С О Ф И Й С К И У Н И В Е Р С И Т Е Т
“СВ. К Л И М Е Н Т О Х Р И Д С К И”

Ф А К У Л Т Е Т П О М А Т Е М А Т И К А И И Н Ф О Р М А Т И К А

Стефан Владимиров Герджиков

АВТОРЕФЕРАТ

на дисертационен труд със заглавие

ЕФЕКТИВЕН АЛГОРИТЪМ ЗА ПРИБЛИЖЕНО ТЪРСЕНЕ В РЕГУЛЯРНИ МНОЖЕСТВА

за придобиване на образователната и научна степен “доктор”

в професионално направление 4.5 “Математика”

по научната специалност 01.01.01 “Математическа логика”

Научен ръководител
доцент д-р Стоян Михов

СОФИЯ, септември 2013

Левенщайн разстоянието възниква във връзка с изследванията на Sellers и Levenshtein в теория на кодирането, [56, 37]. В своите работи те разглеждат модел за предаването на информацията в условията на шум. Идеята на този модел е следната. Под влиянието на шум в съобщение, което се предава, се причиняват грешки. В резултат на това вместо оригиналното съобщение, получателят на съобщението наблюдава изменен вариант на оригинала. Целта на получателя е да разбере какво е било оригиналното съобщение.

Моделът, изложен от Leveshtein и Sellers, разглежда съобщението като последователност от символи, тоест дума, а шума като процес, който влияе последователно на отделните символи на съобщението. Те предполагат, че шумът се проявява в една от следните три атомарни форми:(i) заменя символ с друг символ; (ii) изтрива символ от първоначалното съобщение и (iii) на текущата позиция в съобщението добавя символ. Получателят не разполага с достъп до атомарните прояви на шум, нито знае кога изобщо те са настъпили. Той наблюдава само крайния резултат - последователност от символи, една дума.

Разбира се, разполагайки с тази информация, получателят не е в състояние да възстанови оригиналното съобщение. Дори не може да ограничи възможните оригинални съобщения в разумни граници (те може да са безброй много). Поради тази причина, проблемът се нуждае от допълнително доуточняване.

Първо, в редица естествени приложения, като разпознаване на човешка реч, разпознаване или корекция на писмен текст, получателят знае какъв е езикът, на който е съставено съобщението. Това въвежда допълнително ограничение за самото съобщение. В гореизброените приложения може да предполагаме, че езикът е речник, тоест крайно множество от думи или езикът представлява множеството от всевъзможните последователности от такива думи.

Въпреки допълнителното знание на получателя - той вече има представа от езика на съобщението, което трябва да разкодира, то е практически безполезно. Проблемът е, че шумът, дефиниран по този начин, може да измени всяко съобщение във всяко друго. Така че, получавайки изменението съобщение, получателят по никакъв начин не може да се възползва от допълнителната информация за езика на оригинала.

Поради това се налага второ ограничение, този път за природата на шума. Интуитивно, то се стреми да ограничи агресивния шум и казва следното. Докато съобщението е било предавано, проявите на шум са били най-много b на брой. Тук b е параметър, чиято стойност може да се моделира по различни начини. Той може да бъде абсолютен: каквото и да е полученото съобщение, проявите на шум са били неповече от b. Може да бъде гъвкав параметър: неговата стойност зависи (по някакъв начин) от полученото съобщение. Компромисен вариант между тези две крайности е параметърът b да се определя като фиксирана част от дължината на полученото съобщение, например 10% от дължината на резултатното съобщение.

И така, неформално проблемът за приближено търсене е следният: зна-

ейки езика на оригиналното съобщение, полученото съобщение и максималния брой прояви на шум, кои са всевъзможните оригинални съобщения?

Основната трудност при решаването на този проблем е размерът на дадения език, $\mathcal{L}$ и допустимият брой прояви на шум b. Ако $\mathcal{L} = \{U\}$ е съставен от единствена дума, то, всъщност, каквото и съобщение V да се получи, то трябва да се сравни с U и да се установи може или не може V да е резултат от до b прояви на шум в U. Този проблем може да се реши алгоритъма на Ukkonen, който попада в класа от алгоритми Динамично програмиране, [60] за време $O(b \max(|U|, |V|))$ или с автоматна техника за оптимално време $O(b + \min(|U|, |V|))$, [43].

Задачата се усложнява, когато езикът $\mathcal{L}$ е произволно крайно множество. Случаят $b = 0$, разбира се, е тривиален. Ако $b = 1$ и $\mathcal{L}$ е множеството от всички поддуми на текст T, проблемът може да се реши за време $O(|V|+occ+\log|T|\log\log|T|)$, където occ е броят на допустимите оригинални съобщения, [15]. Приложена, обаче за по-големи стойности на параметъра b, концепцията от [15] води до времева сложност $O(3^b \log^b |V|)$ и изисква допълнителна памет от порядъка на $O((\log|\mathcal{T}|)^b)$, [16]. Това показва, че при големи стойности на параметъра b тези алгоритми не са практични, а за безкрайни регулярни езици са неприложими.

В тези случаи е приложим алогритъмът на Oflazer, [50]. При него се използва краен детерминиран автомат, с който представя езика $\mathcal{L}$ и докато обработва думата V симулира алгоритъма на Ukkonen, [60], за да отсее невъзможните кандидати колкото се може по-рано. С нарастването на параметъра b, обаче, множеството от генерираните на първите b стъпки кандидати расте експоненциално и отново алгоритъмът има ограничени възможности. Този ефект се намалява ако се използва алгоритъма на Михов и Schulz, [42]. Идеята е, че полученото съобщение V може да е резултат от (i) най-много $\frac{b}{2}$ прояви на шум, отговорни за първата половина на V или (ii) най-много $\frac{b}{2}$ прояви на шум, отговорни за втората половина на V. Тази идея позволява на алгоритъма да започне генерирането на кандидати при горна граница $\frac{b}{2}$ вместо b, което е съществено преимущество пред алгоритъма на Oflazer. Технически, за осъществяването на тази идея, са необходими два автомата, за да представят езика $\mathcal{L}$ и езика $\mathcal{L}^{rev}$, при който думите са тези на $\mathcal{L}$, четени отдясно наляво. Така, алгоритъмът на Михов и Schulz от [42] на практика прилага два пъти една и съща процедура – веднъж за езика $\mathcal{L}$ и думата V, и веднъж за езика $\mathcal{L}^{rev}$ и четената отзад напред дума V^{rev}. Двойното траверсиране, обаче, оказва незначително забавяне с оглед на съществено намаляване на пространството на търсене, което води до значително ускорение пред алгоритъма на Oflazer.

В предишните два параграфа очертахме две групи алгоритми за приближено търсене. Първата група, [16, 15], разчита на фиксирана горна граница, b, докато алгоритмите от втората група, [50, 42], не правят никакви предположения за нея. Компромисен вариант между тези две крайности е моделът, при който $b = q|V|$, където $q \in (0; 1)$ е фиксиран параметър, а $|V|$ е дължината на полученото съобщение. При тази схема и предположението,

че езикът $\mathcal{L}$ е множеството от поддуми на текст T, Myers предлага ефективен алгоритъм за приближено търсене. В своята същност той обобщава идеята от алгоритъма на Михов и Schulz, като при него получената дума V рекурсивно се разделя на по-къси поддуми. Резултатът от това делене е, че се появяват повече заявки, но допустимият праг за тях е значително по-малък от първоначалната граница $q|V|$. Фактът, че езикът $\mathcal{L}$ има ясна, линейна структура – той е масив от символи – позволява тази идея да се реализира в ефективен алгоритъм. В [47] Myers описва алгоритъм с очаквана сложност $O(q|V||T|^{\varepsilon(p)} \log |T|)$, където $0 < \varepsilon(p) < 1$ е изпъкнала, растяща функция, която е нула в нулата. Този алгоритъм обаче изисква думата V да бъде достатъчна дълга, поне $\log^2 |T|$ и още изисква допълнителна памет, която нараства поне линейно с дължината на текста T. Тези ограничения на алгоритъма на Myers са съществени и пречат на неговото директно обобщаване.

Известно обобщение на алгоритъма на Myers, представлява алгоритъмът на Navarro и Baeza-Yates, [49], който е приложим за крайни езици $\mathcal{L}$. За съжаление той няма потенциала на алгоритъма на Myers и единственото, което постига е, че започва генерирането на кандидати със малка грешка. Веднага след първата стъпка, обаче, границата става максималната, $q|V|$. В подхода на Navarro и Baeza-Yates, [49], отново се използва индекс, чиято големина зависи от големината на езика.

В настоящата работа разглеждаме проблема за приближено търсене при предположения, че езикът $\mathcal{L}$ е *регулярен език*, а горната граница $b = q|V|$ се определя от получената дума V и предварително фиксиран параметър $q \in (0; 1)$.

Тъй като всеки краен език е в частност регулярен, тоест може да се представи чрез краен автомат, тази постановка е по-обща от разгледаните от Myers,[47], Baeza-Yates, [49], Chan et at., [15], Cole et at., [16]. Допълнително ще разгледаме по-общ модел, в който шумът може да се проявява не само посредством замяна на отделни символи с други единични символи, изтриване или добавяне на символ, а когато последователност от символи може да се замени с друга последователност от символи.

За да може да изложим нашия подход по-точно, полезно е да имаме математически термини за конкретните явления, които изучаваме. Ще предполагаме, че е дадена азбука Σ – крайно множество от букви – и всички думи, т.е. оригиналното и полученото съобщения са записани над тази азбука. Със Σ^* отбелязвам множеството на всички думи над Σ.

За дадена дума $V = v_1 \circ v_2 \circ \cdots \circ v_n$ с $I_i^j(V)$ ще означаваме поддумата на V с начална позиция i и крайна позиция j, тоест $v_i \circ v_{i+1} \circ \cdots \circ v_j$.

Език е произволно множество от думи $\mathcal{L} \subseteq \Sigma^*$. С $Inf(\mathcal{L})$ ще означаваме езика от всички поддуми на думи от езика $\mathcal{L}$. В нашето изследване ще се занимаваме с регулярни езици. Това е клас от езици, които могат да бъдат разпознавани посредством крайно устройство без използването на допълнителна памет. За формалната дефиниция на понятието регулярен език ще си послужим с конструктивен подход, който описва устройствата, които разпознават тези езици, [27]:

Дефиниция 1 *(Дефиниция 1.2.1)*[1] *Краен автомат е* $\mathcal{A} = \langle \Sigma, Q, I, \Delta, T \rangle$, *където* Σ *е азбука,* Q *е крайно множество от състояния,* $I \subseteq Q$ *и* $T \subseteq Q$ *са съответно множество от начални и множество от крайни състояния,* $\Delta \subseteq Q \times \Sigma \times Q$ *е релация на преходите.*

$\mathcal{A}$ *се нарича детерминиран, ако* $|I| = 1$ *и* Δ *е графика на частична функция* $\delta : Q \times \Sigma \to Q$.

За да определим език на краен автомат ни трябва още една дефиниция:

Дефиниция 2 *(Дефиниция 1.2.2) Нека* $\mathcal{A} = \langle \Sigma, Q, I, \Delta, T \rangle$ *е краен автомат. За преход* $t = \langle p', a, p'' \rangle \in \Delta$ *с* $\iota(t) = p'$ *означаваме началното състояние на прехода, с* $\tau(t) = p''$ *– крайното състояние, а с* $\lambda(t) = a$ *– етикета на предхода* t. *Път* π *в автомата* $\mathcal{A} = \langle \Sigma, Q, I, \Delta, T \rangle$ *е крайна последователност от преходи* $\pi = t_1 t_2 \dots t_n$, *където* $t_k \in \Delta$ *и за всяко* $1 \leq k < n$ *е в сила, че* $\tau(t_k) = \iota(t_{k+1})$. *Означаваме с* $|\pi| = n$ *дължината на пътя* π *и още въвеждаме:*

$$\iota(\pi) = \iota(t_1), \quad \tau(\pi) = \tau(t_n),$$
$$\lambda(\pi) = \lambda(t_1) \circ \lambda(t_2) \circ \dots \circ \lambda(t_n)$$

за началното и крайното състояние на π *и съответно за етикета на* π. *Множеството от всички пътища в* $\mathcal{A}$ *означавем с* $\Pi(\mathcal{A})$.

Сега може да определим понятието език на краен автомат:

Дефиниция 3 *(Дефиниция 1.2.3) Път* π *в краен автомат* $\mathcal{A} = \langle \Sigma, Q, I, \Delta, T \rangle$ *се нарича успешен тогава и само тогава, когато* $\iota(\pi) \in I$ *и* $\tau(\pi) \in T$. $Acc(\mathcal{A})$ *е множеството от успешни пътища в* $\mathcal{A}$.

Език, определен от автомата $\mathcal{A}$, *се нарича* $\mathcal{L}(\mathcal{A})$, *който се състои точно от онези думи в* Σ^*, *които са етикети на успешни пътища в* $\mathcal{A}$.:

$$\mathcal{L}(\mathcal{A}) = \{\lambda(\pi) \,|\, \pi \in Acc(\mathcal{A})\}$$

И така за нас регулярен език ще бъде език, определен посредством краен (детерминиран) автомат. Важен резултат за нас ще бъде, че за всеки регулярен език $\mathcal{L}$, езикът $Inf(\mathcal{L})$ също е регулярен, [27].

Сега може да се върнем към нашата първоначална задача, изучаването на проявата на шум. Проявата или отсъствието на (конкретна форма на) шум ще определим като двойка думи, която ще наричаме *операция*. Интуитивно, първата дума съответства на сегмент от оригиналното съобщение, а втората – на образа на този сегмент след като е бил подложен евентуално на шум. Тогава, отсъствието на шум при дадена буква σ съответства на операцията (σ, σ), а множеството на всички такива операции бележим с:

$$Id = \{(\sigma, \sigma) \,|\, \sigma \in \Sigma\}.$$

[1] В скоби до дефинициите, лемите, следствията и твърденията, са указани техните съответствия в дисертационния труд

Проявата на конкретна форма на шум ще бъде операция (X, Y), за която $X \neq Y$. Както отбелязахме, тази дефиниция обобщава по-горе разгледаните конкретни прояви на шум – замяна на буква с друга буква (σ_1, σ_2), изтриване на буква (σ_1, ε), добавяне на буква (ε, σ_2).

Оттук нататък ще предполагаме, че е дадено крайно множество от операции $Op = Id \cup U$, където $\mathcal{U}$ са операции, които представляват проява на шум. Допълнително всяка проява на шум, тоест операция $op \in U$ ще бъде наказвана с някаква цена, която се определя от функция $c : Op \to \mathbb{N}$ с условието:

$$c(op) = 0 \iff op \in Id.$$

Това съответства на интуицията, че отсъствието на шум не се наказва, докато проявата на каквато и да е форма на шум трябва да бъде наказано. В конкретния случай на Левенщайн разстояние, всяка проява на шум се наказва с цена 1.

Двойката (Op, c) наричаме обобщено ортографско разстояние. Основното понятие за нашите разглеждания е понятието *подравняване*:

Дефиниция 4 *(Дефиниция 1.3.4) За дадено обобщено ортографско разстояние (Op, c), подравняване наричаме всяка последователност $\omega \in Op^*$ от операции. Цена на подравняването $\omega = op_1 op_2 \dots op_n$, където $op_i = (X_i, Y_i)$, е сумата от цените на отделните операции, тоест:*

$$c(\omega) = \sum_{k=1}^{n} c(op_k).$$

Лявата и дясната проекция на ω се определят като:

$$l(\omega) = X_1 \circ X_2 \cdots \circ X_n$$
$$r(\omega) = Y_1 \circ Y_2 \circ \cdots \circ Y_n,$$

Казваме, че ω подравнява (думата) U с (думата) V, или още ω е подравняване на U с V ако:

$$l(\omega) = U \text{ и } r(\omega) = V.$$

Това, което моделира горната дефиниция, е процесът на привнасяне на шум в оригиналното съобщение. Цената на подравняването съответства на интуицията колко шумен е бил този процес. В частност, ако $c(op) = 1$ за всяка $op \in Op \setminus Id$, цената на подравняването отчита точно броя на проявите на отделните форми на шум. Това наблюдение мотивира и следната дефиниция:

Дефиниция 5 *(Дефиниция 1.3.5) Разстояние (между думи), породено от обобщеното ортографско разстояние (Op, c), е функцията $d : \Sigma^* \times \Sigma^* \to \mathbb{N} \cup \{\infty\}$, която съпоставя на всяка двойка думи $U, V \in \Sigma^*$ цената на най-евтиното подравняване на U и V, определено от Op, тоест:*

$$d(U, V) = \min_{\omega \in Op^*} \{c(\omega) \mid l(\omega) = U \,\&\, r(\omega) = V\}.$$

Следователно, при $c(op) = 1$ за $op \in Op \setminus Id$, ортографско разстоянието $d(U, V)$ моделира точно минималния брой прояви на шум, които може да са изменили евентуалното оригинално съобщение U до полученото съобщение V.

Така може да запишем формално задачата, която разглеждаме като:

Дефиниция 6 *(Дефиниция 1.6.1) По дадени регулярен език $\mathcal{L}$, (определен от краен автомат), обобщено ортографско разстояние (Op, c) и рационално число $q \in (0; 1)$, се търси ефективен алгоритъм, който:*

Вход: $V \in \Sigma^*$
Изход: $\{U \in \mathcal{L} \mid d(U, V) \leq q|V|\}$.

Ние ще подходим към този проблем като първо изучим някои от свойства на подравняванията и след това ще видим как те се пренасят върху думите от езика, (по-точно върху поддуми от думите на езика).

Съществена роля ще играе множеството от операции Λ, което е отговорно за изтриването на информация от оригиналното съобщение и се задава като:

$$\Lambda = \{op \in Op \mid r(op) = \varepsilon\}.$$

Също така важно ще бъде и означението $\rho = \rho(Op)$, което използваме за дължината на най-дългата дясна проекция на операция $op \in Op$:

$$\rho = \rho(Op) = \max\{|r(op)| \mid op \in Op\}$$

Лема 1 *(Лема 6.1.1) Ако $V = V_0 \circ V_1$ е дума, а ω е подравняване с дясна проекция $r(\omega) = V$, то съществуват подравнявания ω_0 и ω_1 и $o \in Op \cup \{\varepsilon\}$ със свойствата:*

$$\begin{aligned} \omega &= \omega_0 \circ o \circ \omega_1 \\ |r(\omega_0)| &\leq |V_0| \leq |r(\omega_0)| + |r(o)| \\ |r(\omega_1)| &\leq |V_1| \leq |r(\omega_1)| + |r(o)| \\ \omega_0 \circ o &\notin Op^* \circ \Lambda. \end{aligned}$$

При това, ако $o \in Op$, то неравенствата са строги.

Лема 1 казва, че всяко подравняване на дума V може да се раздели на две подравнявания с приблизително равни по дължина десни проекции. Прилагайки принципа на Дирихле, това ни дава възможност да наложим и допълнителни ограничения за цените на тези подравнявания, стига да имаме горна граница за разглежданото подравняване на V.

Следното следствие е обобщение на лемата от страница 3 в [47]:

Следствие 1 *(Следствие 6.1.2) Нека $b = b_0 + b_1$ са неотрицателни рационални числа и $V = V_0 \circ V_1$ е дума като $n_0 = |V_0|$ и $n_1 = |V_1|$, където $n_i > 0$ за $i = 0, 1$. Тогава подравняване ω, за което $r(\omega) = V$ има цена $c(\omega) \leq b$ само ако съществува $k < \rho$ и подравнявания ω'_0 и ω'_1, за които $\omega = \omega'_0 \circ \omega'_1$ и:*

1. *или* $r(\omega_0') = I_1^{n_0-k}(V_0)$ *и* $c(\omega_0) \leq b_0$ *и при* $k > 0$ *първата операция на* ω_1' *има дясна проекция с дължина поне* $k+1$,

2. *или* $r(\omega_1') = I_{k+1}^{n_1}(V_1)$ *и* $c(\omega_1) \leq b_1$ *и последната операция на* ω_0' *има дясна проекция с дължина поне* $k+1$.

При $b = q|V|$, разбиване на V "по средата"на V_0 и V_1 и $b_i = q|V_i|$ за $i = 0, 1$ Следствие 1 ни позволява да сведем задачата за думата V към задача за две по-къси думи V_0 и V_1. Все още, обаче, то говори за конкретно подравняване, а на практика ние не знаем кое е това подравняване. Ние знаем само някои негови свойства:(i) то има дясна проекция V и (ii) цената му не надминава $q|V|$. Следните две дефиниции отчитат точно тази информация:

Дефиниция 7 *(Дефиниция 4.2.1) За множество от подравнявания* $\mathfrak{A} \subseteq Op^*$ *и неотрицателно рационално число* $b \in \mathbb{Q}^+$ *дефинираме:*

$$\mathfrak{A}^{\leq b} = \{\omega \in \mathfrak{A} \,|\, c(\omega) \leq b\}$$

Дефиниция 8 *(Дефиниция 4.2.4) За дума* V *и множество от операции* Op*:*

$$\mathfrak{A}_{Op}(V) = \mathfrak{A}(V) = \{\omega \in Op^* \,|\, r(\omega) = V\}.$$

Ясно е, че нас ни интересуват свойствата на подравняванията $\mathfrak{A}^{\leq b}(V)$ за $b = q|V|$. При изучаването на тези свойства, технически удобно, ще е следното означание:

Дефиниция 9 *(Дефиниция 4.2.2) За множество* $\mathfrak{A} \subseteq Op^*$ *и естествено число* $j \in \mathbb{N}$ *дефинираме*

$$\mathfrak{A}[j] = \{\omega \in \mathfrak{A} \,|\, |l(\omega)| = j\}.$$

Всъщност $\mathfrak{A}[j]$ са тези подравнявания в $\mathfrak{A}$ с лява проекция с дължина точно j. Така че $\mathfrak{A} = \cup_{j\in\mathbb{N}}\mathfrak{A}[j]$. При отразяването на операциите Λ, които не влияят на дясната проекция на подравняванията, ще е удобно и ефективно да разглеждаме подравняванията $\mathfrak{A}[j]$ в нарастващ ред относно j. Следната лема ни позволява да постигнем точно това:

Лема 2 *(Лема 4.2.3) Нека* $\mathfrak{A}$ *е множество от подравнявания, а* $b \in \mathbb{Q}^+$ *е неотрицателно рационално число. Ако* $\mathfrak{B} = \mathfrak{A} \circ \Lambda^*$*, то за всяко* $j \in \mathbb{N}$ *са в сила равенствата:*

$$\mathfrak{B}[j] = \mathfrak{A}[j] \cup \bigcup_{op\in\Lambda} \mathfrak{B}[j - |l(op)|] \circ op$$
$$\mathfrak{B}^{\leq b}[j] = \mathfrak{A}^{\leq b}[j] \cup \bigcup_{op\in\Lambda} \left(\mathfrak{B}^{\leq b}[j - |l(op)|] \circ op\right)^{\leq b}.$$

Сега може да видим как влияят и останалите операции на подравняванията $\mathfrak{A}(V)$:

Лема 3 *(Лема 6.1.3) Нека $\rho = \rho(Op)$ е дължината най-дълга дясна проекция на операция $op \in Op$, $V \in \Sigma^*$ е дума с дължина $|V| = n > 0$. Тогава:*

$$\mathfrak{A}(V) = \left(\bigcup_{j=n-\rho}^{n-1} \bigcup_{op=(U,I_j^n(V))\in Op} \mathfrak{A}(I_1^{j-1}(V)) \circ op \right) \circ \Lambda^*$$

Лема 3 описва как се получават подравняванията $\mathfrak{A}(V)$ от подравняванията за собствени представки на V. Ограниченията за цената на подравняванията, $\leq b$, се въвеждат лесно в това равенство:

$$\mathfrak{A}^{\leq b}(V) = \left(\left(\bigcup_{j=n-\rho}^{n-1} \bigcup_{op=(U,I_j^n(V))\in Op} \mathfrak{A}^{\leq b}(I_1^{j-1}(V)) \circ op \right)^{\leq b} \circ \Lambda^* \right)^{\leq b}.$$

Предвид Лема 2 операцията конкатенация с Λ^* не представлява трудност. Всъщност, при изчисляването на подравняванията $\mathfrak{A}^{\leq b}(V)$ могат да участват само краен брой операции $op \in \Lambda$, защото всяка такава операция имат положителна цена.

Сега може да обобщим Следствие 1 за множества от подравнявания $\mathfrak{A}(V)$. За целта ще ни бъдат необходими следните две означения:

$$\overrightarrow{\mathfrak{A}}_k(V) = \mathfrak{A}(V) \cap (\{op \in Op \,|\, |r(op)| > k\} \circ Op^* \cup \{\varepsilon\})$$
$$\overleftarrow{\mathfrak{A}}_k(V) = \mathfrak{A}(V) \cap (Op^* \circ \{op \in Op \,|\, |r(op)| > k\} \cup \{\varepsilon\})$$

Тоест, това са онези подравнявания, чиято първа/последна операция, ако такава има, има дясна проекция с дължина поне $k+1$ символа.

Лема 4 *(Лема 6.1.5) Нека $\rho = \rho(Op)$ е дължината на най-дълга дясна проекция на операция $op \in Op$, а $V = V_0 \circ V_1$ е дума, където $V_0, V_1 \in \Sigma^*$ имат дължини $n_0 > \rho$ и $n_1 > \rho$ ип $= n_0 + n_1$. Нека още $b = b_0 + b_1$ за неотрицателни рационални числа $b_0, b_1 \in \mathbb{Q}^+$. Означаваме с:*

$$\overrightarrow{\mathfrak{B}}_k = \mathfrak{A}(V_0, b_0 \xrightarrow{k} V, b) = \begin{cases} \left(\mathfrak{A}^{\leq b_0}(I_1^{n_0-k}(V_0)) \circ \mathfrak{A}(I_{n_0-k+1}^{n}(V_0 \circ V_1))\right)^{\leq b} & \text{ако } k = 0 \\ \left(\mathfrak{A}^{\leq b_0}(I_1^{n_0-k}(V_0)) \circ \overrightarrow{\mathfrak{A}}_k(I_{n_0-k+1}^{n}(V_0 \circ V_1))\right)^{\leq b}, & \text{иначе} \end{cases}$$

$$\overleftarrow{\mathfrak{B}}_k = \mathfrak{A}(V, b \xleftarrow{k} V_1, b_1) = \left(\overleftarrow{\mathfrak{A}}_k(I_1^{n_0+k}(V_0 \circ V_1)) \circ \mathfrak{A}^{\leq b_1}(I_{k+1}^{n_1}(V_1))\right)^{\leq b}$$

Тогава:

$$\mathfrak{A}^{\leq b}(V) = \bigcup_{k=0}^{\rho-1} \overrightarrow{\mathfrak{B}}_k \cup \bigcup_{k=1}^{\rho-1} \overleftarrow{\mathfrak{B}}_k.$$

С помощта на Леми 2 и 3, изразите описани в Лема 4 могат да се пресмятат ефективно. За съжаление, за решаването на нашата задача, този подход ще съдържа много излишества. Те се дължат на факта, че много от подравнявания ще имат една и съща лява проекция и това, което е по-лошо е,

че голяма част от тези леви проекции няма да имат нищо общо с дадения език, $\mathcal{L}$. За да преодолеем тези усложнения, ще ни помогнат следните две дефиниция:

Дефиниция 10 *(Дефиниция 4.3.1) Списък от разстояния наричаме всяка частична функция* $L : \Sigma^* \to \mathbb{N}$.

Дефиниция 11 *(Дефиниция 4.3.2) За език* $\mathcal{L}$ *и множество от подравнявания* $\mathfrak{A}$ *казваме, че списъкът от разстояния* $L : \Sigma^* \to \mathbb{N}$ $\mathcal{L}$*-представя* $\mathfrak{A}$ *точно когато:*

$$Dom(L) = Inf(\mathcal{L}) \cap \{l(\omega) \mid \omega \in \mathfrak{A}\}$$
$$L(U) = \min\{c(\omega) \mid \omega \in \mathfrak{A} \text{ и } l(\omega) = U\}.$$

С други думи, L е дефинирана за онези леви проекции от множеството $\mathfrak{A}$, които се явяват поддуми в езика $\mathcal{L}$. Условието $L(U)$ да е минимално измежду $c(\omega)$ за $\omega \in \mathfrak{A}$ и $l(\omega) = U$ е пряко свързано с дефиницията за ортографско разстояние $d(U, V)$, където се намесва подобен минимум. Всъщност, целта ни ще бъде да намерим списъка от разстояния L, който $\mathcal{L}$-представя множеството от подравнявания $\mathfrak{A}^{\leq q|V|}(V)$. Тогава за всяка дума U от $Dom(L)$ ще знаем, че $L(U) = d(U, V) \leq q|V|$. Освен това ще знаем, че $Dom(L)$ е точно сечението на $Inf(\mathcal{L})$ с множеството от левите страни на подравнявания в $\mathfrak{A}^{\leq q|V|}(V)$. Така че, за да решим задачата от Дефиниция 6 ще остане да намерим $Dom(L) \cap \mathcal{L}$. Тази последна стъпка ще може да се осъществи ефективно с помощта на краен детерминиран автомат за езика $\mathcal{L}$.

Две атомарни операции за списъци от разстояния, отразяват операциите: конкатенация и обединение на подравнявания:

Лема 5 *(Следствие 4.3.5) Нека* $\mathcal{L}$ *е език,* $\mathfrak{A}$ *е множество от подравнявания и* $op \in Op$ *е операцията* $op = (X, Y)$*. Ако* $L : \Sigma^* \to \mathbb{N}$ $\mathcal{L}$*-представя* $\mathfrak{A}$ *и* $b \in \mathbb{Q}^+$*, то списъкът от разстояния* $L' : \Sigma^* \to \mathbb{N}$*, определен чрез:*

$$\begin{aligned} Dom(L') &\subseteq Dom(L) \circ X \\ L'(U \circ X) &= \begin{cases} L(U) + c(op) \text{ ако } L(U) + c(op) \leq b \text{ и } U \circ X \in Inf(\mathcal{L}) \\ \neg! \text{ иначе} \end{cases} \end{aligned}$$

$\mathcal{L}$*-представя множеството от подравнявания* $(\mathfrak{A} \circ op)^{\leq b}$.

Трябва да обърнем внимание, че описаната "конструкция"съществено зависи не само от представянето на функцията L, но и от представянето на езика $\mathcal{L}$. Необходимо е въпросът $U \circ X \in Inf(\mathcal{L})$ да има ефективно решение за дадения език. В нашия случай това е така поради факта, че от краен автомат за език $\mathcal{L}$, може ефективно да намерим краен автомат, представящ езика $Inf(\mathcal{L})$.

Лема 6 *(Лема 4.3.6) Нека $\mathcal{L}$ е език, а $\mathfrak{A}_1$ и $\mathfrak{A}_2$ са множества от подравнявания. Ако L_1 и L_2 са списъци от разстояния, които $\mathcal{L}$-представят $\mathfrak{A}_1$ и $\mathfrak{A}_2$, то списъкът от разстояния $L : \Sigma^* \to \mathbb{N}$, определен чрез:*

$$L(U) = \begin{cases} \min\{L_1(U), L_2(U)\} \text{ ако } U \in Dom(L_1) \cap Dom(L_2) \\ L_1(U) \text{ ако } U \in Dom(L_1) \setminus Dom(L_2) \\ L_2(U) \text{ иначе} \end{cases}$$

$\mathcal{L}$-представя множеството от подравнявания $\mathfrak{A} = \mathfrak{A}_1 \cup \mathfrak{A}_2$.

Посредством списъци от разстояния, представящи множества от подравнявания относно даден език, отстраняваме (почти) всички излишества, които съдържат множествата от подравнявания. За единствен недостатък, може да се смята, че дефиниционната област на списъците от разстояния са поддуми, а не например думи, от езика, $\mathcal{L}$.

Сега може да пристъпим към изложението на алгоритъма, който решава задачата, определена в Дефиниция 6. Идеята следва принципа разделяй и владей. Дадената дума V се разделя на по-къси поддуми V_α за които се решава задача поддобна на оригиналната. Решенията за по-късите думи V_α се комбинират, използвайки Лема 4, за да се намерят решения за по-дългите поддуми. Накрая се решава задачата и за дадената дума V. В идеологията си, този подход е същият като представения от Myers, [47], но се различава в своята общност и също техника на пресмятане.

Някои от основните детайли са такива. Нека $\rho = \rho(Op)$ е дължината на най-дългата дясна проекция на операция $op \in Op$. Поддумите V_α се дефинират рекурсивно като $V_\varepsilon = V$. Ако V_α вече е дефинирана и удовлетворява условията, че $|V_\alpha| > 4(\rho - 1)$ и $q|V_\alpha| \geq 1$, определяме $V_{\alpha 0}$ и $V_{\alpha 1}$, за които:

$$V_\alpha = V_{\alpha 0} \circ V_{\alpha 1} \text{ и } |V_{\alpha 0}| - |V_{\alpha 1}| \in \{0, 1\}.$$

По този начин получаваме дървовидна структура, за чиито върхове може да си мислим като за думите V_α. Листата в тази дървовидна структура имат свойството, че $q|V_\alpha| < 1$ и $|V_\alpha| \leq 4(\rho - 1)$.

С всяка дума V_α свързваме ρ^2 подзадачи от вида:

`Дадено:` V_α, k_0, k_1 `като` $0 \leq k_1, k_0 < \rho$
`Търси се:` $\{U \in Inf(\mathcal{L}) \mid d(U, I_{k_0+1}^{|V_\alpha|-k_1}(V_\alpha)) \leq \ q|V_\alpha|\}$

С други думи, поставяме оригиналната задача, но за езика $Inf(\mathcal{L})$ и за няколко поддуми на V_α, които се получават от V_α като се махнат първите няколко и последните няколко букви. От ограниченията за $k_0, k_1 < \rho$, следва, че се премахват най-много $2(\rho - 1)$ букви. Така че всяка от получените поддуми има дължина между $|V_\alpha| - 2(\rho - 1)$ и $|V_\alpha|$.

Решенията на всяка конкретна подзадача се представят чрез списъци от разстояния $L(\alpha, k_0, k_1)$. За намирането на такива списъци за листата V_α се използва (детерминиран) краен автомат за $Inf(\mathcal{L})$ при $q|V_\alpha| < 1$ или се използва предварително изчислен индекс за онези дължини V_α, за които

$q|V_\alpha| \geq 1$, но $|V_\alpha| \leq 4(\rho - 1)$. Броят на тези думи, а също и съответните решения, които изискват тези думи е краен, определен от глобалната константа ρ.

При изчисляването на списъците $L(\alpha, k_0, k_1)$ за вътрешни върхове V_α се използва Лема 4. Последователно се генерират списъци от разстояния $L_r(\alpha, k_0, j)$, които $\mathcal{L}$-представят множествата от подравнявания:

$$\cup_{k=0}^{\rho-1} \mathfrak{A}(I_{k_0+1}^{|V_{\alpha 0}|}(V_{\alpha 0}), q|V_{\alpha 0}| \xrightarrow{k} I_{k_0}^{|V_{\alpha 0}|+j}(V), q|V_\alpha|).$$

При тяхното пресмятане се прилагат свойствата за конкатенация и обединение на списъци от разстояния, които се контролират чрез детерминиран краен автомат за езика $Inf(\mathcal{L})$, а тяхното програмно представяне се осъществява чрез подходяща структура от данни, позволяваща ефективно им поддържане. По подобен начин се пресмятат и списъци от разстояния $L_l(\alpha, k_1, j)$, които се грижат за подравняванията от вида "$\leftarrow$". При тях се налага използването на краен детерминиран автомат за езика $Inf^{rev}(\mathcal{L})$, а също операциите следва да бъдат разглеждани/четени отляво надясно.

След като се намерят списъците от разстояния $L_r(\alpha, k_0, |V_{\alpha 1}| - k_1)$ и $L_l(\alpha, k_1, |V_{\alpha 0}| - k_0 + 1)$ те се обединяват, за да получим списъка $L(\alpha, k_0, k_1)$. Тук е необходимо всяка една от думите в дефиниционните области на двата изходни списъка от разстояния да бъде разгледана по веднъж.

По този начин, при фиксирана дума V_α и знаейки списъците от разстояния $L(\alpha 0, k_0', k_1')$ и $L(\alpha 1, k_0'', k_1'')$ пресмятаме списъците от разстояния $L(\alpha, k_0, k_1)$. Прилагайки тази процедура рекурсивно, намираме списъка от разстояния $L(\varepsilon, 0, 0)$. От думите, които са в дефиниционната област на този списък, използвайки краен детерминиран автомат за езика $\mathcal{L}$ пресяваме тези поддуми на $\mathcal{L}$, които са думи на $\mathcal{L}$. Това представлява и отговорът на задачата, определена в Дефиниция 6.

Алгоритъмът, който идейно изложихме по-горе, може да се конкретизира и формализира, така че за него да се покаже следната ефективност:

Твърдение 1 *(Твърдение 6.2.8) Ако V е дума с дължина $|V| = N$, времето за намирането списъка $L(\varepsilon, 0, 0)$ е $O(T_1 + T_2)$, където:*

$$T_1 = \rho \sum_{k=0}^{\rho-1} \sum_{\alpha} \left(\sum_{j=1}^{|V_{\alpha 0}|} |L_l(\alpha, k, j)| + \sum_{j=0}^{|V_{\alpha 1}|} |L_r(\alpha, k, j)| \right)$$

$$T_2 = \sum_{k_0, k_1 = 0}^{\rho-1} \sum_{\alpha} \sum_{U \in L(\alpha, k_0, k_1)} |U|.$$

Да обърнем внимание, че времето T_1 е оптимално за генерирането на множества с толкова елементи. Времето T_2 обаче, не е. Това се дължи на неефективността на стъпката по обединение на два списъка при окончателното построяване на $L(\alpha, k_0, k_1)$. Този недостатък може да бъде избегнат в случай, че езикът $\mathcal{L}$ е краен. Тогава, списъците от разстояния може да се поддържат подредени, първо по дължината на думата, а между думите

с равна дължина да се поддържа лексикографска наредба. За осъществяването на тази идея е необходим суфиксен масив, [34], който е с дължина пропорционално на сумата от дължините на думите от $\mathcal{L}$. Също така в този случай може да използваме вместо два инфиксни автомата, една единствена структура, [12], която да осъществява ефективното траверсиране по време на търсенето.

Така полчуваме следния резултат:

Твърдение 2 *(Твърдение 6.2.9) Ако $\mathcal{L}$ е краен език с обща големина $||\mathcal{L}|| = \sum_{U\in\mathcal{L}} |U|$, то има структура от данни с общ размер $O(||\mathcal{L}||)$, с която намирането на списъка $L(\varepsilon, 0, 0)$ за произволна дума V с дължина N може да се осъществи за време:*

$$O(T_1 + T_2')$$

където:

$$\begin{aligned} T_1 &= \rho \sum_{k=0}^{\rho-1} \sum_{\alpha} \sum_{j=0}^{|V_{\alpha 1}|} |L_r(\alpha, k, j)| + \sum_{\alpha} \sum_{j=1}^{|V_{\alpha 0}|} |L_l(\alpha, k, j)| \\ T_2' &= \sum_{k_0,k_1=0}^{\rho-1} \sum_{\alpha} |L(\alpha, k_0, k_1)|. \end{aligned}$$

Въпреки че скицираният по-горе алгоритъм е оптимален относно информацията, която генерира, големината на генерираните списъци от разстояния може да нараства експоненциално при определени думи V. Ако това се случва често, както е при алгоритмите на Oflazer, [50], или Михов и Schulz, [42], предложеният алгоритъм качествено няма да се различава от тях. Това, което ще покажем, е, че това не е така. Интуитивно, основната разлика между изложения по-горе алгоритъм и предишните алгоритми, е, че новият алгоритъм не губи излишно време по генерирането на поддуми, които изглеждат безперспективни за крайния резултат.

Подходът, който ще приложим, е вероятностен. Така, оценявайки очакваното време за работа на алгоритъма, имплицитно ще обосновем и изказаната в предния параграф хипотеза.

За целта ни трябва известна подготовка:

Дефиниция 12 *(Дефиниция 1.8.1) За дума $U \in \Sigma^*$ от вида $U = u_1u_2\ldots u_n$, където $u_j \in \Sigma$ определяме i-тип на U $||U||_i$ като:*

$$||U||_i = |\{j \mid u_j = \sigma_i\}|.$$

Типът на думата U е векторът $||U|| \in \mathbb{N}^{|\Sigma|}$, зададен чрез:

$$||U|| = (||U||_1, ||U||_2, \ldots, ||U||_{|\Sigma|}).$$

Всъщност типът на дадена дума кодира количеството букви от даден вид като игнорира техния ред.

Следващата стъпка е количественото кодиране на качествената информация, зададена в обобщеното ортографско разстояние (Op, c). За целта използваме пораждащи функции:

Дефиниция 13 *(Дефиниция 7.1.1) За азбука* Σ, $\Sigma = \{\sigma_1, \ldots, \sigma_{|\Sigma|}\}$, *и обобщено ортографско разстояние* (Op, c) *определяме множествата* $Op_\varepsilon = Op \cap (\{\varepsilon\} \times \Sigma^*)$ *и* $Op_i = Op \cap (\sigma_i \Sigma^* \times \Sigma^*)$. *Пораждащите функции* $f_\varepsilon(t_1, \ldots, t_{|\Sigma|}, z)$ *и* $f_i(t_1, \ldots, t_{|\Sigma|}, z)$ *зададени от* (Op, c) *се дефинират като:*

$$f_\varepsilon(t_1, \ldots, t_{|\Sigma|}, z) = \sum_{(\varepsilon, V) \in Op_\varepsilon} \boldsymbol{t}^{||V||} z^{c(op)} \text{ и}$$

$$f_i(t_1, \ldots t_{|\Sigma|}, z) = \sum_{(U,V) \in Op_i} \boldsymbol{t}^{||V|| - ||U||} z^{c(op)}.$$

Определяме пораждащата функция $f_\Sigma : \mathbb{R}^{|\Sigma|} \times \mathbb{R} \to \mathbb{R}^{|\Sigma|}$ *като:*

$$f_\Sigma(\boldsymbol{t}; z) = (f_1(\boldsymbol{t}; z), f_2(\boldsymbol{t}; z), \ldots f_{|\Sigma|}(\boldsymbol{t}; z)).$$

Означението $\boldsymbol{t}^{\boldsymbol{s}}$, където $\boldsymbol{t} \in \mathbb{R}^n$, а $\boldsymbol{s} \in \mathbb{Z}^n$, е съкратен запис на $\boldsymbol{t}^{\boldsymbol{s}} = \prod_{i=1}^n t_i^{s_i}$. Всъщност, отделните членове на всяка пораждаща функция имат две части. Множител, който кодира типовата информация на операцията – как се променя типа, прилагайки дадена операция; и множител $z^{c(op)}$, който кодира цената на тази операция.

Следващата стъпка е свързана с това да ограничим (отгоре) броя пъти, които дадена поддума от езика ще бъде генерирана. Това мотивира и следната дефиниця:

Дефиниция 14 *(Дефиниция 7.1.3) За дума* $U \in \Sigma^*$, *рационално число* $q \in (0; 1)$ *и вектор* $\boldsymbol{s} \in \mathbb{N}^{|\Sigma|}$, *с* $\mathcal{A}(U, q; s)$ *означаваме множеството от подравнявания* $\omega \in Op^*$, *за които:*

$$l(\omega) = U, \quad ||r(\omega)|| = \boldsymbol{s} \text{ и } c(\omega) \leq q|r(\omega)|.$$

Определяме $a(U, q; \boldsymbol{s}) = |\mathcal{A}(U, q; \boldsymbol{s})|$ *и дефинираме* $\mathcal{A}(U, q)$ *като обединението на множествата* $\mathcal{A}(U, q; \boldsymbol{s})$, *когато* $\boldsymbol{s}$ *пробягва векторите* $\mathbb{N}^{|\Sigma|}$. *Накрая* $Q(U, q; \boldsymbol{t})$ *е пораждащата функция:*

$$Q(U, q; \boldsymbol{t}) = \sum_{\boldsymbol{s}} a(U, q; \boldsymbol{s}) \boldsymbol{t}^{\boldsymbol{s}},$$

където отново $\boldsymbol{s} = (s_1, \ldots, s_{|\Sigma|})$ *пробягва* $\mathbb{N}^{|\Sigma|}$.

Идеята е, че вместо думите V_α, които принуждават генерирането на кандидат U в алгоритъма, ние броим нещо по-грубо, но с по-ясна комбинаторна структура. Именно, интересуваме се от подравняванията, които могат да свидетелстват за такова генериране. Появата на параметъра $\boldsymbol{s}$ съответства на това, че отъждествяваме думите с еднакъв тип, а ограничението за цената идва именно от факта, че алгоритъмът не разглежда "скъпите"подравнявания.

Следващата лема дава възможност да оценим стойностите $a(U, s; q)$ по нетривиален начин:

Лема 7 *(Лема 7.1.4) Нека $U \in \Sigma^*$ и $q \in (0;1)$ са фиксирани. Тогава съществува функция $b_{U,q} : \mathbb{Z}^{|\Sigma|} \times \mathbb{R} \to \mathbb{R}$ със свойствата:*

1. *за всяко $\boldsymbol{s} \in \mathbb{Z}^{|\Sigma|}$ и всяко $z \in (0;1)$:*

$$a(U, q; \boldsymbol{s}) \leq b_{U,q}(\boldsymbol{s}, z).$$

2. *за всеки реалнозначен вектор $\boldsymbol{t} \in \mathbb{R}^{|\Sigma|}$ и всяко $z \in \mathbb{R}$, за които $f_\varepsilon(z^{-q}\boldsymbol{t}; z) < 1$, е в сила равенството:*

$$\sum_{\boldsymbol{s} \in \mathbb{Z}^{|\Sigma|}} b_{U,q}(\boldsymbol{s}, z)\boldsymbol{t}^{boldsymbols} = \frac{(z^{-q}\boldsymbol{t})^{||U||}}{1 - f_\varepsilon(z^{-q}\boldsymbol{t}; z)} \left(\frac{f_\Sigma(z^{-q}\boldsymbol{t}; z)}{1 - f_\varepsilon(z^{-q}\boldsymbol{t}; z)} \right)^{||U||}.$$

С други думи, въвеждайки допълнителен параметър z и интерпретирайки вектора t като вектор от вероятностите на отделните букви, получаваме оценка отгоре за очаквания брой подравнявания, които могат да допринесат за генерирането на поддумата U.

Следващата дефиниция вече се интересува не от свойствата на поддумите на езика, а от думите, V - получените съобщения.

Дефиниция 15 *За дума $V \in \Sigma^*$, език $\mathcal{L}$ и рационално число $q \in (0;1)$ дефинираме $Gen_{\mathcal{L}}(V, q)$като*

$$Gen_{\mathcal{L}}(V, q) = \{\omega \in Op^* \mid l(\omega) \in Inf(\mathcal{L}) \,\&\, r(\omega) = V \,\&\, c(\omega) \leq q|V|\}.$$

Означаваме $gen_{\mathcal{L}}(V, q) = |Gen_{\mathcal{L}}(V, q)|$.

Тази дефиниция задава чрез $gen_{\mathcal{L}}(V, q)$ горна граница за генерираните кандидати за V, а оттук, индиректно, и за сложността на алгоритъма при вход V.

Използвайки Лема 7 може да оценим $gen_{\mathcal{L}}(V)$. Последната подготвителна стъпка е количествена характеризация за регулярния език $\mathcal{L}$. Тя може да се получи като на даден краен (детерминиран) автомат $\mathcal{A} = \langle \Sigma, Q, I, \Delta, T \rangle$ с език $\mathcal{L}(\mathcal{A}) = \mathcal{L}$ и преходи Δ съпоставим характеристичната функция, виж също [17]:

$$g_{\mathcal{A}}(t) = \sum_{\langle p', U, p'' \rangle \in \Delta^*} \boldsymbol{t}^{||U||}.$$

Тоест на всеки маршрут в автомата от състояние p' до състояние p'' с етикет U се съпоставя по един член в тази производяща функция от вида $\boldsymbol{t}^{||U||}$.

Сега може да оценим и величината $gen_{\mathcal{L}}(V, q)$:

Лема 8 *(Лема 7.1.6) Нека $\mathcal{A} = \langle \Sigma, Q, I, \Delta, T \rangle$ е краен (детерминиран) автомат и $q \in (0;1)$ е рационално число. Определяме функциите $v_i : \mathbb{R}^{|\Sigma|} \times \mathbb{R} \to \mathbb{R}$ и $\boldsymbol{v} : \mathbb{R}^{|\Sigma|} \times \mathbb{R} \to \mathbb{R}^{|\Sigma|}$ като:*

$$v_i(\boldsymbol{t}, z) = z^{-q} t_i \frac{f_i(z^{-q}\boldsymbol{t}, z)}{1 - f_\varepsilon(z^{-q}\boldsymbol{t}; z)}$$
$$\boldsymbol{v}(\boldsymbol{t}, z) = (v_1(\boldsymbol{t}, z), \ldots, v_m(\boldsymbol{t}, z)).$$

Тогава за всеки реалнозначен вектор $\boldsymbol{t} \in \mathbb{R}_+^{|\Sigma|}$ *и реално число* $z \in (0;1)$ *със свойството, че* $f_\varepsilon(z^{-q}t, z) < 1$, *е в сила:*

$$\sum_{V\in\Sigma^*} gen_{\mathcal{L}(\mathcal{A})}(V,q)\boldsymbol{t}^{||V||} \leq \frac{g_{\mathcal{A}}(\boldsymbol{v}(\boldsymbol{t},z))}{1-f_\varepsilon(z^{-q}\boldsymbol{t};z)}$$
$$\sum_{V\in\Sigma^*}\sum_{\omega\in Gen_{\mathcal{L}(\mathcal{A})}(V,q)} |l(\omega)|\boldsymbol{t}^{||V||} \leq \sum_{i=1}^{|\Sigma|} \frac{v_i(\boldsymbol{t};z)}{1-f_\varepsilon(z^{-q}t;z)} \frac{\partial g_{\mathcal{A}}}{\partial v_i}(\boldsymbol{v}(\boldsymbol{t};z)).$$

За да се разбере смисъла на тази оценка, векторът t може да се интерпретира като разпределение на буквите Σ. Тогава при предположение за независимост, изразът от лявата страна в първото неравенство е сумата от очакваните стойности $gen_{\mathcal{A}}(., q)$, когато V описва всички думи с дължина 0, 1, 2 и т.н. От друга страна, изразът вдясно е стойност на друга производяща функция, която изобщо казано може да не е сходяща в точката $v(t; z)$, но зависи единствено от дадения език, ортографското разстояние и параметъра q. Въпроса за сходимост ще обсъдим по-долу.

Като следствия от Лема 8 може да получим оценки за очакваната сложност на нашия алгоритъм:

В общия случай имаме:

Твърдение 3 *(Твърдение 7.2.2) Нека* $pr : \Sigma \to (0;1)$ *е вероятност и* $t_i = pr(\sigma_i)$ *за* $\sigma_i \in \Sigma$. *Нека* $q \in (0;1)$ *и* $z \in (0;1)$ *са такива, че* $f_\varepsilon(z^{-4q}\boldsymbol{t}; z) < 1$ *и* $f_\varepsilon(z^{-2q}\boldsymbol{t}; z) < 1$, *а функциите* $v_i(t; z)$ *и* $v_i^{(2)}(\boldsymbol{t}; z)$, $\boldsymbol{v}(\boldsymbol{t}; z)$ *и* $\boldsymbol{v}^{(2)}(\boldsymbol{t}; z)$ *са зададени чрез:*

$$v_i(t;z) = \frac{z^{-2q}t_i f_i(z^{-2q}\boldsymbol{t};z)}{1-f_\varepsilon(z^{-2q}\boldsymbol{t};z)}$$
$$\boldsymbol{v}(\boldsymbol{t};z) = (v_1(\boldsymbol{t};z), v_2(\boldsymbol{t};z), \ldots, v_{|\Sigma|}(\boldsymbol{t};z))$$
$$v_i^{(2)} = \frac{z^{-4q}t_i f_i(z^{-4q}\boldsymbol{t};z)}{1-f_\varepsilon(z^{-4q}\boldsymbol{t};z)}$$
$$\boldsymbol{v}^{(2)}(\boldsymbol{t};z) = (v_1^{(2)}(\boldsymbol{t};z), v_2^{(2)}(\boldsymbol{t};z), \ldots, v_{|\Sigma|}^{(2)}(\boldsymbol{t};z)).$$

Тогава ако $\mathcal{A}$ *краен (детерминиран) автомат с език* $\mathcal{L} = \mathcal{L}(\mathcal{A})$ *с пораждаща функция* $g_{\mathcal{A}}$ *и* $N \in \mathbb{N}$, *то очакваното време за решаването на задачата в* $\mathbb{E}_{V\in\Sigma^N} T(V)$ *е:*

$$\mathbb{E}_{V\in\Sigma^N} T(V) \leq 2c_0 N \left(\frac{g_{\mathcal{A}}(\boldsymbol{v}^{(2)}(\boldsymbol{t};z))}{1-f_\varepsilon(z^{-4q}\boldsymbol{t};z)} + \sum_{i=1}^{|\Sigma|} \frac{v_i(\boldsymbol{t};z)}{1-f_\varepsilon(z^{-2q}\boldsymbol{t};z)} \frac{\partial g_{\mathcal{A}}}{\partial v_i}(\boldsymbol{v}(\boldsymbol{t};z)) \right),$$

където $c_0 = c_0(Op)$ *е константа, зависеща единствено от ортографското разстояние* (Op, c).

Както обяснихме по-горе, състоятелността на горната теорема, зависи съществено от това дали функцията $g_{\mathcal{A}}$ и нейните производни се схождат при съответните стойности на параметрите. Едно достатъчно условие за това е следното:

Лема 9 *(Лема 7.3.3) Ако* $\mathcal{A}$ *е краен (детерминиран) автомат,* $z \in (0;1)$ *и* $\boldsymbol{t} \in \mathbb{R}_+^{|\Sigma|}$ *удовлетворяват условията:*

1. $f_\varepsilon(z^{-q}\boldsymbol{t}; z) < 1$.

2. *за всяко състояние* $k \in Q$ *е изпълнено, че:*

$$\sum_{\langle k,\sigma_i,j\rangle\in\Delta} z^{-q} t_i \frac{f_i(z^{-q}\boldsymbol{t}; z)}{1 - f_\varepsilon(z^{-q}\boldsymbol{t}; z)} < 1,$$

то $g_{\mathcal{A}}(\boldsymbol{v}(\boldsymbol{t}, z))$ *и* $\sum_{j=1}^{|\Sigma|} v_j(\boldsymbol{t}; z) \frac{\partial g_{\mathcal{A}}}{\partial v_j}(\boldsymbol{v}(\boldsymbol{t}, z))$ *са сходящи.*

Съвсем конкретно, ситуацията описана в горната лема настъпва за следния клас автомати при Левенщайн разстояние:

Следствие 2 *(Следствие 7.3.4) Нека* (Op, c) *е Левенщайн разстояние*[2] *и* $q \leq \frac{1}{2}$. *Ако* $|\Sigma| \geq 16$, $t_i = \frac{1}{|\Sigma|}$ *за* $i = 1 \ldots |\Sigma|$ *и* $\mathcal{A}$ *е детерминиран автомат, в който за всяко състояние* k*:*

$$|\{i : !\delta(k, \sigma_i)\}| \leq \frac{\sqrt{|\Sigma|}}{4},$$

то има число $z \in (0, 1)$, *за което* $g_{\mathcal{A}}(\boldsymbol{v}(\boldsymbol{t}; z))$ *се сходжда.*

В последната глава на дисертационния труд е предложен нов подход за дефиниране на близост между думи. За разлика от ортографското разстояние, което предполага наличието на фиксирани операции и цени, предложеният подход определя тази информация на базата на мултимножество от примери $\mathcal{I} = \{(U_i, V_i)\}_i$ и речник от думите за конкретния език $\mathcal{D}$. Примерите представляват двойки от думи (U_i, V_i) като се предполага, че $U_i \in \mathcal{D}$ е близка до наблюдаваната дума V_i, но не е известно защо или как.

Новият метод определя структурата както на речника $\mathcal{D}$ така и на множеството $\mathcal{N} = \{V_i\}_i$ от наблюдавани грешни думи. В тази стуктура са отчетени всички *характерни поддуми* в съответното множество, които се срещат в различен ляв контекст [12] и следователно могат да се разширят по различен начин до дума от съответното множество. Тази структура поражда естествени конструкции, които позволяват всяка дума V_i и всяка дума U_i да бъдат разглеждани йерархично, като съставени от по-къси характерни поддуми.

Методът следва три принципа:

1. повечето букви в думата U_i и V_i съвпадат, иначе не бихме ги възприели като ортографски близки;

2. редът на буквите в U_i и V_i като цяло се запазва, тоест локално може да има транспозиции и пермутации на букви, но не се наблюдава обща тенденция представките да се трансформират в наставки и обратно;

3. дължината на думите U_i не се отклонява съществено от дължината на съответната ѝ дума V_i.

[2] Тоест, $Op = \{(X, Y) \mid X, Y \in \Sigma \cup \{\varepsilon\}$ но не е вярно, че $X = Y = \varepsilon\}$ и $c((X, Y)) = 1$ винаги, когато не е вярно, че $X = Y$.

Въз основа на тези три принципа моделираме съответствието на йерархичните структури за U_i и V_i за конкретна двойка $(U_i, V_i) \in \mathcal{I}$. Това ни позволява да извелечем операции, които свидетелстват за промяната на U_i до V_i, като отчитат контекста на структурата на примерите и тази на дадения език. Допълнително, използвайки прост емпиричен подход, оценяваме вероятността на всяка от дефинираните операции. По този начин получаваме количествена информация за това, колко съществена е дадена операция при описването на примерите $\mathcal{I}$.

В работата е показана конкретна алгоритмична реализация на скицирания по-горе метод, която може да бъде осъществена ефективно:

Лема 10 *(Лема 8.1.17) Нека е дадено мултимножество от примери* $\mathcal{I} = \{(U_i, V_i)\}_i$ *и речник* $\mathcal{D}$*, а* $\mathcal{N} = \{V \mid \exists U((U, V) \in \mathcal{I})\}$*. Тогава операциите, определени от* $\mathcal{I}$ *и* $\mathcal{D}$ *заедно с техните вероятности могат да бъдат определени от предложения метод за време:*

$$O(\sum_{(U_i,V_i)\in\mathcal{I}} |U_i|^3 |V_i|).$$

Изложен е и метод за търсене, който по дадена нова дума V предлага кандидати $U \in \mathcal{D}$, които са близки до V. При това се отчитат следните характеристики:

1. структурата на думата V в рамките на глобалната структура на примерните думи $\mathcal{N}$.

2. извлечените при първата фаза на метода операции и техните вероятности.

3. конкатенацията на операциите трябва да бъде подравняване за (U, V).

При определянето на структурата на думата V в точка 1, методът се стреми да отчете всевъзможните характерни поддуми от йерархичната структура на V, ако V би била дума от множеството $\mathcal{N}$. Така, на практика се взимат предвид поддуми с произволна дължина, които са общи за V и примерните думи V_i, определени от множеството $\mathcal{I}$.

Въз основа на точка 2 предложеният алгоритъм допълнително класира думите U според това колко е вероятно те да бъдат близки до V. По този начин, методът се стреми действително да отговори на въпроса, коя дума от речника $\mathcal{D}$ всъщност представлява думата V. Това би била именно думата, класирана на първо място.

Предложената в дисертационния труд алгоритмична реализация на този метод определя структурата на думата V за линейно време $O(|V|)$. След това генерира хипотези *Hypotheses*, тоест подравнявания, които спазват ограниченията от точка 1, но подравняват само представки на думите U и V. При това те се генерират в намаляващ ред според тяхната вероятност.

Основният резултат е следният:

Твърдение 4 *(Твърдение 8.2.9) При дадена дума V намирането на сортиран списък от близките до V думи в речника $\mathcal{D}$ изисква:*

$$O(|V| + \sum_{h \in Hypotheses} (|U(h)| + \log |Hypotheses|))$$

където $Hypotheses$ е множеството от генерирани хипотези, а $U(h)$ е поддумата, с която се разширява хипотезата h.

В работата са обсъдени допълнителни фактори, които са интересни за конкретни приложения и могат да се използват за контролирането на растежа на множеството $Hypotheses$. По този начин може да се контролира и ефективността на предложения алгоритъм.

Адекватността и приложимостта на скицирания по-горе метод е засвидетелствана с резултати, получени върху експериментални данни, включващи два различни типа задачи и обхващащи два различни (естествени) езика.

Резюме на получените резултати и декларация за оригиналност на труда

Авторът смята, че основни приноси на дисертационния труд са следните резултати:

1. **Общ апарат за изучаване на свойствата на подравнявания от думи и ефективното генериране на кандидати за корекция**. Този апарат, основаващ се на концепцията за множествата от подравнявания и списъци от разстояния, е разработен в Глава 4.

2. **Нов алгоритъм за приближено търсене в произволно регулярно множество от думи.** Това е алгоритъмът по схемата разделяй и владей, който е представен подробно в Глава 5.

3. **Нов алгоритъм за приближено търсене за произволно ортографско разстояние.** Това е алгоритъмът представен в Глава 6, който използва схемата разделяй и владей и решава проблема за приближено търсене за произволно ортографско разстояние.

4. **Практическа ефективност на предложените алгоритми.** Предложеният алгоритъм има свойството, че постепенно увеличава допустимото разстояние. По този начин броят на генерираните кандидати се поддържа разумно малък. Допълнителен аргумент за практическата ефективност на алгоритъма са и резултатите от Твърдения 1 и 2, които казват, че всеки кандидат за корекция в този алгоритъм се генерира веднъж, (по-точно този брой е $O(1)$ – ограничен от глобална константа).

5. **Теоретична ефективност на предложения алгоритъм.** Предложен е вероятностен подход, който теоретично аргументира ефективността на алгоритъма, Твърдение 3.

6. **Нов подход за дефиниране на близост между думи.** Глава 8 представя нов подход за дефиниране на близост между думи, който отчита цялостната структура на езика, а също и контекстната информация в рамките на самите думи. Концептуалните предимства на този подход са потвърдени експериментално, а ефективността на разработения алгоритъм е оценена в Твърдение 4.

Идеята, която стои зад Резлутатите 1–5 развива предишни идеи на Myers, [47], Baeza-Yates и Navarro, [49], Mihov и Schulz, [42]. За нейната реализация са използвани класически резултати от теорията на крайните автомати и частично предишни резултати на Blumer et al., [11, 12], Ko и Aluru, [34]. За постигането на Резултат 5 е използван алгебричен подход, предложен от Eilenberg, [17], класически резултати от линейната алгебра, теория на графите, теория на крайните автомати.

Резултат 6 е следствие от оригинална математическа интерпретация на структурата, предложена от Blumer et al., [12]. За нейната алгоритмична реализация са приложени идеи на Aho и Corasick, [7], а също и общ метод, предложен от Hart et al., [24, 25], от който новият подход съумява да се възползва.

Резултатите 1–6 са отразени в една самостоятелна и няколко статии в съавторство със Стоян Михов, Петър Митанкин, Klaus Schulz и Владислав Ненчев:

1. **Some algebraic properties of alignments of words, S. Gerdjikov, Comptes rendu de l'Academie bulgare des Sciences, 65(10):1311–1319, 2012**,

   Тази статия е самостоятелна. Тя представя основните стъпки, които водят до Резултати 1 и 5. По-точно, тази статия въвежда и описва основните свойства на списъците от разстояния и множества от подравнявания, които са в основата на Твърдения 1 и 2. Тя също представя Лема 8, която се използва съществено при доказателството Твърдение 3.

2. **WallBreaker - overcoming the wall effect in similarity search, S. Gerdjikov, S. Mihov, P. Mitankin, and K. U. Schulz, ACM Proceedings of the 2013 Joint EDBT/ICDT Workshops, 2013,**

   и нейната пълна версия:

   Good parts first - a new algorithm for approximate search in lexica and string databases, S. Gerdjikov, S. Mihov, P. Mitankin, and K. U. Schulz, ArXiv, 2013
   e-prints:http://adsabs.harvard.edu/abs/2013arXiv1301.0722G.

   Тези две статии са в съавторство със Петър Митанкин, Стоян Михов, и Klaus Schulz. Алгоритъмът за приближено търсене, описан в статиите, е резултат, постигнат в рамките на семинара, воден от Стоян Михов, където Петър Митанкин и авторът едновременно работеха и представяха своите идеи за решаването на този проблем. Така, заедно, те достигнаха до две еквивалентни решения в случая на Левенщайн разстояние. Различията бяха в линейните структури от данни, които се използваха за представянето на множеството от думи.

   Понататъшните изследвания на Петър Митанкин доведоха до окончателния вариант, представен в тези статии, в който се използва още по-компактна структура, именно тази от [12, 29].

   В същото време дисертантът обобщи метода за произволно ортографско разстояние, Резултат 3.

   Тези части от статиите [20] и [21] представляват основата за постигането на Резултати 2, 3 и 4, а експерименталните резултати в [20] и независимото оценяване по време на форума Workshop on Scalable Similarity Search Strings/Join, Genoa, 2013, емпирично потвърждават Резултат 4.

3. **Extraction of spelling variations for noisy text correction. S. Gerdjikov, S. Mihov, and V. Nenchev, In Proceedings of 12th International Conference on Documents Analysis and Recognition 2013, 2013, p.324–328.**

   Тази статия е в съавторство със Стоян Михов и Владислав Ненчев. В нея авторът има основен принос при създаването, разработването и конкретната реализация на описания в статията подход, който е всъщност Резлутат 6.

Някои от тези резултати бяха представени на международни форуми:

- **WallBreaker - overcoming the wall effect in similarity search, S. Gerdjikov, S. Mihov, P. Mitankin, and K. U. Schulz, on the EDBT/ICDT Workshop for Scalable String Similarity Search/Join, Genoa, Italy, 2013.** (презентация на С. Герджиков)

  При тази изява авторът представи основните идеи от статиите [21] и [20] с акцент върху Резултата 4 – практическата ефективност на алгоритъма за приближено търсене.

- **Extraction of spelling variations for noisy text correction. S. Gerdjikov, S. Mihov, and V. Nenchev on the 12th International Conference on Documents Analysis and Recognition, Washington, DC, USA, 2013.** (постер, представен от С. Герджиков)

  С този постер авторът илюстрира резултатите от [22]. В дискусиите с експерти, с научни интереси в обработката на исторически текстове и OCR-корекция, авторът представи Резултата 6 от различни гледни точки, като по този начин мотивира неговата широка приложимост.

- **On Modernisation of Historical Texts. S. Gerdjikov, Computability in Europe 2012, Cambridge, UK, 2012.** (презентация С. Герджиков)

  В тази презентация авторът представи Резултат 6 в контекста на нормализация на английски текстове от 17 век.

Съществени части от Резултати 5 и 6 бяха представени на Пролетните сесии на Факултета по математика и информатика на Софийския Университет:

- **Доколко са регулярни правописните промени в българския език от 19 век до днес? С. Герджиков, Пролетна сесия на Факултета по математика и информатика на Софийския Университет, 2012** (презентация пред катедра Математическа логика и приложенията ѝ)

  В тази презентация авторът представи Резултат 6 върху конкретната задача за нормализация на български език от 19 век.

- **Комбинаторен етюд: "Подравнявания на думи" , С. Герджиков, Пролетна сесия на Факултета по математика и информатика на Софийския Университет, 2011** (презентация пред катедра Математическа логика и приложенията ѝ)

  В тази презентация авторът представи комбинаторния резултат от Лема 8, който води до Резултат 5.

Въз основа на предишните параграфи, авторът заявява, че настоящата дисертация е оригинален научен труд. Употребата на предишни резултати е отразена по честен начин като съответните източници са цитирани според условията на авторските права на техните автори, и/или издатели и/или други притежатели на конкретните авторски права.

Благодарен съм на Стоян Михов за поставения изследователски проблем и за ръководения от него семинар. Признателен съм на участниците в този семинар за ползотворните дискусии, възможността за непосредствено излагане на нови идеи и споделения им опит. На подготовката за поредните сбирки на семинара и на дискусиите по време на семинара се дължат голяма част от изложените в дисертационния труд резултати. Други резултати, получени в същата благоприятна научна среда намериха друго място за изява, но най-интересни остават тези сбирки, които завършват не с решение, а с въпрос, пораждащ нови творчески изследвания, хипотези и разсъждения – причина за следващата сбирка и следващата дискусия на семинара.

Библиография

[1] http://www.speech.cs.cmu.edu/cgi-bin/cmudict.

[2] http://www.gutenberg.org.

[3] http://1641.tcd.ie.

[4] http://trec.nist.gov/pubs/trec5/t5_proceedings.html.

[5] http://www.statmt.org/moses/.

[6] http://www.impact-project.eu.

[7] A. V. Aho and M. J. Corasick. Efficient string matching: an aid to bibliographic search. *Commun. ACM*, 18:333–340, June 1975.

[8] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[9] R. A. Baeza-Yates and G. Navarro. Faster approximate string matching. *Algorithmica*, 23(2):127–158, 1999.

[10] A. Baron and P. Rayson. Automatic standardisation of texts containing spelling variations how much training data do you need? In *Proceedings of the Corpus Linguistics Conference, CL2009*, 2009.

[11] A. Blumer, J. Blumer, D. Haussler, A. Ehrenfeucht, M. T. Chen, and J. Seiferas. The smallest automation recognizing the subwords of a text. *Theoretical Computer Science*, 40:31 – 55, 1985. Eleventh International Colloquium on Automata, Languages and Programming.

[12] A. Blumer, J. Blumer, D. Haussler, R. McConnell, and A. Ehrenfeucht. Complete inverted files for efficient text retrieval and analysis. *Journal of the Association for Computing Machinery*, 34(3):578–595, 1987.

[13] L. Boytsov. Super-linear indices for approximate dictionary super-linear indices for approximate dictionary super-linear indices for approximate dictionary searching. In *Proceedings of the 5th International Conference on Similarity Search and Applications*, 2012.

[14] E. Brill and R. C. Moore. An improved error model for noisy channel spelling correction. In *Proc. 38th ACL*, pages 286–293, 2000.

[15] H.-L. Chan, T.-W. Lam, W.-K. Sung, S.-L. Tam, and S.-S. Wong. Compressed indexes for approximate string matching. In *ESA'06: Proceedings of the 14th conference on Annual European Symposium*, pages 208–219, London, UK, 2006. Springer-Verlag.

[16] R. Cole, L.-A. Gottlieb, and M. Lewenstein. Dictionary matching and indexing with errors and don't cares. In *STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 91–100, New York, NY, USA, 2004. ACM.

[17] S. Eilenberg. *Automata, Languages and Machines*, volume A. Academic Press, Inc. Orlando, Fl, USA, 1974.

[18] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, pages 652–673, 1998.

[19] S. Gerdjikov. Some algebraic properties of alignments of words. *Comptes rendu de l'Academie bulgare des Sciences*, 65(10):1311–1319, 2012.

[20] S. Gerdjikov, S. Mihov, P. Mitankin, and K. Schulz. Good parts first - a new algorithm for approximate search in lexica and string databases. ArXiv e-prints:http://adsabs.harvard.edu/abs/2013arXiv1301.0722G, 2013.

[21] S. Gerdjikov, S. Mihov, P. Mitankin, and K. Schulz. Wallbreaker - overcoming the wall effect in similarity search. In *ACM Proceedings of the 2013 Joint EDBT/ICDT Workshops*, 2013.

[22] S. Gerdjikov, S. Mihov, and V. Nenchev. Extraction of spelling variations for noisy text correction. In *Proceedings of 12th International Conference on Documents Analysis and Recognition*, pages 324–328, 2013.

[23] D. Gusfield. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology.* Cambridge University Press, 1997.

[24] P. Hart, N. Y. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics SSC4*, 4(2):100–107, 1968.

[25] P. Hart, N. Y. Nilsson, and B. Raphael. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Newsletter*, 37:28–29, 1971.

[26] D. S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, 24:664–75, 1977.

[27] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison-Wesley Publishing Company, 1979.

[28] R. Horn and C. Johnson. *Norms for Vectors and Matrices.* Cambridge University Press, 1990.

[29] S. Inenaga, H. Hoshino, A. Shinohara, M. Takeda, and S. Arikawa. On-line construction of symmetric compact directed acyclic word graphs. In *Proc. of 8th International Symposium on String Processing and Information Retrieval (SPIRE'01)*, pages 96–110. IEEE Computer Society, 2001.

[30] S. Inenaga, H. Hoshino, A. Shinohara, M. Takeda, S. Arikawa, G. Mauri, and G. Pavesi. On-line construction of compact directed acyclic word graphs. *Word Journal Of The International Linguistic Association*, 146(2):1–12, 2005.

[31] J. Kärkkäinen and P. Snaders. Simple linear work suffix array construction. In *In proceedings of the 30th International Colloquium Automata, Languages and Programming.*, pages 81–88, Cancun, Mexico, 2003. IEEE Computer Society.

[32] S. Kleene. Representation of events in nerve nets and finite automata. *Automata Studies*, (34):3–41, 1956.

[33] J. Klovstad and L. Mondshein. The caspers linguistic analysis system. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 23(1):118–123, 1975.

[34] P. Ko and S. Aluru. Space efficient linear time construction of suffix arrays. *Journal of Discrete Algorithms*, (3):143–156, 2005.

[35] S. Koeva. Grammar dictionary of the bulgarian language. description of the concept for organization of the linguistic data. *Bulgarian language*, 6:49–58, 1998.

[36] K. Kukich. Techniques for automatically correcting words in texts. *ACM Computing Surveys*, pages 377–439, 1992.

[37] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 1966.

[38] D. J. Lipman and W. R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–41, 1985.

[39] M. G. Maaß. Linear bidirectional on-line construction of affix trees. In *Proc. of 11th Ann. Symp. on Combinatorial Pattern Matching (LNCS1848)*, pages 320–334. Springer-Verlag, 2000.

[40] E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of the Association for Computing Machinery*, 23(2):262–272, 1976.

[41] S. Mihov, P. Mitankin, A. Gotscharek, U. Reffle, and C. Schulz, K. U. Ringlstetter. Using automated error profiling of texts for improved selection of correction candidates for garbled tokens. In *AI 2007: Advances in Artificial Intelligence*, pages 456–465. Springer Berlin Heidelberg, 2007.

[42] S. Mihov and K. U. Schulz. Fast approximate search in large dictionaries. *Computational Linguistics*, 30(4):451–477, 2004.

[43] P. Mitankin, S. Mihov, and K. U. Schulz. Deciding word neighborhood with universal neighborhood automata. *Theoretical Computer Science*, 412(22):2340 – 2355, 2011.

[44] M. Mohri, P. Moreno, and E. Weinstein. General suffix automaton construction algorithm and space bounds. *Theoretical Computer Science*, 410(37):3553–3562, 2009.

[45] M. Mohri and M. Riley. An efficient algorithm for the n-best-strings problem. In *Proceedings ICSLP*, 2002.

[46] M. Mor and A. S. Fraenkel. A hash code method for detecting and correcting spelling errors. *Commun. ACM*, 25(12):935–938, 1982.

[47] E. W. Myers. A sublinear algorithm for approximate keyword searching. *Algorithmica*, 12:345–374, 1994.

[48] P. Nabende. *Applying Dynamic Bayesian Networks in Transliteration Detection and Generation*. PhD thesis, University of Groningen, 2011.

[49] G. Navarro and R. Baeza-Yates. A hybrid indexing method for approximate string matching. *Journal of Discrete Algorithms*, 1(1):205–239, 2000.

[50] K. Oflazer. Error-tolerant finite-state recognition with applications to morphological analysis and spelling correction. *Computational Linguistics*, 22(1):73–89, 1996.

[51] S. J. Puglisi, W. F. Smyth, and A. H. Turpin. A taxonomy of suffix array construction algorithms. *ACM Comput. Surv.*, 39(2), 2007.

[52] U. Reffle. Efficiently generating correction suggestions for garbled tockens of historical language. *Natural Language Engineering*, 17(2):265–282, 2011.

[53] E. S. Rista and P. N. Yianilos. Learning string-edit distance. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(5):522–532, 1998.

[54] K. Schulz, S. Mihov, and P. Mitankin. Fast selection of small and precise candidate sets from dictionaies for text correction tasks. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 471–75, Washington, DC, USA, 2007. IEEE Computer Society.

[55] K. U. Schulz and S. Mihov. Fast string correction with Levenshtein automata. *IJDAR*, 5(1):67–85, 2002.

[56] F. J. Sellers. Bit loss and gain correction code. *Information Theory, IRE Transactions on*, 8(1):35–38, 1962.

[57] J. Stoye. Affixbäume. Master's thesis, Universität Bielefeld, May 1995.

[58] J. Stoye. Affix trees. Technical Report 2000-04, Universität Bielefeld, Technische Fakultät, 2000.

[59] K. Toutaniva and R. C. Moore. Pronunciation modeling of improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 144–151, 2002.

[60] E. Ukkonen. Algorithms for approximate string matching. *Information Control*, 64:100–18, 1985.

[61] E. Ukkonen. On-line construction of suffix-trees. *Algorithmica*, 14(3):249–260, 1995.

[62] http://www.uibk.ac.at/anglistik/projects/icamet.

[63] M. S. Watermann. *Computational Biology: Maps, Sequences, Genomes.* Chapman and Hall, London, England, 1995.

[64] M. S. Watermann, M. Eggert, and E. Lander. A new algorithm for best subsequence alignments with application to trna-rrna comparison. *J. Mol. Biol.*, 197:723–728, 1987.

[65] M. S. Watermann and M. Vingron. Rapid and accurate estimates of statistical significance for sequence data base searches. In *Proc. Natl. Academy Sciences*, volume 81, pages 4625–4628, 1994.

[66] P. Weiner. Linear pattern matching algorithms. In *Proceedings of 14th IEEE Annual Symposium on Switching and Automata Theory*, pages 1–11, 1973.

[67] S. Wu and U. Manber. Fast text searching: allowing errors. *Communications of the ACM*, 35(10):83–91, 1992.