Некоторые методы очистки словаря запросов поиска

Some methods for language model pruning

Карпенко М. П. (m.karpenko@rambler-co.ru)
Протасов С. В. (s.protasov@rambler-co.ru)
Рамблер Интернет Холдинг

В докладе рассматривается система очистки статистической модели языка. С помощью нескольких этапов очистки система удаляет из модели данные, содержащие опечатки. Описаны эксперименты и различные методики для подавления данных, содержащих опечатки. Результаты экспериментов показывают рост эффективности работы системы коррекции ошибок поиска Рамблера.

1. Введение

В данном докладе мы рассматриваем метод очистки внутреннего словаря, который составляется полностью автоматически на основе запросов пользователей и текстов, извлеченных из сайтов в сети Интернет. Система коррекции ошибок (далее «опечаточник») в запросах пользователей к поисковой системе Rambler является одной из компонент поисковой системы и использует данный словарь для моделирования языка запросов пользователей.

Одно из первых упоминаний о проблематике исправления орфографических ошибок можно найти в работе Дамеро [7]. В ней коррекция ошибок предполагает поиск проверяемого слова в эталонном словаре и в случае, если слово отсутствует в словаре, то предлагаются близкие варианты. В случае поиска по Web для коррекции ошибок в запросах вариант ручной проверки малоэффективен, поэтому используется статистическая языковая модель на основе запросов пользователей [3][4]. Питер Норвиг в своей статье «How to Write a Spelling Corrector»[1] приводит пример такого подхода и отмечает, что данный подход прост (28 строчек кода), но недостаточно эффективен. Отмечается, что если данные содержат опечатки, то опечаточник не будет их исправлять. Эрик Брилл и Роберт Мур в своей статье [2] предлагают решить эту проблему с помощью более сложной модели ошибок. Благодаря усложнению модели ошибок опечаточники правильно исправляют «thay» на «they», которое имеет меньший рейтинг чем «that». Модель ошибок, зачастую, основывается на понятии расстояния, введенным Левенштейном[11], и использует функцию вычисления расстояния между строками на основе алгоритма Вагнера и Фишера [8]. Варианты с использованием модели ошибок описаны в работах Цобеля и Дарта[9][10]. В них описывается, что модель ошибок учитывает несколько

параметров: статистические данные о реальных опечатках, фонетическую близость слов, а так же близость на клавиатуре. Они также проводили сравнение алгоритмов анализа строк по произношению, и отметили, что вариант Soundex[1], плохо подходит для общей задачи коррекции опечаток. Помимо модели ошибок хорошим средством повышения качества работы опечаточника является использование контекста, о чем отмечают Маерс и Дамерау в своей статье [12]. Суть метода заключается в анализе слов используемых по-соседству с проверяемым. Но для построения контекстной расширенной модели требуются значительные объемы данных.

Несмотря на наличие расширенной модели языка и достаточно полной и точной модели ошибок, остаются слова с опечаткой, которые не могут исправиться, так как их рейтинг в модели языка слишком велик и модель ошибок не может это компенсировать. А так же слова с более чем одной ошибкой, которые часто исправляются на слова с одной опечаткой. С точки зрения быстродействия сложно и малоэффективно создавать модель ошибок, которая бы покрывала эти проблемные места. Поэтому далее мы рассматриваем технологии предварительной очистки модели языка, которая не работает в режиме онлайн, а используется при подготовке исходных данных.

Отдельной обширной темой являются различные способы сглаживания моделей языка: Good-Turing, Back-Off, интерполяция через удаление. Вопросы сглаживания языковых моделей не рассматриваются в данной статье и информацию о них можно найти с трудах Джефри Сампсона [18] и Манина и Шютце[19]. В данной статье мы описываем вариант повышения качества не через сглаживание, а через очистку модели языка от слов и словосочетаний, содержащих опечатки.

Для обучения (тренировки) модели используется статистика запросов пользователей, из которой извлекается языковая модель и модель ошибок. Пользователи поисковой системы совершают опечатку примерно в 15 процентах случаев (среди половины случаев, когда они набирают запрос вручную). Более половины этих опечаток наша система исправляет успешно. Однако, только в 20 процентах случаев, пользователи обращают внимание на заведомо правильное исправление и «кликают» на него. Можно выделить следующие типы опечаток:

1. Опечатки в отдельных словах
   - удаление (агенство вместо аген**т**ство) (8% случаев)
   - перестановка (ко**к**ши вместо кошки) (4% случаев)
   - вставка (пар**р**аллельный вместо параллельный) (4% случаев)
   - замена (однокласс**в**ики вместо одноклассники) (80% случаев)
   - вставка пробела как частный случай вставки (апель син вместо апельсин)
2. Опечатки в последовательностях — склейка слов (недвижимостьв москве вместо недвижимость в москве)
3. Смысловые опечатки («купить акации» вместо «купить акции»)
4. Раскладка клавиатуры (ckjy вместо слон)
5. Латиница (varezhka вместо варежка).

[1] http://ru.wikipedia.org/wiki/Soundex

От 80% до 90% всех опечаток отстоят от оригинала на одно изменение символа.

Опечаточник в работе использует два основных компонента:

- Языковая модель. Состоит из униграммной и биграммной моделей.
- Модель ошибок. Модель ошибок представляет из себя модуль, который на основе статистики предсказывает вероятность той или иной опечатки. Например, замена «а» на «о» более вероятна, чем «а» на «б».

Пользовательские опечатки являются источником шума в статистических моделях и приводят к падению точности исправлений, чрезмерному потреблению оперативной памяти серверов и падению скорости работы опечаточника.

В данном докладе мы рассмотрим итерационный алгоритм очистки, который по сути является разновидностью EM-алгоритма [13][15]. В каждом шаге мы уточняем рейтинги слов, добиваясь занижения рейтинга у слов с ошибкой и повышая рейтинги слов без ошибок.

Более полная информация об опечатках позволяет более точно сконструировать не только языковую модель, но и модель ошибок.

Кроме корпуса запросов, мы используем дополнительные сигналы, такие как — было ли нажато исправление, а также дистрибутивная схожесть между опечаткой и кандидатом на исправление. [4].

В качестве метрик качества работы опечаточника мы используем размеченный корпус опечаток, проверенный вручную. Он состоит из более чем 20 000 уникальных запросов пользователей в течение выбранного дня к поисковой системе Rambler. Данный тестовый корпус позволяет примерно оценить изменения характеристик опечаточника: качества (точность, полнота) и скорости работы. Ключевым показателем для нас является информация о количестве кликов и показов опечаточника при контрольном эксперименте на части аудитории.

2. Цели и задачи

Опечатки в данных для обучения создают много проблем. Во-первых, 60%-90% объема памяти занимаемого моделью языка — это опечатки. Во-вторых, становится сложнее исправлять популярные опечатки. Например, слово с популярной опечаткой уже есть в словаре и имеет рейтинг, ориентируясь на который, мы ошибочно предполагаем, что оно правильно написано. В-третьих, сложнее исправлять запросы с более чем одной опечаткой.

Цель очистки модели языка, состоит в том, чтобы улучшить показатели системы:

- Качество работы системы. Увеличивающийся CTR при незначительном уменьшении полноты.
- Скорость работы. Очистка позволит подбирать меньше кандидатов на исправление и это значительно ускорит работу системы.

- Объем языковой модели. Увеличение объема модели ограничивается быстродействием и объёмом используемой памяти для хранения модели. Чем меньше модель занимает места, тем больше обучающих данных мы можем использовать.

3. Опечатки можно разделить на две проблемные группы

- Низкочастотные опечатки. Рейтинг ошибочного варианта написания слова в модели в 2–3 раза ниже правильного написания слова.
- Частотные опечатки. Рейтинг ошибочного варианта близок или даже больше правильного написания слова. Например, слово «береме**нн**ая» в модели языка имеет рейтинг на 10 % больше, чем у «береме**н**ая».

Для низкочастотных и частотных ошибок мы используем разные подходы, которые опишем далее.

4. Удаление низкочастотных опечаток

Согласно закону Ципфа в модели языка значительную часть составляют малочастотные слова, и среди них чаще всего встречаются опечатки. Самый простой способ обработки малочастотных слов — удалить их, но при этом теряется много ценной информации в виде редких слов и словосочетаний.

Данный этап ориентирован на удаление низкочастотных ошибок, их влияние на качество сказывается например в случаях двойных ошибок. Так как слово с двойной опечаткой более вероятно исправится на слово с одинарной опечаткой.

4.1. Простое удаление «невероятных» слов и словосочетаний

Языковая модель представляет собой словари униграм, биграмм и триграмм с рейтингами, характеризующими частотность этих слов. Суть очистки от «невероятных» слов состоит в том, чтобы найти в словаре слова, которые не могут быть результатом работы опечаточника. В общем случае рейтинг исправления слова (ска**чч**ать) «c» на вариант «w» (скачать) определяется по формуле Байеса:

$$Ratio(c \to w) = P_L(w) * P_E(w \mid c) \qquad (1)$$

В которой:

- «c» — «ска**чч**ать» — невероятное слово, которое мы хотим удалить из словаря, частотность 2.
- «w» — «скачать» — вариант исправления, частотность 45 354.
- $P_L(w)$ — языковая модель. Вероятность варианта «w», характеризуемая его частотностью.

- $P_E(w \mid c)$ — Модель ошибок. Вероятность изменения строки «c» до строки «w»
- $q \in L(c, 1)$— множество исправлений «c» с расстоянием 1 по Левеншнейну[2]

В качестве примера для «q» мы будем использовать «ска**ччч**ать». Чтобы «c» никогда не появлялось на выходе опечаточника, нам для любых «q» нужно потребовать условие:

$$P_L(c) * P_E(c \mid w) < P_L(w) P_E(w \mid q) \tag{2}$$

То есть «q» = «ска**ччч**ать» не будет исправляться на «c» = «ска**чч**ать», так как есть лучший вариант «w» = «скачать»

$$P_L(c) max P_E(c \mid q) < P_L(w) min P_E(w \mid q) \tag{3}$$

Так как вероятность $max P_E(c \mid q)$ не может быть больше 1, поэтому

$$P_L < P_W min P(w \mid q)$$
$$\frac{1}{min P_E(w \mid q)} < \frac{P_L(w)}{P_L(c)} \tag{4}$$

У модели ошибок $P_E(w \mid q)$ всегда есть некоторая минимально возможная оценка $min P_E(w \mid q)$ (например 1/1000) и если частотность «w» = «скачать» в (45 354/2) раз больше частотности «c» = «ска**чч**ать», то мы можем смело удалять «c»= «ска**чч**ать»

Результаты очистки представлены ниже в таблице 1.

Таблица 1. Изменение параметров после удаления низкочастотных опечаток

Тип модели языка	F мера	Скорость работы (запросов/сек)	Объем модели. (млн. элементов)
Изначальный вариант	0.430	11.2	7.24
Очищенный вариант	0.430	19.3	3.91

Такой подход позволяет удалить 46 % модели языка без потери качества, что приводит к повышению скорости на 72 %.

4.2. Самоочистка

В нашем случае статистическая языковая модель строится на основе запросов пользователей к поисковой системе Rambler. Из этой статистики запросов формируется языковая модель в виде словарей униграмм, биграмм

[2] http://ru.wikipedia.org/wiki/Расстояние_Левенштейна

и триграмм. Рейтинг слова в словаре является частотностью слова или словосочетания в статистике запросов. Среди этих запросов встречаются запросы с опечаткой. Предположим, что есть запрос «w», в котором две опечатки (ск**оча-а**ть) и правильный вариант исправления «c» (скачать), при этом в словаре есть похожий на запрос вариант «c'» (скач**аа**ть). В случае использования, описанной выше формулы (1), если:

$$P_L(c')P_E(\text{c'}|\,w) > P_L(c)P_E(c\,|\,w) > P_L(w)P_E(w\,|\,w) \qquad (5)$$

То результатом исправления будет неправильный вариант «c'», при этом настроить модель ошибок, для корректной обработки подобных опечаток очень проблематично. Доля случаев подобных опечаток составляет 2–3 %, и чтобы их правильно исправлять, необходимо очистить словарь от опечаток.

Разбиваем данные для обучения модели языка на несколько частей (например 4), после чего строим для каждой части языковой модели и свою систему проверки, которую используем для проверки изначальных данных (списка запросов). Если запрос определился системой как опечатка в 1 из 4-х случаев, то его рейтинг при обучении будет 1/4. Если запрос определился системой как опечатка в 4-х из 4-х случаев, то мы удаляем его из обучающих данных. Операцию самоочистки можно проводить несколько раз, при этом ее эффективность с каждым разом снижается, так как данных, которые будут определяться как опечатки, становится меньше. На первом этапе из общей модели выделяется 15–20 % запросов, которые считаются неправильными и требуют коррекции.

Таблица 2. Изменение характеристик после самоочистки

Этапы очистки	F мера	Скорость работы (запросов/сек)	Объем модели языка (млн. элементов)
Изначальный вариант	0.430	19.3	3.91
Первый этап	0.430	20.3	3.84
Второй этап	0.431	21.0	3.81

Такой подход позволяет лучше исправлять опечатки с расстоянием 2, а так же позволяет удалить низкочастотные опечатки.

4.3. Использование результатов поиска

Мы так же можем использовать количество результатов поиска по запросу. На рис. 1 представлена статистика результатов поиска по словарю униграмм модели языка.

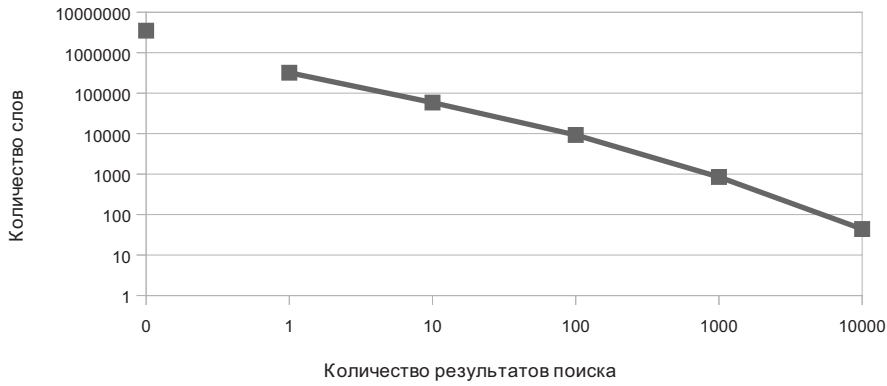

Рис. 1. График количества результатов поиска у слов модели языка

Как видно из графика, у значительной части словаря нет результатов поиска, поэтому мы можем удалить их из словаря. Этот вариант может удалить часть правдоподобной информации, но ее наличие или отсутствие никак не сказывается на качестве опечаточника в контексте работы поисковой системы.

5. Удаление высокочастотных опечаток

Это наиболее проблемные опечатки, так как их сложней обнаружить и исправить в автоматическом режиме. При этом, если опечатка популярна, ее чаще вводят пользователи. Основная проблема в таких случаях состоит в том, что мы не можем исправить запрос. Например, в модели языка есть два слова:

- «бессо**н**ица» имеет рейтинг 33 735
- «бессо**нн**ица» его рейтинг 34 471

Согласно формуле (1), если:

$$P_L(c)P_E(c \mid w) < P_L(w)P_E(w \mid w) \qquad (6)$$

То запрос с опечаткой не исправится. Модель ошибок, предполагает, что вероятность замены «н» на «нн» велика, но не перекрывает высокий рейтинг слова с опечаткой.

Для удаления этих опечаток используется два шага:

- Выявление опечатки с помощью метода общего контекста [6][12] и анализа поведения пользователей [4]

- Исправление опечатки, используя дополнительные материалы, в которых подобные опечатки маловероятны.

5.1. Выявление высокочастотных опечаток

Для обнаружения высокочастотных ошибок используются ниже описанные методики:

5.1.1. Соотношение вариантов

Для наиболее частотных слов модели языка строятся варианты их ошибок, среди которых ищутся варианты с большим рейтингом. Если вариант опечатки имеет достаточно большой рейтинг и при этом модель ошибок показывает, что данная замена очень вероятна, то считаем данное слово кандидатом в «высокочастотные опечатки», таких кандидатов набирается 14% от объема языковой модели.

5.1.2. Использование общего контекста

Подобный подход, описывается в работе Кейси Уайтлоу [16]. Суть идеи состоит в том, что если слово и вариант опечатки близки с точки зрения модели ошибок и при этом у них общий контекст, то с достаточной степенью уверенности можно утверждать, что данный вариант опечатки действительно является опечаткой данного слова, а не отдельным абсолютно другим словом. Под общим контекстом понимается одинаковые слова и словосочетания, используемые совместно со словом и его вариантом опечатки.

Например:

- программы ***для android***
- програмы ***для android***
- праграммы ***для android***

В данном случае у слова «программы» и вариантов ошибок: «програмы», «праграммы» есть общий контекст «для android», мы можем утверждать, что данные варианты действительно являются опечатками слова «программы».

К контексту накладывается некоторые полезные ограничения:

- контекстом не являются слова короче 3–4 букв, но в сочетании слов их можно использовать («для android», «для машины», «тур в»)
- контекстом не являются слова общеупотребительной лексики (который, около …)

Рис. 2. Статистика общих контекстов

На Рис. 2. Представлена статистика, которая показывает, что достаточно часто в (60%) случаев, у слова и варианта исправления есть как минимум одно слово употребляемое совместно с данным словом и вариантом исправления.

Например у слов «автомобилей» и «автамобилей» есть несколько слов контекстов:

- ***покупка*** автомобилей/авт**а**мобилей
- ***продажа*** автомобилей/авт**а**мобилей
- ***настройка*** автомобилей/авт**а**мобилей
- ***ремонт*** автомобилей/авт**а**мобилей
- ***запчасти для*** автомобилей/авт**а**мобилей

У пары слов «автомобилей» и «авт**а**мобилей»представлено 5 контекстов, по этому мы можем предположить, что одно из слов является ошибкой. Слово «автомобилей» будет считаться правильным, так как у него будет больше:

- Рейтинг языковой модели.
- Количество слов контекстов.

При этом похожие слова как «рамблер» и «трамблер» означают абсолютно разные вещи, но в написании отличаются на одну букву. Благодаря контексту мы можем определить их в разные группы и не исправлять одно на другое.

Слово «рамблер» употребляется в следующих словосочетаниях:

- ***почта*** рамблер
- ***поиск*** рамблер
- рамблер ***фото***
- рамблер ***тв программа***
- рамблер ***знакомства***
- рамблер ***игры***
- …

А слово «трамблер» больше употребляется в случаях:

- ***купить*** трамблер
- ***цена*** трамблер
- трамблер ***на ваз 21***
- ***ремонтировать*** трамблер
- …

Как видим из примера, пересечений контекстов в данном случае нет.

5.2. Исправление высокочастотных опечаток

Так как опечатки высокочастотные и прошли несколько стадий анализа, то их количество уменьшается в разы и при желании их даже можно проверить вручную. Например:

- девчонка/девч**е**нка
- лестница/ле**сн**ица

Но есть более эффективный способ. Полученный список кандидатов анализируется с помощью дополнительных источников данных:

- Материалы, в которых вероятность ошибок наименьшая (новостные, литературные тексты)
- Списки названий:
  - Списки фамилий, имен
  - Списки географических объектов, адресов
  - Списки лекарственных средств
  - Списки названий компаний

Анализируется наличие данного и исправленного слова в данных списках, если слово есть в списках, то мы его не правим. А так же анализируется общий контекст у исправления и слова кандидата в высокочастотные опечатки. Считать данный вариант опечаткой или нет определяется порогом для соотношения объемов контекста. Вариант, у которого больше контекста, считается правильным [16].

Например, как было показано выше, у слов «автомобилей» и «автамобилей» есть 6 контекстов. Таким образом мы можем предположить, что эти слова связаны и одно из них является ошибкой другого. Так как:

1. $\frac{P_L(\text{автомобилей})}{P_L(\text{автамобилей})} = \frac{0{,}0000546}{0{,}0000017} = 321 > 20$
2. Количество слов контекста у «автомобилей» — 114, а количество слов у «автамобилей» — 23 и 114/23=4,96 > порогового значения 3.

Доработка модели языка материалами, в которых вероятность ошибок наименьшая заключается в следующем:

- Ищем пересечение моделей языка
- Повышаем рейтинг словам, общим для моделей.

Вместо добавления новых слов мы используем дополнительные данные как критерий, повышающий рейтинг правильных слов над ошибочными.

Результаты исправления высокочастотных опечаток на корпусе 20 000 запросов представлены ниже.

Таблица 3. Изменение параметров после удаления высокочастотных опечаток

Вариант словаря	F мера	Скорость работы (запросов/сек)	Объем модели (млн. элементов)
Изначальный вариант	0.431	21.0	3.81
Доработанный вариант	0.442	21.3	3.77

6. Выводы

Описанные методы позволяют значительно сократить объем модели языка, при этом повысив ее качество.

В нашем случае:

- Очистка позволила удалить 48% модели языка
- Улучшить качество работы на 2.7%
- Увеличить скорость работы на 87%

Так как качество опечаточника увеличивается при росте модели языка, то при фиксированном объеме быстрой оперативной памяти сервера очистка модели позволяет повысить качество за счет большего объема исходных данных.

Литература

1. *Norvig P.* How to Write a Spelling Corrector // http://norvig.com/
2. *Eric Brill, Robert C. Moore.* An Improved Error Model for Noisy Channel Spelling Correction. //Association for Computational Linguistics Stroudsburg, PA, USA, 2000
3. *Silviu Cucerzan, Eric Brill.* Spelling correction as an iterative process that exploits the collective knowledge of web users. // Conference on Empirical Methods in Natural Language Processing, 2004
4. *Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Sebastiano Vigna.* Query Suggestions Using Query-Flow Graphs // Workshop on Web Search Click Data, ACM, 2009

5. *Kristina Toutanova, Robert C. Moore.* Pronunciation Modeling for Improved Spelling Correction // Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6–12, 2002 стр. 144–151
6. *Andre R. Golding, Dan Roth.* Applying Winnow to Context-Sensitive Spelling // Correction.International Conference on Machine Learning (ICML) pp. 182–190, 1996
7. *Damerau, F.J.* A technique for computer detection and correction of spelling errors // Journal of the Royal Statistical Society 39 (1): 1–21., 1964
8. *Cherkassky, V., N. Vassilas, G. L. Brodt, R. A. Wagner, and M. J. Fisher.* The string to string correction problem. ACM New York, NY, USA, 1974
9. *Justin Zobel, Philip W. Dart*: Phonetic String Matching: Lessons from Information Retrieval // SIGIR '96 Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, 1996
10. *Justin Zobel, Philip W. Dart*: Finding Approximate Matches in Large Lexicons // Software—Practice & Experience, Volume 25 Issue 3, March 1995
11. *Levenshtein V.* Binary codes capable of correcting deletions, insertions and reversals. // Доклады совета физиков, Vol. 10, No. 8. , pp. 707–710 1966
12. *Mayes, E., F. Damerau,* et al. Context Based Spelling Correction. //Information Processing and Management: an International Journal Volume 27 Issue 5, 1991
13. *Dempster, A., N. Laird,* et al. . Maximum likelihood from incomplete data via the EM algorithm. //Journal of the Royal Statistical Society. Series B (Methodological), Vol. 39, No. 1. pp. 1–38. 1977
14. *Mu Li, Muhua Zhu,Yang Zhang,Ming Zhou.* Exploring Distributional Similarity Based Models for Query Spelling Correction //ACL-44 Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, 2006
15. *Adam L. Berget, Stephen A. Della Piertra, Vincent J. Della Pietra.* Maximum Entropy Approach to Natural Language Processing. //Computational Linguistics Volume 22 Issue 1, March 1996
16. *Casey Whitelaw, Ben Hutchinson, Grace Y Chung and Gerard Ellis.* Using the Web for Language Independent Spellchecking and Autocorrection // EMNLP '09 Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 — Volume 2 2009
17. *Renato De Mori.* Spoken dialogues with computers //Academic Press, 1998.
18. *Geoffrey Sampson.* Empirical linguistics //Continuum International (London and New York), 2001
19. *Maning, Schutze.* Foundations of statistical natural language processing // MIT Press. Cambridge, MA: May 1999