Every Bit Counts: A New Version of Non-binary VT Codes with More Efficient Encoder

Tuan Thanh Nguyen*, Kui Cai*, and Paul H. Siegel†

*Singapore University of Technology and Design, Singapore 487372
†University of California, San Diego, La Jolla, CA 92093, USA
Emails: {tuanthanh_nguyen, cai_kui}@sutd.edu.sg, psiegel@ucsd.edu

***Abstract*—In this work, we present a new version of non-binary VT codes that are capable of correcting a single deletion or single insertion. Moreover, we provide the first-known linear-time algorithms that encode user messages into these codes of length n over the q-ary alphabet for $q \geqslant 2$ with at most $\lceil \log_q n \rceil + 1$ redundant symbols, while the optimal redundancy required is at least $\log_q n + \log_q(q-1)$ symbols. Our designed encoder reduces the redundancy of the best known encoder of Tenengolts (1984) by at least $2 + \log_q(3)$ redundant symbols, or equivalently $2\log_2 q + 3$ redundant bits.**

I. Introduction

Codes correcting deletions and insertions are important for many data storage systems such as the bit-patterned media magnetic recording systems [1] and racetrack memory devices [2]. Insertions and deletions may also occur due to the synchronization errors in communication systems [3] and mobile data [4]. Furthermore, the problem of correcting such errors has recently received significantly increased attention due to the DNA-based data storage technology, which suffers from deletions and insertions with extremely high probability [5]–[9]. Designing codes for correcting deletions or insertions is well-known to be a challenging problem, even in the most fundamental settings with only a single error.

Over the q-ary alphabet, $q \geqslant 2$, consider a channel model that suffers from at most one deletion or one insertion, and suppose that the optimal redundancy required to correct such errors is $\mathrm{r}_{\mathrm{opt}}$, then two crucial coding theory problems are:

P1: Code Design. Can one design the largest possible code $\mathcal{C}$, with the redundancy $\mathrm{r}_{\mathcal{C}}$, such that $\mathrm{r}_{\mathcal{C}} \to \mathrm{r}_{\mathrm{opt}}$?

P2: Encoder/Decoder Design. Can one design an efficient encoder ENC (and a corresponding decoder DEC) that encodes arbitrary user messages into codewords in $\mathcal{C}$ with nearly-optimal redundancy $\mathrm{r}_{\mathrm{ENC}}$, $\mathrm{r}_{\mathrm{ENC}} \to \mathrm{r}_{\mathcal{C}}$?

While the problems of giving nearly-optimal explicit constructions of codes (P1) and designing nearly-optimal encoders for such codes (P2) over the binary alphabet have been settled for more than 50 years, the approach fails to be extended to the case of q-ary alphabet for any fixed $q > 2$ (refer to Table I for a summary of literature results). In particular, to correct a single deletion or single insertion, we have the celebrated class of Varshamov-Tenengolts (VT) codes. In 1965, Varshamov and Tenengolts introduced the binary VT codes to correct asymmetric errors [10], and Levenshtein subsequently showed that such codes can be used for correcting a deletion or insertion with a simple linear-time decoding algorithm [11]. For codewords of length n, the binary VT codes incur $\log_2(n+1)$ redundant bits, while the optimal redundancy, provided in [11], is at least $\log_2 n$ bits. Curiously, even though the binary VT codes and efficient decoding algorithm was known since 1965, a linear-time encoder for such codes was only proposed by Abdel-Ghaffar and Ferriera in 1998 [12], which used $\lceil \log(n+1) \rceil$ redundant bits. We observe that, over the binary alphabet, (P1) and (P2) are solved asymptotically optimal:

$$\mathrm{r}_{\mathrm{opt}} \geqslant \log_2 n, \ \mathrm{r}_{\mathcal{C}} = \log_2(n+1), \text{and } \mathrm{r}_{\mathrm{ENC}} = \lceil \log_2(n+1) \rceil.$$

For the non-binary alphabet, in 1984, a non-binary version of the VT codes was proposed by Tenengolts [13], and the constructed codes can correct a single deleted or inserted symbol in the q-ary alphabet with a linear-time decoder for any $q > 2$. The construction of Tenengolts retains the attractive properties of the binary VT codes, such as the simple decoding algorithm. For codewords of length n, such codes incur at most $\log_q n + 1$ redundant symbols. In the same paper, Tenengolts also provided an upper bound for the cardinality of any q-ary codes of length n correcting a deletion or insertion, which is at most $q^n/((q-1)n)$, and hence, the minimum redundancy required is at least $\log_q n + \log_q(q-1)$ symbols. Unlike the binary case, designing an efficient encoder that encodes arbitrary user messages into Tenengolts' code is a challenging task (refer to Section III-A for detailed discussion). To overcome the challenge, several attempts have been made in three variations:

- *Targeting a specific value of q.* When $q = 4$, Chee *et al.* [14] presented a linear-time quaternary encoder that corrects a single deletion or insertion with $\lceil \log_4 n \rceil + 1$ redundant symbols. The redundancy is asymptotically optimal. Unfortunately, the approach fails to be extended to the case of q-ary alphabet for arbitrary $q > 2$.

Alphabet size	Optimal Redundancy	Redundancy of the Largest Code $\mathcal{C}$	Redundancy of the best encoder for $\mathcal{C}$
Binary Σ_2	$\mathrm{r}_{\mathrm{opt}} \geqslant \log_2 n$	$\log_2(n+1)$ Levenshtein [11]	$\lceil \log_2(n+1) \rceil$ Abdel-Ghaffar and Ferriera [12]
Non-binary Σ_q	$\mathrm{r}_{\mathrm{opt}} \geqslant \log_q n + \log_q(q-1)$	$\log_q n + 1$ Tenengolts [13]	$> \log_q n + \log_2 n$ Abroshan *et al.* [15]

TABLE I: Related works for binary/non-binary codes correcting a single deletion or single insertion. Redundancy is measured in bits for binary codes, and in symbols for non-binary codes.

- *Using more redundancy.* Abroshan *et al.* [15] presented a systematic encoder that maps user messages into a single q-ary VT code as constructed in [13] with complexity that is linear in the code length. Unfortunately, the redundancy of this encoder is more than $\log_q n + \log_2 n$ symbols (see Section II).
- *Relaxing the condition for output codewords*. In [13], Tenengolts provided a systematic encoder that requires $\lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$ symbols, which is the best-known encoder for codes that correct a single deletion or insertion. In term of redundancy, a natural question is: can one construct a linear-time encoder with at most r redundant symbols, where $\log_q n + \log_q(q-1) \leqslant r < \lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$? In addition, The drawback of the encoder in [13] is that the codewords obtained from this encoder are not contained in a single q-ary VT code. Note that to correct a single deletion or insertion, it is not necessary that all the codewords must belong to the same coset of q-ary VT codes. Nevertheless, when the words share the same parameters, Abroshan *et al.* [15] demonstrated that these codes can be adapted to correct multiple insertion/deletion errors, in the context of *segmented edits* [16]–[18].

Motivated by the code design problem above, we present a new version of non-binary VT codes that give asymptotically optimal solutions for (P1) and (P2), as best as over binary alphabet, summarized as follows:

$$\mathrm{r}_{\mathrm{opt}} \geqslant \log_q n + \log_q(q-1), \mathrm{r}_{\mathcal{C}} = \log_q n + 1, \text{and } \mathrm{r}_{\mathrm{ENC}} = \lceil \log_q n \rceil + 1.$$

Our code construction method supports both systematic and non-systematic linear-time encoders that encode user messages into these codes. Our constructed codes have the same cardinality and redundancy, as compared to the best known q-ary single deletion/insertion codes constructed by Tenengolts [13]. On the other hand, our proposed code construction method supports more efficient encoding and decoding procedures (in other words, it enables an easier method to solve (P2)). Consequently, our best encoder uses at most $\lceil \log_q n \rceil + 1$ redundant symbols, and hence, it reduces the redundancy of the best known encoder of Tenengolts [13] by at least $2 + \log_q(3)$ redundant symbols, or equivalently $2 \log q + 3$ redundant bits.

II. Preliminary

Let Σ_q denote an *alphabet* of size q. For any positive integer $m < n$, we let $[m, n]$ denote the set $\{m, m+1, \ldots, n\}$ and $[n] = [1, n]$.

Given two sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, we let $\boldsymbol{xy}$ denote the *concatenation* of the two sequences. In the special case where $\boldsymbol{x}, \boldsymbol{y} \in \Sigma_q^n$, we use $\boldsymbol{x}||\boldsymbol{y}$ to denote their *interleaved sequence* $x_1 y_1 x_2 y_2 \ldots x_n y_n$. For a subset $I = \{i_1, i_2, \ldots, i_k\}$ of coordinates, we use $\boldsymbol{x}|_I$ to denote the vector $x_{i_1} x_{i_2} \ldots x_{i_k}$. A sequence $\boldsymbol{y}$ is said to be a *subsequence* of $\boldsymbol{x}$, if there exists a subset of coordinates I such that $\boldsymbol{y} = \boldsymbol{x}|_I$. Let $\boldsymbol{x} \in \Sigma_q^n$. We define the error ball $\mathcal{B}^{\mathrm{InDel}}(\boldsymbol{x})$ to be the set of all sequences $\boldsymbol{y}$ that can be obtained from $\boldsymbol{x}$ via a single insertion or single deletion.

Definition 1. Let $\mathcal{C} \subseteq \Sigma_q^n$. We say that $\mathcal{C}$ corrects a single insertion or single deletion error if and only if $\mathcal{B}^{\mathrm{InDel}}(\boldsymbol{x}) \cap \mathcal{B}^{\mathrm{InDel}}(\boldsymbol{y}) = \varnothing$ for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}$.

For a code $\mathcal{C} \subseteq \Sigma_q^n$, the code rate is measured by the value $\mathrm{r}_{\mathcal{C}} = \log_q |\mathcal{C}|/n$. In this work, not only are we interested in constructing large error-correction codes, we desire efficient encoder that maps arbitrary user data into these codes.

Definition 2. The map $\textsc{Enc} : \Sigma_q^k \to \Sigma_q^n$ is a *single-deletion-insertion-encoder* if there exists a *decoder* map $\textsc{Dec} : \Sigma_q^{n+1} \cup \Sigma_q^n \cup \Sigma_q^{n-1} \to \Sigma_q^k$ such that the following conditions hold:

- For all $\boldsymbol{x} \in \Sigma_q^k$, we have $\textsc{Dec} \circ \textsc{Enc}(\boldsymbol{x}) = \boldsymbol{x}$,
- If $\boldsymbol{c} = \textsc{Enc}(\boldsymbol{x})$ and $\boldsymbol{c}' \in \mathcal{B}^{\mathrm{InDel}}(\boldsymbol{c})$, then $\textsc{Dec}(\boldsymbol{c}') = \boldsymbol{x}$.

Hence, we have that the code $\mathcal{C} = \{\boldsymbol{c} : \boldsymbol{c} = \textsc{Enc}(\boldsymbol{x}), \boldsymbol{x} \in \Sigma_q^k\}$ and $|\mathcal{C}| = q^k$. The *redundancy of the encoder* is measured by the value $n - k$ (in symbols) or $(n-k)\log_2 q$ (in bits).

We now introduce the binary VT codes [10], [11].

Definition 3. The *VT syndrome* of a binary sequence $\boldsymbol{x} \in \{0,1\}^n$ is defined to be $\mathrm{Syn}(\boldsymbol{x}) = \sum_{i=1}^n i x_i$.

Construction 1 (Binary VT codes [10])**.** *For $a \in \mathbb{Z}_{n+1}$, let*

$$\mathrm{VT}_a(n) = \{\boldsymbol{x} \in \{0,1\}^n : \mathrm{Syn}(\boldsymbol{x}) = a \pmod{n+1}\}. \quad (1)$$

Theorem 1 (Levenshtein, 1965 [11])**.** *For $a \in \mathbb{Z}_{n+1}$, $\mathrm{VT}_a(n)$ can correct a single deletion or a single insertion. There exists $a \in \mathbb{Z}_{n+1}$ such that $\mathrm{VT}_a(n)$ has at least $2^n/(n+1)$ codewords, and the redundancy of the code is at most $\log_2(n+1)$ bits.*

Over the nonbinary alphabet, in 1984, Tenengolts [13] generalized the binary VT codes to q-ary VT codes for any fixed q-ary alphabet. Crucial to the construction of Tenengolts in [13] was the concept of the *signature vector* defined as follows.

Definition 4. The *signature vector* of a q-ary vector $\boldsymbol{x}$ of length n is a binary vector $\alpha(\boldsymbol{x})$ of length $n-1$, where $\alpha(x)_i = 1$ if $x_{i+1} \geq x_i$, and 0 otherwise, for $i \in [n-1]$.

Construction 2 (q-ary VT codes as proposed in [13])**.** *Given $n, q > 0$, for $a \in \mathbb{Z}_n$ and $b \in \mathbb{Z}_q$, set*

$$\mathrm{T}_{a,b}(n;q) \triangleq \Big\{\boldsymbol{x} \in \mathbb{Z}_q^n : \alpha(\boldsymbol{x}) \in \mathrm{VT}_a(n-1) \text{ and } \sum_{i=1}^n x_i = b \pmod{q}\Big\}.$$

Theorem 2 (Tenengolts, 1984 [13])**.** *The set $\mathrm{T}_{a,b}(n;q)$ forms a q-ary single deletion/insertion correction code and there exists a and b such that the size of $\mathrm{T}_{a,b}(n;q)$ is at least $q^n/(qn)$. There exists a systematic encoder $\textsc{Enc}_{\mathrm{T}}$ with redundancy $\lceil \log_2 n \rceil + 3\lceil \log_2 q \rceil + 3$ (bits) or $\lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$ (symbols).*

On the other hand, the codewords obtained from the encoder $\textsc{Enc}_{\mathrm{T}}$ are not contained in a single q-ary VT code $\mathrm{T}_{a,b}(n;q)$. Recently, Abroshan *et al.* [15] presented a systematic encoder that maps binary messages into $\mathrm{T}_{a,b}(n;q)$. Unfortunately, the redundancy of the encoder is as large as $\log_2 n(\log_2 q + 1) + 2(\log_2 q - 1)$ bits, and hence, more than $\log_2 n + \log_q n$ symbols.

A. Paper Organisation and Main Contributions

In Section III, we present a new version of non-binary VT codes that are capable of correcting a single deletion or a single insertion. Our decoding algorithm may be considered simpler than Tenengolts' method, and more importantly, our proposed code construction method supports more efficient encoding and decoding procedures.

In Section IV, we present a linear-time encoder that encodes user messages into the codes constructed in Section III. For codewords of length n over the q-ary alphabet, our designed encoder uses at most $\lceil \log_q n \rceil + 1$ redundant symbols. Our encoder can also enable more efficient design of the non-binary segmented edits correcting codes. The efficiency of our proposed encoders, compared to previous works, is illustrated in Table II.

III. A New Version of q-ary VT Codes

Note that any code that corrects k deletions if and only if it can correct k insertions, as established by Levenshtein [19]. Therefore, for simplicity, throughout this paper, we present the decoding algorithm to correct a deletion only. Crucial to our construction is the concept of q-ary *differential vector*.

Definition 5. Given $\boldsymbol{x} \in \Sigma_q^n$. The *differential vector* of $\boldsymbol{x}$, denoted by $\mathrm{Diff}(\boldsymbol{x})$, is a sequence $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) \in \Sigma_q^n$ where:

$$\begin{cases} y_i &= x_i - x_{i+1} \pmod q, \text{ for } 1 \leqslant i \leqslant n-1, \\ y_n &= x_n. \end{cases}$$

Clearly, $\mathrm{Diff}(\boldsymbol{x})$ is a one-to-one function. From $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$, we can obtain $\boldsymbol{x} = \mathrm{Diff}^{-1}(\boldsymbol{y})$ as follows.

$$\begin{cases} x_n &= y_n, \text{ and} \\ x_i &= \sum_{j=i}^{n} y_j \pmod q, \text{ for } n-1 \geqslant i \geqslant 1. \end{cases}$$

A. A Natural Idea from Binary VT Codes

Recall the design of the binary VT codes $\mathrm{VT}_a(n)$ from Construction 1 to correct a single deletion or insertion. A natural question is whether there exists a simple VT syndrome over q-ary codewords to correct single deletion or insertion for arbitrary $q > 2$. Observe that, in the construction of Tenengolts [13] (refer to Construction 2), the VT syndrome is enforced over the signature of each codeword, which is a binary sequence. That is a drawback leading to the difficulty of designing an efficient encoder as in the binary case. Consequently, to encode arbitrary messages into $\mathrm{T}_{a,b}(n;q)$ by enforcing the VT syndrome over the binary signature sequences, Abroshan *et al.* [15] required more than $\log_q n + \log_2 n$ redundant symbols. A natural solution should be obtained by enforcing a single VT syndrome over all q-ary sequences.

On the other hand, we observe that, imposing VT syndrome directly over every q-ary codeword is not sufficient to correct a deletion or insertion. For example, it is easy to verify that the following two sequences $\boldsymbol{z}1 = \boldsymbol{x}213\boldsymbol{y}$ and $\boldsymbol{z}2 = \boldsymbol{x}132\boldsymbol{y}$, where $\boldsymbol{x}, \boldsymbol{y}$ are arbitrary sequences, have the same VT syndrome, however, share a common sequence in the single error ball as $\boldsymbol{z}' = \boldsymbol{x}13\boldsymbol{y}$.

Surprisingly, imposing the VT syndrome over the differential vector of every q-ary codeword allows us to correct a deletion or an insertion, and that is the main contribution of this work.

B. Codes Construction

Lemma 1. *Given $\boldsymbol{x} \in \Sigma_q^n$ and let $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) \in \Sigma_q^n$. Suppose that $\boldsymbol{x}'$ is obtained via $\boldsymbol{x}$ by a deletion at symbol x_i for $1 \leqslant i \leqslant n$. We then have:*

(i) If $2 \leqslant i \leqslant n$, then $y_{i-1}y_i$ is replaced by $y_{i-1} + y_i \pmod q$,

(ii) If $i = 1$, then y_1 is deleted in $\mathrm{Diff}(\boldsymbol{x})$.

Proof. We have $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$, where $y_i = x_i - x_{i+1} \pmod q$ for $1 \leqslant i \leqslant n-1$ and $y_n = x_n$.

If $i = 1$, i.e. x_1 is deleted in $\boldsymbol{x}$, we then have $\boldsymbol{x}' = x_2x_3 \dots x_n$. Clearly, $\mathrm{Diff}(\boldsymbol{x}') = y_2y_3 \dots y_n$, or y_1 is deleted in $\mathrm{Diff}(\boldsymbol{x})$.

If $2 \leqslant i \leqslant n$, a deletion at x_i affects y_{i-1}, y_i in $\mathrm{Diff}(\boldsymbol{x})$, as $y_{i-1} = x_{i-1} - x_i \pmod q$ and $y_i = x_i - x_{i+1} \pmod q$. We observe that the change in $\mathrm{Diff}(\boldsymbol{x}')$ is then

$$\begin{aligned} \mathrm{Diff}(\boldsymbol{x}')_{i-1} &= x_{i-1} - x_{i+1} \pmod q \\ &= (x_{i-1} - x_i) + (x_i - x_{i+1}) \pmod q \\ &= y_{i-1} + y_i \pmod q. \end{aligned}$$

We conclude that $y_{i-1}y_i$ is replaced by $y_{i-1} + y_i \pmod q$. ■

Example 1. Consider $\Sigma_4 = \{0, 1, 2, 3\}$, and $\boldsymbol{x} = 0211301$. We then have $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) = 2102331$. Suppose that the symbol 2 is deleted in $\boldsymbol{x}$, resulting $\boldsymbol{x}' = 011301$, and $\mathrm{Diff}(\boldsymbol{x}') = 302331$. In this example, we observe that, x_2 is deleted in $\boldsymbol{x}$, and the resulting $y_1y_2 = 21$ in $\mathrm{Diff}(\boldsymbol{x})$ is replaced by $3 = y_1 + y_2$.

Construction 3 (New version of q-ary VT codes)**.** *Given $n > 0$. For $q \geqslant 2, a \in \mathbb{Z}_{qn}$, set*

$$\mathrm{VT}_a^*(n;q) \triangleq \left\{\boldsymbol{x} \in \Sigma_q^n : \mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x})) = a \pmod{qn}\right\}.$$

The following lemma is crucial to the correctness of our error-decoding algorithm.

Lemma 2 (Parity check lemma)**.** *Given $n > 0$, $q \geqslant 2$, and $a \in \mathbb{Z}_{qn}$. Consider $\boldsymbol{x} \in \Sigma_q^n$ such that $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x})) = a \pmod{qn}$. We then have $\sum_{i=1}^{n} x_i \equiv a \pmod q$.*

Proof. Let $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$, where $y_i = x_i - x_{i+1} \pmod q$ for $1 \leqslant i \leqslant n-1$ and $y_n = x_n$. Suppose that $\mathrm{Syn}(\boldsymbol{y}) = a + kqn$ for some positive integer k. We have

$$\begin{aligned} \mathrm{Syn}(\boldsymbol{y}) &= \sum_{i=1}^{n-1} iy_i + ny_n \\ &\equiv \sum_{i=1}^{n} i(x_i - x_{i+1}) + nx_n \pmod q \\ &\equiv \sum_{i=1}^{n} x_i \pmod q. \end{aligned}$$

Since $\mathrm{Syn}(\boldsymbol{y}) = a + kqn$, it implies $\sum_{i=1}^{n} x_i \equiv a \pmod q$. ■

Encoder	Redundancy (in symbols)	Encoding/Decoding Complexity	Receiver Information on Code's Parameters	Encoder Output	Remark
Encoder $\mathrm{ENC_T}$ proposed by Tenengolts [13] using $\mathrm{T}_{a,b}(n;q)$	$\lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$	$O(n)$	not available	not in $\mathrm{T}_{a,b}(n;q)$	systematic
Encoder $\mathrm{ENC_A}$ proposed by Abroshan *et al.* [15] using $\mathrm{T}_{a,b}(n;q)$	$> \log_q n + \log_2 n$	$O(n)$	VT Syndrome (a) and parity check (b)	in $\mathrm{T}_{a,b}(n;q)$	systematic
Encoder ENC_1 proposed in this work using $\mathrm{VT}_a^*(n;q)$	$\lceil \log_q n \rceil + 3 + \lceil \log_q 3 \rceil$	$O(n)$	not available	not in $\mathrm{VT}_a^*(n;q)$	systematic
Encoder ENC_2 proposed in this work using $\mathrm{VT}_a^*(n;q)$	$\lceil \log_q n \rceil + 1$	$O(n)$	VT Syndrome (a) and parity check (a)	in $\mathrm{VT}_a^*(n;q)$	non-systematic

TABLE II: Efficient encoders for q-ary codes correcting single deletion or insertion proposed in this work and and those in literature. For each design category, the most desirable option is highlighted in blue. Particularly, our proposed encoder ENC_2 incurs the least redundancy of $\lceil \log_q n \rceil + 1$ symbols. Here, the receiver information on code's parameters plays an important role in error-detecting and error-correcting procedure. For example, it may provide more efficient basis for the design of segmented deletion/insertion correcting codes (see [16]–[18]).

Theorem 3. *The code $\mathrm{VT}_a^*(n;q)$ can correct a single deletion or single insertion in linear time. In other words, there exists a linear-time decoder $\mathrm{DEC}_{\mathrm{error}} : \Sigma_q^{n-1} \cup \Sigma_q^{n+1} \to \Sigma_q^n$ such that if $\boldsymbol{x}'$ is obtained from $\boldsymbol{x} \in \mathrm{VT}_a^*(n;q)$ after a deletion or an insertion, we can recover $\boldsymbol{x} = \mathrm{DEC}_{\mathrm{error}}(\boldsymbol{x}')$. In addition, there exists $a \in \mathbb{Z}_{qn}$, such that $\left|\mathrm{VT}_a^*(n;q)\right| \geqslant \frac{q^n}{qn}$.*

Proof. Observe that the lower bound is verified by using the pigeonhole principle. It remains to show that the code $\mathrm{VT}_a^*(n;q)$ can correct a single deletion in linear time.

For a codeword $\boldsymbol{x} \in \mathrm{VT}_a^*(n;q)$, let $\boldsymbol{x}'$ be obtained from $\boldsymbol{x}$ after a deletion at x_i. According to Lemma 2, we can obtain the value of the deleted symbol as follows: $x_i = a - \sum_{j=1}^{n-1} x'_j \pmod q$.

It remains to determine the value of i, i.e. the location of the deleted symbol. Let $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x})$ and $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}')$. We then compute:

$$\Delta = \mathrm{syn}(\boldsymbol{y}) - \mathrm{Syn}(\boldsymbol{y}') = a - \mathrm{Syn}(\boldsymbol{y}') \pmod{qn}, \text{ and}$$
$$s = \sum_{j=1}^{n-1} y'_j, \text{ i.e. the sum of symbols in } \boldsymbol{y}'.$$

Observe that both a and $\boldsymbol{y}'$ are known, hence, the values of Δ and s can be determined.

Let s_R be the sum of symbols on the right of error symbol y_i in $\boldsymbol{y}$. We show how $\boldsymbol{y}$ can be recovered from $\boldsymbol{y}'$ and thus $\boldsymbol{x}$ can be recovered based on Δ and s, which are computable at the decoder. We now have the following cases.

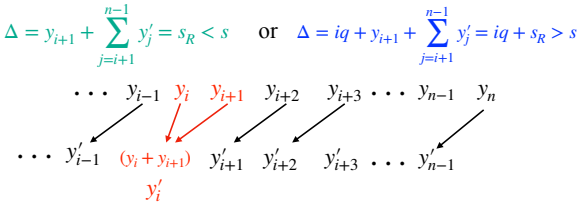
Case 1. If $i = 1$, we must have $\Delta = x_1 + \sum_{j=1}^{n-1} y'_j = x_1 + s$.

Case 2. If $2 \leqslant i \leqslant n$, according to Lemma 1, $y_{i-1}y_i$ is replaced by $y_{i-1} + y_i \pmod q$.

- (2a) If $y_i + y_{i+1} \leqslant q - 1$, then it is easy to verify that $\Delta = s_R < s$.
- (2b) If $q \leqslant y_i + y_{i+1} \leqslant 2(q-1)$, then it is easy to verify that $\Delta = iq + s_R > s$.

Therefore, given the computed values Δ and s, we can distinguish (2a) and (2b). Moreover, observe that both s_R and $iq + s_R$ are monotonic functions in the index i. Particularly, it is easy to verify that s_R is decreasing in the index i while $iq + s_R$ is increasing function in the index i. Hence, Δ is decreasing in the case (2a) while it is increasing in the case (2b). In other words, given the value of x_i, there is a unique value of i according to the value of Δ. It is easy to see that, in the case when the deleted symbol belongs to a run of identical symbols, we then have more than one options for the index i. Nevertheless, we obtain the same codeword. Consequently, to locate the error in $\boldsymbol{y}$, for (2a), the decoder scans $\boldsymbol{y}'$ and simply searches for the first index h where $\sum_{j=h}^{n-1} y'_j > \Delta$, while for (2b), the decoder scans $\boldsymbol{y}'$ and simply searches for the largest index h where $qh + \sum_{j=h}^{n-1} y'_j < \Delta$. The error location in $\boldsymbol{x}$ is then $i = h + 1$.

In conclusion, the code $\mathrm{VT}_a^*(n;q)$ can correct a single deletion (or equivalently, a single insertion) in linear time. ■



Claim: Given Δ and s, we can locate the deleted symbol x_i.

Remark 1. The advantage of our proposed codes design, besides the simplicity of the constraint, is that it enables $O(\log n)$ time to find the error location, since Δ is a monotonic function in the index i. In addition, one may construct $\mathrm{VT}_a^*(n;q)$ using different variations of the differential function $\mathrm{Diff}(\boldsymbol{x})$. In general, for all values p, $1 \leqslant p \leqslant q-1$ and $\gcd(p,q) = 1$, this coding method works for all *p-transformation vector* $\Gamma_p(\boldsymbol{x})$, defined as follows.

$$\begin{cases} y_i & = p(x_i - x_{i+1}) \pmod q, \text{ for } 1 \leqslant i \leqslant n-1, \\ y_n & = px_n. \end{cases}$$

Another variation of the differential vector was used in [20], [21] for binary codes to correct a burst of at most two deletions.

Example 2. Given $n = 10, q = 4, a = 0$, $\Sigma_4 = \{0,1,2,3\}$. Consider a codeword $\boldsymbol{x} = 0103112013 \in \mathrm{VT}_a^*(n;q)$. We obtain $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{x}) = 3112032323$. It is easy to verify that $\mathrm{Syn}(\boldsymbol{y}) = 120 \equiv 0 \pmod{40}$ and $\sum_{i=1}^{10} x_i \equiv 0 \pmod 4$.

Suppose that we receive $\boldsymbol{x}' = 013112013$, i.e. a deletion occurs at $x_3 = 0$. We then obtain $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}') = 322032323$. Now, to correct $\boldsymbol{x}$ and find out the value of i, we follow the decoding procedure in Theorem 3 as follows.

- From $\boldsymbol{x}'$, the decoder finds the value of the deleted symbol, which is $a - \sum_{i=1}^{n-1} x'_i = 0 \pmod 4$.
- From $\boldsymbol{y}' = \mathrm{Diff}(\boldsymbol{x}') = 322032323$, the decoder computes:

$$\Delta = a - \mathrm{Syn}(\boldsymbol{y}') = 0 - 104 = 16 \pmod{40},$$
$$s = \sum_{i=1}^{n-1} y'_i = 3+2+2+3+2+3+2+3 = 20.$$

- Since $\Delta < s$, the decoder concludes that it belongs to the case (2a) where the deletion is not at the first position, i.e. $i \neq 1$, and $y_{i-1} + y_i < q = 4$.
- Find the error location in $\boldsymbol{y}$. It can be observed that $\sum_{h=2}^{9} y'_i = 17 > \Delta = 16$ while $\sum_{h=3}^{9} y'_i = 15 < \Delta$. The decoder then concludes that the error in $\boldsymbol{y}$ is at the $h = 2$ position, and hence, the error in $\boldsymbol{x}$ is at $i = h + 1 = 3$.
- To correct $\boldsymbol{x}$, it inserts the symbol 0 to the third position.

C. Systematic Encoder

It is easy to show that our constructed codes $\mathrm{VT}_a^*(n;q)$, from Construction 3, also support systematic linear-time encoder. The design is similar to the construction of the systematic encoder proposed by Tenengolts [13]. For message $\boldsymbol{x} \in \Sigma_q^k$, the encoder appends the information of the VT syndrome of the differential vector of $\boldsymbol{x}$ (of length $t+1 = \lceil \log_q k \rceil + 1$) into its suffix. In addition, there is a marker of length two, serves as separators between the data part and the redundancy part (refer to [13]). We illustrate the main idea of the encoder as follows.

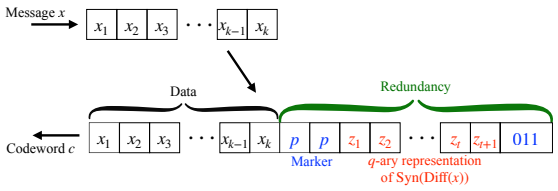


Fig. 1: Systematic encoder ENC_1. Here $t = \lceil \log_q k \rceil$. The combination 011 at the end of the code sequence plays the role of the comma between transmitted sequences. The marker pp, where $p = x_k + 1 \pmod q$ serves as separators between the data part and the redundancy part, while the $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{x}))$ is represented by $z_1 z_2 \ldots z_t z_{t+1}$.

IV. More Efficient Encoder and Decoder

In this section, we present a linear-time encoder that encodes user data into $\mathrm{VT}_a^*(n;q)$ with $\lceil \log_q n \rceil + 1$ redundant symbols.

Encoder 2. Given n, q, and $a \in \mathbb{Z}_{qn}$, set $t \triangleq \lceil \log_q n \rceil$ and $k \triangleq n - t - 1$. The user message is of length k.

INPUT: $\boldsymbol{x} \in \Sigma_q^m$
OUTPUT: $\boldsymbol{c} \triangleq \mathrm{ENC}_2(\boldsymbol{x}) \in \mathrm{VT}_a^*(n;q)$

(I) Set $S \triangleq \{q^{j-1} : j \in [t]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$. In other words, the set S includes the nth index and all the indices that are powers of q.
(II) Set $\boldsymbol{y} = y_1 y_2 \ldots y_n \in \Sigma_q^n$, where $\boldsymbol{y}|_I = \boldsymbol{x}$ and $\boldsymbol{y}|_S = 0$. In other words, the symbols in $\boldsymbol{x}$ are filled into $\boldsymbol{y}$ excluding indices in S (refer to Figure 2) and $y_j = 0$ for $j \in S$.
(III) Compute the difference $a' \triangleq a - \mathrm{Syn}(\boldsymbol{y}) \pmod{qn}$. In the next step, we modify $\boldsymbol{y}$, by setting suitable values for y_j where $j \in S$, to obtain $\mathrm{Syn}(\boldsymbol{y}) = a \pmod{qn}$. Since $0 \leqslant a' \leqslant qn - 1$, we find α, $0 \leqslant \alpha < q - 1$, to be the number such that $\alpha n \leqslant a' < (\alpha + 1)n$.
(IV) The values for y_j where $j \in S$ are set as follows.
- Set $y_n = \alpha$, and $a'' = a' - \alpha n < n$.
- Let $z_{t-1} \ldots z_1 z_0$ be the q-ary representation of a''. Clearly, since $a'' < n$, the q-ary representation of a'' is of length at most $t = \lceil \log_q n \rceil$. We then have $a'' = \sum_{i=0}^{t-1} z_i q^i$.
- Set $y_{q^{j-1}} = z_{j-1}$ for $j \in [t]$.

(V) Set $\boldsymbol{c} = \mathrm{Diff}^{-1}(\boldsymbol{y})$. In other words, we set $c_n = y_n$ and $c_i = \sum_{j=i}^{n} y_j \pmod q$ for $1 \leqslant i \leqslant n$.
(VI) Output $\boldsymbol{c}$.

Example 3. Consider $n = 10, q = 3$ and $a = 0$. Then $t = \lceil \log_3 10 \rceil = 3$ and $k = 10 - 3 - 1 = 6$. Suppose that the message is $\boldsymbol{x} = 220011$ and we compute $\mathrm{ENC}_2(\boldsymbol{x}) \triangleq \boldsymbol{c} \in \mathrm{VT}_0^*(10;3)$.

(I) Set $S = \{1, 3, 9, 10\}$ and $I = \{2, 4, 5, 6, 7, 8\}$.
(II) The encoder first sets $\boldsymbol{y} = y_1 2 y_3 20011 y_9 y_{10}$. It then sets $y_1 = y_3 = y_9 = y_{10} = 0$ to obtain $\boldsymbol{y} = 0202001100$ and computes $a' = a - \mathrm{Syn}(\boldsymbol{y}) = 0 - 27 = 3 \pmod{30}$.
(III) Since $0 < a' = 3 < 10$, the encoder sets $\alpha = 0$ and $a'' = a' = 3$. It then sets $y_{10} = \alpha = 0$.
(IV) The 3-ary representation of 3 is then 010. Therefore, the encoder sets $y_1 = 0$, $y_2 = 1$, and $y_9 = 0$ to obtain $\boldsymbol{y} = 0212001100$. We can verify that $\mathrm{Syn}(\boldsymbol{y}) = 0 \pmod{30}$.
(V) The encoder outputs $\boldsymbol{c} = \mathrm{Diff}^{-1}(\boldsymbol{y}) = 1121222100$.

Theorem 4. *Encoder 2 is correct and has redundancy* $\lceil \log_q n \rceil + 1$ *symbols. In other words,* $\mathrm{ENC}_2(\boldsymbol{x}) \in \mathrm{VT}_a^*(n;q)$ *for all* $\boldsymbol{x} \in \Sigma_q^{n - \lceil \log_q n \rceil - 1}$.

Proof. It suffices to show that $\mathrm{Syn}(\mathrm{Diff}(\boldsymbol{c})) = a \pmod{qn}$. From Step (V) of the Encoder 2, $c = \mathrm{Diff}^{-1}(\boldsymbol{y})$, in other words, $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{c})$. It remains to show that $\mathrm{Syn}(\boldsymbol{y}) = a \pmod{qn}$.

Recall that from Step (I) of Encoder 2, $S \triangleq \{q^{j-1} : j \in [t]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$. Therefore,

$$\begin{aligned}
\mathrm{Syn}(\boldsymbol{y}) &= \sum_{j \in S} j y_j + \sum_{j \in I} j y_j \pmod{qn} \\
&= \sum_{j \in [t]} q^{j-1} y_j + n y_n + \sum_{j \in I} j y_j \pmod{qn} \\
&= a'' + n\alpha + (a - a') \pmod{qn} \\
&= a' - \alpha n + n\alpha + a - a' \pmod{qn} \\
&= a \pmod{qn}
\end{aligned}$$

■

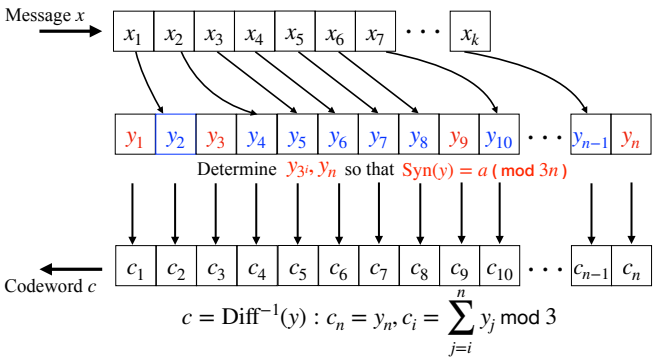


Fig. 2: Linear-time encoder to encode arbitrary messages into $\mathrm{VT}_a^*(n;q)$. In this example, $q = 3$ and the VT syndrome $\mathrm{Syn}(\boldsymbol{y})$ is computed in modulo $3n$ while each symbol is computed in modulo 3. The message is of length $k = n - \lceil \log_3 n \rceil - 1$.

Decoder 2. Given n, q, and $a \in \mathbb{Z}_{qn}$, $t \triangleq \lceil \log_q n \rceil$ and $k \triangleq n - t - 1$. Given $\boldsymbol{c} = \mathrm{ENC}_1(\boldsymbol{x})$ for some message $\boldsymbol{x} \in \Sigma_q^k$.
INPUT: $\boldsymbol{c}' \in \Sigma_q^{n-1} \cup \Sigma_q^n \cup \Sigma_q^{n+1}$
OUTPUT: $\boldsymbol{x} = \mathrm{DEC}_2(\boldsymbol{c}') \in \Sigma_q^k$

(I) The decoder follows the error-decoding procedure in Theorem 3 to obtain $\boldsymbol{c} \triangleq \mathrm{DEC}_{\mathrm{error}}(\boldsymbol{c}') \in \Sigma_q^n$.
(II) Set $\boldsymbol{y} = \mathrm{Diff}(\boldsymbol{c}) \in \Sigma_q^n$, $y_i = c_i - c_{i+1} \pmod q$ for $1 \leqslant i \leqslant n-1$ and $y_n = c_n$.
(III) Set $S \triangleq \{q^{j-1} : j \in [t]\} \cup \{n\}$ and $I \triangleq [n] \setminus S$.
(IV) Output $\boldsymbol{x} = \boldsymbol{y}|_I \in \Sigma_q^k$.

V. Conclusion

We have presented a new construction method of non-binary VT codes that are capable of correcting a single deletion or single insertion. We have further proposed efficient linear-time encoders that encode user messages into these codes of length n. Particularly, for codewords of length n, over the q-ary alphabet for $q > 2$, our best designed encoder uses $\lceil \log_q n \rceil + 1$ redundant symbols, which improves the redundancy of the best known encoder for single deletion or single insertion correction codes, proposed by Tenengolts [13], by $2 + \log_q(3)$ redundant symbols, or equivalently $2 \log_2 q + 3$ redundant bits. Our constructed codes also support systematic linear-time encoding and decoding procedures.

Acknowledgment

The work of Tuan Thanh Nguyen and Kui Cai is supported by the Singapore Ministry of Education Academic Research Funds Tier 2 MOE2019-T2-2-123 and T2EP50221-0036.

References

[1] Y. Ng, B. V. K. V. Kumar, K. Cai, S. Nabavi, and T. C. Chong, "Picketshift codes for bit-patterned media recording with insertion/deletion errors," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 2268-2271, Jun. 2010.
[2] W. Kang *et al.*, "Complementary skyrmion racetrack memory with voltage manipulation," *IEEE Electron Device Lett.*, vol. 37, pp. 924-927, 2016.
[3] L. Dolecek and V. Anantharam, "Using Reed-Muller RM$(1, m)$ codes over channels with synchronization and substitution errors," *IEEE Trans. Inf. Theory*, vol. 53, no. 4, pp. 1430-1443, Apr. 2007.
[4] S. Agarwal, D. Starobinski, and A. Trachtenberg, "On the scalability of data synchronization protocols for PDAs and mobile devices," *IEEE Netw.*, vol. 16, no. 4, pp. 22-28, Jul. 2002.
[5] S. Yazdi, H. M. Kiah, E. R. Garcia, J. Ma, H. Zhao, and O. Milenkovic, "DNA-based storage: Trends and methods", *IEEE Trans. Molecular, Biological, Multi-Scale Commun.*, vol. 1, no. 3, pp. 230-248, 2015.
[6] R. Heckel, G. Mikutis, and R. N. Grass, "A characterization of the DNA data storage channel," *Sci. Rep.*, vol. 9, no. 1, pp. 1-12, Jul. 2019.
[7] K. Cai, Y. M. Chee, R. Gabrys, H. M. Kiah and T. T. Nguyen, "Correcting a Single Indel/Edit for DNA-Based Data Storage: Linear-Time Encoders and Order-Optimality," in *IEEE Transactions on Information Theory*, vol. 67, no. 6, pp. 3438-3451, June 2021, doi: 10.1109/TIT.2021.3049627.
[8] R. Gabrys, V. Guruswami, J. Ribeiro and K. Wu, "Beyond Single-Deletion Correcting Codes: Substitutions and Transpositions", in *IEEE Transactions on Information Theory*, Aug 2022, doi: 10.1109/TIT.2022.3202856.
[9] T. T. Nguyen, K. Cai, K. A. Schouhamer Immink, and H. M. Kiah, "Capacity-approaching constrained codes with error correction for DNA-based data storage," *IEEE Trans. Inf. Theory*, vol. 67, no. 8, pp. 5602-5613, Aug. 2021.
[10] R. R. Varshamov and G. M. Tenengolts, "Codes which correct single asymmetric errors", *Automatica i Telemekhanica*, vol. 26, no. 2, pp. 288-292, 1965.
[11] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals", *Doklady Akademii Nauk SSSR*, vol. 163, no. 4, pp. 845-848, 1965.
[12] K. A. S. Abdel-Ghaffar and H. C. Ferreira, "Systematic encoding of the Varshamov-Tenengolts codes and the Constantin-Rao codes", *IEEE Trans. Inf. Theory*, vol. 44, no. 1, pp. 340-345, 1998.
[13] G. Tenengolts, "Nonbinary codes, correcting single deletion or insertion", *IEEE Trans. Inf. Theory*, vol. 30, no. 5, pp. 766-769, 1984.
[14] Y. M. Chee, H. M. Kiah, and T. T. Nguyen, "Linear-time encoders for codes correcting a single edit for DNA-based data storage", in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Paris, France, Jul. 2019, pp. 772-776.
[15] M. Abroshan, R. Venkataramanan and A. G. I. Fabregas, "Efficient Systematic Encoding of Non-binary VT Codes," *2018 IEEE International Symposium on Information Theory (ISIT)*, 2018, pp. 91-95.
[16] M. Abroshan, R. Venkataramanan, and A. G. i Fabregas, "Coding for segmented edit channels," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 3086-3098, Apr. 2018.
[17] Z. Liu and M. Mitzenmacher, "Codes for deletion and insertion channels with segmented errors," *IEEE Trans. Inf. Theory*, vol. 56, no. 1, pp. 224-232, Jan. 2010.
[18] K. Cai, H. M. Kiah, M. Motani and T. T. Nguyen, "Coding for Segmented Edits with Local Weight Constraints," *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 1694-1699.
[19] V. I. Levenshtein, "Binary codes capable of correcting spurious insertions and deletions of ones," *Prob. Inf. Trans.*, vol. 1, no. 1, pp. 8-17, Jan. 1965.
[20] V. Levenshtein, "Asymptotically optimum binary code with correction for losses of one or two adjacent bits," *Problemy Kibernetiki*, vol. 19, pp. 293-298, 1967.
[21] S. Wang, J. Sima and F. Farnoud, "Non-binary Codes for Correcting a Burst of at Most 2 Deletions," *2021 IEEE International Symposium on Information Theory (ISIT)*, 2021, pp. 2804-2809, doi: 10.1109/ISIT45174.2021.9517917.