быть несколько (лексические энграммы, морфологические метки, семантика, векторные модели семантики...).

Ещё несколько замечаний

- В реальной модели добавляют веса:

$$\hat{s} = \text{argmax}_s \, \alpha \log p(s \to t) + \beta \log p(s)$$

- Веса α, β настраивают по обучающей выборке.
- Моделей $\log p(s)$ может быть несколько (лексические энграммы, морфологические метки, семантика, векторные модели семантики...).
- Каждая из них берётся со своим весом, которые тоже надо настраивать.