ТЕХНИЧЕСКИЕ НАУКИ

Дударев Олег Кимович
старший преподаватель
ФГБОУ ВО «Сибирский государственный аэрокосмический
университет имени академика М.Ф. Решетнева»
г. Красноярск, Красноярский край

НЕЧЕТКАЯ ОЦЕНКА ЗНАНИЙ В ТЕСТИРОВАНИИ

Аннотация: *в статье рассматриваются актуальные проблемы тестирования, описываются недостатки существующих систем тестирования. Автором предлагается подход, основанный на нечеткой оценке правильности ответов в тестировании с результатом в виде процента правильности.*

Ключевые слова*: оценка знаний, тестирование, сравнение строк, алгоритмы нечеткого сравнения строк.*

Важным элементом в обучении является контроль знаний и умений учащегося. При этом эффективность всей учебной работы во многом зависит от его организации и нацеленности. В последние годы тесты знаний и способностей получили широкое распространение в различных областях общественно-экономической жизни. В связи с широким распространением web-технологий появляется все больше систем с широкими возможностями по автоматизации процесса обучения и контроля качества знаний и умений. Наиболее распространенные OpenSource системы: ATutor, Claroline, Dokeos, LAMS, Moodle, OLAT, OpenACS и Sakai. Каждая из них имеет свои достоинства и недостатки, однако наибольшее распространение получила система LMS Moodle. К недостаткам тестирования с вводом ответа системы LMS Moodle можно отнести:

– нет возможности учесть возможные опечатки в ответе;

– частично правильные ответы так же задаются в явном виде либо с помощью управляющих символов;

– нет возможности генерации новых вопросов из уже добавленных.

Большинство недостатков, связанных с анализом правильных ответов можно решить при помощи нечеткой оценки, где нет как такового правильного или не правильного ответа, а все ответы имеют некоторую количественную оценку. При этом ответы можно сравнивать при помощи различных алгоритмов нечеткого сравнения строк и получать количественную оценку схожести ответов, выраженную в процентах. Наиболее распространенные алгоритмы сравнения строк используют расстояния Хэмминга, Левенштейна, Джаро и др.

При оценке результатов тестирования возникает необходимость сравнивать правильный ответ с ответом, который ввел пользователь. При этом недостаточно просто сравнить ответы посимвольно, так как при этом опечатка в одном символе может быть засчитана как неверный ответ. Для сравнения ответов нужно использовать алгоритмы, позволяющие количественно оценить похожесть двух строк, при этом возникает необходимость вычисления расстояний между сравниваемыми строками.

Для сравнения строк обычно используют метрики, оценивающие минимальное количество действий (операция редактирования), необходимых для преобразования одной строки в другую. К элементарным операциям редактирования относятся операции замены, вставки и удаления символа, последние две из которых иногда объединяют в одну.

Существует множество различных подходов к выбору функции похожести строк. Одной из классических мер является расстояние Левенштейна. Функция Левенштейна – это мера разницы двух последовательностей символов относительно минимального числа элементарных операций редактирования, необходимых для перевода одной строки в другую в случае, когда операции имеют одинаковый вес.

Вычислив расстояние Левенштейна, можно найти процент совпадения строк по следующей формуле:

$$\text{percen\_of\_simil} = \frac{\text{max\_len} - \text{d}}{\text{max\_len}} \cdot 100 \ ,$$

где percent_of_simil – процент совпадения; max_len – длина наибольшей строки; d – расстояние Левенштейна.

Другим подходом является алгоритм нечеткого сравнения строк. Он использует в качестве аргументов две строки и параметр сравнения – максимальную длину сравниваемых подстрок. Результатом работы алгоритма является число, лежащее в пределах от 0 до 1.0 соответствует полному несовпадению двух строк, а 1 – полной (в определённом ниже смысле) их идентичности.

Сравнение строк происходит по следующей схеме. Пусть, например, в качестве аргументов заданы две строки «test» и «text» и некоторая максимальная длина подстрок, скажем, 4. Функция сравнения составляет все возможные комбинации подстрок с длинной вплоть до указанной и подсчитывает их совпадения в двух сравниваемых строках. Количество совпадений, разделённое на число вариантов, объявляется коэффициентом схожести строк и выдаётся в качестве результата работы алгоритма.

Расстояние Джаро – расстояние, вычисляемое для двух строк, равное:

$$d_j = \frac{1}{3}\left(\frac{m}{s_1} + \frac{m}{s_2} + \frac{m-t}{m}\right),$$

где m – число схожих символов, t – число транспозиций – количество схожих, но отличающихся позицией в строке символов, деленное на 2.

Два символа являются схожими, если находятся друг от друга на расстоянии не более

$$\frac{\max(s_1, s_2)}{2} - 1 .$$

Данная метрика позволяет сравнивать строки различной длины, а также учитывать перестановки в строках.

Особый интерес представляет функция similar_text из стандартной библиотеки языка программирования php, который широко используется в web-программировании и написании online-тестов. Функция вычисляет степень похожести двух строк по алгоритму, описанному Oliver [1993]. Эта реализация

алгоритма не использует стека, использованного в оригинале, вместо этого применяются рекурсивные вызовы, что в некоторых случаях может ускорить процесс. Сложность алгоритма составляет O(N**3), где N – длина более длинной из двух строк, что делает ее гораздо медленнее функции Левенштейна.

Было проведено сравнение алгоритмов с учетом распространенных ошибок при вводе текста: перестановка символов, неверный ввод символа, пропуск символа, ввод лишнего символа, ошибка регистра символов.

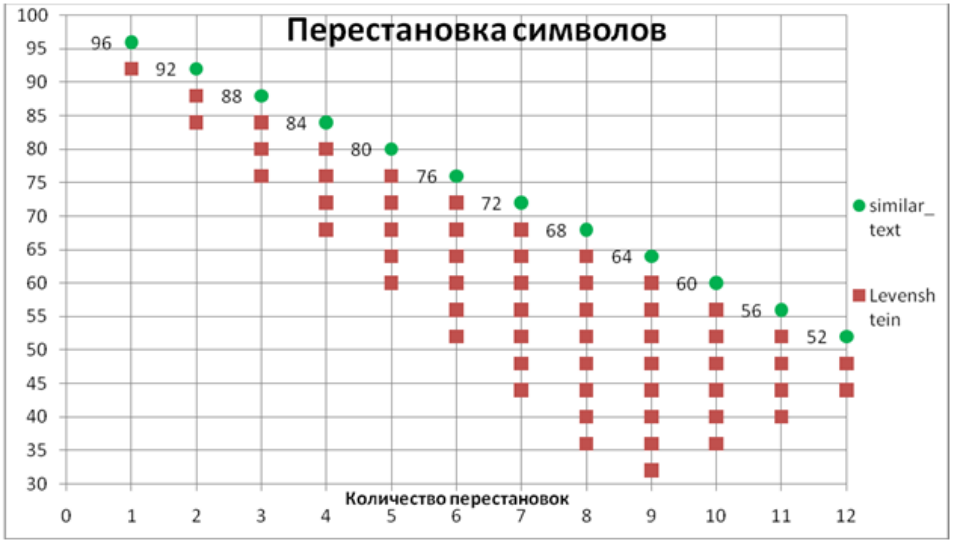

Рис. 1. Пример сравнения работы функции similar_text и levenshtein при перестановке символов

Выводы: Функция Левенштейна является одной из самых мощных функций для сравнения строк, в зависимости от реализуемой задачи, функция позволяет задавать стоимость вставки символа, стоимость удаления, а также стоимость замены, при этом есть возможность задать функцию вычисляющую сложность трансформации. Все эти настройки позволяют применять функцию при решении множества задач. К примеру, если в задаче важно, чтобы длины строк совпадали, иначе решение будет сильно отличаться от верного, можно задать высокую стоимость вставки, при этом процент совпадения таких строк будет гораздо ниже. При всей своей мощности функция довольно быстрая, сложность алгоритма этой функции равна O(m*n), т.е. пропорциональна произведению длин строк str1 и str2, поэтому эта функция намного более быстрая, чем функция similar_text,

сложность алгоритма которой составляет O(N**3), где N – длина более длинной из двух строк. Однако в некоторых случаях, все же лучше использовать функцию similar_text, это касается, прежде всего, коротких строк, так как при использовании функции Левенштейна при транспозиции получаются довольно большие расстояния. Например, при использовании функции Левенштейна строки «привет» и «рпивет» будут схожи всего на 66,66%, в то время как функция similar_text даст 83,33%. Так же зависимость положения транспозиции может быть не важна в некоторых задачах. Например, если последовательность 123456789 является единственно верной, то совершенно неважно, где были допущены ошибки, однако функция Левенштейна даст различные проценты совпадения для последовательностей 214356789(66,67%) и 213456798(55,56%), хотя в обоих случаях было сделано 2 перестановки. В этих случаях можно использовать расстояние Дамерау-Левенштейна, в котором добавлена операция транспозиции или функцию similar_text. Но если нет одной единственно верной последовательность, и ее нужно выбрать из набора наиболее подходящих, как это сделано в MS Word при проверке правописания, то расстояние Левенштейна может дать этот необходимый набор последовательностей. В остальном эти функции схожи, за исключением линейности функции Левенштейна при пропуске символов, в отличие от функции similar_text, которая с увеличением количества пропущенных символов дает меньший процент совпадения строк в переводе на один символ.

Если рассматривать задачу с точки зрения положения ошибки в строке, следует рассмотреть расстояние Джаро и Алгоритм нечеткого сравнения строк, которые даже для одной ошибки дают различные результаты в зависимости от ее положения, что не наблюдается у рассмотренных ранее функций. Алгоритм нечеткого поиска все время дает больший результат при приближении ошибки к началу и концу строки, следовательно, можно сделать вывод, что этот алгоритм хорошо подходит для сравнения строк, в которых наиболее важно совпадение в середине строки. Например, это может быть задание на установление правильной последовательность дат, в котором, как правило, начальное и конечное событие установить довольно просто. Интересной особенностью расстояния

Джаро является его поведение при пропуске или вводе лишнего символа в строке. Чем дальше от начала строки расположена ошибка такого рода, тем больше процент совпадения таких строк, следовательно, можно сделать вывод, что алгоритм хорошо подходит для сравнения строк, в которых наиболее важно совпадение в начале строки. Для строк небольшой длины желательно использовать расстояние Джаро-Винклера, в котором используется масштабный коэффициент.

Результатом проделанной работы стала уникальная система web-тестирования, реализующая поставленные задачи по анализу результатов с помощью функций нечеткой оценки, с возможностью выбора алгоритма сравнения строк для каждого теста. Таким образом, можно задать различную степень строгости сравнения вводимых ответов с эталоном, в зависимости от целей тестирования.

Список литературы

1. Navarro G. A Guided Tour to Approximate String Matching. ACM Computing Surveys. – 2001. – V. 33(1). – P. 31–88.

2. Андреев А.В. Практика электронного обучения с использованием Moodle / А.В. Андреев, С.В. Андреева, И.Б. Доценко. – Таганрог: Изд-во, ТТИ ЮФУ, 2008. – 146 с.

3. Ахмедова Е.В., Филиппова О.В. Методические аспекты тестирования как одной из форм контроля при обучении // Информационный портал ИГХТУ, 2007.

4. Бойцов Л.М. Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска // Труды Всероссийской конференции RCDL'2004.