Опечатки

Сергей Губанов
Яндекс
esgv@yandex-team.ru

12 ноября 2015 г.

План

План

Задача

- $q =$ одноклассники;

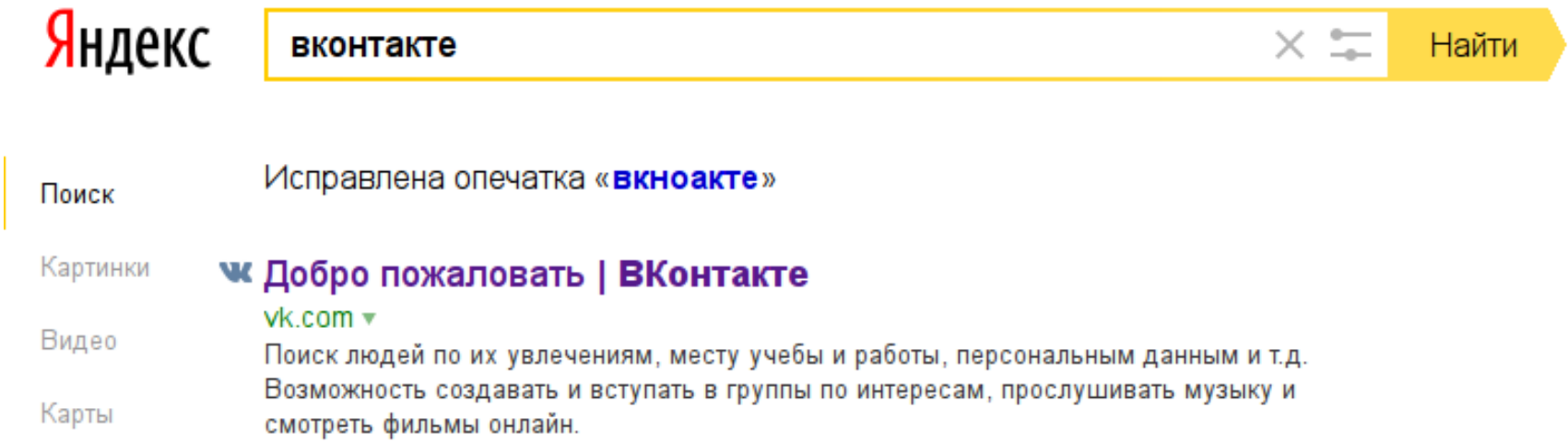
Задача

- $q =$ одноклассники;
- $c^* =$ одноклассники;

Задача

- $q =$ одноклаcники;
- $c^* =$ одноклаccники;
- $c^* = c^*(q)$.

Задача


Яндекс
вконтакте
Найти
Поиск
Картинки
Видео
Карты
Исправлена опечатка «вкноакте»
Добро пожаловать | ВКонтакте
vk.com
Поиск людей по их увлечениям, месту учебы и работы, персональным данным и т.д.
Возможность создавать и вступать в группы по интересам, прослушивать музыку и
смотреть фильмы онлайн.

Специфика

- Домен – поисковые запросы (это сложно);
- 10-12% опечаток в поисковом потоке;
- Постоянно появляются новые слова (нельзя сделать словарь).
- Считается решенной и/или неинтересной задачей (мало статей, еще меньше хороших статей).

Разные типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;

Разные типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;

Разные типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;
- *нфтвуч* $\rightarrow$ *yandex* – раскладка клавиатуры;

Разные типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;
- *нфтвуч* $\rightarrow$ *yandex* – раскладка клавиатуры;
- *гитхаб теано* $\rightarrow$ *github theano* — транслитерация;

Разные типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;
- *нфтвуч* $\rightarrow$ *yandex* – раскладка клавиатуры;
- *гитхаб теано* $\rightarrow$ *github theano* — транслитерация;
- Остальное и смешанные ошибки.

Разные типы ошибок

10-12% от поискового потока.

- **66.7%** – обычные;
- **13.7%** – segmentation;
- **9.1%** – раскладка клавиатуры;
- **2.1%** – транслитерация;
- **8.4%** – остальное и смешанные ошибки.

Длинный хвост

- эрнест хаменгуэль
- нелигитивность
- девцвиница
- санбелютень

Длинный хвост

- эрнест хаменгуэль
- нелигитивность
- девцвиница
- санбелютень
- вклньаьке
- одноикикассн

Длинный хвост

вконтваке	вкантактй	ваконтакт
вконьакье	вкуонтакте	вконтттакте
вкорнтакте	вконитакте	вконтакнте
вконтактое	вконтавккте	выконтакте
вконтакоте	вклонтакте	вуконтакте
воконтакте	вкоетакте	вконтакто
вкомтакте	вкюнтакте	вкотаккте
вкоеткте	вкогтактн	вконате
вконьекте	вконтвкье	контокте
вконтаьке	вкорнтаке	ваконтакт
вконтвуте	вкоентакт	вконетакт

Длинный хвост

- Таких опечаток не так мало (в килограммах);

Длинный хвост

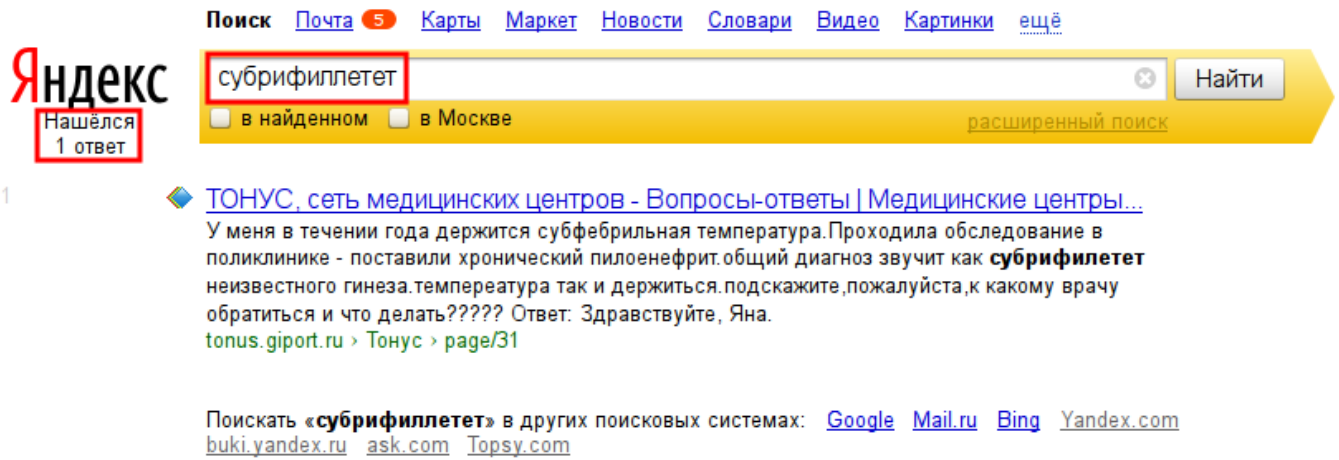
- Таких опечаток не так мало (в килограммах);
- Таких опечаток очень, очень много (в штуках);

Длинный хвост

- Таких опечаток не так мало (в килограммах);
- Таких опечаток очень, очень много (в штуках);
- Больше всего влияют на поиск;
- Сложнее всего исправлять.

Наборы

- **Обычно**: случайные K запросов из лога.
- **Сложно**: случайные K запросов из лога, для которых мало найденных документов.



Обычно vs. Сложно

- Запросов с опечаткой:
  - **Обычно:** 10%;

Обычно vs. Сложно

- Запросов с опечаткой:
  - **Обычно:** 10%;
  - **Сложно:** >50%.

Обычно vs. Сложно

- Запросов с опечаткой:
  - **Обычно:** 10%;
  - **Сложно:** >50%.
- Примеры:
  - **Обычно:** barbe (barbie), batle (battle);

Обычно vs. Сложно

- Запросов с опечаткой:
  - **Обычно:** 10%;
  - **Сложно:** >50%.
- Примеры:
  - **Обычно:** barbe (barbie), batle (battle);
  - **Сложно:** яцилокант (целакант), фокебок (фейсбук), карбонатотетраамминкобальта (карбонатотетрааминкобальта).

Обычно vs. Сложно

- Запросов с опечаткой:
  - **Обычно:** 10%;
  - **Сложно:** >50%.
- Примеры:
  - **Обычно:** barbe (barbie), batle (battle);
  - **Сложно:** яцилокант (целакант), фокебок (фейсбук), карбонатотетрааммминкобальта (карбонатотетрааминкобальта).

Метрика качества

Обычный классификатор.

- `nop` – ошибки не было, и мы ничего не исправили (TN);
- `good` – была ошибка и мы ее исправили (TP);
- `false` – ошибки не было, но мы что-то исправили (FP);
- `nosug` – была ошибка, но мы ее не исправили (FN);

```
precision = good / (good + false)
recall = good / (good + nosug)
```

Метрика качества

Обычный классификатор (не совсем).

- `nop` – ошибки не было, и мы ничего не исправили (TN);
- `good` – была ошибка и мы ее исправили (TP);
- `false` – ошибки не было, но мы что-то исправили (FP);
- `nosug` – была ошибка, но мы ее не исправили (FN);

```
precision = good / (good + false)
recall = good / (good + nosug)
```

Метрика качества

Обычный классификатор (не совсем).

- `nop` – ошибки не было, и мы ничего не исправили (TN);
- `good` – была ошибка и мы ее исправили (TP);
- `false` – ошибки не было, но мы что-то исправили (FP);
- `nosug` – была ошибка, но мы ее не исправили (FN);
- `bad` – была ошибка, и мы ее исправили, но неправильно (new!).

```
precision = good / (good + false + bad)
recall = good / (good + nosug + bad)
```

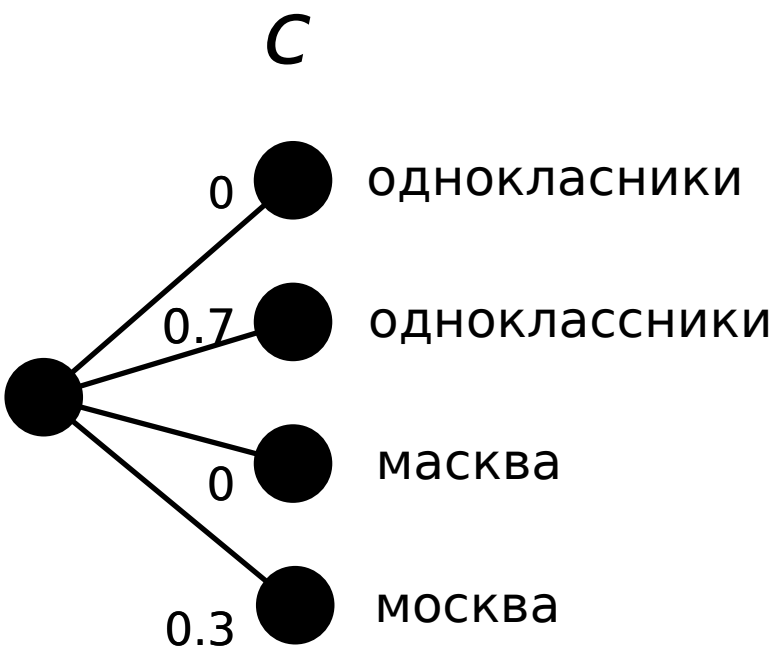
План

Мат. модель

Нужна модель того, как пользователи ошибаются.

Мат. модель

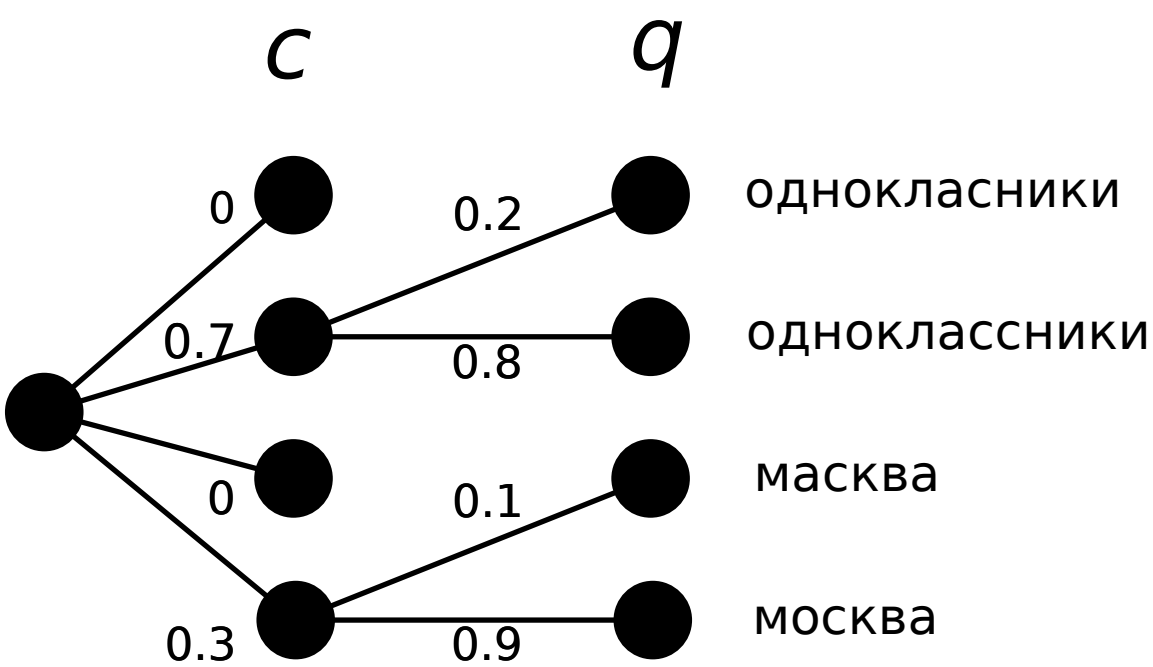
Нужна модель того, как пользователи ошибаются.



$P(c)$

Мат. модель

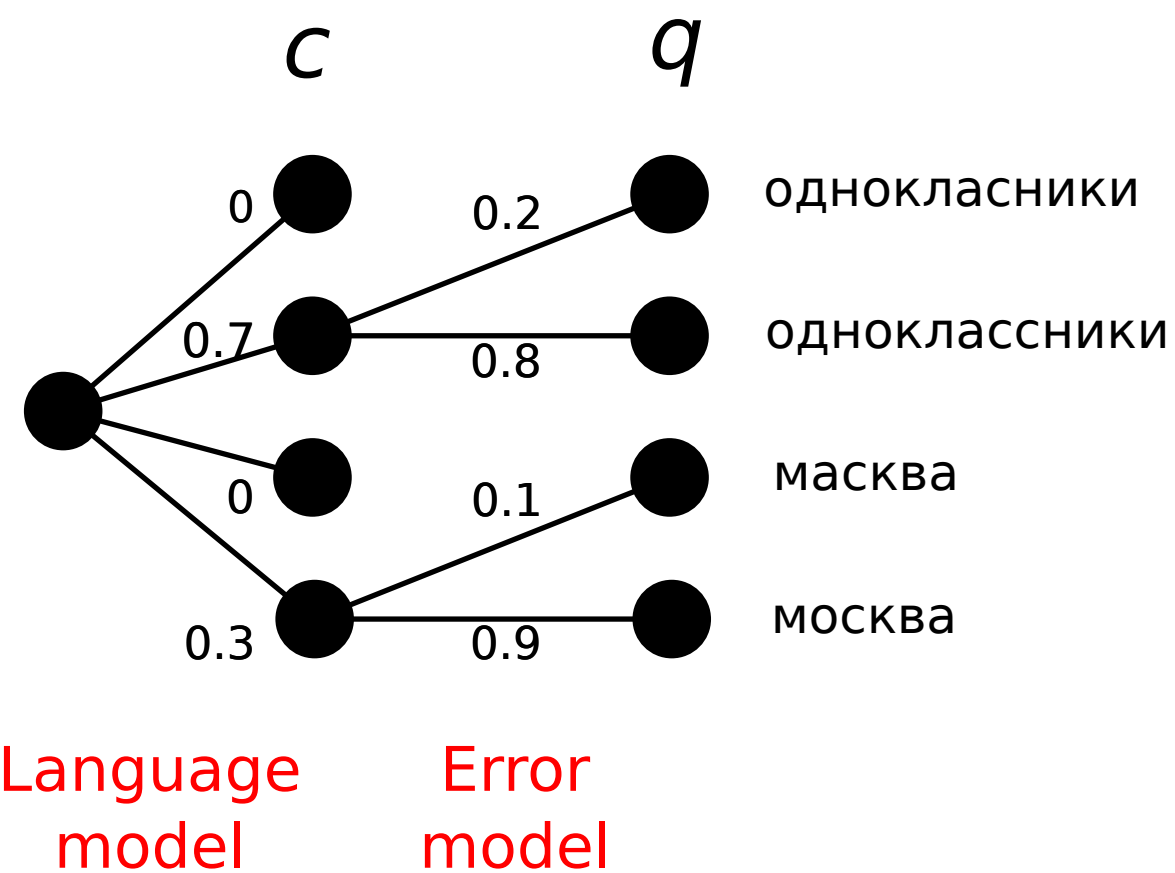
Нужна модель того, как пользователи ошибаются.



$P(c)$ $P(q|c)$

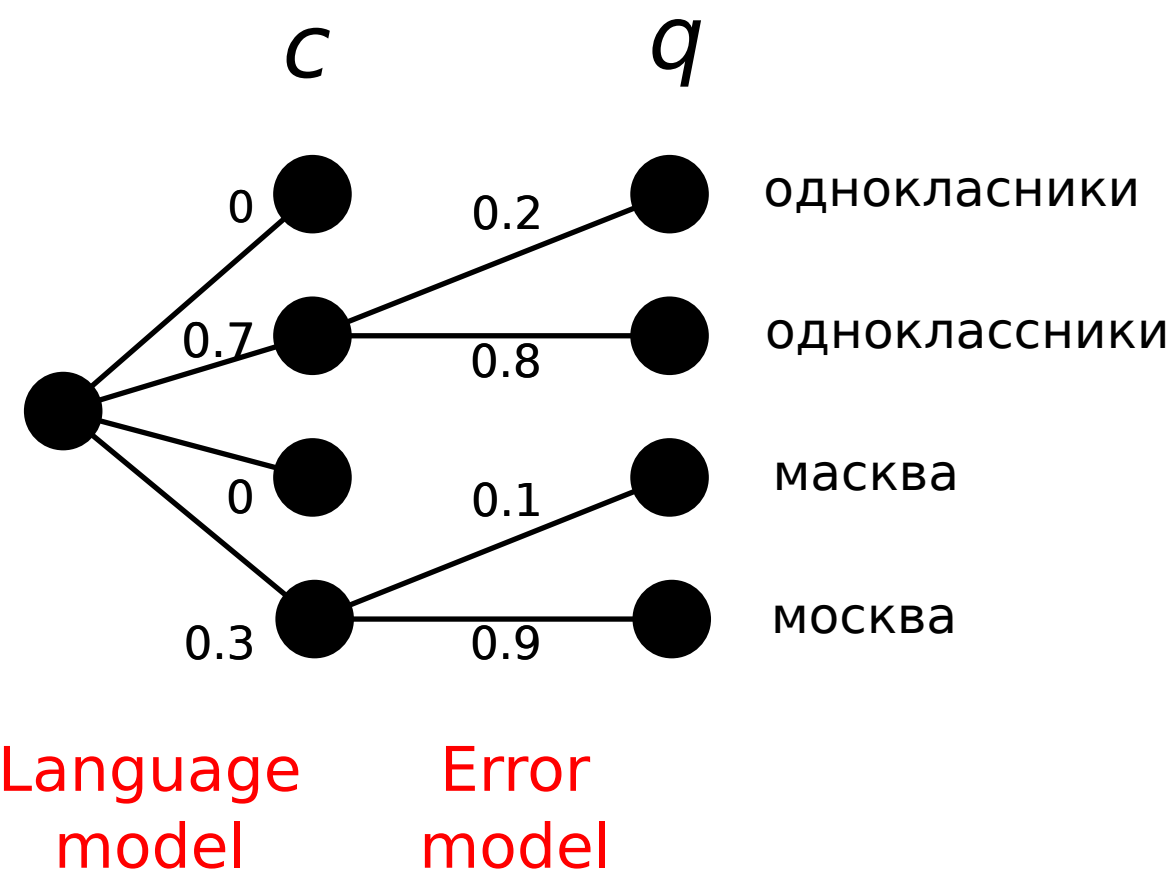
Мат. модель

Нужна модель того, как пользователи ошибаются.



Мат. модель

Нужна модель того, как пользователи ошибаются.



Называется **noisy channel model**

Noisy channel model

- (q, c)

Noisy channel model

- $(q, c) \sim P(q, c)$

Noisy channel model

- $(q, c) \sim P(q, c)$
- Пусть знаем $P(q, c)$;

Noisy channel model

- $(q, c) \sim P(q, c)$
- Пусть знаем $P(q, c)$;
- $c^*(q)$ - ?

Bayes decision rule

$$c^*(q) = \arg\max_c P(c|q);$$

Bayes decision rule

$$c^*(q) = \arg\max_c P(c|q);$$

- Если так, то $P\Big(c^*(q) = c\Big)$ максимальна;

Bayes decision rule

$$c^*(q) = \arg\max_c P(c|q);$$

- Если так, то $P_{(q,c)}\Big(c^*(q) = c\Big)$ максимальна;

Bayes decision rule

$$c^*(q) = \arg\max_c P(c|q);$$

- Если так, то $P_{(q,c)}\Big(c^*(q) = c\Big)$ максимальна;
- Что можно доказать;

Bayes decision rule

$$c^*(q) = \arg\max_c P(c|q);$$

- Если так, то $P_{(q,c)}\Big(c^*(q) = c\Big)$ максимальна;
- Что можно доказать;
- Лучший возможный спеллчекер!

Bayes decision rule

$$c^*(q) = \arg\max_c P(c|q);$$

- Если так, то $P_{(q,c)}\Big(c^*(q) = c\Big)$ максимальна;
- Что можно доказать;
- Лучший возможный спеллчекер!
- Но есть одна проблема...

Проблема

- $P(q, c) - ?$

Проблема

- $P(q, c)-?$
- $P(c|q)$ было бы достаточно, но его тоже не знаем.

Правило Байеса

$$c^*(q) = \arg\max_c P(c|q)$$

Правило Байеса

$$c^*(q) = \arg\max_c P(c|q)$$

$$P(c|q) = P(q, c) \Big/ P(q);$$

Правило Байеса

$$c^*(q) = \arg\max_c P(c|q) = \arg\max_c P(q, c).$$

$$P(c|q) = P(q, c) \Big/ P(q);$$

Правило Байеса

$$c^*(q) = \arg\max_c P(c|q) = \arg\max_c P(q, c).$$

$$P(c|q) = P(q, c) \Big/ P(q);$$

$$P(q, c) = P(q|c) \cdot P(c) \to \max_c .$$

Интерпретация

$$\underbrace{P(q|c)}_{\text{Error model}} \cdot \underbrace{P(c)}_{\text{Language model}} \rightarrow \max_{c}$$

Интерпретация

$$\underbrace{P(q|c)}_{\text{Error model}} \cdot \underbrace{P(c)}_{\text{Language model}} \to \max_c$$

Будем писать так:

$$P_{\text{dist}}(c \to q) \cdot P_{\text{LM}}(c) \to \max_c$$

Интерпретация

$$\underbrace{P(q|c)}_{\text{Error model}} \cdot \underbrace{P(c)}_{\text{Language model}} \to \max_c$$

Будем писать так:

$$P_{\text{dist}}(c \to q) \cdot P_{\text{LM}}(c) \to \max_c$$

А оценивать — так:

- $-\log P_{\text{dist}}(c \to q) \propto \text{Levenshtein}(q, c)$;

Интерпретация

$$\underbrace{P(q|c)}_{\text{Error model}} \cdot \underbrace{P(c)}_{\text{Language model}} \to \max_c$$

Будем писать так:

$$P_{\text{dist}}(c \to q) \cdot P_{\text{LM}}(c) \to \max_c$$

А оценивать — так:

- $-\log P_{\text{dist}}(c \to q) \propto \text{Levenshtein}(q, c)$;
- $P_{\text{LM}}(c) = \text{freq}(c) \Big/ \sum_w \text{freq}(w)$

Интерпретация

$$\underbrace{-\log P_{\text{dist}}(c \to q)}_{\text{Штраф за расстояние}} + \underbrace{-\log P_{\text{LM}}(c)}_{\text{Штраф за мал. частоту}} \to \min_{c}$$

Интерпретация

$$\underbrace{-\log P_{\text{dist}}(c \to q)}_{\text{Штраф за расстояние}} + \underbrace{-\log P_{\text{LM}}(c)}_{\text{Штраф за мал. частоту}} \to \min_c$$

Минимизируем сумму штрафов:

- c не очень редкое;
- c не очень далекое от q.

Интерпретация

$$\underbrace{-\log P_{\text{dist}}(c \to q)}_{\text{Штраф за расстояние}} + \underbrace{-\log P_{\text{LM}}(c)}_{\text{Штраф за мал. частоту}} \to \min_c$$

Минимизируем сумму штрафов:

- c не очень редкое;
- c не очень далекое от q.

Пример:

c	freq	$-\log P_{\text{LM}}$	$-\log P_{\text{dist}}$	$\sum$
масква	70K	19.30	0	19.30
москва	47M	9.892	5	**14.892**

Таблица: $q =$ масква

Спеллечекер за 5 мин.

Многословные запросы:

- $P_{\mathrm{LM}}(w_1, ..., w_n)$ – известно как (см. «Language model»)

Спеллечекер за 5 мин.

Многословные запросы:

- $P_{\mathrm{LM}}(w_1, \ldots, w_n)$ – известно как (см. «Language model»)
- $P_{\mathrm{dist}}(c_1, \ldots, c_n \to q_1, \ldots, q_n) = \sum_i P_{\mathrm{dist}}(c_i \to q_i)$

Спеллчекер за 5 мин.

Многословные запросы:

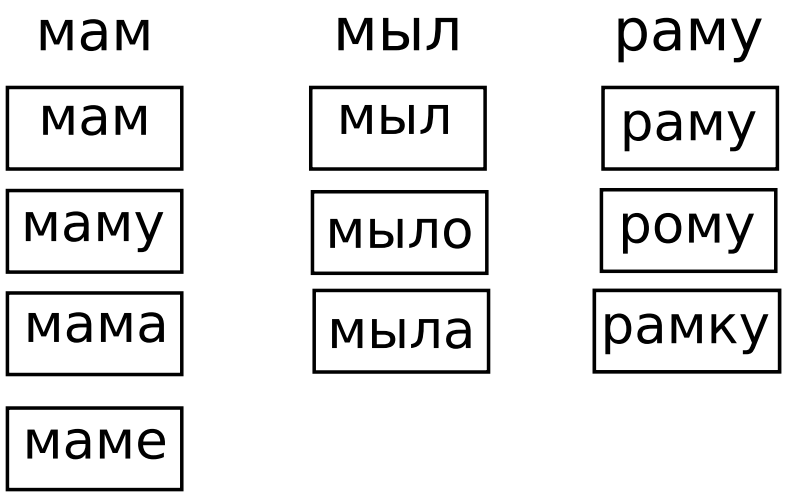
- $P_{\mathrm{LM}}(w_1, \ldots, w_n)$ – известно как (см. «Language model»)
- $P_{\mathrm{dist}}(c_1, \ldots, c_n \to q_1, \ldots, q_n) = \sum_i P_{\mathrm{dist}}(c_i \to q_i)$

мам мыл раму

Спеллчекер за 5 мин.

Многословные запросы:

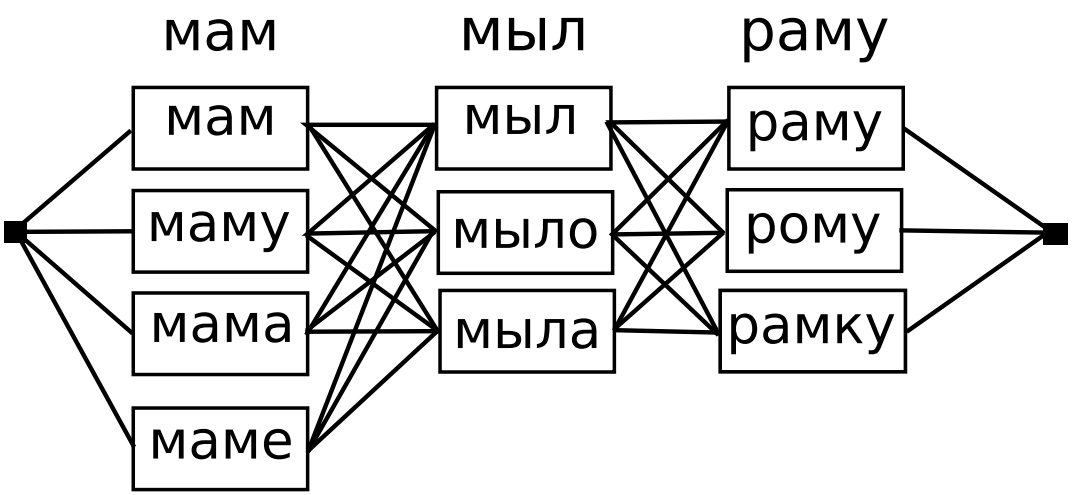
- $P_{\mathrm{LM}}(w_1, \ldots, w_n)$ – известно как (см. «Language model»)
- $P_{\mathrm{dist}}(c_1, \ldots, c_n \to q_1, \ldots, q_n) = \sum_i P_{\mathrm{dist}}(c_i \to q_i)$



Спеллчекер за 5 мин.

Многословные запросы:

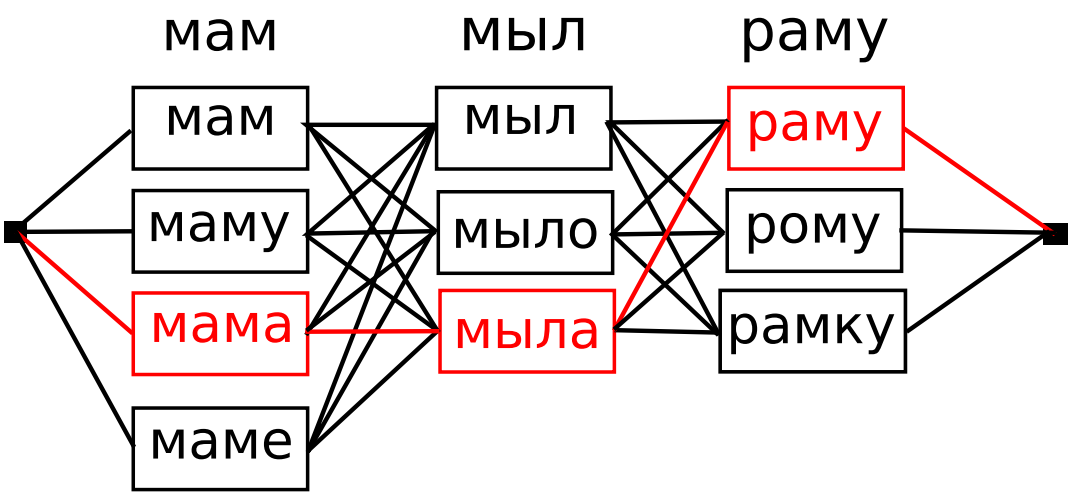
- $P_{\mathrm{LM}}(w_1, \ldots, w_n)$ – известно как (см. «Language model»)
- $P_{\mathrm{dist}}(c_1, \ldots, c_n \to q_1, \ldots, q_n) = \sum_i P_{\mathrm{dist}}(c_i \to q_i)$



Спеллчекер за 5 мин.

Многословные запросы:

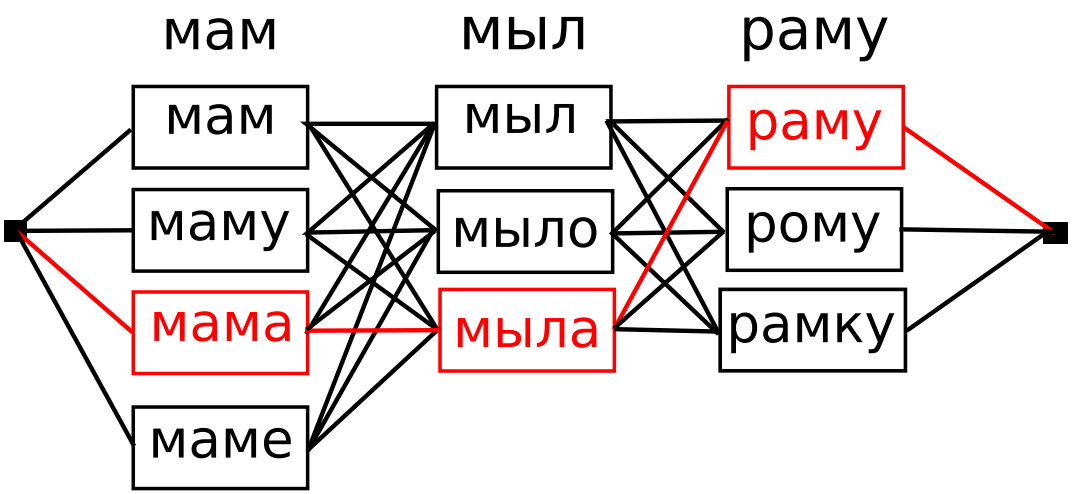
- $P_{\mathrm{LM}}(w_1, \ldots, w_n)$ – известно как (см. «Language model»)
- $P_{\mathrm{dist}}(c_1, \ldots, c_n \to q_1, \ldots, q_n) = \sum_i P_{\mathrm{dist}}(c_i \to q_i)$



Спеллечекер за 5 мин.

Многословные запросы:

- $P_{\mathrm{LM}}(w_1, \ldots, w_n)$ – известно как (см. «Language model»)
- $P_{\mathrm{dist}}(c_1, \ldots, c_n \to q_1, \ldots, q_n) = \sum_i P_{\mathrm{dist}}(c_i \to q_i)$



Алгоритм:

- Перебираем все возможные пути (исправления) c;
- Для каждого считаем $P_{\mathrm{dist}}(c \to q) \cdot P_{\mathrm{LM}}(c)$;
- Выбираем лучший.

План

План

Расстояние Левенштейна

- Минимальное число операций редактирования
  - Удаление символа: drow $\rightarrow$ row
  - Вставка символа: row $\rightarrow$ crow
  - Замена символа: crow $\rightarrow$ cron

Расстояние Левенштейна

- Минимальное число операций редактирования
  - Удаление символа: drow $\rightarrow$ row
  - Вставка символа: row $\rightarrow$ crow
  - Замена символа: crow $\rightarrow$ cron
- На самом деле нет.

Расстояние Левенштейна

- Минимальное число операций редактирования
  - Удаление символа: drow $\rightarrow$ row
  - Вставка символа: row $\rightarrow$ crow
  - Замена символа: crow $\rightarrow$ cron
- На самом деле нет.
- О расстоянии Левенштейна стоит рассуждать как о joint segmentation.

Joint segmentation

- d|e|o|e|n|-|a|n|t
  d|e|p|e|n|d|e|n|t
  0+0+1+0+0+1+1+0+0 = 3

Сегменты не больше 1 буквы (меньше можно); Получается побуквенное *выравнивание*;

Joint segmentation

- d|e|o|e|n|-|a|n|t
  d|e|p|e|n|d|e|n|t
  0+0+1+0+0+1+1+0+0 = 3
  S I S

  S = substitution
  I = insertion

  Сегменты не больше 1 буквы (меньше можно); Получается побуквенное *выравнивание*;
- Это не так очевидно, но лучшее выравнивание и лучшая последовательность правок — это одно и то же.

Joint segmentation

- ```
  d|e|o|e|n|-|a|n|t
  d|e|p|e|n|d|e|n|t
  0+0+1+0+0+1+1+0+0 = 3
      S       I S

      S = substitution
      I = insertion
  ```

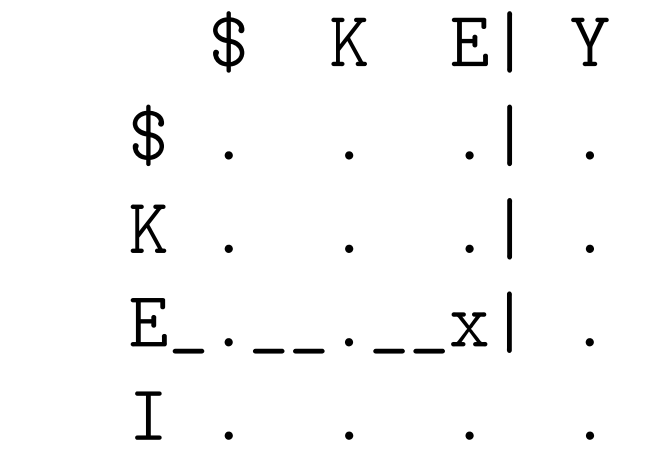
  Сегменты не больше 1 буквы (меньше можно); Получается побуквенное *выравнивание*;
- Это не так очевидно, но лучшее выравнивание и лучшая последовательность правок — это одно и то же.
- Выравнивание *монотонно* и *генеративно*

  (последовательность — нет).
  ```

Как считать Левенштейна

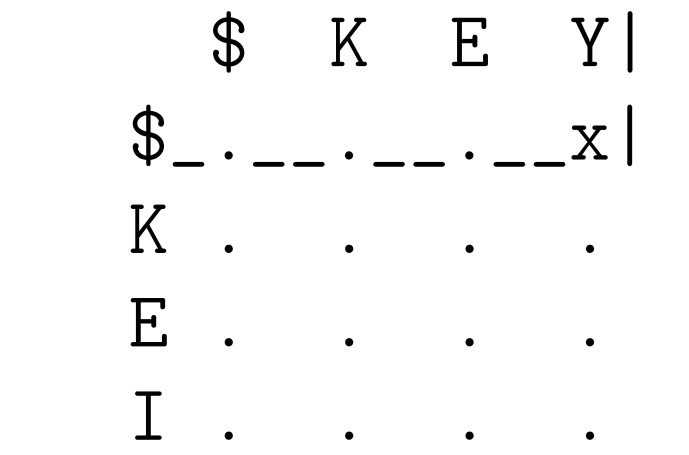
		destination string			
		$	K	E	Y
source string	$	.	.	.	.
	K	.	.	.	.
	E	.	.	.	.
	I	.	.	.	.

Инвариант




```
x = dist(KE, KE)
```

Инвариант



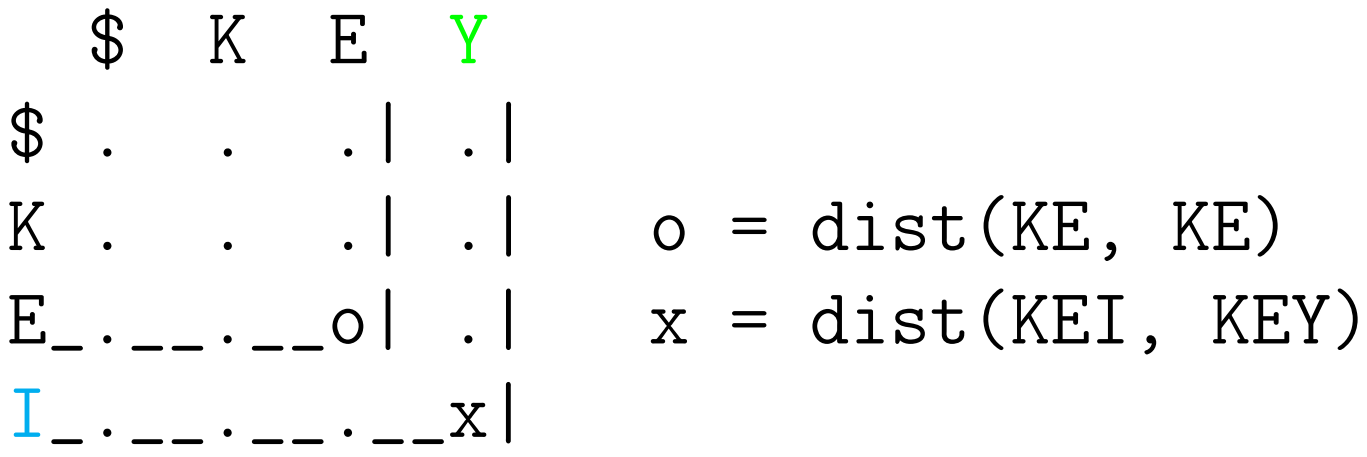

```
x = dist("", KEY)
```

Инициализация

	$	K	E	Y
$	0	x	x	x
K	x	x	x	x
E	x	x	x	x
I	x	x	x	x

```
d[0][0] = 0
d[i][j] = +INF
```

Релаксация




```
o -- знаем
x = min(x, o + penalty(I, Y))
```

Релаксация

```
   $  K  E  Y
$  .  .  .| .|
K  .  .  .| .|    o = dist(KEI, KE)
E  .  .  .| .|    x = dist(KEI, KEY)
I  .  .  o| x|
   =======+--+
```

```
o -- знаем
x = min(x, o + penalty("", Y))
```

Релаксация

	$	K	E	Y
$	.	.	.	.
K	.	.	.	.
E	.	.	o	o
I	.	.	o	x

>x< зависит от [o]

Порядок вычислений

	$	K	E	Y
$>0<	x	x	x	x
K	x	x	x	x
E	x	x	x	x
I	x	x	x	x

Порядок вычислений

	$	K	E	Y
$[0]	x	x	x	
K>1<	x	x	x	
E	x	x	x	x
I	x	x	x	x

Порядок вычислений

	$	K	E	Y
$	0	x	x	x
K[1]		x	x	x
E>2<		x	x	x
I	x	x	x	x

Порядок вычислений

	$	K	E	Y
$	0	x	x	x
K	1	x	x	x
E[2]		x	x	x
I>3<		x	x	x

Порядок вычислений

	$	K	E	Y
$	[0]	>1<	x	x
K	1	x	x	x
E	2	x	x	x
I	3	x	x	x

Порядок вычислений

	$	K	E	Y
$	[0]	[1]	x	x
K	[1]	>0<	x	x
E	2	x	x	x
I	3	x	x	x

Порядок вычислений

```
   $  K  E  Y
$  0  1  x  x
K[1][0] x  x
E[2]>1< x  x
I  3  x  x  x
```

Порядок вычислений

	$	K	E	Y
$	0	1	2	3
K	1	0	1	2
E	2	1	0	1
I	3	2	1	1

Порядок вычислений

	$	K	E	Y
$	0	1	2	3
K	1	0	1	2
E	2	1	0	1
I	3	2	1	>1<

Трансфемы

Трансфема (от «transformation unit») — это пара (коротких) строк и вес.

ре ← е	7.6296
с ← сь	3.3078
с ← т	8.4283
с ← тс	3.0278
стк ← ск	7.2816

Трансфемное расстояние

- Все то же самое, только длина сегмента может быть больше 1.

  e|l|e|f |na|t
  e|l|e|ph|an|t

Трансфемное расстояние

- Все то же самое, только длина сегмента может быть больше 1.

  e|l|e|f |na|t
  e|l|e|ph|an|t

- Левенштейн:

  e|l|e|f|n|a|-|t
  e|l|e|p|h|a|n|t

Трансфемное расстояние

- Все то же самое, только длина сегмента может быть больше 1.

  e|l|e|f |na|t
  e|l|e|ph|an|t

- Левенштейн:

  e|l|e|f|n|a|-|t
  e|l|e|p|h|a|n|t

- Штраф за разбиение = сумма штрафов за трансфемы;
- Трансфемное расстояние = минимально возможный штраф.

Трансфемное расстояние

- Все то же самое, только длина сегмента может быть больше 1.

  e|l|e|f |na|t
  e|l|e|ph|an|t

- Левенштейн:

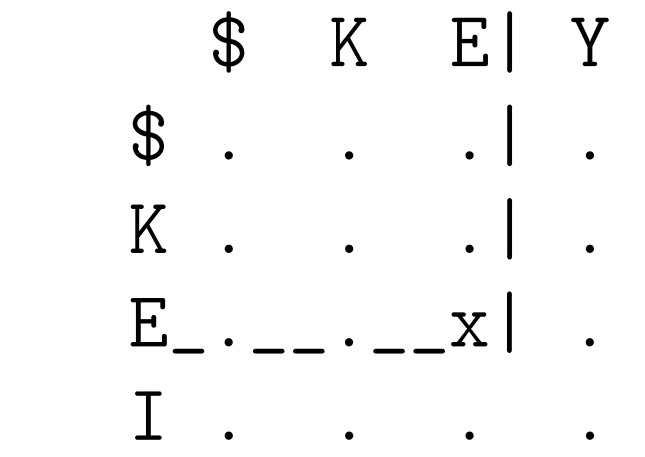
  e|l|e|f|n|a|-|t c|л|o|н|-
  e|l|e|p|h|a|n|t -|п|o|н|и

- Штраф за разбиение = сумма штрафов за трансфемы;
- Трансфемное расстояние = минимально возможный штраф.

Как считать

		destination string			
		$	K	E	Y
source string	$	.	.	.	.
	K	.	.	.	.
	E	.	.	.	.
	I	.	.	.	.

Инвариант



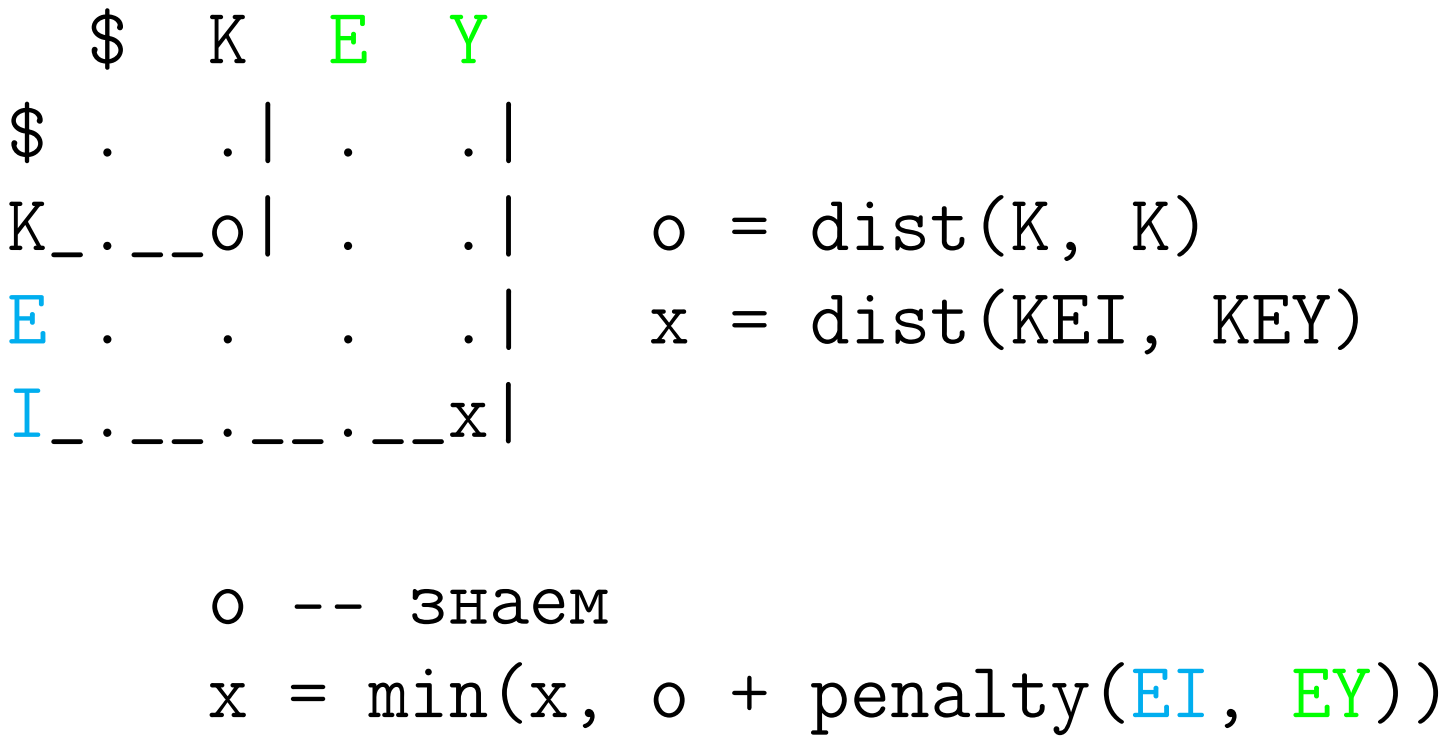

```
x = dist(KE, KE)
```

Инициализация

	$	K	E	Y
$	0	x	x	x
K	x	x	x	x
E	x	x	x	x
I	x	x	x	x

```
d[0][0] = 0
d[i][j] = +INF
```

Релаксация



Релаксация

	$	K	E	Y
$	.	.	.	.
K	.	o	o	o
E	.	o	o	o
I	.	o	o	x

>x< зависит от [o]

Порядок вычислений

	$	K	E	Y
$	0	4	8	c
K	1	5	9	d
E	2	6	a	e
I	3	7	b	f

Порядок вычислений

	$	K	E	Y
$	0	1	2	3
K	4	5	6	7
E	8	9	a	b
I	c	d	e	f

План

Генератор гипотез

$q =$ ремофломуцин

Text	Frequency	Distance
римофломуцин	1760	6.702
ринофломуцин	3461	13.37
римофламуцин	120	8.540
ринофламуцин	105	15.21
римофломуцил	19	12.78
римафламуцил	145	16.57
ринофлуимуцил	107609	26.93
ринофлоимуцин	160	17.01

Генератор гипотез

$q =$ ремофломуцин

Text	Frequency	Distance
римофломуцин	1760	6.702
ринофломуцин	3461	13.37
римофламуцин	120	8.540
ринофламуцин	105	15.21
римофломуцил	19	12.78
римафламуцил	145	16.57
ринофлуимуцил	107609	26.93
ринофлоимуцин	160	17.01

$$q \to c_1, \ldots, c_{30}.$$

Генератор гипотез

$q =$ ремофломуцин

Text	Frequency	Distance
римофломуцин	1760	6.702
ринофломуцин	3461	13.37
римофламуцин	120	8.540
ринофламуцин	105	15.21
римофломуцил	19	12.78
римафламуцил	145	16.57
ринофлуимуцил	107609	26.93
ринофлоимуцин	160	17.01

$$q \to c_1, \ldots, c_{30}.$$

- Нет гипотезы — не будет подсказки;
- 30 лучших (иначе будет медленно).

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
  2. Изменять до T символов в q;

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
  2. Изменять до T символов в q;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
     Медленно;
  2. Изменять до T символов в q;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
     Медленно;
  2. Изменять до T символов в q;
     Медленно и/или плохо;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
     Медленно;
  2. Изменять до T символов в q;
     Медленно и/или плохо;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;
     Сложно

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
     Медленно;
  2. Изменять до T символов в q;
     Медленно и/или плохо;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;
     Сложно и все еще плохо.

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
     Медленно;
  2. Изменять до T символов в q;
     Медленно и/или плохо;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;
     Сложно и все еще плохо (или медленно).

Решение 1: Левенштейн

- Все такие c_i, что $\mathrm{levenshtein}(q, c_i) < T$;
  1. Перебрать все слова;
     Медленно;
  2. Изменять до T символов в q;
     Медленно и/или плохо;
  3. Построить инвертированный индекс на буквенных n-граммах и решить t-threshold problem;
     Сложно и все еще плохо (или медленно).
- При любой реализации: много гипотез при больших T, много пропусков при малых T.

Решение 2: soundex

- Soundex — функция `str` $\rightarrow$ `str`; выдает фонетическую сигнатуру.
  Т.е. похожие по звучанию слова получают похожие (или одинаковые) сигнатуры;

Решение 2: soundex

- Soundex — функция `str` $\rightarrow$ `str`; выдает фонетическую сигнатуру.
  Т.е. похожие по звучанию слова получают похожие (или одинаковые) сигнатуры;
- К прмр: вкнт вс глсн;
  (Вообще, soundex — конкретная функция, но мы так будем называть целый класс);
- На основании soundex можно построить генератор гипотез.

Решение 2: soundex

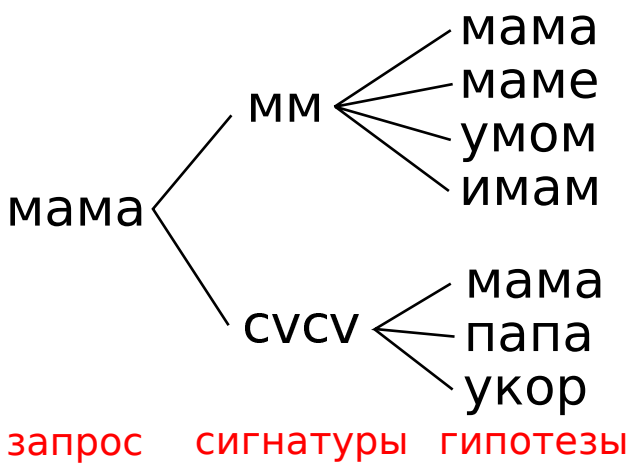
- Строим обратный индекс по сигнатурам: для каждой сигнатуры s — список слов $W(s) = (w_1, ..., w_k)$ с такой сигнатурой.

Решение 2: soundex

- Строим обратный индекс по сигнатурам: для каждой сигнатуры s — список слов $W(s) = (w_1, ..., w_k)$ с такой сигнатурой.
- Для запроса q считаем сигнатуру $s(q)$, и выдаем слова с такой же сигнатурой: $W(s(q))$;

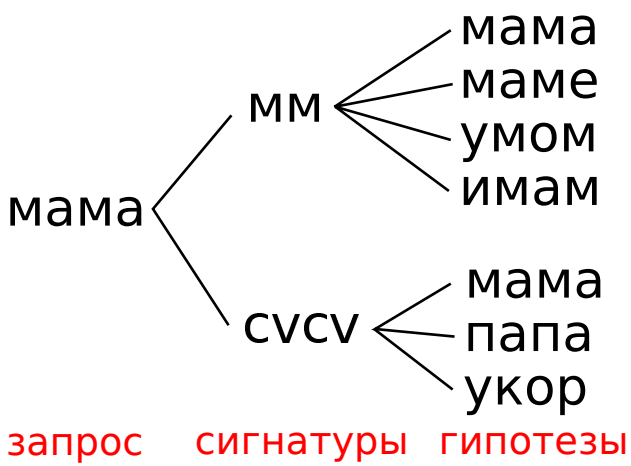
Решение 2: soundex

- Строим обратный индекс по сигнатурам: для каждой сигнатуры s — список слов $W(s) = (w_1, ..., w_k)$ с такой сигнатурой.
- Для запроса q считаем сигнатуру $s(q)$, и выдаем слова с такой же сигнатурой: $W(s(q))$;
- Функций-сигнатур может быть несколько.



Решение 2: soundex

- Строим обратный индекс по сигнатурам: для каждой сигнатуры s — список слов $W(s) = (w_1, ..., w_k)$ с такой сигнатурой.
- Для запроса q считаем сигнатуру $s(q)$, и выдаем слова с такой же сигнатурой: $W(s(q))$;
- Функций-сигнатур может быть несколько.



- Для коротких слов – очень много гипотез.

Качество

- Размеченный корпус: (опечатка, исправление);

Качество

- Пара запросов: (мам мыл раму, мама мыла раму);

Качество

- Пара запросов: (мам мыл раму, мама мыла раму);
- Пары слов: (мам, мама), (мыл, мыла);

Качество

- Пара запросов: (мам мыл раму, мама мыла раму);
- Пары слов: (мам, мама), (мыл, мыла);
- N-best recall: процент пар слов, для которых правильное слово есть в списке гипотез для исходного слова.

Качество: baseline

	N-best recall	
	Обычно	Сложно
Levenshtein	85%	
Soundex	93%	

Качество: baseline

	N-best recall	
	Обычно	Сложно
Levenshtein	85%	75%
Soundex	93%	59%

Качество: baseline

	N-best recall	
	Обычно	Сложно
Levenshtein	85%	75%
Soundex	93%	59%
Transfeme	95%	89%

Генерация гипотез

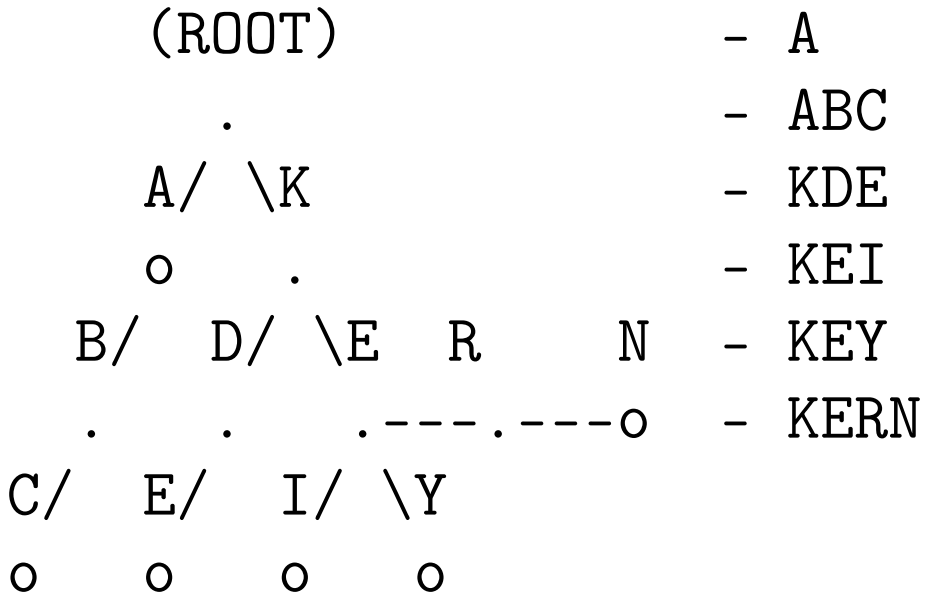
- $c_1, \ldots, c_{30}$ – такие, что $\mathrm{dist}(q, c_i) \to min$.

Генерация гипотез

- $c_1, \ldots, c_{30}$ – такие, что $\mathrm{dist}(q, c_i) \to min$.
- Как искать? Перебирать — долго.

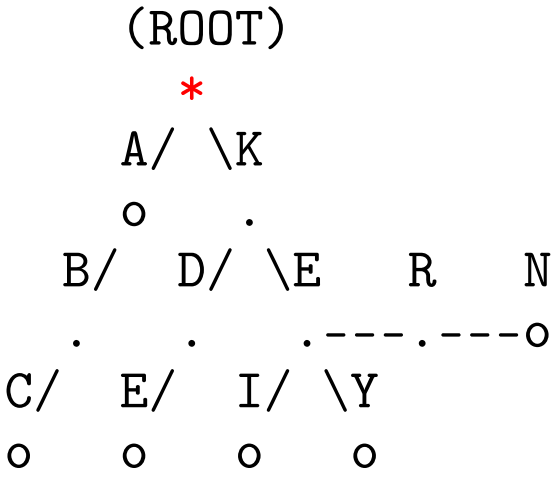
Генерация гипотез

- $c_1, ..., c_{30}$ – такие, что $\mathrm{dist}(q, c_i) \to min$.
- Как искать? Перебирать — долго.
- Trie of words



- Будем одновременно «съедать» запрос и спускаться в трае.

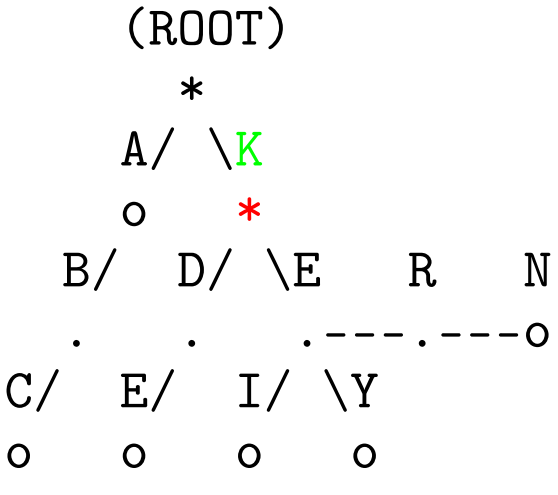
Генерация гипотез



Query	|k e e i
Hypothesis	
Penalty	0
Transfemes	

|NOP	0|
|INS	10|
|SUB	10|
|DEL	10|
|ei <- ey	5|
|i <- y	7|
|k <- g	9|

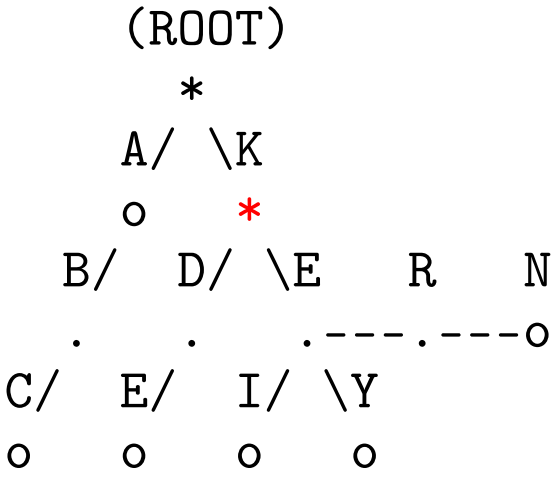
Генерация гипотез




```
Query          k|e e i
Hypothesis     k
Penalty         0
Transfemes
  NOP k <- k    0
```

```
+------------+
|NOP          0|
|INS         10|
|SUB         10|
|DEL         10|
|ei <- ey     5|
|i  <- y      7|
|k  <- g      9|
+------------+
```

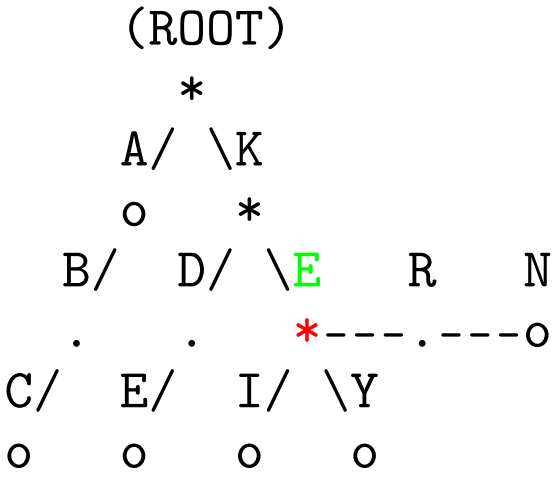
Генерация гипотез



Query	k|e e i
Hypothesis	k
Penalty	0
Transfemes	
NOP k <- k	0

NOP	0
INS	10
SUB	10
DEL	10
ei <- ey	5
i <- y	7
k <- g	9

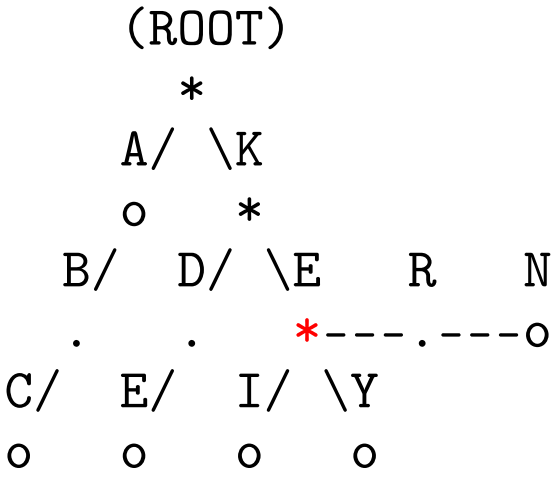
Генерация гипотез



Query	k e|e i
Hypothesis	k e
Penalty	0
Transfemes	
NOP k <- k	0
NOP e <- e	0

NOP	0
INS	10
SUB	10
DEL	10
ei <- ey	5
i <- y	7
k <- g	9

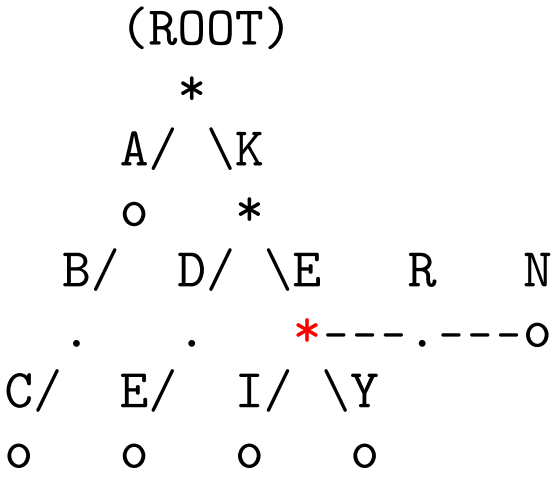
Генерация гипотез



Query	k e|e i
Hypothesis	k e
Penalty	0
Transfemes	
NOP k <- k	0
NOP e <- e	0

|NOP 0|
|INS 10|
|SUB 10|
|DEL 10|
|ei <- ey 5|
|i <- y 7|
|k <- g 9|

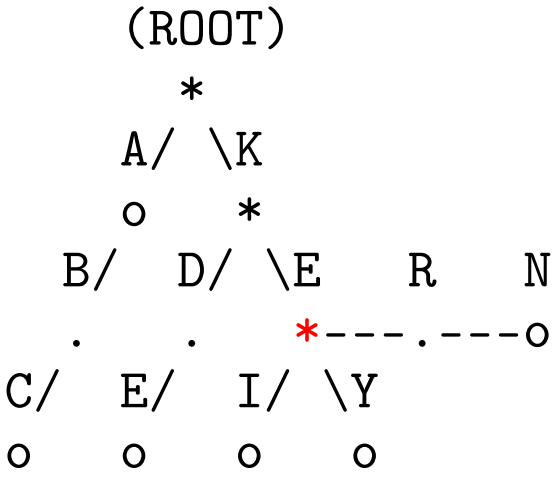
Генерация гипотез



Query	k e e|i
Hypothesis	k e
Penalty	10
Transfemes	
NOP k <- k	0
NOP e <- e	0
DEL e <-	10

NOP	0
INS	10
SUB	10
DEL	10
ei <- ey	5
i <- y	7
k <- g	9

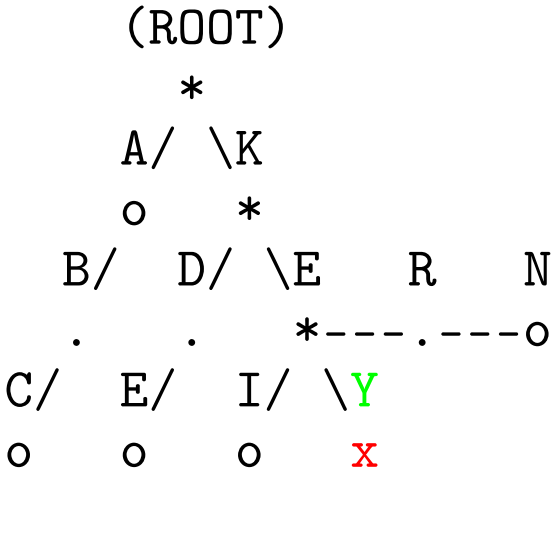
Генерация гипотез




```
Query          k e e|i
Hypothesis     k e
Penalty        10
Transfemes
  NOP k <- k    0
  NOP e <- e    0
  DEL e <-     10
```

```
+------------+
|NOP          0|
|INS         10|
|SUB         10|
|DEL         10|
|ei <- ey     5|
|i  <- y      7|
|k  <- g      9|
+------------+
```

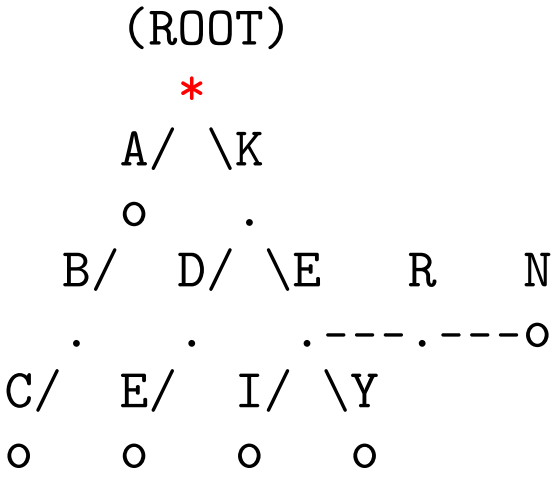
Генерация гипотез




```
Query          k e e i|
Hypothesis     k e y
Penalty        17
Transfemes
  NOP k <- k    0
  NOP e <- e    0
  DEL e <-     10
      i <- y    7
```

```
+------------+
|NOP          0|
|INS         10|
|SUB         10|
|DEL         10|
|ei <- ey     5|
|i  <- y      7|
|k  <- g      9|
+------------+
```

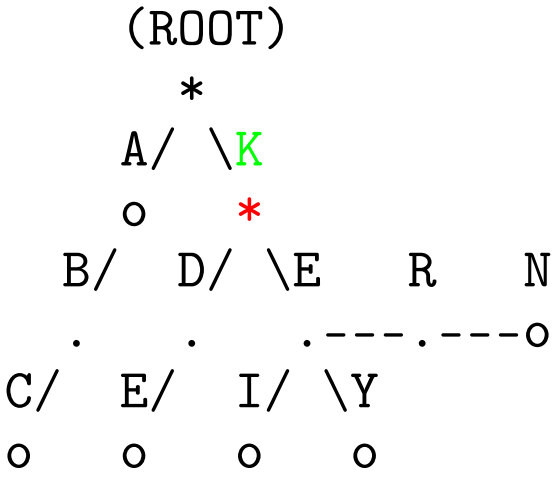
Генерация гипотез



Query	|k e e i
Hypothesis	
Penalty	0
Transfemes	

|NOP	0|
|INS	10|
|SUB	10|
|DEL	10|
|ei <- ey	5|
|i <- y	7|
|k <- g	9|

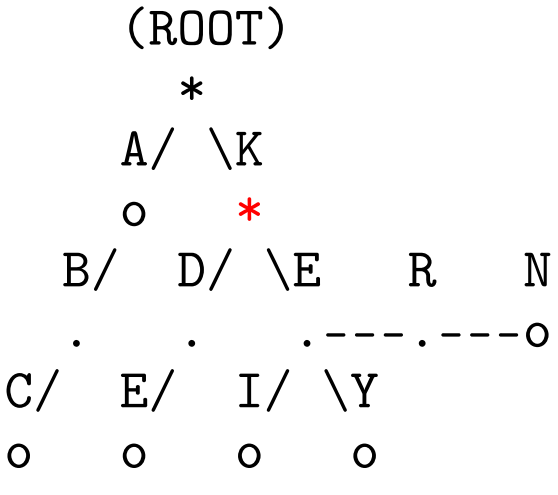
Генерация гипотез




```
Query          k|e e i
Hypothesis     k
Penalty         0
Transfemes
  NOP k <- k    0
```

```
+------------+
|NOP          0|
|INS         10|
|SUB         10|
|DEL         10|
|ei <- ey     5|
|i  <- y      7|
|k  <- g      9|
+------------+
```

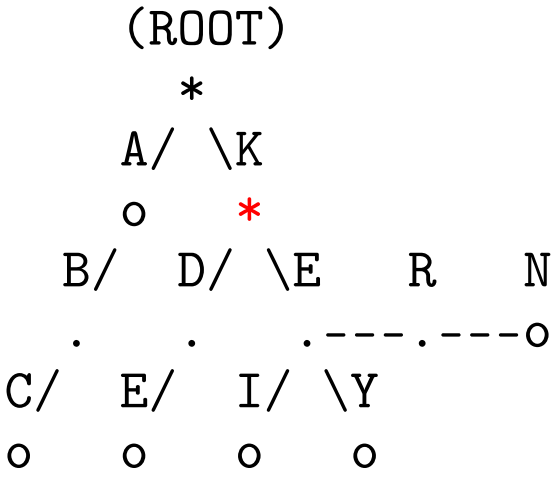
Генерация гипотез



Query	k|e e i
Hypothesis	k
Penalty	0
Transfemes	
NOP k <- k	0

NOP	0
INS	10
SUB	10
DEL	10
ei <- ey	5
i <- y	7
k <- g	9

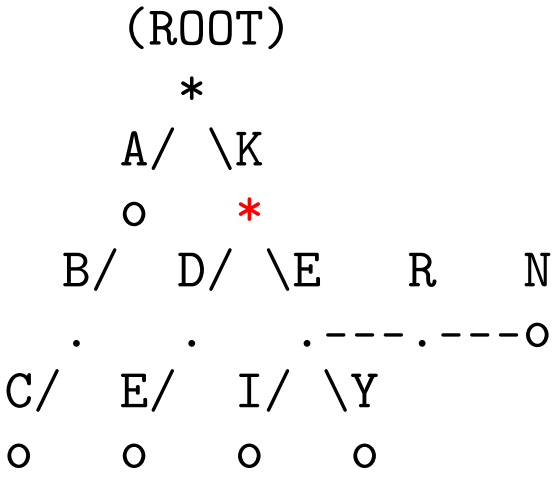
Генерация гипотез




```
Query          k e|e i
Hypothesis     k
Penalty        10
Transfemes
  NOP k <- k    0
  DEL e <-     10
```

```
+------------+
|NOP          0|
|INS         10|
|SUB         10|
|DEL         10|
|ei <- ey     5|
|i  <- y      7|
|k  <- g      9|
+------------+
```

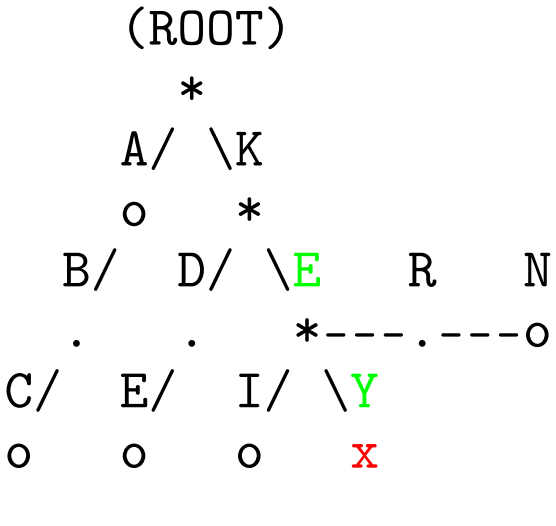
Генерация гипотез



Query	k e|e i
Hypothesis	k
Penalty	10
Transfemes	
NOP k <- k	0
DEL e <-	10

NOP	0
INS	10
SUB	10
DEL	10
ei <- ey	5
i <- y	7
k <- g	9

Генерация гипотез




```
Query           k e e i|
Hypothesis      k e y
Penalty         15
Transfemes
  NOP k <- k     0
  DEL e <-      10
      ei<-ey     5
```

```
+------------+
|NOP          0|
|INS         10|
|SUB         10|
|DEL         10|
|ei <- ey     5|
|i  <- y      7|
|k  <- g      9|
+------------+
```

Поиск Дейкстры / в ширину

```
queue.enqueue("", "", 0.0)
while not queue.empty():
    src, dest, dist = queue.dequeue_min_dist()

    if src == query and trie_has_word(dest):
        print(dest, dist)

    for (a, b, d) in transfemes:
        if not is_prefix(src + a, query):
            continue
        if not trie_has_prefix(dest + b):
            continue
        queue.enqueue(src + a, dest + b, dist + d)
```

Поиск Дейкстры / в ширину

```
queue.enqueue("", "", 0.0)
while not queue.empty():
    src, dest, dist = queue.dequeue_min_dist()

    if src == query and trie_has_word(dest):
        print(dest, dist)

    for (a, b, d) in transfemes:
        if not is_prefix(src + a, query):
            continue
        if not trie_has_prefix(dest + b):
            continue
        queue.enqueue(src + a, dest + b, dist + d)
```

Первые напечатанные гипотезы — самые близкие.

Tips, tricks

1. $$\mathrm{dist}(c \to q) \to \min;$$

Tips, tricks

1. $$\mathrm{dist}(c \to q) - \log \mathrm{freq}(c) \to \min;$$

Tips, tricks

1.

$$\mathrm{dist}(c \to q) - \log \mathrm{freq}(c) \to \min;$$

То же, что и

$$-\log P_{\mathrm{dist}}(c \to q) - \log P_{\mathrm{LM}}(c) \to \min;$$

Tips, tricks

1.

$$\mathrm{dist}(c \to q) - \log \mathrm{freq}(c) \to \min;$$

То же, что и

$$-\log P_{\mathrm{dist}}(c \to q) - \log P_{\mathrm{LM}}(c) \to \min;$$

Алгоритм:

- Собрать $\mathrm{freq}(c)$;
- В каждую вершину трая записать максимальную частоту в поддереве;
- Считать queue score немножко по-другому.

Tips, tricks

2. $-\log P_{\text{dist}}(c \to q) - \log P_{\text{LM}}(c) \to \max;$

Tips, tricks

2. $-\log P_{\text{dist}}(c \to q) - \lambda \log P_{\text{LM}}(c) \to \max;$

Tips, tricks

2. $-\log P_{\text{dist}}(c \to q) - \lambda \log P_{\text{LM}}(c) \to \max$;
3. Pruning — исследуем ограниченное число гипотез
   - для каждой позиции в запросе;
   - для каждой глубины в трае.

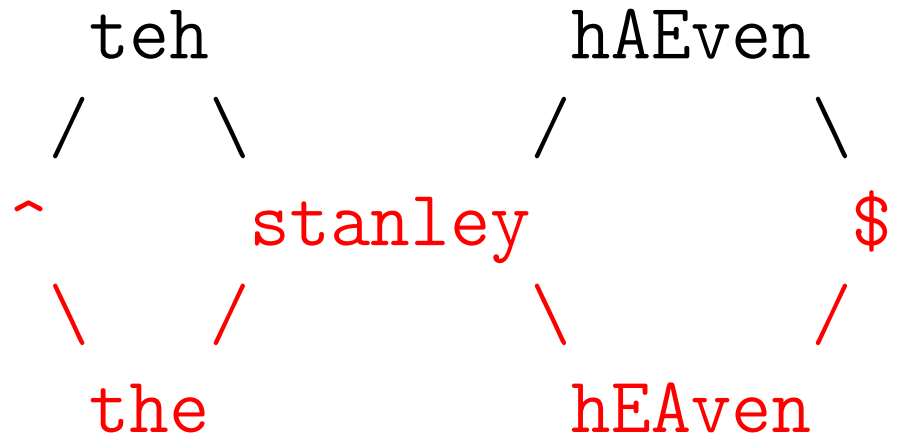
Результаты (еще раз)

	N-best recall	
	Обычно	Сложно
Levenshtein	85%	75%
Soundex	93%	59%
Transfeme	95%	89%

План

Оптимизатор Дейкстры

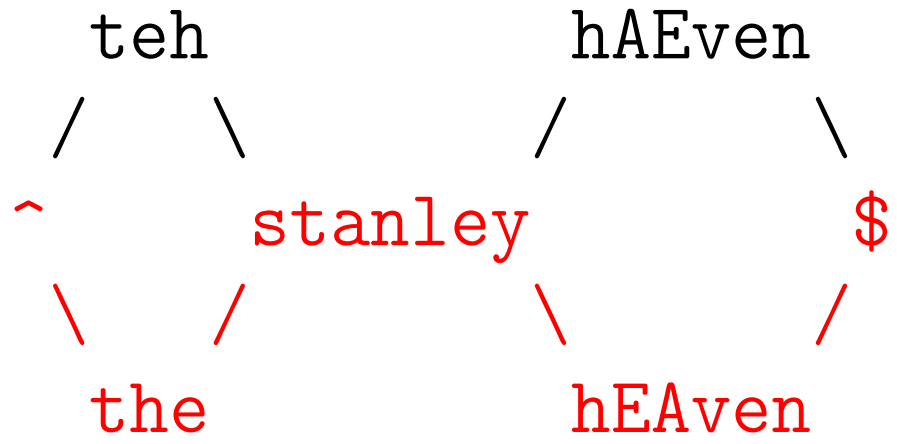
Помните lattice?



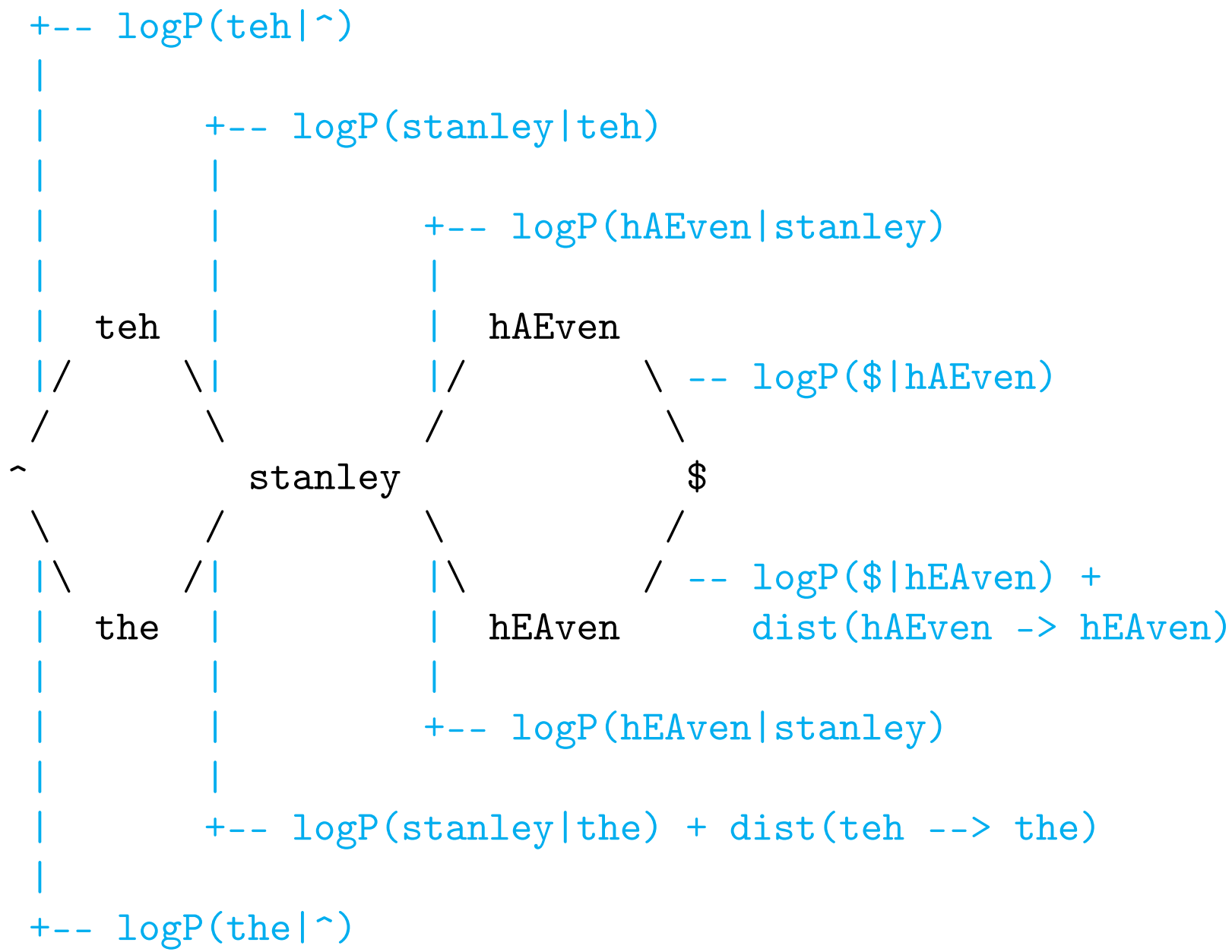
- Noisy-channel baseline: найти путь c в lattice, для которого $\mathrm{LM}(c) + \mathrm{dist}(q, c)$ минимально.

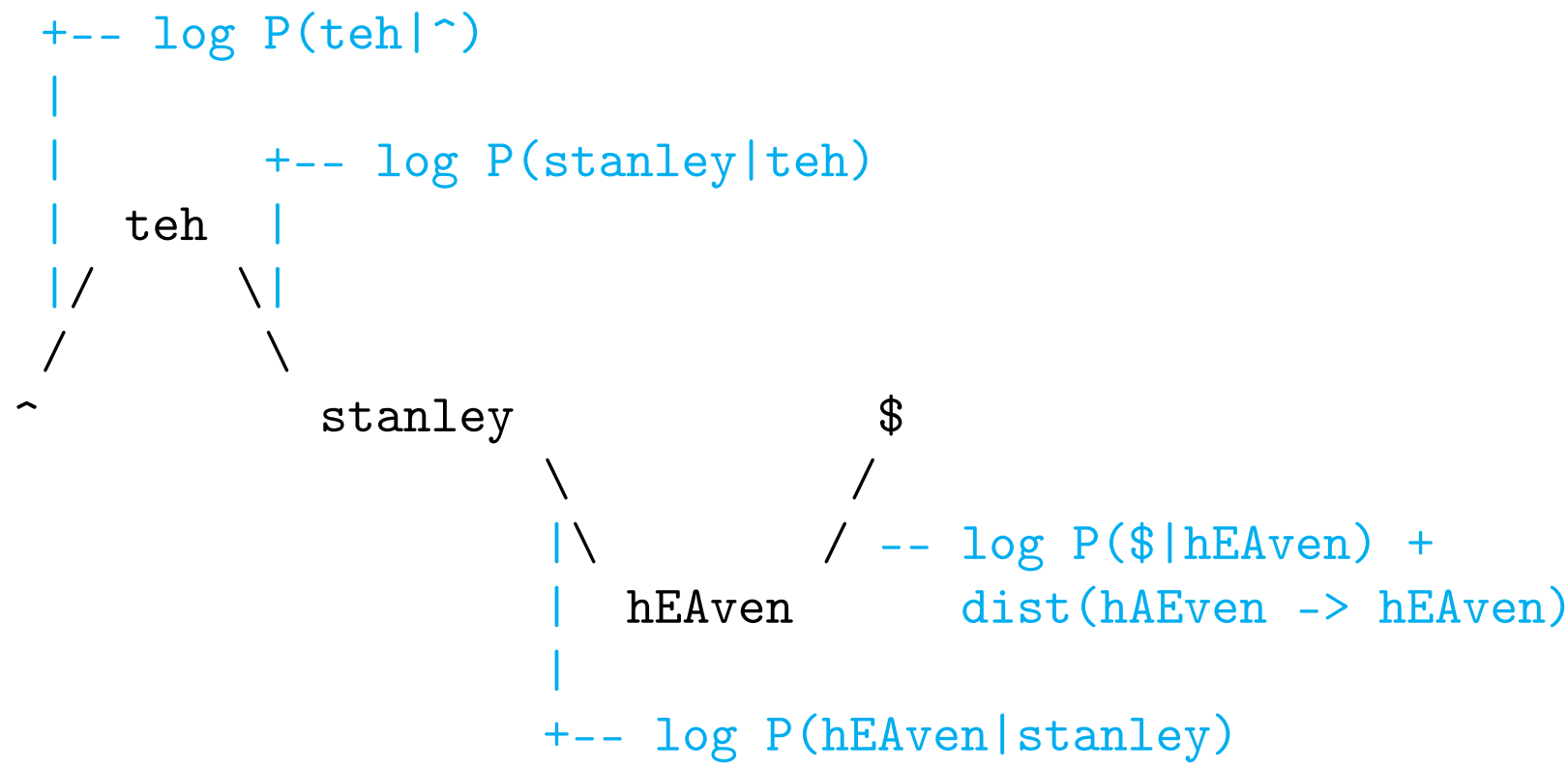
Оптимизатор Дейкстры

Помните lattice?



- Noisy-channel baseline: найти путь c в lattice, для которого $\mathrm{LM}(c) + \mathrm{dist}(q, c)$ минимально.
- Перебирать все пути – медленно.


+-- logP(teh|^)
+-- logP(stanley|teh)
+-- logP(hAEven|stanley)
teh
hAEven
-- logP($|hAEven)
^
stanley
$
-- logP($|hEAven) +
the
hEAven
dist(hAEven -> hEAven)
+-- logP(hEAven|stanley)
+-- logP(stanley|the) + dist(teh --> the)
+-- logP(the|^)




```
  logP(teh|^)
+ logP(stanley|teh)
+ logP(hEAven|stanley) + dist(hAEven -> hEAven)
+ logP($|hEAven)
-------------------------------------------------
= logP(^teh stanley hEAven$) + dist(hAEven -> hEAven)

          total LM                    total dist
```

Оптимизатор Дейкстры

- Профит: вместо полного перебора путей можно запустить поиск кратчайшего пути.

Свойства:

- Каждый путь в графе соответствует исправлению.
- Любое допустимое исправление имеет путь в графе.
- Цена исправления равна сумме длин ребер.
- Длины ребер положительны.
- Лучшее исправление соответствует кратчайшему пути.

Оптимизатор Дейкстры

- Профит: вместо полного перебора путей можно запустить поиск кратчайшего пути.
- Проблема: так можно делать только для биграмм или униграмм.

Свойства:

- Каждый путь в графе соответствует исправлению.
- Любое допустимое исправление имеет путь в графе.
- Цена исправления равна сумме длин ребер.
- Длины ребер положительны.
- Лучшее исправление соответствует кратчайшему пути.

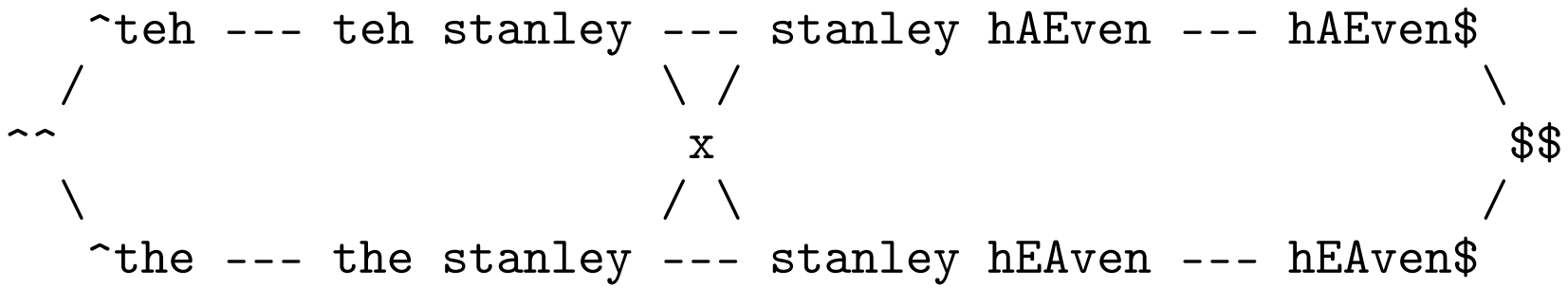
Оптимизатор Дейкстры

- Профит: вместо полного перебора путей можно запустить поиск кратчайшего пути.
- Проблема: так можно делать только для биграмм или униграмм.
- Решение: контекстим граф.

Свойства:

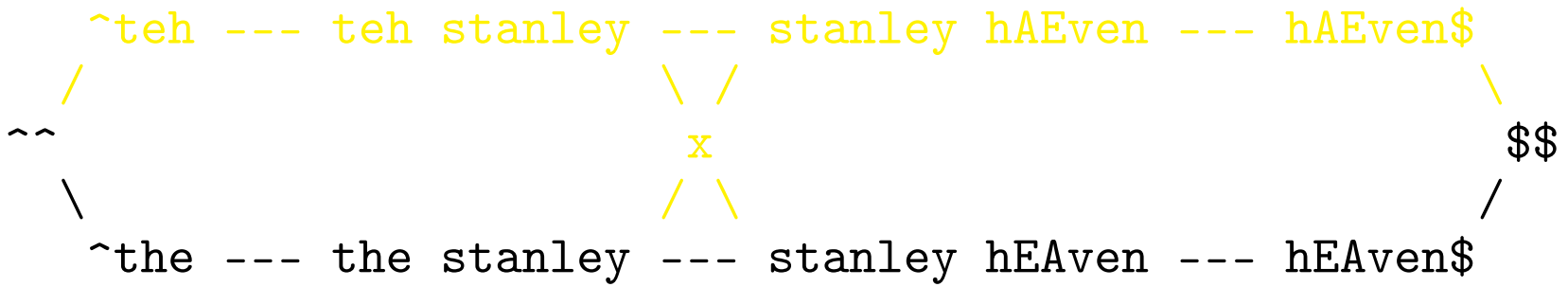
- Каждый путь в графе соответствует исправлению.
- Любое допустимое исправление имеет путь в графе.
- Цена исправления равна сумме длин ребер.
- Длины ребер положительны.
- Лучшее исправление соответствует кратчайшему пути.

Оптимизатор Дейкстры



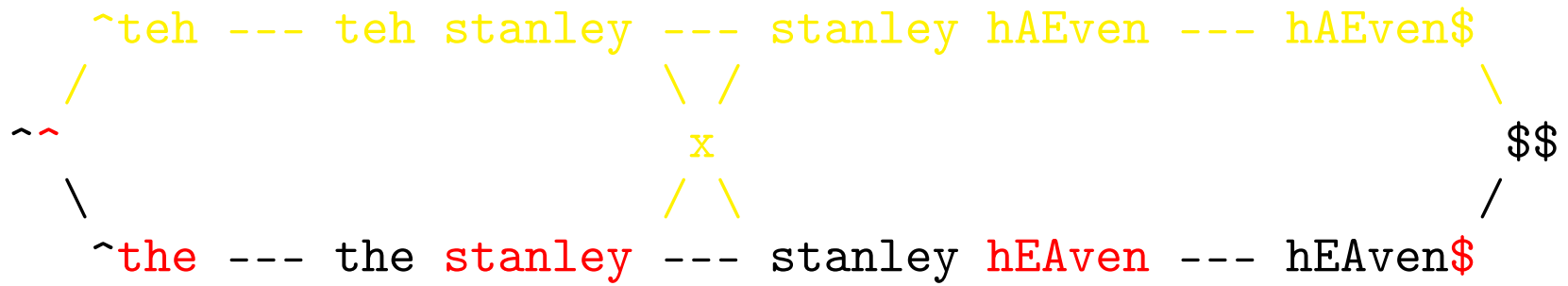
- Трансформируем граф (N-граммная LM $\Longrightarrow$ N-1 слов в вершине);
- Пишем веса на ребрах;
- Ишем Дейкстрой лучший путь.

Оптимизатор Дейкстры



Каждый путь в графе соответствует исправлению (остальные свойства очевидны).

Оптимизатор Дейкстры



Каждый путь в графе соответствует исправлению (остальные свойства очевидны).

Как контекстить граф

^ ^ ^ ^

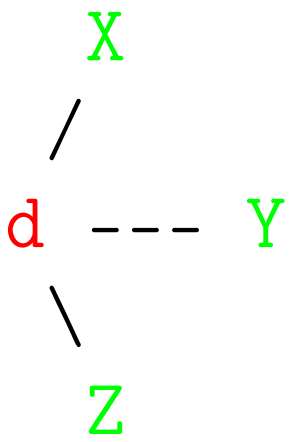
Как контекстить граф

```
a b c d --- ?
```

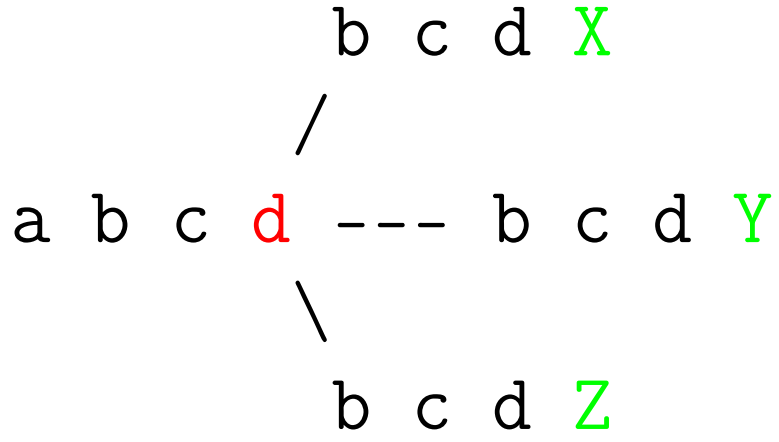
Как контекстить граф

```
a b c d --- ?
```

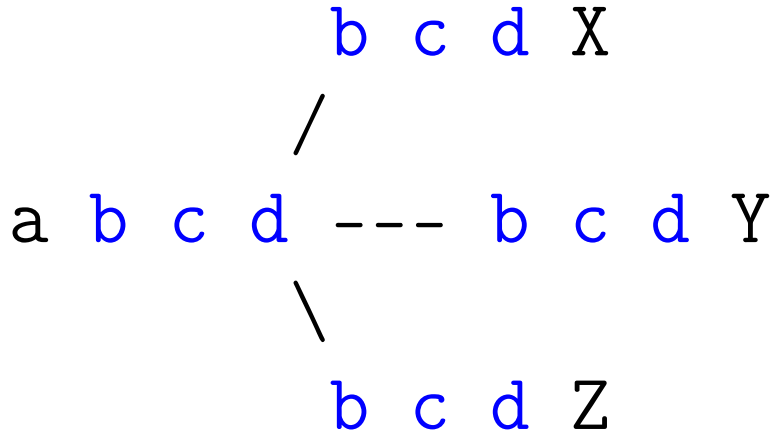
Как контекстить граф



Как контекстить граф



Как контекстить граф



План

Реранкер

1. Берем 30 лучших путей согласно noisy channel;
2. Считаем фичи.
3. Запускаем matrixnet;
4. Готово.

Реранкер

- 90% top-30 recall.

Реранкер

- 90% top-30 recall.
- Фичи: общий вес по языковой модели, общее расстояние, количество слов/символов, sparse индикаторы для трансфем, и т.д.

План

Типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;

Типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;

Типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;
- *нфтвуч* $\rightarrow$ *yandex* – раскладка клавиатуры;

Типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;
- *нфтвуч* $\rightarrow$ *yandex* – раскладка клавиатуры;
- *гитхаб теано* $\rightarrow$ *github theano* — транслитерация;

Типы ошибок

10-12% от поискового потока.

- *эльфиливая башня* $\rightarrow$ *эйфелева башня* – обычные;
- *мин юст* $\rightarrow$ *минюст* – segmentation;
- *нфтвуч* $\rightarrow$ *yandex* – раскладка клавиатуры;
- *гитхаб теано* $\rightarrow$ *github theano* — транслитерация;
- Остальное и смешанные ошибки.

Проблемы для noisy channel

(Их нет.)

Пайплайн

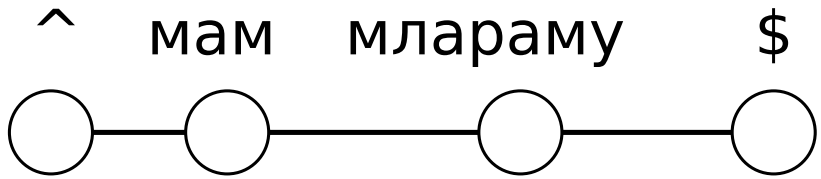
мам млараму

Пайплайн

- Токенизация;

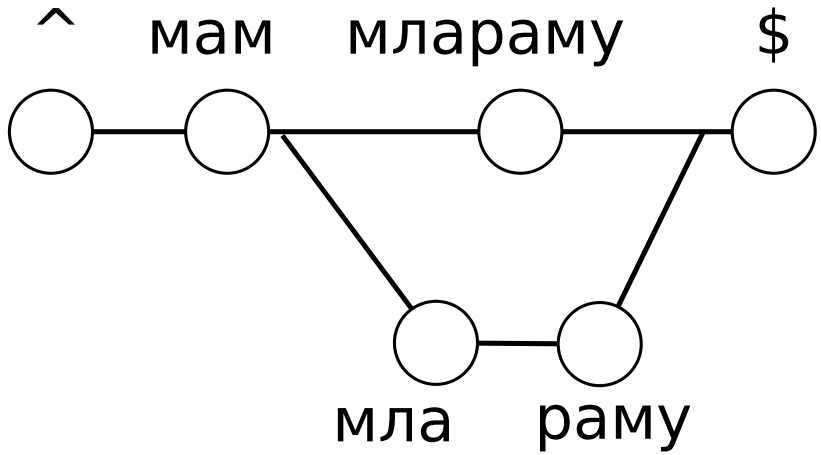
Пайплайн

- Токенизация;
- Word lattice;



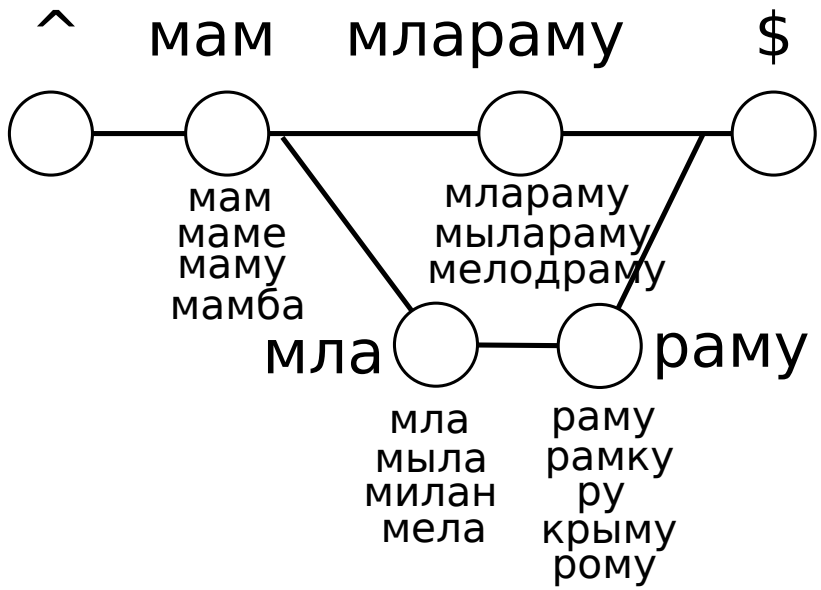
Пайплайн

- Токенизация;
- Word lattice;
- New!;



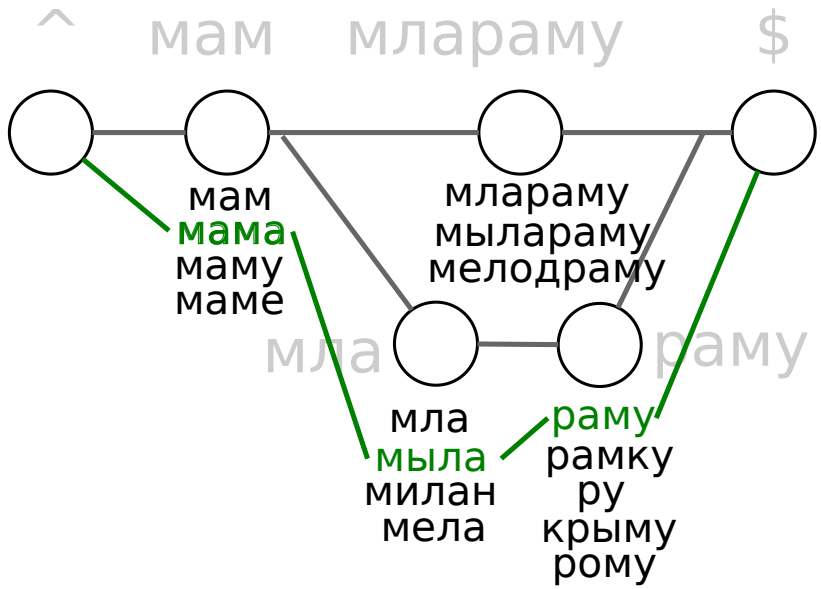
Пайплайн

- Токенизация;
- Word lattice;
- New!;
- Генерация гипотез;



Пайплайн

- Токенизация;
- Word lattice;
- New!;
- Генерация гипотез;
- Ранкер.



Типы ошибок

- Новые пути в lattice для ошибок сегментации;
- Новые трансфемы для транслита и keyboard layout;
- И т.д.

Проблемы для ранкера

- В запросе и исправлении по несколько слов;

Проблемы для ранкера

- В запросе и исправлении по несколько слов;
- В запросе и исправлении может быть разное число слов;

Проблемы для ранкера

- В запросе и исправлении по несколько слов;
- В запросе и исправлении может быть разное число слов;
- В разных исправлениях может быть разное число слов;

Проблемы для ранкера

- В запросе и исправлении по несколько слов;
- В запросе и исправлении может быть разное число слов;
- В разных исправлениях может быть разное число слов;
- Расстояние надо считать по-другому, и прочие проблемы.

Решения для ранкера

- Считать sentence-level фичи;

Решения для ранкера

- Считать sentence-level фичи;
- Считать средние значения фичей по предложению;

Решения для ранкера

- Считать sentence-level фичи;
- Считать средние значения фичей по предложению;
- Использовать structured learning (см. "transition-based dependency parser"и статью http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.2325&rep=rep1&type=pdf)

Решения для ранкера

- Считать sentence-level фичи;
- Считать средние значения фичей по предложению;
- Использовать structured learning (см. "transition-based dependency parser"и статью http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.68.2325&rep=rep1&type=pdf)
- Да хоть свертки нейросетями.

План

References

- https://class.coursera.org/nlp
- Brill, Eric, and Robert C. Moore.
  "An improved error model for noisy channel spelling correction."
  Proceedings of the 38th Annual Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2000.
  http://www.aclweb.org/anthology/P00-1037
- Duan, Huizhong, and Bo-June Paul Hsu.
  "Online spelling correction for query completion."
  Proceedings of the 20th international conference on World wide web. ACM, 2011.
  http://research-srv.microsoft.com/pubs/148103/WWW1

References

- Cucerzan, Silviu and Brill, Eric.
"Spelling Correction as an Iterative Process that Exploits the Collective Knowledge of Web Users."
EMNLP. Vol. 4., 2004. http://anthology.aclweb.org/W/W04/W04-3238.pdf
- Gao, Jianfeng and Li, Xiaolong and Micol, Daniel and Quirk, Chris and Sun, Xu
"A large scale ranker-based system for search query spelling correction"
Proceedings of the 23rd International Conference on Computational Linguistics, 2010. http://anthology.aclweb.org/C/C10/C10-1041.pdf

References

- Whitelaw, Casey and Hutchinson, Ben and Chung, Grace Y and Ellis, Gerard.
"Using the web for language independent spellchecking and autocorrection."
Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2, 2009. http://www.aclweb.org/anthology/D/D09/D09-1093.pdf

Вопросы?

Приходите к нам

galinskaya@yandex-team.ru