Поиск и Выделение Географических Адресов в Связных Текстах.

Сильчев А.В.

***Аннотация*— В данной статье изложены результаты квалификационной работы, выполненной автором в Лаборатории ОИТ во время обучения по программе дополнительного образования на факультете ВМК МГУ им. М.В. Ломоносова. Рассматривается задача поиска и выделения географических адресов в связных текстах на русском языке. Автор предложил и реализовал алгоритм нечеткого поиска на основе вычисления расстояний между строками.**

***Ключевые слова*—геокодинг, обработка текстов.**

I. Введение

В некоторых приложениях возникает необходимость проанализировать произвольный текст на наличие в нем географических адресов. Например, это могут быть web-приложения и сервисы, которые читают содержимое новостных сайтов (RSS-ленты) и формируют наглядное представление о том, что и где произошло. Так, например, из текста новости можно получить адрес произошедшего события и обозначить его на карте с кратким описанием этого события. Это может быть полезным для быстрого обзора новостей и событий, произошедших на интересующей нас территории. «Что интересного произошло в таком-то районе за последние сутки?»

Общего решения этой проблемы не существует, существуют лишь различные методы и подходы к ее решению, обладающие теми или иными достоинствами и недостатками. В частности различные алгоритмы обладают разной скоростью обработки информации и разной степенью точности.

Дополнительную сложность этой проблеме добавляют морфологические особенности языка, отсутствие четкого формата записи адреса, а так же ошибки, возникающие при наборе текста. Если с морфологией языка и отсутствием формата записи адреса еще можно как-то смириться, то ошибки наборщика нередко настолько осложняют задачу, что даже человеку, читающему текст, трудно понять что имелось в виду на самом деле, не говоря уж об алгоритмах.

В данной работе рассмотрен реализованный автором алгоритм нечеткого поиска, в основе которого лежит понятие расстояния между двумя строками, который может быть использован в работе описанных выше web-сервисов.

В данной работе рассматривается следующая задача - необходимо реализовать алгоритм, который из произвольного текста выделяет города Российской Федерации. Иначе говоря, на входе имеется произвольный текст, на выходе – список городов, упоминавшихся в этом тексте. При этом текст может быть произвольным, произвольной длины. Данный алгоритм должен выявлять названия городов, независимо от падежа, в котором они входят в текст.

II. Обзор существующих алгоритмов

Для начала введем некоторые обозначения, которые будут использоваться везде далее по тексту.

Прежде всего, рассматриваемые последовательности – одномерные и составлены из символов. Таким образом, под строкой символов X будем понимать последовательность символов $x_1 x_2 \ldots x_n$ такая строка будет иметь длину n, этот факт будем обозначать следующим образом $|X| = n$.

На данный момент в существующей литературе рассматривается достаточно большое количество разнообразных подходов к решению задачи о нечетком поиске. Большая часть этих алгоритмов оперирует с таким понятием как степень схожести двух строк. Для этого некоторым образом определяется функция d, вычисляющая так называемое «расстояние» между двумя строками. Здесь необходимо отметить, что под расстоянием понимается не вполне привычное понятие, такое как, например, расстояние между двумя точками. Функция d должна удовлетворять следующим условиям:

1) $d(x, y) \geq 0, \forall x, y$ - неотрицательность;

2) $d(x, y) = d(y, x), \forall x, y$ - симметричность;

3) $d(x, y) = 0 \Leftrightarrow x = y$ - тождественность;

4) $d(x, z) \leq d(x, y) + d(y, z), \forall x, y, z$ - неравенство треугольника.

Статья получена 20 сентября 2013.

Сильчев А.В. – выпускник программы дополнительного образования факультета Вычислительной Математики и Кибернетики Московского государственного университета им. Ломоносова.

Здесь под переменными x, y, z понимаются некоторые последовательности символов (строки). Таким образом, функция d двум своим аргументам ставит в соответствие некоторое число. В зависимости от того, каким образом задана функция d, по результату ее вычисления можно говорить остепени схожести ее аргументов. Функция d называется метрикой.

Теперь можно сформулировать задачу о нечетком сопоставлении строк, и она будет выглядеть следующим образом:

Пусть даны две последовательности – образец X, $|X| = m$, и текст $Y, |Y| = n; m, n > 0$ и $m \le n$. Пусть даны также целое $k \ge 0$ и функция расстояния d. Требуется найти все подстроки s текста y такие, что d(x, s) $\le$ k.

Расстояние Хемминга [1] между двумя строками одинаковой длины определяется как число позиций, в которых символы не совпадают. Это эквивалентно минимальной цене преобразования первой строки во вторую в случае, когда разрешена только операция замены с единичным весом. Если допускается сравнение строк разной длины, то, как правило, требуются также вставка и удаление. Если придать им тот же вес, что и замене, минимальная общая цена преобразования будет равна одной из метрик, предложенных Левенштейном [2]. Другая метрика равна минимальной цене преобразования в случае, когда разрешены только вставка и удаление. Первую из этих метрик называют расстоянием Левенштейна, а вторую - расстоянием редактирования.

Необходимо заметить, что не все функции, измеряющие расстояние между строками, являются метриками. Например, если цены операций редактирования являются функциями конкретных символов, неравенство треугольника может не выполняться.

Метод вычисления расстояния между двумя строками, основанный на методе динамического программирования был предложен несколькими исследователями. Чаще всего он именуется как метод Вагнера-Фишера [3]. Идея метода динамического программирования состоит в том, чтобы последовательно оценивать расстояния между все более длинными префиксами строк до получения окончательного результата. Промежуточные результаты вычисляются итеративно и хранятся в массиве длины $(m+1)\times(n+1)$ что приводит к времени и памяти O(mn). Метод Вагнера и Фишера, основанный на структуре, которую они назвали следом (trace), позволяет вычислить самую длинную общую под-последовательность прямо по заполненному массиву расстояний.

Более подробно этот метод рассмотрен далее.

Пусть $d_{i,j}$ есть расстояние между префиксами строк *x* и *y*, длины которых равны, соответственно, *i* и *j*, то есть:

$d_{i,j} = d(x(1,i), y(1,j))$

Цену преобразования символа *a* в символ *b* обозначим через *w(a,b)*. Таким образом, *w(a,b)* – это цена замены одного символа на другой, когда $a \ne b$, $w(a, \varepsilon)$ – цена удаления *a*, а $w(\varepsilon, b)$ – цена вставки b. Заметим, что в случае, когда выполнены нижеследующие условия, *d* является расстоянием Левенштейна:

$w(a, \varepsilon) = 1$

$w(\varepsilon, b) = 1$

$w(a, b) = 1, если\ a \ne b$

$w(a, b) = 0, если\ a = b$

В процессе вычислений значения $d_{i,j}$ записываются в массив *(m+1)* × *(n+1)*, а вычисляются они с помощью следующего рекуррентного соотношения.

$$d_{ij} = \min\{d_{i-1,j} + w(x_i, \varepsilon), d_{i,j-1} + w(\varepsilon, y_i), d_{i-1,j-1} + w(x_i, y_i)\}$$

Хиршберг модифицировал метод Вагнера и Фишера так, что временные затраты остались равными O(mn), но затраты памяти стали линейными, не квадратичными. В его методе итеративно вычисляются длины самой длинной общей под-последовательности последовательно увеличивающихся префиксов строк, а не расстояния между ними. Экономия памяти происходит за счет того, что в каждый заданный момент времени в памяти хранятся только две строки массива, требуемого динамическому программированию. Потеря остальной части массива, однако, приводит к более сложному методу выделения самой длинной общей под-последовательности двух строк.

Цель Хиршберга [4] состояла в том, чтобы найти самую длинную общую под-последовательность *lcs(x,y)* для строк *x* и *y*. Поэтому вместо расстояний между строками определялись длины самых длинных общих под-последовательностей у все более и более длинных префиксов.

Алгоритм анализа текста приведен на рисунке ЧЧЧ. Образец сканируется в цикле for параллельно с текстом слева на права по одному символу за раз. На итерации i с образцом сравнивается подстрока y(i+1, i+m). Самая правая позиция в тексте, достигнутая за предыдущие итерации, задается значением j, то есть j является максимальным из чисел r+tm[r, k + 1], где $0 \le r < i$. Если i < j, вызывается процедура merge, которая находит несовпадения между x(1, j - i) и y(i+1,j) и устанавливает b равным найденному числу. Если оно не превышает k, вызывается процедура extend. Она сканирует текст впереди от y[j + 1], пока либо не будет найдено k + 1 несовпадений, либо не будет достигнуто y[i+m] с не

больше чем k несовпадениями, то есть найдено вхождение образца, начинающееся с y[i+1].

```
- инициализация
tm[0 ... n - m, 1 ... k + 1] = m + 1
r = 0
j = 0
- сканирование текста
for i = 0 to n - m
  b = 0
  if i < J
    merge(I, R, J, B)
  IF B < K + 1
    R=I
      extend(I, J, B)
```

Рис.1 Обработка текста

III. Алгоритм решения

В ходе предварительных исследований различных подходов к решению поставленной задачи было принято следующее решение – использовать готовый список городов Российской Федерации, что должно обеспечить достаточно высокую точность анализа текста. Таким образом, одной из подзадач в данной работе является получение этого списка.

При анализе исходного текста будет использовано несколько алгоритмов, ключевым из которых будет алгоритм, выявляющий степень похожести двух строк на основе понятия расстояния между строками.

Автором данной работы были проведены исследования того, как изменяются окончания названий городов в зависимости от того, в каком падеже они употребляются в тексте. В результате данных исследований был установлен ряд зависимостей.

Для начала введем некоторые определения. Словарь – полный список городов, названия городов в словаре даны в именительном падеже в единственном числе, такую форму слова далее по тексту для краткости будем называть инфинитивом (infinitive).

Далее будут рассмотрены различные изменения инфинитива на примере нескольких городов, исходя из того, каким образом название города может быть употреблено в тексте.

В таблице 1 представлены различные словоформы городов «Москва», и двух сел «Долгое» и «Мещанское», здесь необходимо подчеркнуть, что все дальнейшие заключения верны для названий городов, заканчивающихся на гласную букву.

Падеж	Словоформы		
Именительный	Москва	Долгое	Мещанское
Родительный	Москвы	Долгого	Мещанского
Дательный	Москве	Долгому	Мещанскому
Винительный	Москву	Долгое	Мещанское
Творительный	Москвой	Долгим	Мещанским
Предложный	Москве	Долгом	Мещанском

Заметим, что все приведенные в таблице формы названий являются всевозможными, т.е. в тексте никакие иные формы употребляться не могут. Так, например, употребление этих слов во множественном числе исключено, нельзя сказать «в Москвах». Так же легко видеть, что длина инфинитива отличается от остальных форм не более чем на 1 символ.

Пусть X – инфинитив, Y – одна из остальных форм, тогда ||X| - |Y|| <= 1, причем |Y| - |X| >= 0, другими словами, длина словоформы по отношению к инфинитиву может только увеличиться или остаться прежней, но не уменьшиться.

Немаловажной характеристикой является амплитуда изменения окончания – максимальное количество символов, на которое может отличаться инфинитив от любой из своих словоформ. Для более ясного представления рассмотрим несколько примеров:

Пусть X=”Москва”, Y_1 =”Москвы”, окончания отличаются на 1 символ, получаем A_1=1 , аналогичным образом поступим со всеми остальными словоформами, для Y_4 =”Москвой” A_4=2 . Найдем амплитуду изменения окончания, согласно вышеприведенному определению:

$$A(X) = \max(\{A_i\}), i = 1,...,5$$
$$A(\text{"}Москва\text{"}) = \max(\{1,1,1,2,1\}) = 2$$

Проанализировав все слова из словаря, заканчивающиеся на гласную букву, можно найти максимум из всех амплитуд. Он оказывается равен 3.

Теперь рассмотрим изменения названий городов, оканчивающихся на согласную букву.

Падеж	Словоформы		
Именительный	Серпухов	Новгород	Воронеж
Родительный	Серпухова	Новгорода	Воронежа
Дательный	Серпухову	Новгороду	Воронежу
Винительный	Серпухов	Новгород	Воронеж
Творительный	Серпуховым	Новгородом	Воронежем
Предложный	Серпухове	Новгороде	Воронеже

Из этой таблицы видно, что винительный падеж можно не принимать во внимание, т.к. он не изменяет форму инфинитива, таким образом, им можно пренебречь в дальнейших рассуждениях. Также из этой таблицы видно, что 0<|Y|-|X|<=2 в этом неравенстве следует обратить особое внимание на левую часть, смысл его в том, что словоформа всегда больше длины инфинитива, но не более чем на два символа (см. правую часть).

Более того, инфинитив целиком и полностью содержится в любой своей словоформе, это обстоятельство в дальнейшем значительно поможет при сравнении строк. Легко видеть, что максимум из всех амплитуд у слов, заканчивающихся на согласную букву равен 2. (A=2)

Отдельного рассмотрения требует случай, когда название города заканчивается на «й». В таблице.3 рассмотрен этот случай:

Падеж	Словоформы		
Именительный	Горный	Мирный	Солнечный
Родительный	Горного	Мирного	Солнечного
Дательный	Горному	Мирному	Солнечному
Винительный	Горный	Мирный	Солнечный
Творительный	Горным	Мирным	Солнечным
Предложный	Горном	Мирном	Солнечном

Отдельное рассмотрение требуется ввиду того что «й» с одной стороны является согласной буквой, с другой стороны, образованные от такого инфинитива словоформы не обладают свойствами, которыми обладают словоформы, образованные от инфинитива, заканчивающегося на согласную. В частности 0<=|Y|-|X| <=1, причем инфинитив никогда целиком и полностью не содержится в словоформе. Кроме того A=3.

Сводка полученных результатов:

Характеристика окончания слова	R	A
Гласная	1	2
Согласная	2	2
«й»	1	3

Здесь R – максимум из модулей разностей длин инфинитива и его словоформы. Данные результаты будут необходимы в дальнейшем при анализе текста.

Также необходимо сделать несколько замечаний относительно того, какие строки можно смело отбросить и не подвергать никакому анализу, т.к. они заведомо не могут являться названиями городов. Это все слова, для которых |X| <= 2, как правило, это предлоги и союзы. Количество городов, названия которых состоят из двух символов крайне мало для того, чтобы ими можно было пренебречь в пользу повышения скорости обработки текста.

Дополнительно в словарь включены такие слова как «Столица» и «Северная столица», так как они довольно часто употребляются как замена слов «Москва» и «Санкт-Петербург» соответственно.

IV. Создание словаря

Одной из подзадач в данной работе является получение и подготовка к использованию полного списка городов Российской Федерации. Наиболее достоверный и актуальный список имеется у ФГУП «Почта России», это предприятие предоставляет web-сервис, который находится по адресу http://info.russianpost.ru/servlet/department. Этот сервис позволяет по запросу индекса получить название почтового отделения, соответствующего этому индексу. Исходя из того, что индекс является шестизначным числом, можно легко получить список городов, перебрав всевозможные значения от 000000 до 999999.

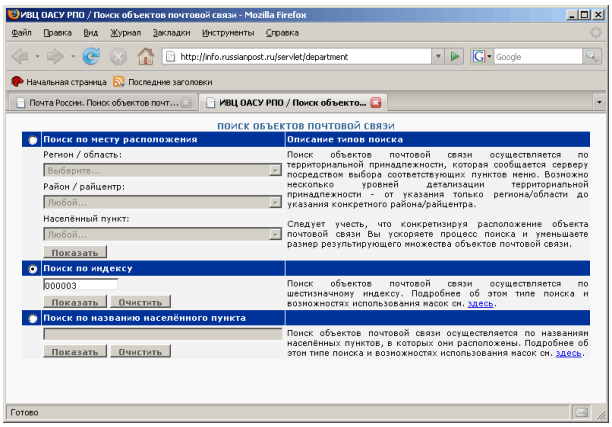

Рис. 2 Сервис Почта России

Автором работы была написана программа, которая перебирала всевозможные индексы, получала ответ с сервера, проводила разбор html-страницы (формат фиксирован), таким образом был получен список всех городов.

Следующим шагом была подготовка словаря, в ходе которой были удалены дубликаты и записи, соответствующие нескольким почтовым отделениям, находящимся в одном и том же городе – «Москва-1», «Москва-2» и т.д.

Далее список был разбит на группы следующим образом, что города, начинающиеся на одну и ту же букву, были отнесены к одной группе. Каждая группа была записана в отдельный файл, что позволило несколько ускорить поиск в данном списке.

Для более удобной работы с преобразованным списком был создан небольшой сервисный фреймворк (пакет). Этот пакет представляет собой набор классов, написанных на языке Java, в них реализованы методы, позволяющие производить элементарные операции со списком, такие как полнотекстовый поиск, выборка с

условием по количеству букв и т.д. Основным недостатком такого подхода к разбиению является тот факт, что размеры некоторых файлов сильно отличаются друг от друга, так, к примеру, количество городов на букву «Щ» гораздо меньше, чем городов на букву «Н». Наиболее привлекательным в данном случае решением было бы использование сбалансированных списков, что может являться предметом дальнейшего рассмотрения.

V. Определение расстояния между строками

Перед анализом исходного текста он подвергается предварительной обработке, из него удаляются лишние пробелы и символы табуляции. Далее текст разбивается на составляющие его слова. Как говорилось выше при проходе текста можно исключить те слова, которые заведомо не являются названиями городов, это все слова X, такие, что |X|<=2, к ним относятся предлоги и союзы. Так же к исключаемым словам необходимо отнести предлоги и частицы, длина которых больше чем 2 символа – «перед», «около», «так» и.д. Эти операции помогают ускорить анализ текста, исключая лишние операции сравнений и вычисления метрик.

Как уже говорилось выше, в основе разработанного автором данной работы анализатора текста, лежит алгоритм нечеткого сопоставления строк. Данный алгоритм использует понятие расстояния *d* между строками. Функция *d* определена следующим образом:

$$d(X,Y)=\sum_{i=1}^{n}\frac{1}{i^2} \quad (1)$$

здесь j-ый член суммы равен 0, если символы, стоящие в j-ой позиции в словах X и Y совпадают и $n=\max(|X|,|Y|)$

Примем во внимание тот факт, что:

$$d(X,Y)=\sum_{i=1}^{n}\frac{1}{i^2}<\lim_{n\to\infty}\sum_{i=1}^{n}\frac{1}{i^2}=\frac{\pi^2}{6}$$

Это означает, что значение получаемых метрик, ограничены сверху числом

$$\frac{\pi^2}{6}$$

Для двух одинаковых строк X и У *d(X,Y)*=0 , т.к. каждый член суммы становится равным нулю, согласно данному выше определению (1).

Сумма в правой части равенства (1) при всех ненулевых ее членах растет очень медленно при увеличении *n*, наглядно это можно продемонстрировать на следующем графике (рисунок 3).

Очевидно, что чем больше первых букв в обоих словах совпадают, тем меньше значение *d*. Другими словами, *d* тем менее чувствительна к изменениям (различиям) в двух словах, чем дальше от начала слова эти изменения встречаются.

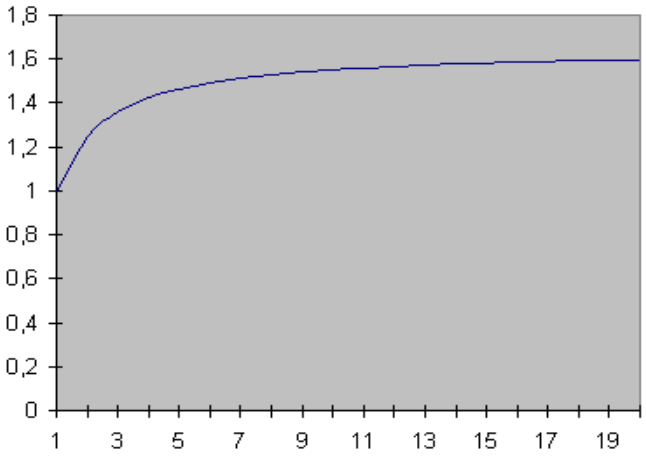

Рис. 3 Чувствительность функции расстояния

Из определения функции *d* видно, что она зависит так же и от длины ее аргументов. Продемонстрируем этот факт: Пусть X=”AAA” Y=”AAB”:

$$d(X,Y)=\frac{1}{9}\approx 0.11$$

Теперь положим X=”AAAA” Y=”AAAB”:

$$d(X,Y)=\frac{1}{16}\approx 0.0625$$

Т.е. две пары слов отличаются на одну букву, но при этом имеют разную длину.

Теперь можно воспользоваться полученными ранее результатами о зависимости характеристик *R* и *A* от того на какую букву (гласную/согласную) оканчивается слово и определить предельные интервалы значения *d*, в зависимости от длины исходного слова.

Для этих целей автором были реализованы две функции, одна из которых вычисляет нижнее предельное значение, другая – верхнее. На вход этим функция передается длина исследуемого слова и последний символ. Поученные значения ограничивают интервал, в который должна попасть метрика, в том случае, когда можно говорить о том, что два слова достаточно близки друг к другу, чтобы считать их «одинаковыми».

Более подробно это выглядит следующим образом,- на основании того, на какую букву заканчивается слово, берется соответствующая амплитуда *A*, и вычисляются суммы, в которых обнулены первые *(n-A)* членов (наихудший случай, когда расхождения начинаются рано) и первые *(n+A-1)* (наилучший случай, когда расхождения начинаются поздно).

Таким образом, получаются значения предельного интервала. Как уже говорилось выше, при попадании значения метрики в этот интервал мы можем считать два слова «одинаковыми».

Реализованный алгоритм работает следующим образом - последовательно перебираются слова исходного текста. Для каждого такого слова начинается перебор слов из

словаря с соответствующей страницы. Под соответствующей слову страницей понимается файл, в котором содержатся названия городов, первая буква которых совпадают с первой буквой слова.

На основе таблицы характеристик *A* и *R* отбрасываются заведомо неподходящие слова, так например если слово из исходного текста – «Сера», то выбранное из словаря слово «Серпухов» будет отброшено, т.к.

|"Серпухов"| - |" Сера"| = 4, а в таблице характеристик *RA* предельное значение *R* для слов, оканчивающихся на согласные равно 2. Таким образом, будут отброшены заведомо «лишние» слова. Если же *R* и *A* удовлетворяют таблице характеристик, тогда вычисляется значение метрики и границы предельно допустимого интервала. В случае если значение метрики попадает в предельно допустимый интервал, то у нас достаточно оснований полагать, что слово из исходного текста достаточно близко к слову из словаря, чтобы считать их «одинаковыми». На рисунке 4 приведена краткая схема работы алгоритма.

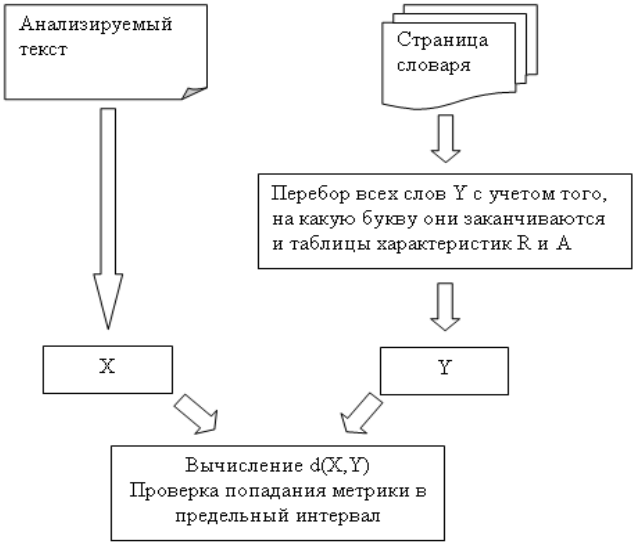

Рис. 4. Алгоритм

А на следующих двух рисунках представлен пример работы программы.

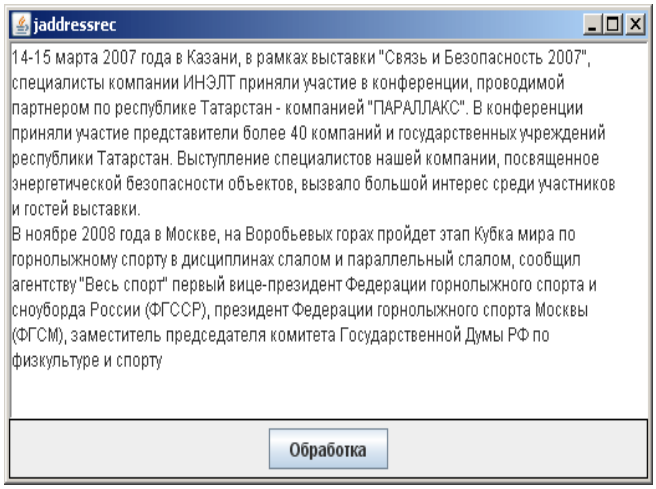

Рис. 5 Исходный пример

На рисунке 5 представлен простой тестовый пример программы, главное окно содержит поле для ввода исходного (анализируемого) текста и кнопку, по нажатию на которую происходит обработка. В текстовом поле введен текст, который взят с одного из новостных сайтов.

После нажатия на кнопку «Обработка» появится окно с результатом – списком найденных в тексте городов - рисунок 6.

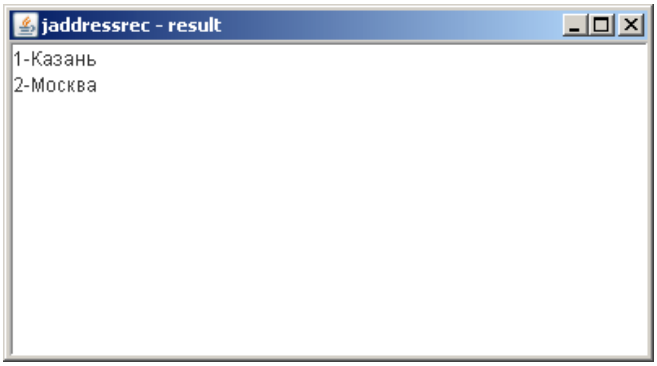

Рис. 6 Результаты работы

VI. Примечание редактора

Мы решили вернуться к этой ранее не опубликованной работе в рамках описания научных тем, предлагаемых для квалификационных работ в лаборатории ОИТ [10]. Вполне годная по времени создания и не утратившая актуальности до сих пор работа. Да, задача на самом деле ставилась когда-то шире. Идея состояла в создании некоторого аналога Yahoo Places (Boss PlaceFinder), но способного работать с русским языком. Финальная цель, например, могла бы заключаться в автоматической конвертации некоторого RSS потока в Geo RSS feed. Автоматическое гео-кодирование для новостных сообщений. Из серии квалификационных работ по этой тематике, работа А.Сильчева была одной из наиболее удачных.

Между прочим, полностью поставленная задача свою актуальность не утратила. Открытые сервисы автоматического гео-кодинга неизвестны авторам. А как возможную модель применения в новых условиях, можно привести, например, контекстно-зависимые вычисления. Например, в [11] описана модель получения данных из социальных сетей на основе информации о сетевой близости. Применение гео-кодинга к получаемым текстам помогло бы еще и позиционировать сообщения и т.д.

VII. Заключение

В работе рассматривается алгоритм автоматического гео-кодинга для тестов на русском языке. Автором данной работы был предложен и практически реализован алгоритм выделения названий городов из новостных текстов.

Предложенный алгоритм не лишен недостатков. В частности, он является чувствительным к разным формам написания названий городов (например, Н. Новгород и Ниж. Новгород и т.д.). К достоинствам алгоритма относятся простота реализации, скорость работы, а также отсутствие необходимости обучения системы. В целом, реализованный алгоритм, показал

приемлемую скорость обработки текста. Так, например, текст размером 3-5Кб обрабатывается около 0.3-0.5 сек, без учета «холодного старта» виртуальной машины Java.

.

Библиография

[1] Гаврилов, А. В. "Проблемы обработки символьной информации в нейронных сетях. Междунар. конф.«SCM-98»." Петербург, июнь (1998).

[2] Бойцов, Л.М.. "Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска." Труды 6 (2004).

[3] Цыганов Н. Метод вычисления обобщенного расстояния редактирования с аффинными штрафами //Материалы научной конференции МИФИ. – 2007. – С. 157-159.

[4] Демидова М. В. АЛГОРИТМЫ НЕЧЕТКОГО СЛОВАРНОГО ПОИСКА С ИСПОЛЬЗОВАНИЕМ ФУНКЦИЙ СХОДСТВА //Сборник трудов Всероссийской научной школы-семинара МОЛОДЫХ УЧЕНЫХ, АСПИРАНТОВ И СТУДЕНТОВ «ИНТЕЛЛЕКТУАЛИЗАЦИЯ ИНФОРМАЦИОННОГО ПОИСКА, СКАНТЕХНОЛОГИИ И ЭЛЕКТРОННЫЕ БИБЛИОТЕКИ».–Таганрог: Изд-во ТТИ ЮФУ, 2008.–156 с. – С. 60.

[5] Дональд Э. Кнут. «Искусство программирования» Том-3 «Сортировка и поиск».

[6] Sun Wu and Udi Manber “Fast Text Searching With Errors” Department of Computer Science University of Arizona Tucson, AZ 85721

[7] Г.Д.Карпова, Ю.К.Пирогова, Т.Ю.Кобзарева, Е.В.Микаэлян. Компьютерный синтаксический анализ: описание моделей и направлений разработок. // Итоги науки и техники (серия “Вычислительные науки”). Т.6. —М.: ВИНИТИ, 1991.

[8] «Нечеткое сопоставление строк». Опубликовано на http://itman.narod.ru/articles/infoscope/string_search.1-3.html

[9] А.Ахо, Дж.Ульман. Теория синтаксического анализа, перевода и компиляции. Т.1. Синтаксический анализ. М.: Мир, 1978

[10] Намиот, Д., & Сухомлин, В. (2013). О проектах лаборатории ОИТ. International Journal of Open Information Technologies, 1(5), 18-21

[11] Namiot, D., & Sneps-Sneppe, M. (2013, March). Wireless Networks Sensors and Social Streams. In Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on (pp. 413-418). IEEE. DOI: 10.1109/WAINA.2013.27

Search and Selection Geographic Addresses in Russian Texts.

Silchev A.V.

Abstract — This article presents the results of the qualifying work done by the author at the Laboratory of the OIT during the study program of additional education at the Faculty of Computational Mathematics and Cybernetics Lomonosov Moscow State University. This paper is devoted to the problem of search and selection of geographical addresses in unstructured texts in Russian. The author proposed and implemented a fuzzy search algorithm based on calculating the distance between the strings.

Keywords — geocoding, text processing.