Санкт-Петербургский
Государственный университет
Факультет прикладной математики и процессов управления

Задание по эмпирическому анализу

Расстояние Дамерау-Левенштейна

Студент
Адамович Анатолий Олегович
331 группа

Преподаватель
Никифоров Константин
Аркадьевич

29 ноября 2021 г.

Содержание

Описание исследуемого алгоритма

При решении задач обработки естественного языка часто возникает необходимость определить, насколько два различных слова *схожи* между собой. Знание об этом часто используется для автоисправления некорректных данных. В качестве примера можно привести всем известные фразы, возникающие вследствие допущения ошибки пользователем при вводе в поисковую систему: *“Возможно, вы имели в виду ...”* или *“Добавлены результаты по запросу ...”*.

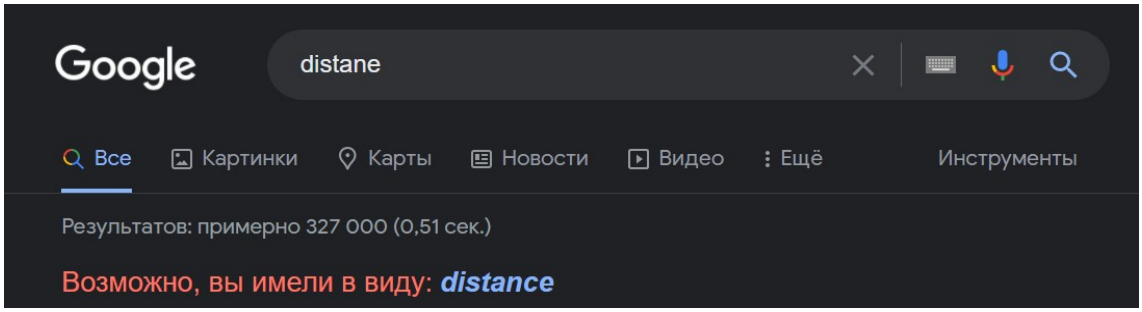


Рис. 1: Пример автокорректировки запроса

Существует несколько количественных метрик, позволяющих оценить *схожесть* слов между собой. В данной работе будет рассмотрена одна из них - **расстояние Дамерау-Левенштейна**.

Краткая история

В 1963 году учёный Фредерик Дамерау рассматривал способы улучшения систем информационного поиска. Его исследования показали, что наиболее частая ошибка - перестановка местами двух букв в слове [1].

В 1965 году Владимир Левенштейн рассматривал способы улучшения передачи двоичной информации по каналам. В ходе исследования он поставил задачу: заменить одну двоичную последовательность на другую за минимальное число шагов, используя операции вставки, замены, удаления [2].

Стоит отметить, что оба ученых не ставили перед собой цели “измерить схожесть двух строк”, однако именно с их именами связано такое важное понятие, как *редакционное предписание*.

Редакционное предписание

Расстояние Дамерау - Левенштейна - это минимальное количество вставок, замен, удалений и транспозиций символов, необходимое для преобразования одного слова в другое. При этом требуется, чтобы к транспонированным символам не применялись другие операции редактирования. Каждое из указанных действий по умолчанию имеет единич-

ную стоимость, однако этот параметр может варьироваться в зависимости от задачи, в которой используется рассматриваемая метрика.

Редакционным предписанием называют непосредственно саму последовательность наикратчайшего числа действий, позволяющих преобразовать одно слово в другое [3].

Рассмотрим в качестве примера расчет расстояния Дамерау-Левенштейна между словами «ИВАНОВ» и «БАННВО». Выполним последовательно следующие действия над словом «ИВАНОВ»:

1. добавим Н в середину: «ИВАН**Н**ОВ»
2. удалим И в начале слова: «ВАННОВ»
3. заменим В на Б в начале слова: «**Б**АННОВ»
4. сделаем транспозицию букв О и В в конце слова: «БАНН**ВО**»

Как можно заметить, потребовалось выполнить 4 действия, чтобы преобразовать одного слово в другое. Можно доказать, что это наименьшее число операций, позволяющих выполнить поставленную задачу. Таким образом, расстояние Дамерау-Левенштейна между «ИВАНОВ» и «БАННВО» равно 4.

Область применения

Как упоминалось ранее, расстояние Дамерау-Левенштейна играет важную роль в автокорректировке пользовательских запросов. Это один из показательных примеров значимости рассматриваемого алгоритма, однако для более детального анализа следует обобщить информацию об его области применения.

В настоящее время широкое распространение получают **алгоритмы нечеткого поиска**, целью которых является отыскание *с заданной точностью схожих* (например, не более 2 различий) текстов или документов. Именно эти алгоритмы лежат в основе современных методах автоматизации перевода, орфографических корректорах и поисковых системах. Ключевое влияние при построении систем нечеткого поиска оказывает выбор наиболее подходящей для рассматриваемой задачи **функции похожести строк**, так как от неё будет зависеть качество и скорость работы системы.

Одной из первых предложенных мер близости для нечеткого поиска было расстояние Левенштейна (впоследствие расстояние Дамерау-Левенштейна), которое и по сей день остается наиболее известной метрикой сравнения строк.[4]

Математический анализ алгоритма

Рекуррентное соотношение

Описываемый ниже подход изложен в статье [8].

Пусть задана строка S_1 длиной N символов и строка S_2 длиной M символов. Предполагается, что $N \geq M$.

Расстояние Левенштейна - минимальное количество операций вставки, удаления, замены символа для преобразования S_1 в S_2. Данное расстояние может быть вычислено рекурсивно:

$$lev_{i,j} = \begin{cases} i & if \;\; j = 0 \\ j & if \;\; i = 0 \\ \min \begin{cases} lev_{i-1,\,j} + 1 \\ lev_{i,\,j-1} + 1 \\ lev_{i-1,\,j-1} + I(w_{1i} \neq w_{2j}) \end{cases} & otherwise \end{cases}$$

Рис. 2: Расстояния Левенштейна: рекуррентная формула

lev_{ij} является расстоянием Левенштейна между i-префиксом строки S_1 и j-префиксом строки S_2. Под *префиксом* в данном случае следует понимать подстроку (например, i-префикс строки T означает первые i символов в строке T). В случае использования расстояния Дамерау-Левенштейна к числу базовых операций добавляется транспозиция. Рекуррентное соотношение в этом случае преобретает следующий вид:

$$dl_{i,j} = \begin{cases} i & if \;\; j = 0 \\ j & if \;\; i = 0 \\ \min \begin{cases} dl_{i-1,\,j} + 1 \\ d_{i,\,j-1} + 1 \\ dl_{i-1,\,j-1} + I\,(w_{1i} \neq w_{2j}) \\ dl_{i-2,\,j-2} + 1 \end{cases} & if \;\; i,j > 1\hat{} \\ & w_{1i} = w_{2j-1}\hat{} \\ & w_{1i-1} = w_{2j} \\ \min \begin{cases} dl_{i-1,\,j} + 1 \\ dl_{i,\,j-1} + 1 \\ dl_{i-1,\,j-1} + I\,(w_{1i} \neq w_{2j}) \end{cases} & otherwise \end{cases}$$

Рис. 3: Расстояния Дамерау-Левенштейна: рекуррентная формула

Стоит отметить, что на практике чаще всего используется **динамический подход** (оптимальное решение подзадач меньшей размерности). В этом случае хранится двумерная матрица D размером $N \times M$, которая в процессе выполнения алгоритма заполняется с помощью представленного выше рекуррентного соотношения. $D[i][j]$ содержит в себе числовое значение, характеризующее расстояние Дамерау-Левенштейна между i-префиксом строки S_1 и j-префиксом строки S_2.

Далее будет рассмотрен динамический подход.

Сложность алгоритма

Сложность алгоритма $O(M * N)$, где M - длина первой строки, а N - длина второй. Для получения расстояния Дамерау-Левенштейна необходимо заполнить двумерную матрицу, что требует $O(M * N)$ затрат по памяти. [12] Функция трудоемкости данного алгоритма зависит не только от размера входных данных, но и от их значений, поэтому данный алгоритм относится к классу *количественно-параметрических по трудоемкости* алгоритмов.

Методика исследования

Дальнейшее изложение основано на статье [9].

В настоящее время реальный анализ некоторого алгоритма предполагает получение функций трудоемкости для лучшего, среднего и худшего случаев как функций длины входа. Проблема такого подхода состоит в том, что данная оценка не позволяет получить какие-либо сведения о поведении алгоритма на конкретных входах. Кроме того, использование оценки в худшем случае приводит к существенному завышению временного прогноза. В связи с этим представляет интерес задача построения практически значимой **интервальной оценки трудоемкости** алгоритма.

Подход авторов статьи [9] заключается в рассмотрении функции трудоемкости (при фиксированной длине входа) как дискретной ограниченной случайной величины. Неизвестное распределение рассматриваемой случайной величны аппроксимируется непрерывным бета-распределением, что позволяет построить доверительный интервал для трудоемкости исследуемого алгоритма и получить её более реальную правую границу при фиксированной длине входа.

Анализ алгоритма при таком подходе разделяется на два этапа: *предварительное исследование*, цель которого состоит в том, чтобы проверить гипотезу о законе распределения значений трудоемкости некоторого алгоритма, и *основное исследование*, в результате которого определяются значение доверительной трудоемкости как функции длины входа.

Результаты проведенных исследований представлены в секции **"Вычислительный эксперимент"**.

Характеристики входных данных

Рассмотрим применение расстояния Дамерау-Левенштейна в качестве функции похожести строк при биоинформационном анализе генетических последовательностей для задачи оценки состояния репродуктивной системы.[5]

В компьютерных программах гены чаще всего представляются в виде строк, содержащих последовательность символов A, C, G, T, где каждой букве сопоставлен определенный *нуклеотид*: аденин, цитозин, гуанин, тимин.

Генетическое строение молекулы ДНК допускает искажение, поэтому расстояние Дамерау-

Левенштейна может быть использовано для оценки состояния репродуктивной системы (например, для оценки мутации рассматриваемого гена).

Для проведения экспериментов и анализа исследуемого алгоритма будем в качестве входных данных использовать последовательности нуклеотидов, представленные в формате строк. На практике длины таких последовательностей могут оказаться гигантскими и в этом случае используется следующий подход: последовательности разбиваются на относительно небольшие по длине группы и далее анализируются. Таким образом, будем предполагать, что на вход алгоритму подаются равные по длине группы нуклеотидов, на которые мы разбили исходные гены.

Способ генерации входных данных

Основной задачей в генерации входных данных является обеспечение репрезентативности выборки, то есть генерации таких входов, которые с некоторой долей вероятности соответствуют области применения исследуемого алгоритма [6]. Данное требование мы частично обеспечиваем тем, что рассматриваем конкретную задачу, связанную с использованием расстояния Дамерау-Левенштейна, - задачу сравнения нуклеотидных последовательностей.

В данном случае алфавит символов, из которых будут состоять генерируемые строки, будет содержать всего 4 элемента: A,C,G,T. Генератор принимает на вход 3 аргумента: длину первой строки, которую необходимо сгенерировать (N), длину второй строки(M) и необязательный параметр seed, характеризующий возможность инициализации генератора случайных чисел (может быть использовано для повторения результатов эксперимента). Далее генератор формирует первую строку: N раз случайным образом выбирает из рассматриваемого алфавита символ и добавляет его в конец строки. Аналогично генерируется вторая строка. При случайном отборе символов из алфавита используется равномерный закон распределения [7].

Способ измерения функции трудоемкости

В связи с тем, что измерение физического времени работы алгоритма имеет ряд недостатков, для текущего анализа применим подход, основанный на использовании счетчика базовых операций. В качестве *базовых* примем операции присваивания и сравнения.

Программная реализация алгоритма

Весь программный код (включая генератор, алгоритм, нахождение оптимального количества экспериментов, предварительный и основной этапы исследования), снабженный комментариями, может быть найден по ссылке:

`https://github.com/AnatolyAdamovich/DamerauLevenshteinDistance`

Вычислительный эксперимент

Практическими значимыми результатами анализа алгоритма является получение сведений, которые могли бы дать возможность прогнозировать ресурсные затраты, требуемые алгоритму для решения задач из выбранной предметной области. В связи с этим было принято решение использовать подход, основанный на построении доверительных интервалов для функции трудоемкости.

Постановка задачи: построить интервальную оценку трудоемкости алгоритма Дамерау-Левенштейна при заданном уровне значимости $\sigma = 0.05$.

Расчет объема выборки

Перед проведением исследования необходимо определить минимальное число экспериментов, которые позволят построить интервальную оценку. С этой целью используем подход, основанный на бета-распределении [10].

1. Проведем предварительный эксперимент: получим значения трудоемкости f_i для $i = \overline{1,k}$, где $k = 50$

2. Определим минимальные и максимальные значения трудоемкости в полученной выборке; сделаем нормирование значений выборки; вычислим выборочное среднее, выборочную дисперсию; вычислим значения параметров α, β; вычислим пределы интегрирования для заданого интервала

3. При решении интегрального уравнения было получено $n_1 = 255$.

4. Повторим шаги 1, 2 с $k = n_1$. Получим $n_2 = 169$

Так как $n_2 < n_1$, то выполнено *условие останова*. Полагаем, что оптимальное число экспериментов равняется 169.

Программная реализация описанного расчета содержится в файле **choose number of experiments.ipynb**.

Этап предварительного исследования

1. Зафиксируем длину входных данных. В данном случае мы рассматриваем задачу сравнения нуклеотидных последовательностей одинаковой длины: $N = M = 60$.

2. В соответствии с проведенным расчетом необходимое число экспериментов - 169.

3. Проведем эксперименты и получим значения трудоемкости f_i. Построим вариационный ряд.

4. В качестве оценки нижней и верхней границы функции трудоемкости при указанной длине входа будем использовать полученные экспериментальным путем данные (в качестве нижней границы возьмем первый элемент вариационного ряда, в качестве верхней - последний элемент).

5. Интервал варьирования трудоемкости $[34520, 34742]$ разобьем на 7 полусегментов.

6. Сделаем нормирование вариационного ряда и построим гистограмму относительных частот.

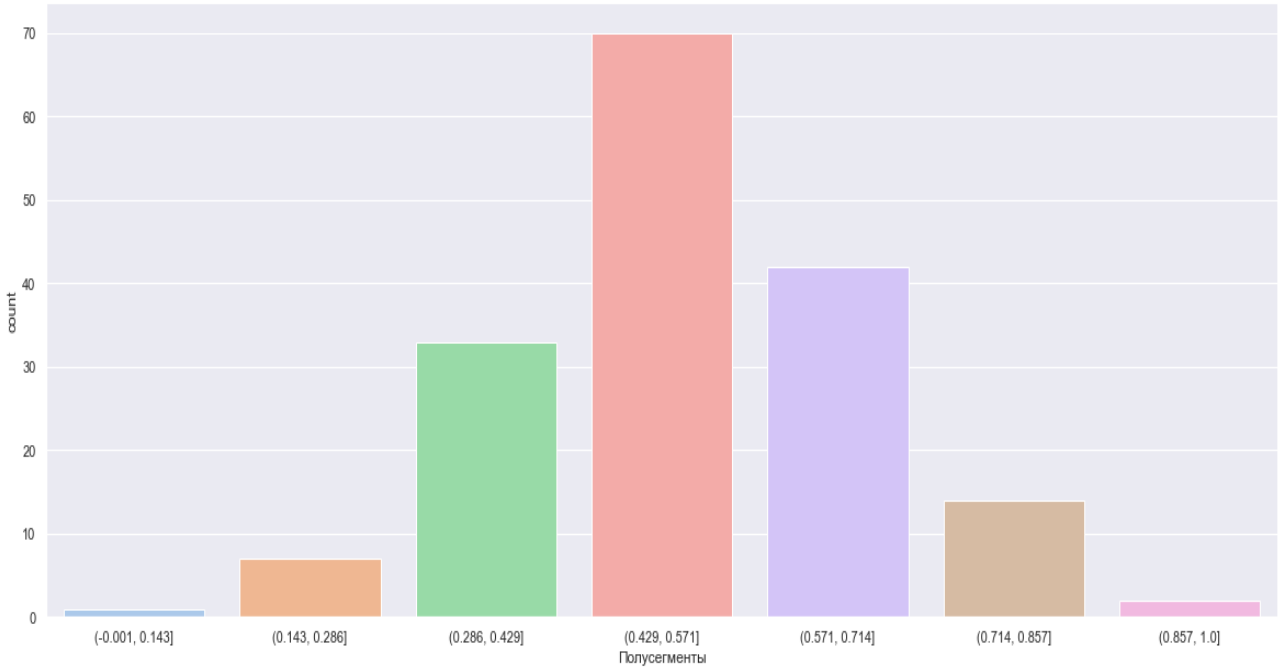


Рис. 4: Гистограмма относительных частот

7. Вычислим значения выборочной средней и выборочной дисперсии:

$\bar{t} = 0.5223, s^2 = 0.0217$

8. Используя метод моментов вычислим параметры α, β аппроксимирующего бета-распределения:

$\alpha = 5.484, \beta = 5.017.$

Выдвигаем нулевую гипотезу:

H_0 : β-распределение не противоречит наблюдаемому в эксперименте распределению относительных частот трудоемкости как случайной величины.

9 Рассчитаем теоретические частоты с помощью интегрирования функции плотности по границам полусегментов нормированного ряда.

10. Рассчитаем наблюдаемое значение критерия: $\chi^2_{current} = 5.937$.

Для вычисления теоретического значения критерия воспользуемся возможностями библиотеки scipy, предназначенного для научных вычислений [11].

Полученное значение:

$\chi^2_{theory} = 9.488$

11. Поскольку наблюдаемое значение критерия меньше теоретического значения, то с заданным уровнем значимости $\sigma = 0.05$ нет оснований опровергнуть нулевую гипотезу.

Таким образом, с вероятностью 0.95 значение трудоемкости исследуемого алгоритма для

фиксированной длины входа (в данном случае N=M=60) будет заключено в интервале $[34520, f_{0.95}]$, где $f_{0.95} = 34689$. Заметим, что этот сегмент варьирования оказался меньше исходного интервала варьирования трудоемкости $[34520, 34742]$ в 1.31 раз.

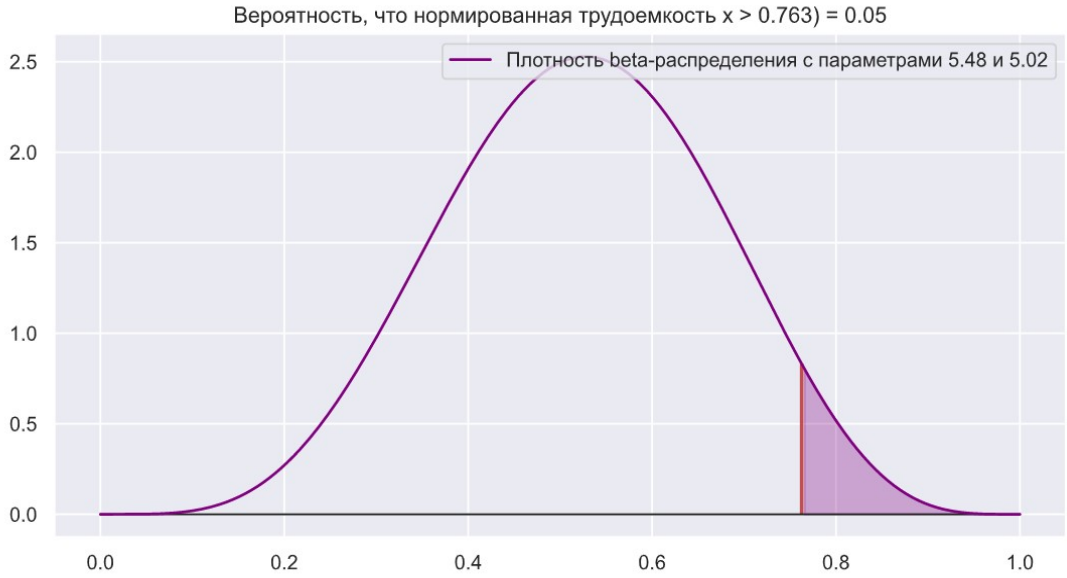


Рис. 5: График плотности бета-распределения

Этап основного исследования

1. Определим сегмент значений длин входа, соответствующий особенностям применения данного алгоритма: $N \in [60 : 250]$

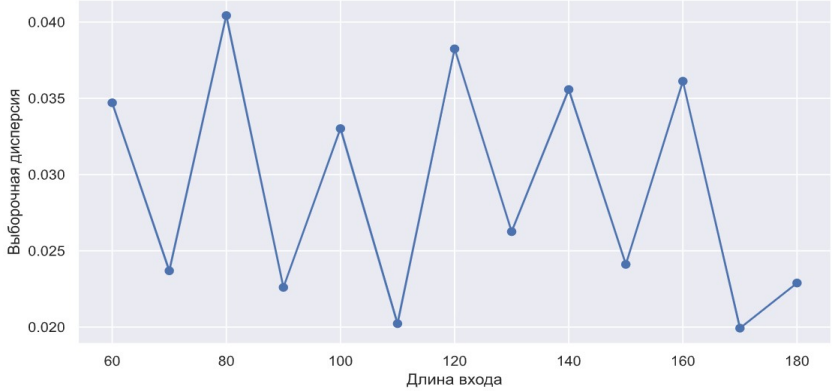
2. Определим сегмент значений длин входа, для которого будут проводиться эксперименты: $N \in [60 : 180]$

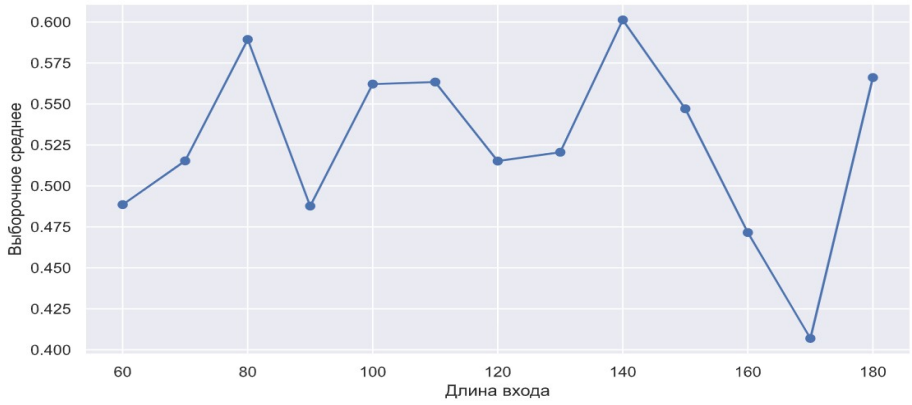
3. Шаг изменения длины входа равен 10.

4. Число экспериментов примем равным 169 в соответствии с проведенным ранее расчетом.

5. Для каждого N_i посчитаем значение выборочной дисперсии и выборочной средней.

Представим полученные результаты на графиках:





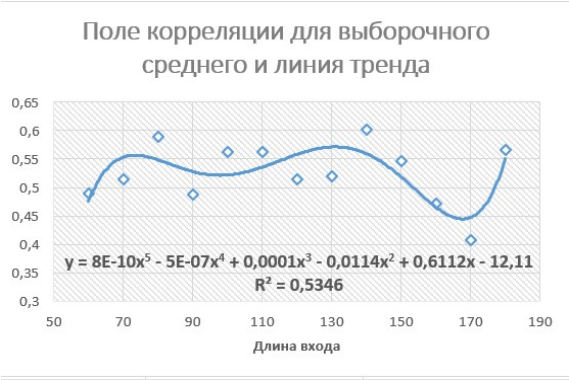
Нетрудно заметить, что данные слабо коррелированы между собой.

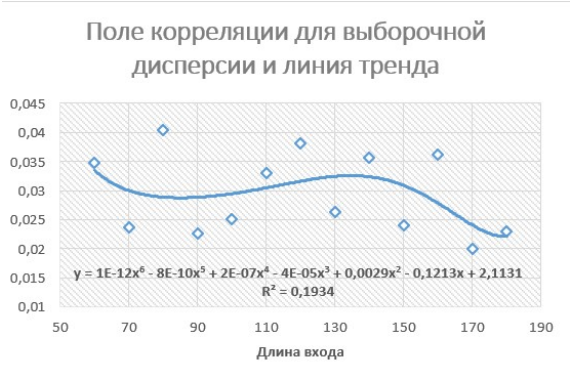
6. Для рассчета параметров бета-распределения как функции длин входа необходимо найти зависимость выборочного среднего и выборочной дисперсии от длины входа. С этой целью воспользуемся методом, описанным авторами статьи [9].

Построим уравнение регрессии для выборочного среднего и выборочной дисперсии на основе проведенных экспериментов. Кроме того, в данном случае нам неизвестны теоретические функции трудоемкости в лучшем и худшем случаях. По этой причине построим уравнения регрессии и для них.

Для построения моделей регрессии использовался MS Excel.

Можно заметить, что регрессия функций трудоемкости с помощью полиномов второй степени дает максимульную точность в смысле коэффициента детерминации R^2, что позволяет нам с приемлемой погрешностью прогнозировать значения функции трудоемкости в лучшем и худшем случае на задачах большой размерности.





Как было отмечено ранее, данные слабо коррелированы между собой. Поскольку MS Excel не позволяет построить полиномиальную регрессию более, чем шестой степени, то по этой причине будем использовать функцию *polyfit* из пакета MATLAB, с помощью которой построим регрессию более высокого порядка для выборочной дисперсии.

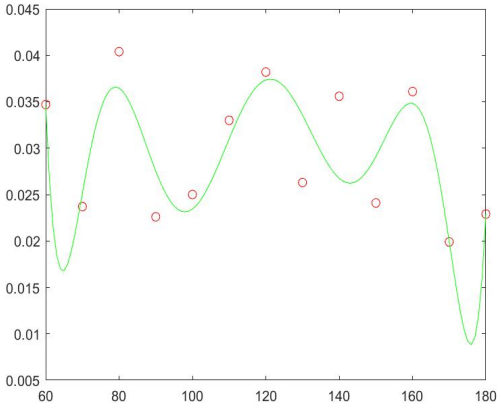


Рис. 6: Полиномиальная регрессия 8 порядка

Полученное значение коэффициента детерминации: $R^2 = 0.6287$

7. Расчитаем параметры бета-распределения как функции длины входа.

8. Вычислим значение левого 0.95-квантиля бета-распределения как функции длины входа.

9. Получим доверительную трудоемкость как функцию длины входа

$f_{0.95} = f_A^{best} + x_{0.95} * (f_A^{worst} - f_A^{best})$, где:

- $f_A^{best}(N) = 9.6296 * N^2 - 3.1575 * N + 62.919$ - оценено полиномиальной регрессией второго порядка ($R^2 = 1$)
- $f_A^{worst}(N) = 9.634 * N^2 - 0.5795 * N + 99.183$ - оценено полиномиальной регрессией второго порядка ($R^2 = 1$)
- $x_{0.95}(N) = B^{-1}(0.95, \alpha(N), \beta(N))$ - 0.95-квантиль бета-распределения как функция длины входа
- $\alpha(N), \beta(N)$ - параметры аппроксимирующего бета-распределения как функции длины входа (подсчитаны с помощью регрессии для выборочного среднего и регрессии для выборочной дисперсии)
- $\overline{t}(N) = 7.743 * 10^{-10} * N^5 - 4.5492 * 10^{-7} * N^4 + 1.0366 * 10^{-4} * N^3 - 0.0114 * N^2 + 0.6112 * N - 12.11$ - выборочное среднее как функция длины входа (оценено полиномиальной регрессией пятого порядка)
- $s^2(N) = 5.82 * 10^{-15} * N^8 - 5.596 * 10^{-12} * N^7 + 2.313 * 10^{-9} * N^6 - 5.366 * 10^{-7} * N^5 + 7.63 * 10^{-5} * N^4 - 0.0068 * N^3 + 0.371 * N^2 - 11.3389 * N + 148.15$ - выборочная дисперсия как функция длины входа (оценено полиномиальной регрессией восьмого порядка)

Анализ полученных результатов

Была получена доверительная трудоемкость как функция длины входа, на основе которой можно получать более точную оценку сложности алгоритма для входа конкретной длины из рассматриваемого сегмента. В силу того, что теоретические функции трудоемкости в лучшем и худшем случае близки друг к другу, для наглядной визуализации результатов было необходимо изменить масштаб представления графика.

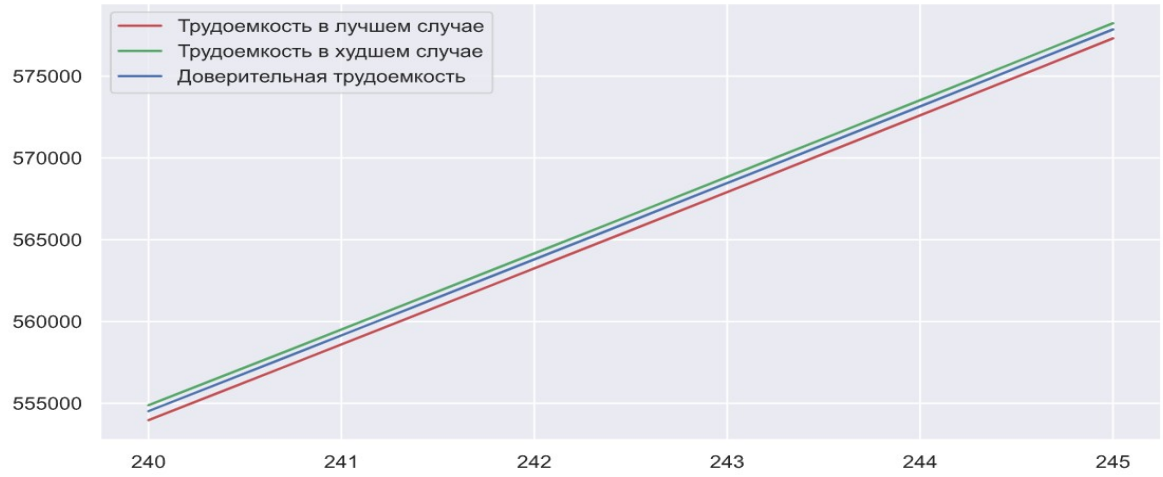


Рис. 7: Результаты

Как можно заметить, нам удалось сократить интервал варьирования. Для более информативной оценки полученных результатов вычислим среднюю абсолютную ошибку между полученными на основе регрессии функциями на сегменте $[60 : 250]$ с шагом изменения длины 10.

$$MAE(f_A^{worst}(N), f_A^{best}(N)) = \frac{1}{20} * \sum_{i \in [60,70,...,250]} (f_A^{worst}(i) - f_A^{best}(i)) = 556.194$$

$$MAE(f_{0.95}(N), f_A^{best}(N)) = \frac{1}{20} * \sum_{i \in [60,70,...,250]} (f_{0.95}(i) - f_A^{best}(i)) = 428.961$$

Характеристики использованной вычислительной среды и оборудования

Исследование проводилось в веб-интерактивной вычислительной среде **Jupyter Notebook**.

Характеристики используемого оборудования:

- процессор: 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz
- оперативная память: 8ГБ

Заключение

В данной работе был проведен анализ алгоритма Дамерау-Левенштейна с помощью подхода, основано на построении интервальной оценки функции трудоемкости с заданной точностью. В ходе исследования было выявлено, что наибольший эффект от данного подхода достигается тогда, когда $\alpha << \beta$. В этом случае сегмент варьирования функции трудоемкости ($[f_A^{best}, f_A^{worst}]$) значительно сокращается и благодаря этому повышается достоверность прогнозирования временной сложности на конкретных входах.

Несмотря на преимущества, данный метод требует глубокого анализа исследуемого алгоритма и выявление его особенностей на первоначальном этапе, потому что в случае, когда сегмент варьирования функции трудоемкости не достаточно большой (то есть когда теоретические функции трудоемкости в лучшем и худшем случае близки к друг другу), использование данного подхода не даст ожидаемый результат. Мы могли убедиться в этом на примере анализа алгоритма Дамерау-Левенштейна. Нам удалось повысить точность прогнозирования за счет вычисления доверительной трудоемкости, однако мы не смогли значительно сократить сегмент варьирования. Стоит также отметить, что получение доверительной трудоемкости на основе предложенного подхода требует больших вычислительных затрат.

Таким образом, можно сделать вывод, что эффективное использование данного метода становится доступным в случае выполнения определенных начальных условий.

Список литературы

1. Fred J. Damerau. A technique for computer detection and correction of spelling errors.
2. В. И. Левенштейн. Двоичные коды с исправлением выпадений, вставок и замещений символов. Доклады Академий Наук СССР, 1965.
3. В.М. Черненький, Ю.Е. Гапанюк. Методика идентификации пассажира по установочным данным (Вестник МГТУ им. Н.Э. Баумана, 2012).
4. Бойцов Л. М. Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска
5. Применение расстояний редактирования при биоинформационном анализе геномов для задач оценки состояния репродуктивной системы. Пономарева Н.С., Реброва Г.Н., Колина Е.А.
6. Планирование экспериментального исследования трудоемкости алгоритмов на основе бета-распределения. Ульянов М.В., Петрушин В.Н.
7. https://docs.python.org/3/library/random.html
8. Amazigh spell checker using Damerau-Levenshtein algorithm and N-gram, Youness Chaabi, Fadoua Ataa Allah. Journal of King Saud University - Computer and Information Sciences, 8 august 2020
9. Доверительная трудоемкость - новая оценка качества алгоритмов. М.В.Ульянов, В.Н.Петрушин, А.С.Кривенцов
10. Планирование экспериментального исследования трудоемкости алгоритмов на основе бета-распределения. В.Н.Петрушин, М.В.Ульянов
11. https://docs.scipy.org/doc/scipy/index.html
12. An Extension of the String-to-String Correction Problem, Robert A. Wagner, Roy Lowrance