ЧЕТИРИМЕРНО КОДИРАНЕ НА СИМВОЛНИ ПОСЛЕДОВАТЕЛНОСТИ И ОЦЕНКА НА СХОДСТВАТА И РАЗЛИКИТЕ МЕЖДУ ТЯХ

Мартин Маринов

Резюме: *Настоящата статия описва алгоритъм за разредено разпределено кодиране (Sparse Distributed Representation или SDR) на текстови данни. По същество, това е модифицирана версия на предходна разработка, като промените в алгоритъма имат следните предимства:*

- *способност за декодиране, без загуба на информация;*
- *значително по-голям капацитет на кодиращото пространство;*
- *възможност за по-подробно описание на приликите и разликите между кодираните стрингове.*

Основният недостатък, в сравнение с първоначалният алгоритъм, е повишена сложност на процедурата за сравнение на кодирани стрингове. Това се дължи на използването на четиримерно вместо двумерно кодиращо пространство.

Ключови думи: *машинна обработка на естествени езици, текстови данни, текстов анализ, автоматичен превод, чистене на данни, подготовка на данни*

FOUR-DIMENSIONAL ENCODING OF CHARACTER SEQUENCES AND EVALUATION OF THEIR SIMILARITIES AND DIFFERENCES

Martin Marinov

Abstract: *This paper describes a string encoding algorithm, which produces sparse distributed representations (SDR) of text data. In essence, this is a modified version of a prior algorithm and the modifications have the following benefits:*

- *the ability to decode data, without loss of information;*
- *greatly increased capacity of the encoding space;*
- *the possibility of performing more detailed comparisons of encoded strings.*

The main disadvantage compared to the prior algorithm is the increased complexity of the procedure for encoded string comparison. This is due to the use of a four-dimensional encoding space, instead of a two-dimensional space.

Key-words: *NLP, languages, text mining, unstructured text, data cleaning, data preparation, sparse data representation, SDR*

ISSN 1311-0829
DOI 10.47978/TUS.2020.70.02.008

1. ВЪВЕДЕНИЕ

Настоящата статия описва модификация на разработен от автора алгоритъм за кодиране на текстови данни. Детайлно резюме на въпросният алгоритъм и мотивацията за разработката му не е включено в настоящата статия, защото то е темата на предходни научни доклади. Първият доклад бе представен по време на Международна Конференция Автоматика 2018 [1]. Вторият доклад бе представен пред международна аудитория, по време ICASC 2019 [2].

С цел да се намали неяснотата, първоначално разработеният алгоритъм ще бъде наричан CP (Current-Prior). Модификацията, която е тема на настоящата статия, ще бъде назована CPPP (Current-Position-Prior-Position). Имената им всъщност описват начина, по който произволни стрингове биват трансформирани в съответните им разредено, разпределени образи.

2. ПРИЛИКИ И РАЗЛИКИ В ПРИНЦИПИТЕ НА ДЕЙСТВИЕ НА CP И CPPP

CP обхожда стрингове символ по символ и използва само подредбата им, за да определи ненулевите елементи в кодиращото пространство. Координатната система на пространството всъщност не използва числа, използва само текстови символи. При CP, координатните оси са две, като текущо изчетените символи се отразяват върху абсцисата, а предходно изчетените се проектират върху ординатата. От там идва и името на алгоритъма, current-prior (текущ и предходен символ).

	-	a	b	c	d	e	f	g	h
-									
a									
b									
c									
d						de		dg	
e					ed	ee		eg	
f									
g						ge			
h									

***Фиг. 1** – edge (CP кодирана)*

		d						e						g					
		0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5
d	0																		
	1																		
	2											de					dg		
	3																		
	4																		
	5																		
e	0																		
	1			ed								ee					eg		
	2																		
	3																		
	4																		
	5																		
g	0																		
	1																		
	2																		
	3											ge							
	4																		
	5																		

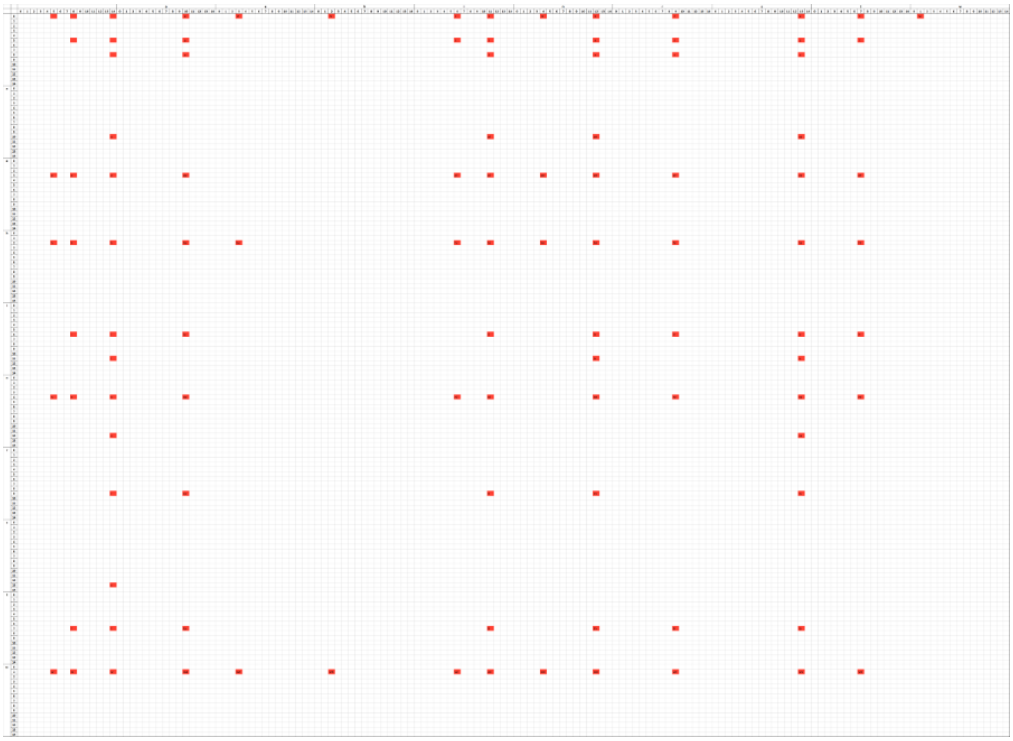
***Фиг. 2** – edge (CPPP кодирана)*

Резултатът от подобно кодиране може да се представи концептуално по различни начини. Табличният е най-четим за хора (фиг.1). За автоматична компютърна обработка е по-ефективно кодираните стрингове да се съхраняват в други структури. Например, думата кодирана от CP и показана на фиг.1 може да се съхрани и като множество от символни чифтове: {' e', ' d', ' g', ' ', 'ed', 'eg', 'ee', 'e ', 'dg', 'de', 'd ', 'ge', 'g '}.

CPPP работи по сходен начин, но освен символни подредби използва и позицията на символите. Тоест, добавят се още две координатни оси към кодиращото пространство, за отчитане на позициите на символите (фиг. 2). Това е отчетено в името му, Current-Position-Prior-Position (текущ символ и позицията му, предходен символ и позицията му). Въпреки че кодиращото пространство има четири измерения в този случай, то все още може да се представи в табличен вид.

На теория, това не затруднява интерпретирането на таблиците от хора, защото кодираните стрингови образи са разредени и разпределени точки в пространството, по дефиниция. Тоест, повторението на осите за позиция на символите не се отразява значително върху това как изглеждат кодираните стрингове в табличен вид. Независимо колко измерения се използват за кодиране, то винаги ще изглежда като разредена матрица, чиито ненулеви елементи са разпределени на привидно случаен принцип (в действителност няма случайност в разпределението, закономерно е).

На практика, затруднение при интерпретацията от човек възниква предимно при нарастването на размерът на кодиращото пространство. Представената информация е логична и консистентна, просто за по-дългите стрингове се накъсва и разпръсва до такава степен в таблицата, че е трудно да се обхване с поглед (фиг. 3).



Фиг. 3. *Фразата “when it rains”, кодирана от CPPP*

В крайна сметка, обаче, целта е тези алгоритми да се ползват от компютри, не хора. Машините са много по-добри в бързата обработка на големи количества информация. Това, че кодираните текстови данни могат да се представят в човешко-четим табличен формат помага преди всичко на разработчиците и потребителите на алгоритмите, като илюстрира принципа им на работа.

3. ПРЕДИМСТВА И НЕДОСТАТЪЦИ НА CPPP

При обработка на думи с повтарящи се букви или срички, CP ги проектира в едни и същи точки на пространството. CPPP не прави така, всеки символ се проектира в уникална за него част от кодиращото пространство. Пряко следствие от това е, че качеството на CPPP кодирането не е чувствително към дължината на стринговете.

Докато при CP е възможно да се насити кодиращото пространство, ако се обработи прекалено дълъг стринг, модифицираният алгоритъм може да трансформира цели изречения, пасажи от текст и дори документи, без да се наруши разредеността на точките в кодиращото пространство. Това води до следните значителни предимства:

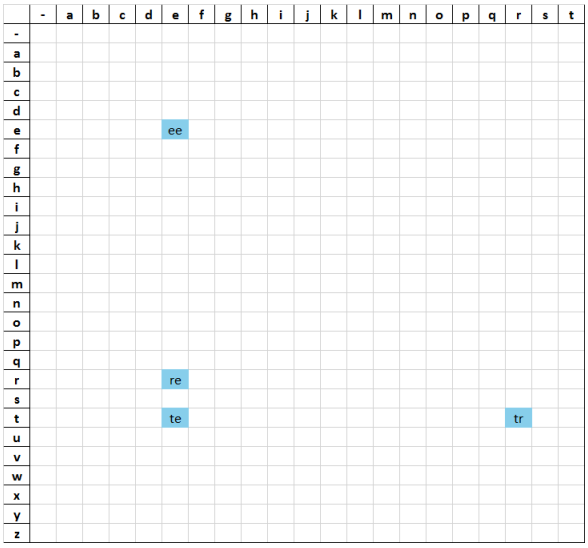
- **Обратимо кодиране**. Възможно е проекциите в кодиращото пространство да се преобразуват отново в стрингове, без загуба на информация.
- **Няма нужда от токенизация**. Текстовите документи (данните) могат да се разглеждат като непрекъсната монолитна поредица от символи, не е нужно да се разделят, чрез интервали, запетаи и подобни разделители.
- **По-голям капацитет**, благодарение на вторичните оси, следящи позициите на изчетените символи. Възможно е не само сравнение на думи, но и сравнение на цели фрази, изречения и параграфи.
- **По-прецизно сравнение на стрингове**. CPPP позволява не само да се определи степента на еднаквост между два стринга, възможно е и да се определи точно в кои части си приличат и в кои се различават.

Въпреки предимствата, при модифицираният алгоритъм възниква следният казус: простата методология за сравнение на стрингови образи, използвана от CP, е неприложима при CPPP.

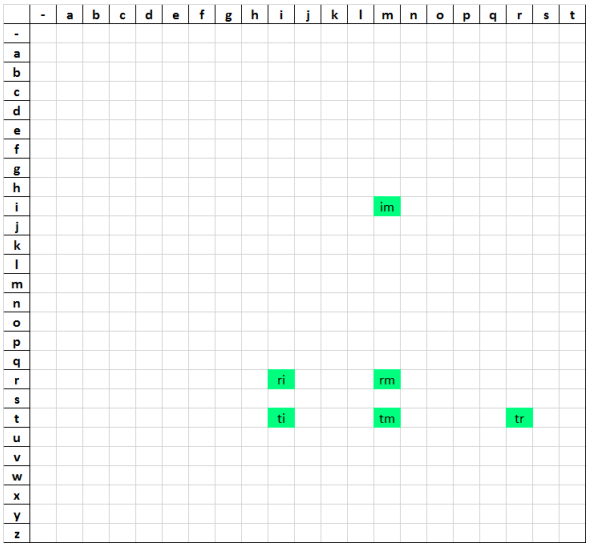
За да се онагледи причината, трябва само да се разгледа кодирането на няколко думи от двата алгоритъма. Подбраните думи са **tree**, **trim** и **tree-trimmer**, защото тези думи са къси, имат дублиращи се букви и третата е съставена от първите две.

При CP кодиране (фиг. 4-6) следните неща не се отразяват правилно:

- Двойното **e** в tree и двойното **m** в trimmer. И в двата случая се прави презаписване в едни и същи точки. За tree точките са **re** и **te**. За trimmer точките са **im**, **rm** и **tm**.
- Повторението на **tr** в tree-trimmer. Информацията за тази последователност от символи се проектира в една единствена точка. Това означава, че се губи и първоначалният стринг не може да се възстанови от образа му в кодиращото пространство.



***Фиг. 4** – tree (CP кодирана)*

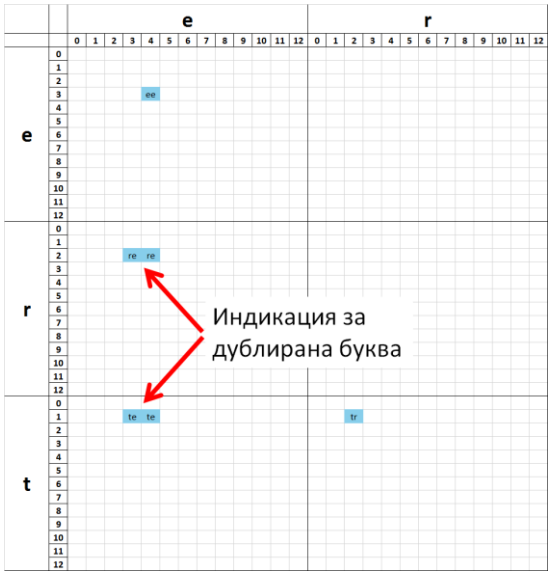


***Фиг. 5** – trim (CP кодирана)*

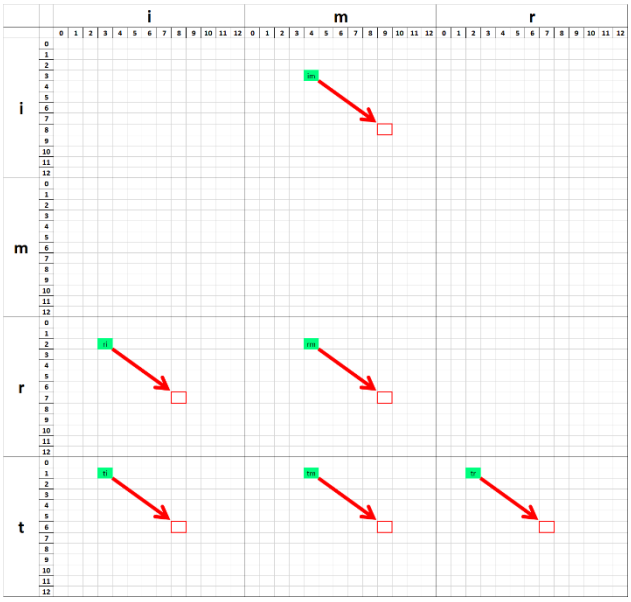
	-	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
-						-e				-i				-m					-r		-t						
a																											
b																											
c																											
d																											
e	e-					ee				ei				em					er		et						
f																											
g																											
h																											
i						ie								im					ir								
j																											
k																											
l																											
m						me													mr								
n																											
o																											
p																											
q																											
r	r-					re				ri				rm					rr		rt						
s																											
t	t-					te				ti				tm					tr		tt						
u																											
v																											

***Фиг. 6** – tree-trimmer (CP кодирана)*

Първото нещо, което усложнява сравняването на CPPP кодирани стрингове, е това че повтарящите се символи създават огледални точки в пространството. На фиг.7 със стрелки е посочен пример за такива точки. Те дават индикация, че буквата **e** се среща повече от веднъж в думата **tree**. Това, че в секторите **re** и **te** има повече от една ненулева стойност е информативно, но обработката на тази информация алгоритмично не е тривиална процедура, като процедурата за сравнение, използвана при CP кодиран текст.



***Фиг. 7** – tree (CPPP кодирана)*



***Фиг. 8** – trim (CPPP кодирана)*

***Фиг. 9** – tree-trimmer (CPPP кодирана)*

Второто усложнение при CPPP кодирането е илюстрирано на фиг. 8-9. Точките отразяващи думата **trim** се изместват в пространството. Това е разбираемо, защото в третият стринг, има друга дума преди **trim**, съответно проекциите на буквите са изместени надясно. Отново, това е логично и информативно, но алгоритмичният му анализ е по-сложен от това, което се прави при CP алгоритъма.

Тези две характеристики на модифицираното кодиращо пространство (огледалните точки и нестационарните проекции) са нещата, които му дават изброените по-рано предимства. Освен това, обаче, не съществува техен аналог при CP кодирането и разработеният за него метод за сравнение не е способен да ги отрази. Нужно е да се използват други техники за сравнение, за да се усвои потенциалът на четиримерното кодиране.

4. ДЕТЕРМИНИСТИЧНИ МЕТОДИ ЗА СРАВНЕНИЕ НА CPPP КОДИРАН ТЕКСТ

От компютърна програма, обработваща естествени езици, може да се очакват само следните възможности:

- откриване на частично или пълно съвпадащи стрингове.
- количествена оценка на честотни и структурни зависимости.
- бърза обработка на голям обем информация.

Ако даден казус, свързан с анализ на текст, може да се формулира по подходящ за машинна обработка начин, компютрите могат да бъдат полезни и да предоставят адекватни решения. На теория е възможно кодираните стрингове да се подават директно към изкуствена невронна мрежа. Тя би трябвало да е способна да открие същите взаимовръзки, като тези от примерите посочени в трета точка. Възможно е, обаче, и да се използват процедури, които не са черна кутия. Описаният в тази точка подход е пример за това.

4.1. Постановка за онагледяване на принципът на действие на детерминистичните методите за сравнение на CPPP кодиран текст

Използван е текстът на стихотворението “Поточе”, автор Елин Пелин. Не е прекалено дълго, затова е подходящо за илюстративни цели. Целият текст за анализ е кодиран чрез CPPP. Освен това, компютърът има достъп до машино-четим речник, в който са изброени отделни думи, също кодирани. Речникът е съставен само от думите на това стихотворение. Може да е съставен и от хиляди думи, но няма смисъл да бъде толкова изчерпателен.

Трябва да се отбележи, че има разликата, между кодирането на речника и данните. Стринговете в речника са кодирани отделно един от друг, но съвкупността от думи в данните се обработва като монолитен стринг. По дефиниция, не се знае кои части от текста съответстват на коя дума преди анализа и за разлика от общоприетата практика, текстът не се токенизира (разделя на части чрез интервали, запетаи и други делители). От една страна няма нужда, щом е CPPP кодиран. От друга страна, разграничаването на думи по такъв тривиален начин е прибързано и предполага, че това което се обработва е добре форматиран текст.

Последното нещо, което трябва да се изясни е, че преди да се определят: коя дума колко често се среща, в какъв контекст се използва, кое от всичките и значения е правилно и прочие, трябва въпросните думи да се открият в текста. Консистентното и качествено търсене (индексиране) е базова стъпка, която се отразява силно върху другите изброени операции. Изложеният в тази точка пример касае само операцията търсене, останалите са извън обхвата на статията.

4.2. Локализиране на думи в обработваният текст

За целта се обхожда всеки елемент в машино-четимият речник (дума) и за всеки от тях се извършват четири ключови стъпки.

4.2.1. Създава се тегловен вектор W

Размерността на този вектор е равна на бройката символи в текста. Инициализира се с всички елементи 0, за всяко търсене на дума. W има за цел да посочва в кои части от текста има силно съвпадение за търсеният стринг.

4.2.2. Проверка за сходни стрингови образи

Ако разгледаме кодираната дума показана на фиг.8, може да използваме символните координати, за да разделим кодиращото пространство на сектори. В рамките на всеки сектор се изчислява близостта Δp на символните чифтове, като разлика между позиционните им координати. Компактно представяне на това е показано в таблица 1. Аналогичен запис, за кодираната дума от фиг.9, е показан в таблица 2. Не всички точки за табл.2 са въведени, от значение са само тези, намиращи се в общите за двата стринга сектори.

Табл. 1 – **trim** (CPPP кодирана)

Сектор	Δp
ri	3 – 2 = 1
ti	3 – 1 = 2
im	4 – 3 = 1
rm	4 – 2 = 2
tm	4 – 1 = 3
tr	2 – 1 = 1

Табл. 2 – **tree-trimmer** (CPPP кодирана)

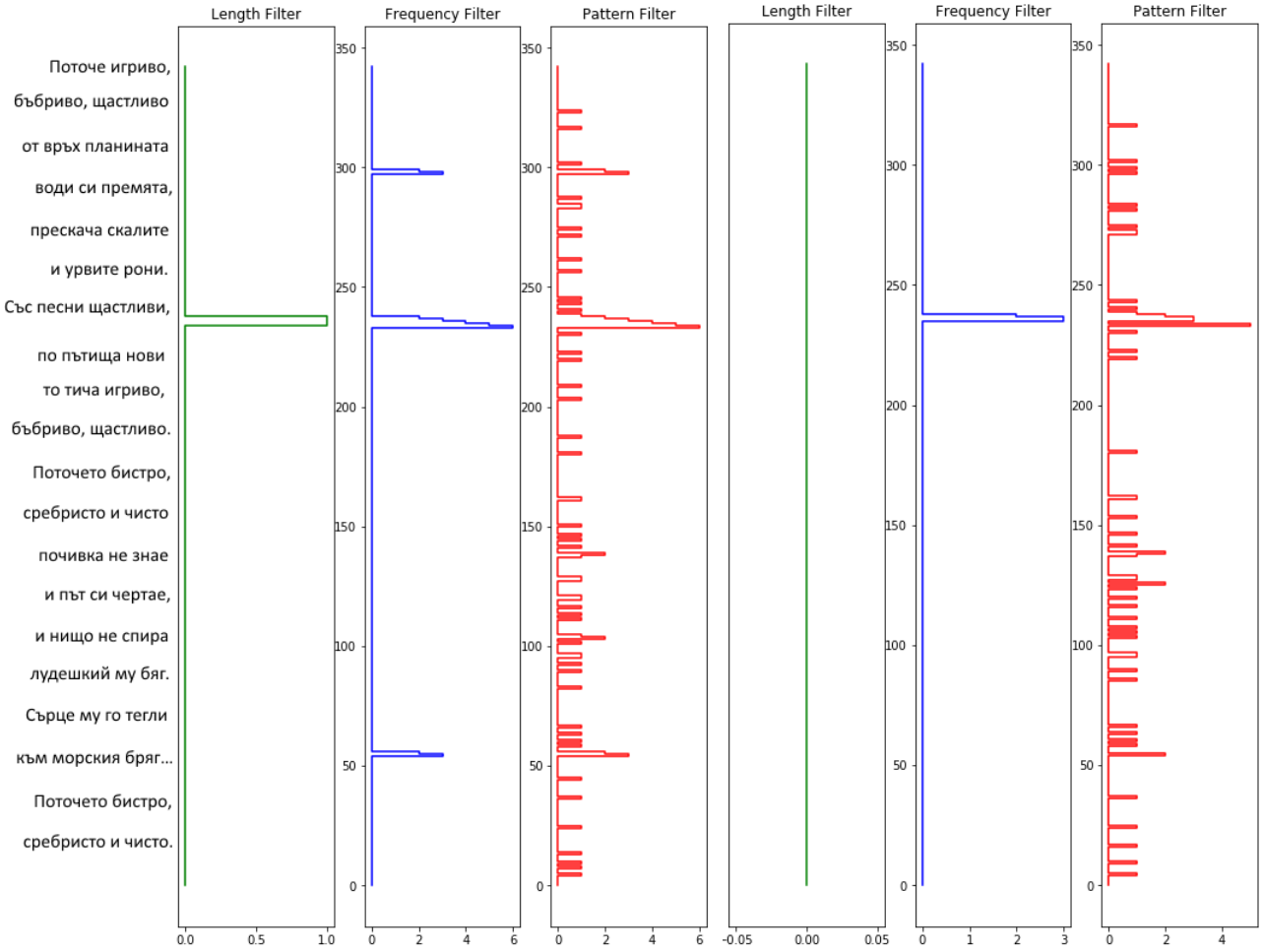
Сектор	Δp	Промяна в W
ri	8 – 2 = 6	
ri	8 – 7 = 1	W_8 += 1
ti	8 – 1 = 7	
ti	8 – 6 = 2	W_8 += 1
im	9 – 8 = 1	W_9 += 1
im	10 – 8 = 2	
rm	9 – 2 = 7	
rm	9 – 7 = 2	W_9 += 1
rm	10 – 2 = 8	
rm	10 – 7 = 3	
tm	9 – 1 = 8	
tm	9 – 6 = 3	W_9 += 1
tm	10 – 1 = 9	
tm	10 – 6 = 4	
tr	2 – 1 = 1	W_2 += 1 **грешка**
tr	7 – 1 = 6	
tr	7 – 6 = 1	W_7 += 1
tr	12 – 1 = 11	
tr	12 – 6 = 6	

Нека предположим, че стрингът от Табл. 1 е произволна дума от речника, а стрингът от Табл. 2 е текстът.

Тогава, за да попълним тегловният вектор W, взимаме точките от даден сектор на речниковата дума (по една точка в този случай) и търсим тези точки в съответните сектори на текста, чиито Δp е най-близко.

4.2.3. Филтриране на слаби съвпадения

Алгоритъмът описан в подточка 4.2.2 е използван, при търсенето на думата **песни** в стихотворението. На фиг. 10 се вижда резултатът (Pattern Filter), който се подлага на допълнителна обработка. Ако всички тегла по-малки от 2 се приемат за шум и се махнат от вектор W, тогава се получава по-сигурна оценка, за местоположението на думата в текста (фиг. 10 - Frequency Filter).



Фиг. 10** – търсене на **песни

Фиг. 11** – търсене на **песен

Понеже описаният алгоритъм търси само частични съвпадения, той може да направи връзка и между две букви от сравняваните стрингове. В някои случаи, ако се премахнат прекалено къси съвпадения, резултатът може да се подобри (фиг. 10 – Length Filter), но това не е гарантирано. Например, ако търсената дума не е **песни**, а базовата форма **песен**, както е по-вероятно да бъде съхранена в речника, тогава силата на съвпадението спада до толкова, че филтърът за дължина нулира всичко в тегловния вектор (фиг. 11).

Работата на тази процедура е да открие всички възможни позиции на дадена дума в текста. Допустимо е да открие повече инстанции от колкото има в действителност, но не е допустимо да изтърва действителните локации. Затова се налага филтрите да са динамични, вместо да ползват фиксирани параметри. Това е все още в процес на разработка, обаче.

4.3. Прецизно сравняване на стрингове

Процедурата описана в точка 4.2 трябва да открива потенциални думи, но сама по себе си не е достатъчно точна, за да се разчита изцяло на нея. Преди всичко тя трябва с малко изчислителни ресурси да прави стотици хиляди сравнения, докато проверява данните за всяка въведена в речника дума. В последствие се прави по-точно сравнение, между субстринговете от текста, белязани като потенциални думи, и предполагаемите им аналози от речника. Прецизната процедура, обаче, е по-сложен алгоритъм, илюстрирането на който би излязло извън рамките на статията.

5. ЗАКЛЮЧЕНИЕ

Модифицираният алгоритъм CPPP има значителни предимства пред базовият CP алгоритъм за кодиране на текстови данни. Най-значими от тях са драстично по-високият капацитет на кодиращото пространство, премахването от нуждата за токенизация на текстови пасажи и възможността да се кодират не само думи, но и фрази, изречения, пасажи от текст, както и цели документи. Тоест, чрез CPPP може да се кодира цял корпус.

Предизвикателствата, които вървят заедно с ползите от модифицираният алгоритъм, са свързани с разработката на по-сложни алгоритми и методи, които са способни да извлекат потенциалните ползи от CPPP кодирането на текст.

ЛИТЕРАТУРА

[1] Годишник на Технически Университет-София, том 68, книга 2, 2018, http://proceedings.tu-sofia.bg/

[2] M. Marinov, A. Efremov, Representing Character Sequences as Sets: A simple and intuitive string encoding algorithm for NLP data cleaning, 2019 IEEE International Conference on Advanced Scientific Computing (ICASC), Romania, 2019, pp. 1-6.

Автор: *Мартин Маринов*, маг. инж., Технически университет-София, Факултет Автоматика, катедра „Автоматизация на непрекъснатите производства”,
e-mail: mu_marinov@abv.bg

Author: *Martin Marinov*, Technical University of Sofia, Faculty of Automatics, department Industrial automation,
e-mail: mu_marinov@abv.bg