РОССИЙСКАЯ АКАДЕМИЯ НАУК
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ имени М.В. Келдыша

Бондаренко А.В., Визильтер Ю.В., Клышинский Э.С., Силаев Н.Ж.,
Максимов В.Ю., Мусаева Т.Н.

Формальный метод нечеткого поиска персональной информации

Москва, 2009

Бондаренко А.В., Визильтер Ю.В., Клышинский Э.С., Силаев Н.Ж., Максимов В.Ю., Мусаева Т.Н.

Формальный метод нечеткого поиска персональной информации

Аннотация
Статья посвящена созданию математического формализма, описывающего процесс транскрипции имен собственных с иностранного языка на русский. Формализм позволяет перейти к программной реализации подобных систем. Для решения задачи нечеткого поиска именных групп в специализированных базах персональных данных предложен метод нечеткого сравнения и поиска слов с использованием их lk-представлений, обеспечивающий гарантированное нахождение соответствий, характеризуемых заданной степенью искажений.

Bondarenko A.V., Vizilter Yu.V., Klyshinsky E.S., Silaev N.J., Maximov V.Yu., Musaeva T.N.

The formal method of personal information fuzzy search

The paper devoted to the mathematical formalism creation describing process of proper names transcription from different foreign languages into Russian. The formalism allows to solve the task of program such systems realization. For fuzzy search of proper names in special bases of personal data the technique of fuzzy words comparison and search is proposed based on lk-representations. This technique guarantees matching of words with fixed degree of relative distortions.

Содержание

Содержание....3
1. ВВЕДЕНИЕ....4
1.1. Биологический нейрон....6
1.2. Формальный нейрон....8
1.3. Моделирование нетривиального (интеллектуального) адаптивного поведения....11
1.4. Бионический нейрон....15
2.РАЗРАБОТКА ПРОСТЫХ БИОНИЧЕСКИХ НЕЙРОННЫХ СЕТЕЙ....20
2.1.Логические элементы....20
2.2. Логический элемент or....20
2.3. Логический элемент and....20
2.4.Логический элемент XOR....21
2.5. Логический элемент not....23
2.6. Преобразователь сигналов из параллельных в последовательные....23
2.7.Десятичный счетчик....24
2.8 Сеть, считывающая состояние нейрона....24
2.9.Анализатор состояния нейрона....25
3.РАЗРАБОТКА МОДЕЛИ АДАПТИВНОГО ПОВЕДЕНИЯ В ЛАБИРИНТЕ....26
3.1 Модель окружающей среды (лабиринт)....26
3.4. Модель нейронной сети....27
ЗАКЛЮЧЕНИЕ....28
СПИСОК ЛИТЕРАТУРЫ....29

1. ВВЕДЕНИЕ

При организации поиска именных групп (ИГ) в базах персональных данных возникают характерные проблемы, связанные с наличием в запросах орфографических и фонетических ошибок, ошибок ввода информации, а также отсутствием единых стандартов транскрипции с иностранных языков. Вследствие указанных причин задача поиска в базах данных, содержащих ИГ, не может быть в полной мере решена только методами проверки на точное соответствие, в связи с чем становится актуальной задача разработки специальных методов и технологий нечеткого поиска.

В настоящее время методы нечеткого поиска применяются для решения целого ряда задач, таких как текстовый и мультимедийный поиск, поиск гомологических цепочек макромолекул в молекулярной биологии, поиск в графах и т. д. Однако универсальной методики их решения не существует, поскольку каждая проблема имеет собственную оригинальную специфику. В данной работе предлагается специализированный метод нечеткого поиска фамильно-именных групп, предназначенный для применения в автоматизированных информационных системах, содержащих персональные данные.

На практике часто встает вопрос о поиске среди иностранных имен, уже переданных на русский язык, и, как следствие, о корректной передаче имен собственных с одного языка на другой. При этом одним из основных требований является адекватная передача звучания имени собственного. Известно несколько методов передачи имен. Наиболее распространенным является *транслитерация*, при которой символу или набору символов из алфавита ставится в соответствие один или несколько символов другого алфавита, причем соответствие проводится скорее по графическому сходству символов. Подобный метод записи позволяет восстановить исходное написание слова, однако не позволяет воспроизвести его звучание лицу, не знакомому с исходным языком.

Более удобным является *словарный метод*, в котором соответствие слов входного языка словам выходного языка задается при помощи некоторого фиксированного словаря. Однако количество имен собственных растет с каждым годом, в связи с чем построить полный и всеобъемлющий словарь на практике не представляется возможным. Возможен вариант постоянного наращивания объемов словаря, однако он требует привлечения специалистов, знакомых с языком, на постоянной основе.

Наиболее удобным является метод *транскрипции*, в котором звучание слова в одном языке записывается средствами другого (в том числе специализированного) языка. Например, при фонетической транскрипции в качестве выходного может использоваться язык записи фонетики, позволяющий отразить все нюансы произношения слова. Однако подобный язык знаком лишь узкому кругу специалистов, в связи с чем, как правило, используют метод *практической транскрипции*, в котором звучание слова записывается алфавитом некоторого существующего языка.

С практической транскрипцией связаны следующие основные проблемы.

- Неполное соответствие фонемного состава двух языков. Так, например, в русском языке отсутствуют арабские горловые звуки, в связи с чем различные с точки зрения араба имена будут переданы на русский язык одинаковым образом.

- Вынужденное использование «запрещенных» в целевом языке морфем. Так, например, в русском языке появляются такие слова, как «парашют», использующие сочетание «шипящая согласная + ю или я».

- Зачастую транскрипция проводится не с исходного языка, а с его транслитерации при помощи латиницы. Подобная ситуация распространена для китайского, арабского, других языков, использующих иные графические системы (иероглифы или вязь). В этом случае при транслитерации теряется часть информации, а транскрипция усугубляет проблему.

- Историческая традиция. Традиционно Hamlet передается как Гамлет, хотя по современным правилам такое имя должно писаться как Хэмлет или Хемлет. Не помогает и фиксация записи в словаре. Так имя Walpol в разное время записывалось в словарях как Вальполь, Вальпол и, наконец, Уолпол.

- Отсутствие единых устоявшихся правил передачи. Как было показано выше, одно и то же имя может корректно быть передано различным образом. Результат зависит от того, какого набора правил передачи из многих существующих мы придерживаемся. Один и тот же учебник может рекомендовать несколько вариантов.

Заметим также, что в ряде случаев в самом исходном языке имя может читаться не в соответствии с правилом, а по сложившейся традиции. Особенно ярко подобная ситуация видна в английском языке, многие фамилии в котором читаются по традиции, принесенной нормандскими завоевателями или шотландскими подданными. Для решения подобной проблемы можно использовать в качестве вспомогательного словарный метод, когда слово сперва ищется в словаре, а при отсутствии соответствия – транскрибируется по правилам.

В данной предметной области отечественная лингвистика подготовила хорошую теоретическую базу, которая основывается на классических работах по практической транскрипции А.В. Суперанской [1], Р.С. Гиляревского и Б.А. Старостина [2], Л.Р. Зиндера [3], в которых дается подробное описание правил транскрипции для большого количества языков. Современный уровень информатизации требует создания программного обеспечения, осуществляющего автоматическую транскрипцию. Для этого необходимо предварительно создать адекватную математическую модель. В зарубежных публикациях на эту тему наиболее распространены подходы, основанные на скрытых Марковских моделях [4] и конечных автоматах [5]. Однако первый подход использует вероятностную передачу имен, что может не гарантировать корректность. Во втором подходе учет контекста передачи организован слишком сложным образом, что не позволяет легко и удобно вносить изменения в правила транскрипции. В связи с этим авторами был

разработан новый метод, основывающийся на декларативном подходе, отделяющем правила транскрипции от формального аппарата.

2. Транскрипция с использованием формально-литеральной записи

Определим *алфавит* $A = \{a_1,a_2,a_3,\ldots a_s\}$, $|A| = s$, $s>1$ как любое конечное множество символов.

Словом **w** в алфавите A длины n назовем конечную последовательность символов алфавита A. $\mathbf{w}=<w_1w_2w_3\ldots w_n>$, $w_i \in A$, $i=1,...,n$.

Параметр литеры определим как пару $\mathbf{p}=<\boldsymbol{\eta},\mathbf{v}>$, где $\boldsymbol{\eta}$ – имя параметра, а **v** – его значение, и будем считать, что два параметра *равны*, если совпадают их имена и значения.

Параметр литеры указывает некоторые характеристики, важные для транскрипции или позволяющие классифицировать символы алфавита по группам. Например: <”ряд“, ”передний“>,<”тип“, ”гласная“>, <”ударение“, ”безударная“>. Заметим, что имя и значение параметра литеры также являются словами, однако их символы могут принадлежать алфавиту, удобному для формирования правил транскрипции. Под литерой здесь понимается некоторое упорядоченное множество, описывающее не только символ, но и множество параметров, приписываемых данному символу по умолчанию при его озвучивании. Будем считать, что литера состоит из символа, однозначно идентифицирующей данную литеру, и набора параметров, либо изначально присущих данной литере, либо отражающих положение литеры в слове.

Введем три алфавита: A_{IN} – *входной алфавит*, содержащий символы входного языка, A_{OUT} – *выходной алфавит* содержащий символы выходного языка и служебный символ ‘#’, и A_{LIT} – *литеральный алфавит*, используемый для идентификации литер. Алфавиты могут быть определены на пересекающихся или непересекающихся множествах символов. В случае передачи иностранных имен средствами русского языка входной алфавит состоит из символов стандартной латиницы, выходной алфавит – из кириллических символов, принятых в русском языке, а литеральный алфавит – из символов входного и выходного алфавитов, а также служебных символов: BEG, END (символы начала и конца слова соответственно), ∅ (пустой символ).

Теперь *литера* может быть формально определена как пара $\mathbf{L}=<c,\mathbf{P}>$, где c – фиксированный символ из A_{LIT}, идентифицирующий данную литеру, а $\mathbf{P}=\{\mathbf{p}\}$ – конечный набор ее параметров.

Формальной литеральной записью (ФЛЗ) слова **w** будем называть его представление в виде упорядоченного множества литер $\mathbf{W}=<\mathbf{L}>$.

При этом будем считать, что различные написания одного и того же символа из A_{IN} (например, строчное и прописное или начальное, срединное, конечное и изолированное) идентифицируются одним и тем же символом из A_{LIT}, однако могут обладать (в зависимости от особенностей применения) различными значениями определенных параметров. Набор используемых

параметров определяется значимостью различения тех или иных написаний при транскрипции и особенностями языка.

Пример литеры: <’A’,{<“тип”, “гласн”>, <”написание”, ”прописн”>, <”ряд”, “задний”>}>, где ‘A’ – литеральный символ, идентифицирующий символ входного алфавита, а множество, заключенное в фигурные скобки – множество параметров данной литеры. Служебные символы, предназначенные для обеспечения процесса транскрипции, будут далее обозначаться несколькими символами и не будут заключаться в апострофы.

Определим также операторы сравнения литер.

Оператор *равенства* = производит сравнение, как литеральных символов, так и наборов параметров литер, то есть две литеры $\mathbf{L}_1$ и $\mathbf{L}_2$ *равны* ($\mathbf{L}_1 = \mathbf{L}_2$), если $c_1=c_2$ и $\mathbf{P}_2 \subseteq \mathbf{P}_1$.

Оператор *условного равенства* ≈ производит сравнение только наборов параметров литер и применяется только в том случае, если в одной из литер литеральный символ принимает значение ∅. Две литеры $\mathbf{L}_1$ и $\mathbf{L}_2$ *условно равны* ($\mathbf{L}_1 \approx \mathbf{L}_2$), если $c_1=\varnothing$ или $c_2=\varnothing$ и $\mathbf{L}_2 \subseteq \mathbf{L}_1$.

Собственно процесс транскрипции осуществляется три этапа:

1. преобразование написания слова на входном языке в формальную литеральную запись;
2. выделение слогов, расстановка переносов и ударений;
3. перевод ФЛЗ слова в запись на кириллице.

Рассмотрим каждый из них подробнее.

2.1. Преобразование написания слова на входном языке в ФЛЗ

Для преобразования слова **w** в его ФЛЗ **W**=<**L**> вводится множество *правил преобразования в ФЛЗ* $\Re_a=\{R_a\}$, где R_a: $a \rightarrow \mathbf{L}$ – собственно правило преобразования, $a \in A_{IN}$, **L** - литера.

Примеры правил преобразования в ФЛЗ:

’A’ → *<’A’,{<“тип”, “гласн”>, <”написание”, ”прописн”>, <”ряд”,“задн”>}>>*

’a’ → *<’A’,{<“тип”, “гласн”>, <”написание”, ”строчн”>, <”ряд”,“задн”>}>>*

’B’ → *<’B’,{<“тип”,“согласн”>,<”написание”,”прописн”>, <”звонкость”,“звонкая”>}>>*

’b’ → *<’B’,{<“тип”,“согласн”>,<”написание”,”строчн”>, <”звонкость”,“звонкая”>}>>*

Курсивом выделена часть правила, относящаяся к литере.

На данном этапе для всех символов входного слова последовательно находятся такие правила, что символ, входящий во входное слово **w**, совпадает с символом из найденного правила. ФЛЗ **W** слова **w** получается путем последовательной конкатенации литер, входящих в полученные правила. Символы, для которых не было найдено соответствия в правилах преобразования в ФЛЗ, считаются знаками препинания и передаются дальше без изменений с соответствующей пометкой. Следует заметить, что появление

подобных символов возможно в одной из двух ситуаций, когда входной символ либо не принадлежит входному алфавиту, либо принадлежит, но набор правил преобразования в ФЛЗ неполон. Кроме того, в начало и конец ФЛЗ добавляются специальные литералы, обозначающие начало и конец слова <BEG, {}> и <END, {}>. Перед началом группы знаков препинания ставится литера конца слова, а после группы – начала слова. Подобный подход позволяет корректно обрабатывать не только знаки препинания, но и символы из других алфавитов, которые не должны транскрибироваться в рамках данного языка.

Пусть i=1,...,n, j=1,...,m, где n – общее количество литер в выходном слове, m – общее количество символов во входном слове, а j не убывает при увеличении n. Тогда $\mathbf{w}=<w_j> \rightarrow \mathbf{W}=\bigcup_{i=1}^{n} \mathbf{L}_i$, причем

1) $\mathbf{L}_1$:=<BEG,{}>,

2) $\mathbf{L}_n$:=<END,{}>,

3) $\mathbf{L}_i$:=**L**, если для w_j $\exists$ ($R_a \in \Re_a$: $w_j \rightarrow \mathbf{L}$),

4) $\mathbf{L}_i$:=<w_j,{}>, если для w_j не $\exists$ ($R_a \in \Re_a$: $w_j \rightarrow \mathbf{L}$),

5) $\mathbf{L}_i$:=<BEG,{}>, если $\mathbf{L}_{i-1}$ получено по правилу 4), а $\mathbf{L}_{i+1}$ получено по правилу 3),

6) $\mathbf{L}_i$:=<END,{}>, если $\mathbf{L}_{i-1}$ получено по правилу 3), а $\mathbf{L}_{i+1}$ получено по правилу 4).

2.2. Выделение слогов и расстановка ударений

На данном этапе определяются закрытые/открытые слоги и ударные/безударные буквы. Любая литера, находящаяся в конце слога, приобретает дополнительный параметр «литера в слоге» со значением «открытая». Для остальных литер значение этого параметра – «закрытая».

Выделение слогов производится по следующему алгоритму. Для алфавита каждого языка может быть задан набор слогообразующих символов входного алфавита. В качестве части слога, присоединяемой к слогообразующемому символу, берется половина символов между двумя слогообразующими. При нечетном количестве символов, средний передается следующему слогу. Исключение делается для приставок, суффиксов и окончаний, в которых разделение на слоги фиксировано. Они присоединяются к остальной части слова как отдельный слог или несколько выделенных фиксированным образом слогов. Написание и деление на слоги таких приставок, суффиксов и окончаний задается отдельной базой правил.

Следует отметить, что известны различные алгоритмы выделения слогов с точки зрения фонетики. Описанный алгоритм был выбран в целях упрощения практической реализации.

Расстановка ударений и выделение слогов не являются обязательными процедурами. Их необходимо производить лишь для тех языков, в которых буквы читаются различным образом в зависимости от того, в какой позиции находится данная буква – в ударной, безударной или в конце слога.

При расстановке ударений в языках, где ударение является критичным, фиксируется номер слога и направление, в котором ведется счет слогов – от начала или от конца слова. Однако если в слове будет меньше слогов, чем было указано в правиле расстановки ударений, то ударение будет ставиться на последний слог при счете слогов с начала или на первый при счете с конца.

2.3. Преобразование ФЛЗ слова в кириллицу

На данном этапе на основе последовательности литер формируется кириллическое представление, отражающее фонетический облик слова.

Подстрокой ФЛЗ назовем подмножество последовательно идущих литер данной ФЛЗ. Обозначим через $\mathbf{W}_l^i$ подстроку ФЛЗ **W** длины *l*, начинающуюся с позиции *i*. В дальнейшем верхний индекс подстроки будет означать позицию, с которой начинается данная подстрока в ФЛЗ, а нижний – длину подстроки. Символом * обозначим произвольное значение позиции.

Под *правилом перевода* будем понимать R_t: $\mathbf{W}_{l1}^* \to \overline{W}_{l2}$, где $\mathbf{W}_{l1}^*$ - литеральная *подстрока-образец*, а $\overline{W}_{l2}$ - *подстрока-результат*, состоящая из символов выходного алфавита. Правило R_t:$\mathbf{W}_{l1}^* \to \overline{W}_{l2}$ *применимо* к подстроке $\mathbf{W}_{l1}^i$, если подстрока-образец $\mathbf{W}_{l1}^*$ сравнима с $\mathbf{W}_{l1}^i$. Под *сравнимостью* понимается нахождение равенства литер из $\mathbf{W}_{l1}^*$ и $\mathbf{W}_{l1}^i$ в одних и тех же позициях подстроки ФЛЗ $\mathbf{W}_{l1}^*$ и образца $\mathbf{W}_{l1}^i$. При этом литеры $\mathbf{L}_1$ и $\mathbf{L}_2$ *равны*, если $\mathbf{L}_1 = \mathbf{L}_2$ или $\mathbf{L}_1 \approx \mathbf{L}_2$.

Будем считать, что перевод подстроки $\mathbf{W}_{l1}^i$ возможен, если $\exists\, R_t \in \Re_t$:$\mathbf{W}_{l1}^* \to \overline{W}_{l2}$ применимое к $\mathbf{W}_{l1}^i$, где $\Re_t = \{R_t\}$ – *база правил перевода*. В этом случае можно говорить о *функции перевода* $\overline{W}_{l2} = f(W_{l1}^i)$, преобразующей ФЛЗ в выходную строку.

С учетом введенных определений и обозначений решение задачи перевода ФЛЗ в кириллическое представление может быть описано следующим образом.

Пусть имеется некоторая ФЛЗ $\mathbf{W}=<\mathbf{L}_1, \mathbf{L}_2, ..., \mathbf{L}_J>$ и набор правил перевода $\Re_t$. Тогда перевод ФЛЗ в кириллическое представление будет заключаться в нахождении и применении упорядоченного подмножества правил $\Re=<R_t : \mathbf{W}_l^* \to \overline{W}_{l2}>$, таких что:

1) $\mathbf{i}=<i_1, i_2, ..., i_n>$, где *n* – число правил в подмножестве $\Re$;
2) $\mathbf{l}=<l_1, l_2, ..., l_n>$;
3) $\sum_{j=1}^{n} l_j = q$ – общая длина выходного слова;
4) $i_1=1$;
5) $i_{k+1}=i_k+l_k$ для $k \in (1,n)$ и $i_n+l_n=q+1$;
6) $\forall k,m\ \exists\, R_t \in \Re$: $\mathbf{W}_l^* \to \overline{W}_{l2}$,

где **i** – упорядоченный список позиций, с которых применимы правила, а **l** – упорядоченный список длин подстрок. Назовем эти условия *условиями полноты покрытия*.

Результатом транскрипции является конкатенация результатов последовательного применения правил перевода

$\overline{W} = \bigcup_{i,l} f(W_l^i)$.

Перейдем к непосредственному описанию алгоритма преобразования ФЛЗ слова в кириллицу. Поскольку данный алгоритм носит рекурсивный характер, описание удобно вести в терминах рекурсивно вызываемых функций.

Определим функцию *findRules*(**W**, *i,* **r**), извлекающую из множества $\Re_t$ подмножество правил **r**, применимых к ФЛЗ **W** с позиции *i*. Параметры **W** и *i* передаются по значению, параметр **r** – по ссылке. Вычисление данной функции состоит в следующем.

Шаг 1. *Для каждого правила в множестве* $\Re_t$ *выполняется:*

Шаг 1.1 Каждое очередное правило рассматривается в качестве кандидата на применение к ФЛЗ **W** с позиции *i*. Если первые несколько последовательно идущих литер в строке-образце найденного правила имеют литеральный символ, равный ∅, то уменьшаем текущую позицию *i* на количество таких литер. Если текущая позиция меньше единицы, то считаем, что правило неприменимо и переходим к следующему правилу. В противном случае переходим к *Шагу 1.2*.

Шаг 1.2. Начиная с полученной текущей позиции, последовательно проверяем сравнимость литер строки с литерами правила. Если хотя бы одна литера строки несравнима с соответствующей литерой правила, то считаем, что правило неприменимо и переходим к следующему правилу (*Шаг 1.1*). Если сравнение всех литер прошло успешно, то данное правило добавляется в множество **r**, после чего производится переход к рассмотрению следующего правила (*Шаг 1.1*).

Шаг 2. На множестве **r** определяется максимальная длина подстроки-образца правила без учета литер, имеющих литеральный символ, равный ∅. После этого все правила, длина подстроки-образца которых меньше найденной максимальной, удаляются из множества **r**.

Определим также рекурсивную функцию *transcript*(**W**, **i**, **l**, $\Re$, $\mathbf{S_W}$), осуществляющую преобразование ФЛЗ в кириллическое написание и получающую в качестве входных параметров, передаваемых по значению, исходную ФЛЗ **W**, списки **i**, **l**, $\Re$, а в качестве передаваемого по ссылке выходного параметра – множество уникальных вариантов кириллической передачи входной ИГ $\mathbf{S_W}$. Алгоритм вычисления данной функции имеет следующий вид.

Шаг 1. Если последнее значение в списке **i** не превышает длину входной ФЛЗ – переход к *Шагу 2*. В противном случае считается, что обработка ФЛЗ завершена, из множества **i** удаляется последнее занесенное значение, а полученный результат транскрипции вычисляется как конкатенация результатов последовательного применения правил перевода и заносится в множество $\mathbf{S_W}$. При этом, если в множестве $\Re$ встречается $R_\varnothing$, то в качестве

результата его применения берется символ ‘#’, обозначающий невозможность адекватной передачи литеры ввиду отсутствия соответствующего правила.

Шаг 2. Текущая позиция *i* устанавливается равной последнему значению, помещенному в множество **i**. Множество правил, применимых к данной позиции, обнуляется. При помощи функции *findRules(***W**, *i,* **r***)* находится множество правил **r**, применимых к текущей позиции.

Шаг 3. Проверяются следующие условия.

Шаг 3.1. Если множество **r** пусто, *i* увеличивается на единицу, после чего в множество **i** добавляется *i*, в множество **l** добавляется 1, а в множество $\Re$ добавляется *пустое правило*, обозначающее отсутствие правила и обозначаемое $R_{\varnothing}$. Подобная ситуация является признаком того, что множество правил $\Re_t$ неполно.

Шаг 3.2. Для всех элементов **r** создаются: список **i'**= **i**, список **l'**=**l** и список $\Re'=\Re$. Далее списки **i'**, **l'** и $\Re'$ пополняются следующим образом. В список **l'** добавляется длина *l* строки-образца найденного правила, за вычетом количества литер с литеральным символом $\varnothing$; в список $\Re'$ добавляется найденное правило; в список **i'** добавляется значение *i*, увеличенное на *l*. После этого производится рекурсивный вызов функции *transcript*(**W**, **i'**, **l'**, $\Re'$, $\mathbf{S_W}$).

Выходом алгоритма является множество $\mathbf{S_W}$. Кроме того, сформированные в результате списки **i**, **l** и $\Re$ могут служить как для проверки работы алгоритма, так и для уточнения базы правил перевода $\Re_t$. Таким образом, указанный алгоритм обеспечивает не только нахождение всех вариантов транскрипции слова и выполнение условий полноты покрытия, но и проверку необходимости пополнения базы правил перевода.

С учетом введенных функций соответствующий алгоритм преобразования ФЛЗ слова в кириллицу имеет следующий вид. Пусть входом алгоритма является ФЛЗ слова **W**. Тогда вычисление его транскрипции производится в два шага.

Шаг 1. Обнулить списки **i**, **l**, $\Re$ и множество $\mathbf{S_W}$, после чего занести в список **i** значение 1.

Шаг 2. Вызвать функцию *transcript*(**W**, **i**, **l**, $\Re$, $\mathbf{S_W}$).

Результатом работы алгоритма будет множество $\mathbf{S_W}$.

Пример транскрипции с французского языка именной группы «Mireille Mathieu». Пусть преобразование в ФЛЗ уже осуществлено, и каждому символу сопоставлена литера. Тогда процесс применения правил перевода будет иметь следующий вид (здесь последовательность строк таблицы соответствует последовательности применения правил).

Позиция во входной строке	Применяемое правило	Результат
Mireille Mathieu	M=M	М
M**i**reille Mathieu	I=И	Ми
Mi**r**eille Mathieu	R=Р	Мир
Mir**eill**e Mathieu	EILL=ЕЙ	Мирей

Mireill**e** Mathieu	E на конце слова = ''	Мирей
Mireille_Mathieu	' '=' '	Мирей_
Mireille **M**athieu	M=M	Мирей М
Mireille M**a**thieu	A=A	Мирей Ма
Mireille Ma**th**ieu	TH=T	Мирей Мат
Mireille Math**ieu**	согласная + IEU=ЬЕ	Мирей Матье

Заметим, что в правилах (E на конце слова = '') и (согласная + IEU=ЬЕ) играют роль параметры, приписанные литерам, а также специальные символы.

На основе предложенного метода была разработана программная технология транскрипции имен собственных. Тестирование показало, что реализованная на базе этой технологии система перевода допускает менее 1% ошибок для таких регулярных языков, как немецкий, румынский и японский. Для английского языка количество ошибок приближается к 2%, но может быть сокращено за счет расширения словаря исключений.

3. Общая проблематика нечеткого поиска

Под *нечетким поиском* понимается поиск по ключевым словам с учётом возможных произвольных ошибок в написании ключевого слова или, напротив, ошибок написания слова в целевом запросе. Достаточно полный и подробный обзор современных методов поиска можно найти, например, в работах [6,7]. Основными аспектами организации текстового поиска являются способ построения *поискового индекса*, выбор *поисковой метрики* и собственно *алгоритмы нечеткого поиска*.

Структуры данных информационного поиска, как правило, относятся к одной из двух основных групп: *векторные и кластерные* модели, либо модели на основе *ключевых слов*. Основная идея *векторных методов* состоит в том, что считаются заданными m поисковых слов, и каждый поисковый объект отображается в вектор, называемый *профилем*, причем величина k-го элемента профиля зависит от частоты вхождения в документ k-го поискового слова, например слова, строки или подстроки длиной n символов (n-граммы). Поисковое выражение также рассматривается, как документ с соответствующим профилем, и ключевым моментом организации поиска является выбор *функции корреляции профилей*. В выборку попадают документы, для которых корреляционное значение превышает пороговое. Недостатком векторных методов является необходимость считывания профилей всех документов. Устранить этот недостаток можно, разбив все документы на группы (*кластеры*) и определив в каждом кластере характерного представителя (*центроид кластера*), что позволяет сначала сравнивать поисковый запрос лишь с центроидами кластеров. В случае релевантности центроида запросу поиск продолжается далее внутри кластера, причем процесс разбиения выборки на кластеры может быть иерархическим (многоуровневым).

При *поиске по ключевым словам*, как правило, производят выборку всех документов, содержащих хотя бы одно ключевое слово, а затем ранжируют результаты поиска по степени соответствия (релевантности). В основе поиска

по ключевым словам лежит использование специализированных *индексных словарей* двух основных типов. *Инвертированный файл* (ИФ) [8] – множество пар <ключевое слово, адрес вхождения ключевого слова в документ>. *Сигнатурные файлы* (СФ) [9] содержат *сигнатуры* данных, представляющие собой их упрощенные профили, в которых каждый элемент кодируется одним битом. Сжатые ИФ [10,11] существенно превосходят СФ по производительности для коротких запросов, но проигрывают им на длинных и очень длинных запросах.

Ключевым элементом организации нечёткого поиска является выбор *меры сходства* слов или обратной функции – *функции расстояния* между словами, часто называемой *метрикой* даже в тех случаях, когда она не является метрикой в строгом математическом смысле. Наибольшее распространение здесь получили *трансформационные метрики,* в области текстового поиска называемые также *расстояниями редактирования*. Наиболее известны поисковые метрики следующих типов.

Расстояние Хемминга [12] между словами одинаковой длины определяется как число позиций, в которых символы не совпадают. *Расстояние Левенштейна* [13], позволяющее сравнивать слова различной длины, равно минимальному числу операций преобразования одного слова в другое, причем допустимы только операции вставки, удаления и замены, которым также присвоена единичная стоимость. При определении *расстояния Дамерау-Левенштейна* [14] перестановка символов принимается за единую операцию редактирования с весом 1.

Известны меры сходства *Джаро* [15] и *Джаро-Уинклера* [16], представляющие собой нормированные коэффициенты, специально разработанные для сравнения коротких строк, например, компонентов ИГ. Также получили распространение метрики, основанные на подсчёте количества общих подстрок равной длины (*n-грамм*).

Большинство практических алгоритмов поиска ключевого слова в словаре основаны на модификациях одного из следующих известных методов: последовательный поиск (полный перебор всех слов словаря), метод расширения выборки (query extension), метод n-грамм, поиск с использованием хеширования, методы на основе неравенстве треугольника (триангуляционные деревья).

Последовательный поиск предполагает последовательный перебор слов из словаря и сравнение каждого из них с запросом в соответствии с принятой метрикой. Данный метод применяется, например, в системах Agrep [17] и Glimpse [18]. *Метод расширения выборки* предполагает построение множества всевозможных «ошибочных» слов, например, получающихся из исходного в результате одной операции редактирования Левенштейна, после чего построенные поисковые запросы ищутся в словаре на точное совпадение. Метод широко используется в программах проверки орфографии, например Ispell, и часто связан с применением морфологического анализа, в частности, стемминга [19]. *Метод n-грамм* [20] основан на том, что для поиска слов

строится инвертированный файл, в котором роль документов играют слова, а роль слов – подстроки длины *n*, называемые n-граммами. *Поиск с использованием хеширования* состоит в подборе отображения (хеш-функции) слова, например, во множество чисел или строк, сохраняющего основные характеристики исходного слова и устойчивого к наиболее распространённым ошибкам. Известным примером является хеш-функция Soundex [7], встроенная в коммерческие СУБД Sybase, MS SQL Server, Oracle. *Триангуляционные деревья* [21] позволяют индексировать множества произвольной структуры, при условии, что на них задана метрика (не обязательно евклидова). В основу построения триангуляционных деревьев положена идея расположения близких в смысле заданной метрики объектов в одинаковых поддеревьях. При поиске в текстовых массивах данный метод менее эффективен, чем при поиске в базах изображений или больших документов.

Перейдем теперь к рассмотрению конкретной задачи нечеткого поиска фамильно-именных групп и описанию разработанного для ее решения специализированного метода.

4. Постановка задачи нечеткого поиска

Пусть имеется *алфавит* $A = \{a_1,a_2,a_3,\ldots a_s\}$, $|A| = s$, $s>1$ – любое конечное множество символов. *Слово* w в алфавите A длины *n* определяется как конечная последовательность символов алфавита A. $w=<w_1w_2w_3\ldots w_n>$, $w_i \in A$, $i=1,...,n$. *Словарем именных компонент* U будем называть заданное конечное множество слов, определённых на алфавите A: $U = \{u_1,u_2,u_3,\ldots u_m\}$, $|U|=m$, $m \geq 1$.

С учетом этого, рассматриваемая в работе задача может быть формально поставлена следующим образом.

Дано: слово-образец w, словарь U, целое $k > 0$ и функция расстояния *d*, являющаяся метрикой.

Требуется: найти все *j* такие, что $d(w, u_j) \leq k$, $u_j \in U$, где *d* – расстояние Левенштейна, определяемое как минимальное количество операций замены, вставки и удаления символов, необходимых для преобразования одного слова в другое. При этом необходимо построить алгоритм поиска, который был бы вычислительно более эффективен, чем метод полного перебора элементов словаря.

5. Метод нечеткого сравнения и поиска слов с использованием lk-представлений

Рассмотрим сначала известный метод n-грамм, применяемый при решении подобных задач [16,20]. В рамках данного метода подстрока длины *n* слова $\mathbf{w}=<w_1w_2w_3\ldots w_m>$ называется *n-граммой* слова **w**. Обозначим множество всех n-грамм слова w как $N_n(\mathbf{w})$. Идея использования n-грамм в задачах нечеткого сравнения слов заключается в следующем. Если слово **v** похоже на слово **u**, то у них должны быть общие подстроки. Следовательно, если расстояние Левенштейна между **v** и **u** не превышает *k*: $d(\mathbf{v}, \mathbf{u}) \leq k$, причем

$||N_n(u)||>k$ и $||N_n(v)||>k$, то $||N_n(u) \cap N_n(v)|| \geq 1$, где норма множества определяется как количество его элементов.

Принципиальным недостатком системы на основе классических n-грамм является то, что они чувствительны к выбору параметров (длина n-граммы, выбор порога фильтрации k). Кроме того, критерии отождествления, вычисляемые на основе n-грамм, зависят не только от расстояния Левенштейна, но также и от расположения несовпадающих символов в сравниваемых словах, что недопустимо в задачах поиска ИГ. Для преодоления указанных недостатков предлагается перейти от n-грамм к более полным представлениям слов. Введем следующие обозначения:

- $\mathbf{w} \oplus \mathbf{v}$ – результат конкатенации слов **w** и **v**;
- $\boldsymbol{\omega}(\mathbf{w},t,l)$ – оператор *выделения подстроки* длины l из слова **w**, начинающейся с символа, находящегося в позиции t.
- $\boldsymbol{\varepsilon}(\mathbf{w},t)=\boldsymbol{\omega}(\mathbf{w},1,t-1) \oplus \boldsymbol{\omega}(\mathbf{w},t+1,l-t)$ – оператор *удаления* из слова **w** символа, находящегося в позиции t;
- $\boldsymbol{\delta}(\mathbf{w},t,a)=\boldsymbol{\omega}(\mathbf{w},1,t-1) \oplus a \oplus \boldsymbol{\omega}(\mathbf{w},t+1,l-t)$ – оператор *вставки* в слово **w** символа a в позиции t;
- $\mu(\mathbf{w})$ – длина (число символов) слова (**w**).

Введем также ряд новых понятий и определений.

Пусть *нормировочным алфавитом* A_{NR} для алфавита A называется любой алфавит, такой что $A_{NR} \cap A = \varnothing$.

l-нормализованным словом для слова **w** называется слово $\mathbf{w}_l^* = \boldsymbol{\omega}(\mathbf{w} \oplus \mathbf{a},1,l)$, где **a** – *нормировочное слово* длины $||A_{NR}||$, содержащее все символы алфавита A_{NR} в лексикографическом порядке, причем $||A_{NR}|| \geq l$.

lk-граммой слова **w** для фиксированных целых неотрицательных значений l и k называется последовательность символов, определяемая по индукции следующим образом:

- *l0-граммой* слова **w** длины является *l-нормализованное* слово
  $\mathbf{w}_l^0 = \mathbf{w}_l^*$;
- *l1-граммой* слова **w** является слово
  $\mathbf{w}_l^1(i) = \boldsymbol{\varepsilon}(\mathbf{w}_l^0,i)$: $i \in \{1,..,l\}$;
- *lk-граммой* слова **w** для любого $l \geq k > 1$ является слово
  $\mathbf{w}_l^k(i) = \boldsymbol{\varepsilon}(\mathbf{w}_l^{k-1}(j),i)$: $i \in \{1,..,l-k-1\}$, $j \in \{1,..,l-k\}$.

При этом параметры l и k имеют смысл *длины нормализованного слова* и *допустимой степени искажения* соответственно. Для фиксированных l и k множество всех lk-грамм слова **w** называется *lk-представлением слова* и обозначается $N_{lk}(\mathbf{w})$, а множество всех lk-грамм слов, входящих в состав словаря U, называется *lk-представлением словаря* и обозначается $N_{lk}(U)$:

$$N_{lk}(U) = \bigcup_{i=1}^{m_U} \{N_{lk}(\mathbf{u}_i)\},$$

где $\mathbf{u}_i \in U$; $m_U = ||U||$.

Функцией lk-принадлежности подстроки **v** слову **w** называется функция

$$\chi_{lk}(\mathbf{w},\mathbf{v}) = \begin{cases} 1, \text{ если } \mathbf{v}\in N_{lk}(\mathbf{w}); \\ 0 - \text{в противном случае.} \end{cases}$$

lk-спектром подстроки **v** по словарю U называется вектор

$$\mathbf{S}_{lk}(\mathbf{w},U)=<\chi_{lk}(\mathbf{w},\mathbf{u}_1),\ldots,\chi_{lk}(\mathbf{w},\mathbf{u}_m)>: \mathbf{u}_i\in N_{lk}(U),\ i=1,..,m,$$

где $m=||N_{lk}(U)||$.

Показателем lk-корреляции слов **w** и **v** по словарю U называется скалярное произведение их спектров:

$$\rho_{lk}(\mathbf{w},\mathbf{v}) = (\mathbf{S}_{lk}(\mathbf{w},U),\mathbf{S}_{lk}(\mathbf{v},U)) = \sum_{i=1}^{m_{NU}} \chi_{lk}(\mathbf{w},\mathbf{u}_i)\times\chi_{lk}(\mathbf{v},\mathbf{u}_i),$$

где $m_{NU}=||N_{lk}(U)||$. Скалярное произведение здесь понимается стандартным образом как сумма покомпонентных произведений векторов. При этом для произвольных **w** и $\mathbf{u}\in U$ выполняется следующее соотношение

$$0 \le \rho_{lk}(\mathbf{w},\mathbf{u}) \le \rho_{max}(l,k) = \rho_{lk}(\mathbf{u},\mathbf{u}).$$

Это позволяет определить соответствующую нормированную характеристику близости двух слов в пространстве lk-представлений.

Нормированным коэффициентом lk-корреляции слов **w** и **v** по словарю U будем называть коэффициент

$$K_{lk}(\mathbf{w},\mathbf{v})=\rho_{lk}(\mathbf{w},\mathbf{v})/\rho_{max}(l,k),$$

относительно свойств которого доказано следующее утверждение.

Утверждение. Пусть даны два слова **w** и **u**, рассматривается словарь $U=\{\mathbf{u}\}$, $l>\max(\mu(\mathbf{w}), \mu(\mathbf{u}))$ и $l>k\ge 0$. Тогда нормированный коэффициент lk-корреляции $K_{lk}(\mathbf{w},\mathbf{u})$ обладает следующими свойствами:

1) $\forall\mathbf{w}: 0\le K_{lk}(\mathbf{w},\mathbf{u})\le 1$;
2) $K_{lk}(\mathbf{u},\mathbf{u})=1$;
3) $\forall\mathbf{w}: K_{lk}(\mathbf{w},\mathbf{u})=0 \Rightarrow d(\mathbf{w},\mathbf{u})>k$,

где d – расстояние Левенштейна.

Доказательство. Положения 1) и 2) следуют из определения.
Докажем свойство 3), эквивалентное тому, что

$$d(\mathbf{w},\mathbf{u})\le k \Rightarrow \rho_{lk}(\mathbf{w},\mathbf{u})>0.$$

Перейдем к l-нормированным строкам $\mathbf{w}_l^*$ и $\mathbf{u}_l^*$. Если $l>\max(\mu(\mathbf{w}),\mu(\mathbf{v}))$, то легко показать, что каждое из перечисленных ниже *элементарных преобразований слова* всегда может быть представлено эквивалентной комбинацией (*суперпозицией*) пары операций удаления и вставки символов:

- *удаление символа* в позиции t: $\delta(\varepsilon(\mathbf{w}_l^*,t),l,\#)$, где $\#\in A_{NR}$;
- *вставка символа* a в позиции t: $\delta(\varepsilon(\mathbf{w}_l^*,l),t,\#)$, где $\#\in A_{NR}$;
- *замена символа* в позиции t на символ b: $\delta(\varepsilon(\mathbf{w}_l^*,t),t,b)$;
- *перестановка пары соседних символов* a и b: $\delta(\varepsilon(\mathbf{w}_l^*,t),t+1,a)$;
- *тождественное преобразование* также может быть представлено при помощи операций *фиктивного удаления* и *фиктивной вставки* нормировочных символов конце слова: $\delta(\varepsilon(\mathbf{w}_l^*,l),l,\#)$, где $\#\in A_{NR}$.

Пусть последовательность элементарных преобразований, переводящая слово **w** в слово **u**, содержит не более k элементарных преобразований перечисленных типов. Представим каждое из этих преобразований парой операций вставки и удаления, как показано выше. Отсортируем преобразования таким образом, чтобы сначала осуществлялись все вставки, а затем – все удаления. Получаем последовательное объединение двух цепочек однотипных преобразований (вставок и удалений), каждая из которых содержит не более k преобразований. Дополняем обе цепочки фиктивными преобразованиями таким образом, чтобы в общей последовательности оказалось в точности k удалений и k вставок. Поскольку описанные эквивалентные преобразования цепочек элементарных преобразований слова можно сделать всегда, то из условия $d(\mathbf{w}_l^*,\mathbf{u}_l^*)\le k$ всегда следует существование последовательности из k вставок $\{\boldsymbol{\varepsilon}_i: i=1,\ldots,k\}$ и k удалений $\{\boldsymbol{\delta}_i: i=1,\ldots,k\}$, такой что:

$$\mathbf{v}_1=\boldsymbol{\varepsilon}_1(\mathbf{w}_l^*),\ \mathbf{v}_2=\boldsymbol{\varepsilon}_2(\mathbf{v}_1),\ \ldots,\ \mathbf{v}_k=\boldsymbol{\varepsilon}_k(\mathbf{v}_{k-1}),$$

$$\mathbf{v}_{k+1}=\boldsymbol{\delta}_1(\mathbf{v}_k),\ldots,\mathbf{v}_{2k-1}=\boldsymbol{\delta}_{k-1}(\mathbf{v}_{2k-2}),\ \mathbf{u}_l^*=\boldsymbol{\delta}_k(\mathbf{v}_{2k-1}).$$

Отсюда $\mathbf{v}_k\in \mathrm{N}_{lk}(\mathbf{w})$ и $\mathbf{v}_k\in \mathrm{N}_{lk}(\mathbf{u})$. Таким образом,

$$\mathrm{N}_{lk}(\mathbf{w})\cap \mathrm{N}_{lk}(\mathbf{u})\ne\varnothing \Rightarrow \rho_{lk}(\mathbf{w},\mathbf{u})>0,$$

что и требовалось доказать.

Данное утверждение позволяет сформулировать численный *критерий lk-отождествимости* слов **w** и $\mathbf{u}\in$U:

$$\mathrm{K}_{lk}(\mathbf{w},\mathbf{u}) \ge \mathrm{K}_{\min}(l,k,x),$$

где $\mathrm{K}_{\min}(l,k,x)$ – минимально допустимое значение нормированного коэффициента lk-корреляции, рассматриваемое как функция от максимально допустимого расстояния $x=d(\mathbf{w},\mathbf{u})$ при фиксированных l и k. При этом важно, что формула для вычисления $\mathrm{K}_{\min}(l,k,x)$ может быть получена в явном виде.

Для описанной выше рекурсивной процедуры построения lk-грамм справедлива следующая оценка максимального показателя lk-корреляции:

$$\rho_{max}(l,k)= k!\mathrm{C}_l^k,$$

где $\mathrm{C}_l^k=\dfrac{l!}{(l-k)!k!}$ – биномиальный коэффициент. Соответственно

$$\rho_{lk}(\mathbf{w},\mathbf{u}) = \rho(l,k,x) = \begin{cases} k!\mathrm{C}_{l-x}^{k-x}, \text{ если } x\le k; \\ 0 \text{ – в противном случае.} \end{cases}$$

Таким образом,

$$\mathrm{K}_{lk}(\mathbf{w},\mathbf{u}) = \rho_{lk}(\mathbf{w},\mathbf{u})/\rho_{max}(l,k) = \frac{k!C_{l-x}^{k-x}}{k!C_l^k}=\frac{C_{l-x}^{k-x}}{C_l^k},$$

откуда

$$\mathrm{K}_{\min}(l,k,x) = \frac{(l-x)!k!}{l!(k-x)!}.$$

Полученную оценку порогового значения lk-корреляции можно проиллюстрировать следующими простыми примерами.

Пусть $l=7$, $k=3$, $x=1$. Тогда

$$\mathrm{K}_{\min}(l,k,x) = \frac{(l-x)!k!}{l!(k-x)!}= \frac{(7-1)!3!}{7!(3-1)!} = 3/7.$$

Пусть l=7, k=2, x=1. Тогда

$$K_{min}(l,k,x) = \frac{(l-x)!k!}{l!(k-x)!} = \frac{(7-1)!2!}{7!(2-1)!} = 2/7.$$

Таким образом, как следует из всего вышеописанного, задача сравнения слов может быть решена с использованием описанного математического аппарата lk-представлений.

Для проведения нечеткого поиска в словаре требуется предварительная подготовка данных, которая заключается в построении lk-представления словаря (*индексного файла*). При этом добавление информации о каждом новом словарном слове требует построения lk-представления данного слова и занесения соответствующих lk-грамм в lk-представление словаря. Для каждой добавленной lk-граммы также хранятся ссылки на все слова, в lk-представлениях которых она встречалась.

Алгоритм нечеткого поиска тестового слова в подготовленных данных состоит из трех основных этапов:

1. Вычисление lk-представления тестового слова.
2. Голосование lk-грамм тестового слова в пользу слов-гипотез из словаря.
3. Подсчет числа голосов и сравнение их с пороговым значением.

Вычисление $\rho_{lk}(\mathbf{w},\mathbf{u}_j)$, $\mathbf{u}_j \in U$ реализуется на втором этапе алгоритма как процедура голосования всех lk-грамм $\mathbf{w}_l^k(i)$ слова $\mathbf{w}$ в пользу слов $\mathbf{u}_j$ с весами, равными $\chi_{lk}(\mathbf{u}_j,\mathbf{w}_l^k(i))$. На третьем этапе полученная сумарная оценка $\rho_{lk}(\mathbf{w},\mathbf{u}_j)$ для всех $\mathbf{u}_j \in U$ сравнивается с порогом $\rho_{min}(l,k,x)$, соответствующим $K_{min}(l,k,x)$.

Оценка вычислительной сложности данного алгоритма имеет порядок $O(||N_{lk}(\mathbf{w})||*\ln(||N_{lk}(U)||))$, что существенно эффективнее алгоритмов прямого поиска в словаре.

6. Метод нечеткого сравнения и поиска многокомпонентных именных групп с использованием lk-представлений

Рассмотрим теперь задачу нечеткого отождествления ИГ, состоящих из нескольких именных компонент (ИК).

Пусть даны две многокомпонентные ИГ (МИГ): *словарная МИГ* $\mathbf{U}=\langle\mathbf{u}_j\rangle=\langle\mathbf{u}_1,\ldots,\mathbf{u}_m\rangle$ и *тестовая МИГ* (*МИГ-запрос*) $\mathbf{W}=\langle\mathbf{w}_i\rangle=\langle\mathbf{w}_1,\ldots,\mathbf{w}_n\rangle$.

Показателем lk-корреляции ИК $\mathbf{w}$ по МИГ $\mathbf{U}$ назовем

$\rho_{lk}(\mathbf{w},\mathbf{U}) = max_j\ \rho_{lk}(\mathbf{w},\mathbf{u}_j)$.

Пусть ИК $\mathbf{w}$ считается *успешно отождествленной* на МИГ $\mathbf{U}$ с точностью x, если

$\rho_{lk}(\mathbf{w},\mathbf{U}) \geq \rho(l,k,x)$.

Показателем lkx-сходства $q_{lkx}(\mathbf{W},\mathbf{U})$ будем называть количество ИК МИГ $\mathbf{W}$, успешно отождествленных с точностью x на МИГ $\mathbf{U}$. МИГ $\mathbf{W}$ будем считать *успешно отождествленной* с МИГ $\mathbf{U}$ с точностью x, если

$q_{lkx}(\mathbf{W},\mathbf{U}) \geq q_{min}$, $q_{min} \leq n$.

Данное выражение описывает *критерий lkx-отождествимости многокомпонентных ИГ*.

Если бы при словарном поиске выполнялось непосредственное попарное сравнение ИК двух МИГ, то вычисление показателя сходства $q_{lkx}(\mathbf{W},\mathbf{U})$ не составляло бы никакой проблемы. Для этого потребовалось бы помимо словаря ИК создавать еще один словарь – многокомпонентных ИГ, размер которого чрезвычайно велик, так же как и время поиска в нем. Однако требования достижения большей вычислительной эффективности приводят к необходимости организации одновременного голосования всех компонент МИГ **W** в пользу всех компонент МИГ **U**, что делает нетривиальной задачу вычисления показателя сходства $q_{lkx}(\mathbf{W},\mathbf{U})$. Сформулируем ее математически.

Суммарным показателем lk-корреляции МИГ **W** по МИГ **U** назовем

$\rho_{lk}(\mathbf{W},\mathbf{U}) = \sum_i \sum_j \rho_{lk}(\mathbf{w}_i,\mathbf{u}_j)$.

Данная статистика вычисляется для МИГ в целом на основе lk-представлений отдельных ИК. При этом

$\forall \mathbf{w},\mathbf{u}: \rho_{lk}(\mathbf{w},\mathbf{u}) \in \mathbf{r}$,

где $\mathbf{r}=<\rho(l,k,0),...,\rho(l,k,l\text{-}1)>$ – заранее известный набор значений, которые могут принимать показатели lk-корреляции отдельной ИК при фиксированных l и k. Значит, вне зависимости от числа отдельных компонент и того, как они отождествляются, суммарный показатель lk-корреляции всегда может быть представлен в следующем виде:

$\rho_\Sigma = \sum_{t=0,..,l\text{-}1} q_t \times \rho_t$,

где $\rho_\Sigma=\rho_{lk}(\mathbf{W},\mathbf{U})$; $\rho_t=\rho(l,k,t)\in\mathbf{r}$; $q_t\geq 0$ – количество ИК МИГ **W**, отождествленных на **U** с точностью t. Отсюда искомый показатель lkx-сходства может быть определен по формуле

$q_{lkx}(\mathbf{W},\mathbf{U}) = \sum_{t=0,..,x} q_t$.

Таким образом, для того, чтобы проверить критерий lkx-отождествимости, необходимо для вычисленного по запросу суммарного показателя сходства ρ_Σ при заранее известном наборе констант **r** определить набор весов $\mathbf{q}=<q_x>$.

Легко заметить, что эта задача эквивалентна задаче представления числа ρ_Σ в системе счисления с основаниями из **r**. К сожалению, такое разложение не будет единственным, поскольку система счисления **r** не является строго позиционной. Иными словами, сформулированная задача является некорректной. Однако, как известно из работ А.Н. Тихонова (например, [22]), некорректные задачи становятся корректными, если удается их *регуляризировать*, то есть дополнить некоторыми дополнительными условиями или критериями. Семантика рассматриваемой задачи позволяет естественным образом определить регуляризирующее условие, связанное с тем, что наличие ненулевых коэффициентов при малых t означает обнаружение в поисковом запросе неискажённых или слабо искажённых именных компонент, а вхождение ненулевых коэффициентов при больших t должно трактоваться как наличие большого количества отличающихся или сильно искажённых компонент. При этом информация о сильном совпадении небольшого количества ИК семантически более значима, чем информация о слабом совпадении большого количества ИК, так как на практике вероятность ошибок редактирования не столь велика. Это позволяет

определить однозначную процедуру решения данной задачи, основанную на том, что в последовательности **r** каждый следующий элемент всегда меньше предыдущего ненулевого

$\forall t < l\text{-}1\colon (\rho_t > 0) \Rightarrow (\rho_{t+1} < \rho_t),\ (\rho_t = 0) \Rightarrow (\rho_{t+1} = 0),$

и приоритет при поиске допустимого разложения отдаётся коэффициентам при малых t. Соответствующая рекурсивная процедура определения коэффициентов разложения имеет традиционный вид:

$p_0 = \rho_\Sigma,\ q_0 = p_0 \text{ div } \rho_0;$

если $\rho_t > 0$: $p_t = p_{t\text{-}1} \bmod \rho_{t\text{-}1},\ q_t = p_t \text{ div } \rho_t,\ t = 1,\ldots,l\text{-}1$

где (a div b) –целочисленное деление a на b, (a mod b) – взятие остатка a по модулю модулю b. После вычисления коэффициентов разложения критерий *lkx*-отождествимости МИГ вычисляется по приведенным выше формулам.

Таким образом, разработанный аппарат lk-представлений позволяет решать задачу сравнения и эффективного поиска в словаре не только отдельных именных компонент, но и многокомпонентных именных групп.

7. Экспериментальные исследования и выбор параметров

Как было показано выше, для гарантированного нахождения слова в словаре, содержащем слова различной длины, необходимо, чтобы используемый параметр нормализации l на единицу превышал дину самого длинного слова в словаре. С другой стороны, на практике для снижения объёма индексного файла целесообразно ограничить значение l минимально возможным числом, определяющим т. н. значимую часть слова. При этом с уменьшением величины l объём индексного файла снижается, однако возрастает число ошибок первого рода.

Для оценки минимально необходимого значения l на исследуемой базе ИГ была проведена серия вычислительных экспериментов, в которых исследовались: распределение записей по количеству именных компонент в МИГ, распределение длин именных компонент, зависимость количества ошибок первого рода при различных l в условиях поиска с точностью до различных значений k. Экспериментально установлено, что значение l=7 является точкой насыщения в процессе уменьшения ошибок первого рода. В связи с этим значение l=7 было принято в качестве базового параметра при построении практических алгоритмов поиска ИГ.

На рис.1 представлена эмпирическая функция распределения длин ИГ в экспериментальных базах. Эмпирические зависимости количества ошибок первого рода (ложного отождествления слова с образцом) от выбранного значения предельной точности поиска представлены на Рис.2-4.

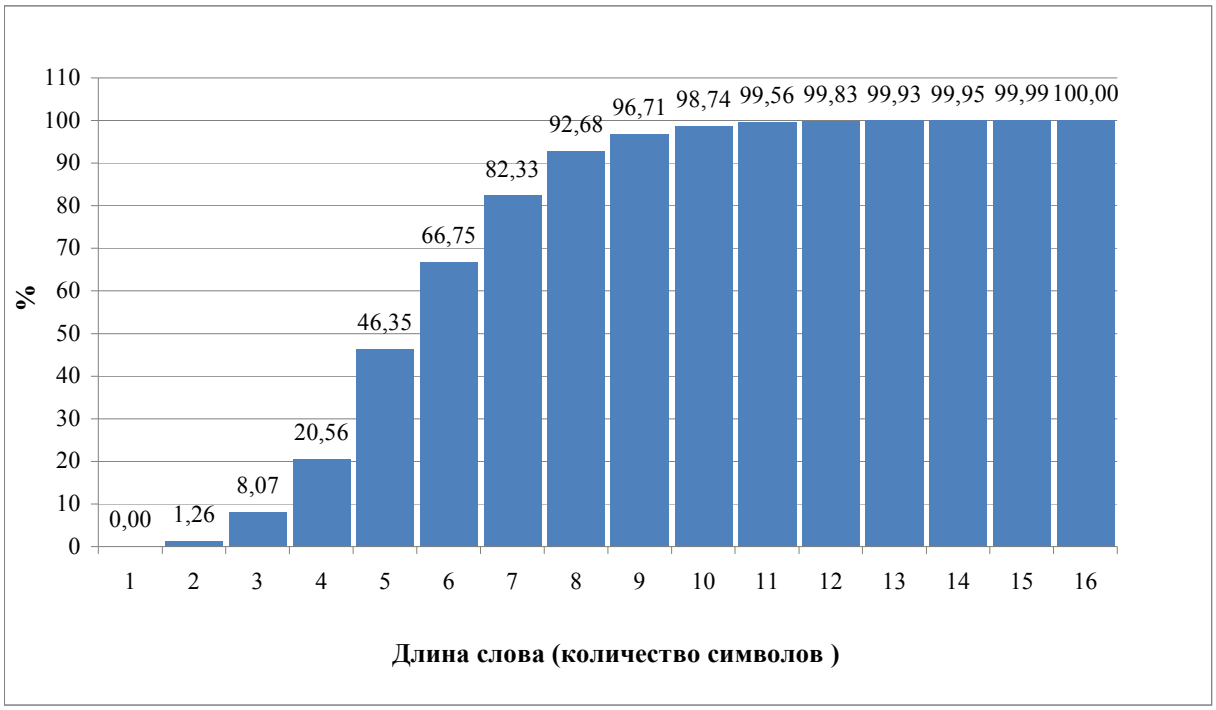


Рис.1. Эмпирическая функция распределения длин слов для ИГ.

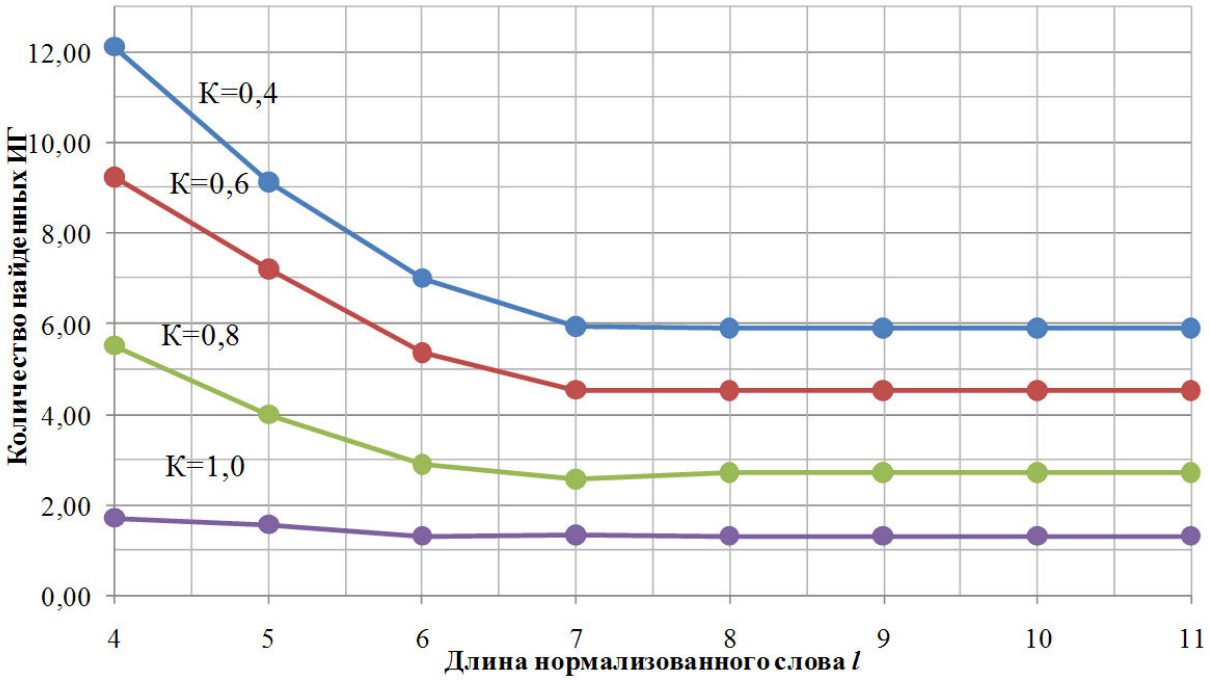


Рис.2. Зависимость количества ошибок первого рода (ложного отождествления слова с образцом) от длины нормализованного слова *l* и порогового значения коэффициента lk-корреляции K при *k*=1.

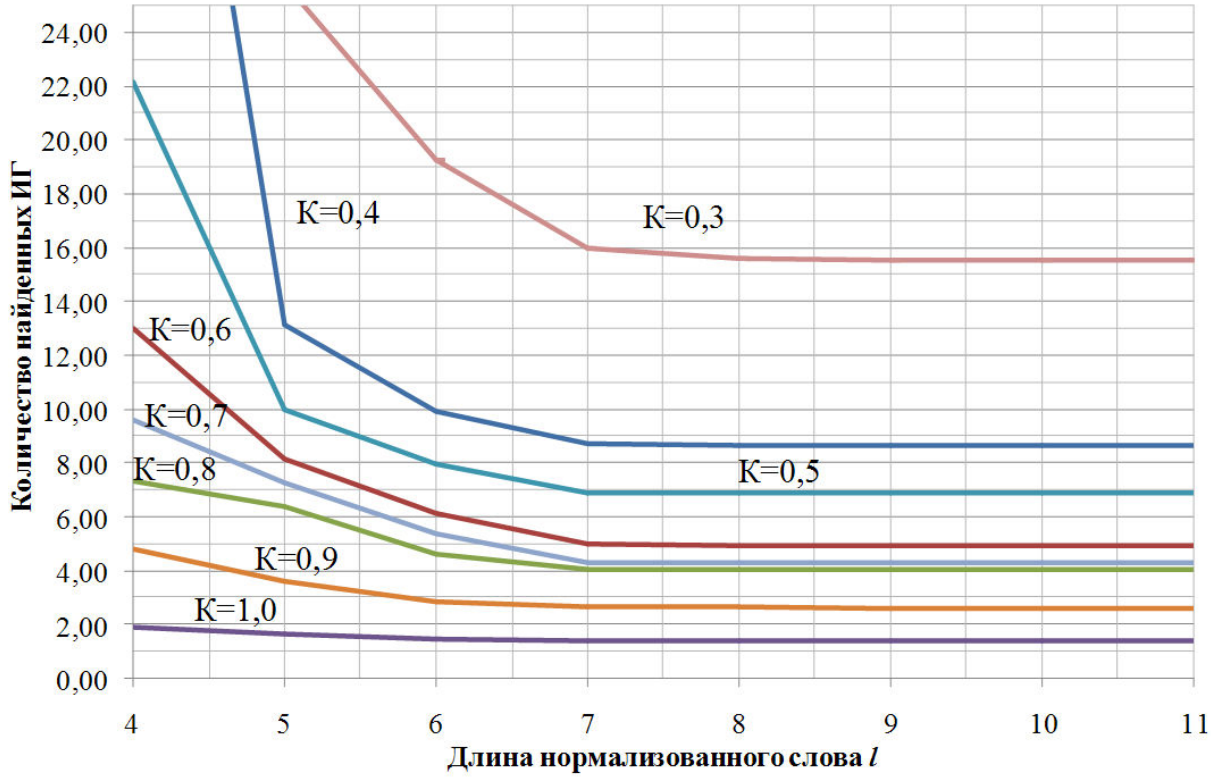
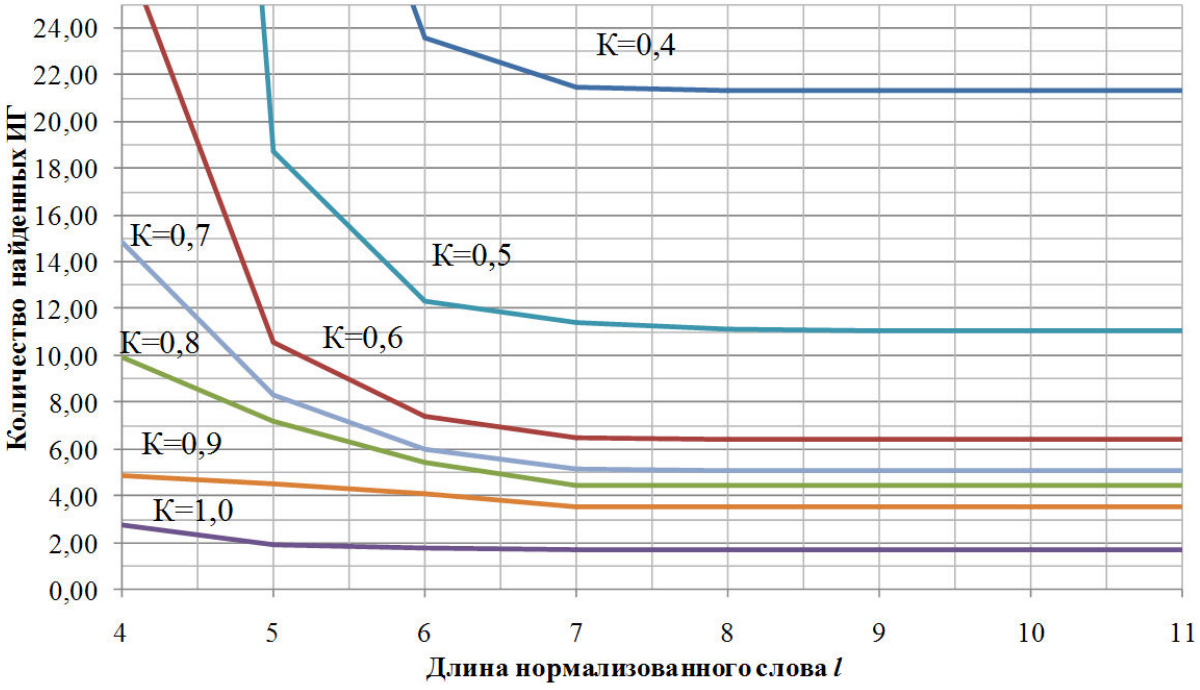


Рис.3. Зависимость количества ошибок первого рода (ложного отождествления слова с образцом) от длины нормализованного слова *l* и порогового значения коэффициента lk-корреляции K при *k*=2.



Р

ис.4. Зависимость количества ошибок первого рода (ложного отождествления слова с образцом) от длины нормализованного слова *l* и порогового значения коэффициента lk-корреляции K при *k*=3.

8. Заключение

В настоящей работе были обсуждены вопросы создания метода машинной транскрипции имен собственных с иностранного языка на русский. Основной упор сделан на передачу фонетического облика имени собственного с использованием возможностей русских фонетики и орфографии.

Предложенный метод позволяет в удобной форме создавать правила для машинной транскрипции. Кроме того, созданный метод в принципе позволяет перейти к решению задачи автоматического порождения правил по имеющейся прецедентной базе, то есть корпусу имен, транскрибированный экспертом. Это должно стать предметом дальнейших исследований.

Далее была рассмотрена задача нечеткого поиска ИГ в специализированных базах персональных данных. Дана формальная постановка задачи, и для ее решения предложен метод нечеткого сравнения и поиска слов с использованием lk-представлений, являющийся дальнейшим развитием ранее предложенного подхода к поиску ИГ [23].

Определены понятия lk-граммы, lk-представления слова $N_{lk}(\mathbf{w})$ и словаря $N_{lk}(U)$, lk-спектра слова по словарю, а также показателя и нормированного коэффициента lk-кореляции слов. Доказаны свойства нормированного коэффициента lk-кореляции. Получены аналитические зависимости пороговых значений показателя lk-кореляции от расстояния Левенштейна. Описан алгоритм нечеткого поиска ИГ с использованием lk-представлений, обеспечивающий гарантированное нахождение соответствий в базе для таких ИГ, степень искажения которых в метрике Левенштейна не превышает заданной. Данный алгоритм имеет сложность порядка $O(||N_{lk}(\mathbf{w})||*\ln(||N_{lk}(U)||))$ операций, что существенно эффективнее алгоритмов прямого поиска в словаре.

С целью определения оптимальных значений параметров алгоритма нечеткого поиска ИГ проведено экспериментальное исследование с использованием фрагментов реальных баз данных, содержащих ИГ. Экспериментально установлено, что значение l=7 является точкой насыщения в процессе уменьшения ошибок первого рода (ложных отождествлений ИГ).

СПИСОК ЛИТЕРАТУРЫ

1. Суперанская А.В., Теоретические основы практической транскрипции / М.: «Наука» 1978., 283 с.

2. Гиляревский Р.С., Старостин Б.А. Иностранные имена и названия в русском тексте. Справочник, 3 изд. / М.: «Высшая школа», 1985., 304 с.

3. Зиндер Л.Р. Общая фонетика – М.: «Высшая школа»,1979 г., 309 с.

4. Rabiner L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings of the IEEE, 77, 257-286 (1989).

5. Knight K., Graehl J. Machine Transliteration // In Proceedings of ACL Workshop on Computational Approaches to Semitic Languages, Philadelphia, USA, 1997

6. Бойцов Л.М. Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска // Труды 6-ой Всероссийской научной конференции “Электронные библиотеки: перспективные методы и технологии, электронные коллекции” - RCDL2004, Пущино, Россия, 2004. http://www.rcdl.ru/papers/2004/paper27.pdf

7. Бойцов Л.М. Использование хеширования по сигнатуре для поиска по сходству // Прикладная математика и информатика, ВМиК МГУ, № 8, 2001. С. 135-154.

8. Zobel J., Moffat A. Inverted Files for Text Search Engines. // ACM Computing Surveys, vol. 38, № 2, 2007.

9. Faloutsos C., Christodoulakis S. Signature files: An access method for documents and its analytical performance evaluation. // ACM Transactions on Information Systems, vol. 2, № 4, 1984.

10. Manber U. A text compression scheme that allows fast searching directly in the compressed file. // Proceedings of the 5th Annual Symposium in Combinatorial Pattern Matching. Asilomar, CA, USA, 5–8 June 1994.

11. Williams H. E., Zobel J. Compressing Integers for Fast File Access. // Computer Journal, vol. 42, № 03, 1996.

12. Hamming R. W. Error–detecting and error-correcting codes. // Bell System

Technical Journal, vol. 29, № 2, 1950.

13. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов. // Доклады Академий Наук СССР. Том 163, № 4, 1965.

14. Damerau F. J. A technique for computer detection and correction of spelling errors. // Communications of the ACM, № 1, 1964.

15. Jaro M. A. Advances in record linking methodology. // Journal of the American Statistical Society, vol. 84, № 406, 1989.

16. Winkler W. E. Overview of Record Linkage and Current Research Directions. // Research Report Series, 2006.

17. Wu S., Manber U. Agrep – A Fast Approximate Pattern-Matching Tool. // Winter USENIX Technical Conference, 1992. (ftp://ftp.cs.arizona.edu/)

18. Wu S., Manber U. GLIMPSE: A Tool to Search Through Entire File Systems. // Winter USENIX Technical Conference, 1994. (ftp://ftp.cs.arizona.edu/)

19. Hull D. A. Stemming Algorithms – A Case Study for Detailed Evaluation. // JASIS, vol. 47, № 1, 1996.

20. Ukkonen E. Approximate String Matching with q-Grams and maximal matches. // Theoretical Computer Science, vol. 92, № 1, 1992.

21. Gusfield D. Algorithms on strings, trees, and sequences: computer science and computational biology. / Cambridge University Press, New York, USA, 1997.

22. Тихонов А.Н., Арсенин В.Я. Методы решения некорректных задач.- М.: Наука,1974.- 222 с.

23. Бондаренко А.В., Герасименко А.А. Об одном алгоритме нечеткого поиска именных компонент в специализированных базах данных // Вестник компьютерных и информационных технологий. № 8 (12), 2005. С.29-34.