Correcting Errors by Natural Redundancy

Anxiao (Andrew) Jiang
CSE Department
Texas A&M University
College Station, TX 77843
ajiang@cse.tamu.edu

Pulakesh Upadhyaya
CSE Department
Texas A&M University
College Station, TX 77843
pulakesh@tamu.edu

Erich F. Haratsch
Seagate Technology
Fremont, CA 94538

Jehoshua Bruck
EE & CNS Dept.
Caltech
Pasadena, CA 91125
bruck@caltech.edu

***Abstract*—For the storage of big data, there are significant challenges with its long-term reliability. This paper studies how to use the natural redundancy in data for error correction, and how to combine it with error-correcting codes to effectively improve data reliability. It explores several aspects of natural redundancy, including the discovery of natural redundancy in compressed data, the efficient decoding of codes with random structures, the capacity of error-correcting codes that contain natural redundancy, and the time-complexity tradeoff between source coding and channel coding.**

I. Introduction

The storage of big data has become increasingly important. Every day a large amount of data – 2.5 billion GB – is generated. However, its long-term reliability has significant challenges. For example, non-volatile memories (NVMs), such as flash memories and phase-change memories, store a substantial portion of big data due to their fast speed, physical robustness and large storage capacity. However they have data retention problems, where charge leakage or cell-level drifting makes data more noisy over time. Operations such as reads and writes cause accumulative disturbance in NVM data. Furthermore, erasures of NVM cells degrade cell quality and make cells more prone to errors over time. There is a strong motivation in elevating the long-term reliability of big-data storage to the next level.

The most effective way to protect data has been error-correcting codes (ECCs). By adding redundancy to data in a disciplined way, errors can be effectively corrected. We call such redundancy *artificial redundancy*. The recent advancement in learning and the availability of big data for study have offered a new opportunity for error correction: *to use the natural redundancy in data for error correction*. By natural redundancy (NR), we refer to the inherent redundancy in data that is not artificially added for error correction, such as various features in languages and images, structural features in databases, etc. Due to practical reasons (e.g., high complexity for compression, and lack of precise models for data), even after compression, lots of redundancy often still exist.

This paper studies how to use the natural redundancy in data for error correction, with a focus on languages and images. It is a topic related to joint source-channel coding and denoising. The idea of using the leftover redundancy at a source encoder to improve the performance of ECCs has been studied within the field of joint source-channel coding (JSCC) [2], [9], [10], [11], [12], [14], [23], [24], [25]. However, not many works have considered JSCC specifically for language-based sources, and exploiting the redundancy in the language structure via an efficient decoding algorithm remains as a significant challenge. Related to JSCC, denoising is also an interesting and well studied technique [1], [4], [5], [6], [18], [21], [22], [26], [32]. A denoiser can use the statistics and features of input data to reduce its noise level for further processing.

This work extends the study in [13], [15], [19], [31], where texts compressed by Huffman coding were studied. In comparison, this work explores several new topics on NR: the effective discovery of NR in texts deeply compressed by LZW coding, as well as NR in compressed images; the combination of NR-decoding with low-density parity-check (LDPC) codes and its performance analysis, for both the case of with and the case of without iterations between the two decoders; the efficient list decoding of random codes; the error-correction capability of ECCs with NR; and the computational-complexity tradeoff between source and channel coding. The experimental results and analysis show that NR-decoding can significantly improve the overall error correction performance.

II. Efficient Natural-Redundancy Discovery

A. Discovery of Natural Redundancy in Languages

Let us illustrate the potential of NR with English texts as an example. Character-wise Huffman coding achieves 4.59 bits/character. A Markov model for 3-grams achieves 3.06 bits/character. When LZW coding is used with a fixed dictionary of 2^{20} patterns (larger than many practical LZW codes), the rate is further reduced to 2.94 bits/character. However, Shannon has estimated that the true entropy of English is upper bounded by 1.34 bits/character [28]. That means over 54% of the data after the above LZW compression is still redundant, which is a great resource for error correction.

We have applied natural language processing (NLP) techniques to English texts to discover different forms of redundancy useful for error correction. We illustrate here one type of such redundancy, *co-location*. Consider the sample text from Wikipedia in Fig. 1 (a). The co-location relationship (i.e., certain phrases appear in similar contexts much more frequently than usual) can be obtained from training data. When tested, it reveals lots of dependency that often spans whole texts. For example, in the sample text, given the phrase "flash memory", its closely related phrases by the co-location relationship are shown in Fig. 1 (b). They exist in different places of the text, not just beside the phrase "flash memory". It means NR can be *global*. The *global NR* is illustrated more clearly in Fig. 1 (c). Let us first partition the sample text into

phrases (such as "Flash memory", "is an", "electronic", $\cdots$) by NLP, and show those phrases as dots (in the same order as in the text) at the bottom of the Tanner graph in Fig. 1 (c). If two phrases have the co-location relationship, they are connected by a red dot at the top through a blue edge and a red edge. Such relationships resemble parity checks in ECCs.

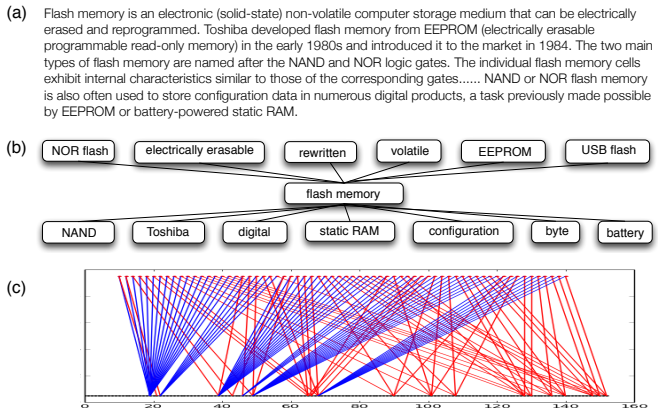

Fig. 1. (a) A sample text (part of which was omitted due to space limit). (b) Phrases in it that have the co-location relationship with "flash memory". (c) Tanner graph for phrases with co-location relationship in the sample text. The numbers on the x-axis are the indexes of the phrases.

B. Error Correction Algorithm and Performance

We study LZW coding that uses a fixed dictionary of 2^{20} patterns, where every 20-bit LZW codeword represents a character string (called a *pattern*). For example, when the paragraph in Fig. 1 (a) is compressed, it is partitioned into patterns "[Flash m], [emory i], [s an ele]$\cdots$", whose LZW codewords are (11011110100001000010), (11101101001100100110), (11001100100000100011) $\cdots$ We have designed an efficient algorithm for correcting erasures in such compressed texts by detecting valid words/phrases and co-location relationships, and using them to filter candidate solutions for each LZW codeword. Then a hard-decision decoding method is used: if all remaining candidate solutions agree on a bit, decode the bit to that value; otherwise, keep it as an erasure. The algorithm does not use any redundancy from ECC, and can be extended to correct errors.

Example 1. *Suppose that an LZW codeword in the compressed text with erasures is 1??0?1$\cdots$, where "?" is an erasure. Suppose that the NR-decoder finds 3 candidate solutions: 110001$\cdots$, 110011$\cdots$, 100011$\cdots$. Then it returns the solution 1?00?1$\cdots$ because the candidate solutions agree on the second erasure, but not the first or the third erasure.* □

We show its performance for the binary-erasure channel (BEC). The output of the NR-decoder has both erasures and errors (which will be further decoded by ECC). Let $\epsilon \in [0,1]$ be the raw bit-erasure rate (RBER) of BEC. After NR-decoding, for an originally erased it, let $\delta \in [0,1]$ denote the probability that it remains as an erasure, and let $\rho \in [0, 1-\delta]$ denote the probability that it is decoded to 0 or 1 incorrectly. Then the amount of noise after NR-decoding can be measured by the entropy of the noise (erasures and errors) per bit: $E_{NR}(\epsilon) \triangleq \epsilon(\delta + (1-\delta)H(\frac{\rho}{1-\delta}))$, where $H(p) = -p\log p - (1-p)\log(1-p)$ is the entropy function. Some typical values of $E_{NR}(\epsilon)$ are shown below. The reduction in noise by NR-decoding is $\frac{\epsilon - E_{NR}(\epsilon)}{\epsilon}$. The table shows that noise is reduced effectively (from 88.0% to 91.6%) for the LZW compressed data (without any help from ECC), for RBER from 5% to 30%, which is a wide range.

ϵ	0.05	0.10	0.15
δ	8.22×10^{-2}	8.67×10^{-2}	9.19×10^{-2}
ρ	9.18×10^{-5}	1.83×10^{-4}	1.82×10^{-4}
$E_{NR}(\epsilon)$	4.18×10^{-3}	8.92×10^{-3}	1.42×10^{-2}
Noise reduction	91.6%	91.1%	90.6%
ϵ	0.20	0.25	0.30
δ	9.76×10^{-2}	1.05×10^{-1}	1.12×10^{-1}
ρ	3.61×10^{-4}	4.48×10^{-4}	7.11×10^{-4}
$E_{NR}(\epsilon)$	2.04×10^{-2}	2.76×10^{-2}	3.60×10^{-2}
Noise reduction	89.8%	89.0%	88.0%

When the compressed texts are encoded as information bits by a systematic ECC, after the above NR-decoding, the ECC-decoder can correct the remaining errors/erasures. We will analyzed such a scheme for LDPC codes in the next chapter, which shows significant performance improvement. For example, for a $(5, 100)$ regular LDPC code, which has rate 0.95 (a typical code rate for storage systems), the scheme can improve the erasure threshold from 0.036 to over 0.3 (an improvement of more than 733.3%).

C. NR-Decoding for Images

We now present a natural-redundancy (NR) decoder for images. In particular, we focus on images of handwritten digits, as in Fig. 2 (a). They are from the National Institute of Standards and Technology (NIST) database, which have $70,000$ images as training or test data. We compress the bi-level images (of size 28×28 pixels) using run-length coding, where the run-lengths of 0s and 1s are compressed by two optimized Huffman codes, respectively. The rate is 0.27 bit/pixel.

Assume the compressed images have erasures. To decode noisy images, we have designed a convolutional neural network for recognizing noisy images, and also a specialized filter based on features of connected components in decompressed images. The NR-decoder is illustrated in Fig. 2 (b). The final step of decoding is: if all candidate solutions agree on a bit, set the bit to that value; otherwise, keep it as an erasure.

The decoding performance can be measured in the same way as for texts. We show E_{NR} in Fig. 2 (c). The NR-decoder reduces noise substantially: it removes noise effectively by over 75% for the compressed images (without any help from ECC), for raw bit-erasure rate (RBER) from 0.5% to 6.5%.

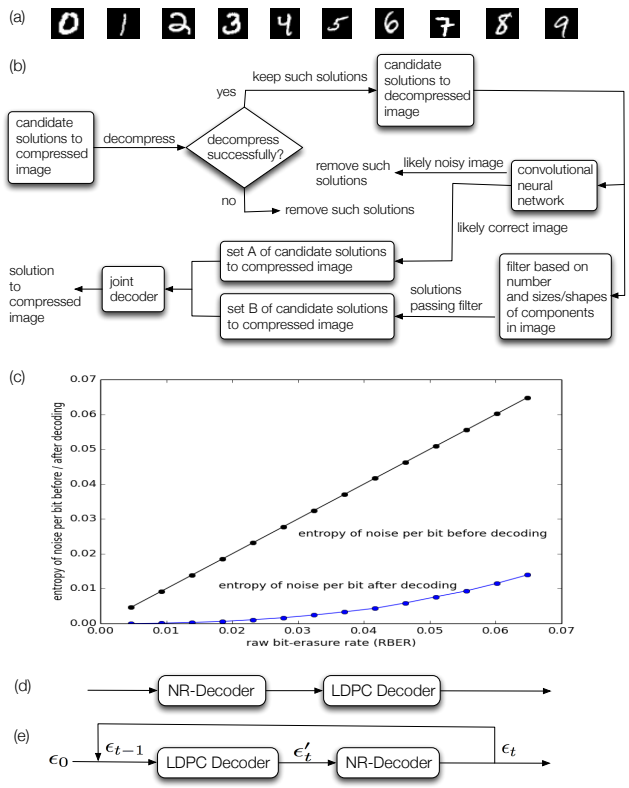

Fig. 2. (a) Examples of handwritten digits. (b) NR-decoder for images. (c) Performance of NR-decoder. (d) A concatenated decoding scheme. (e) An iterative decoding scheme.

III. Combine NR-decoding with LDPC Codes

We protect compressed data (languages or images) as information bits by a systematic LDPC code of rate R. The decoding process is a concatenation of two decoders: first, the NR-decoder decodes the codeword (possibly only its information bits), and outputs a partially corrected codeword with updated soft information; then, the LDPC decoder takes that as input, and uses belief propagation (BP) for decoding. See Fig. 2 (d). We present a theoretical analysis for the decoding performance, and show that the NR-decoder can substantially improve the performance of LDPC codes.

Consider a binary-erasure channel (BEC) with erasure probability ϵ_0. (BSC can be analyzed similarly.) Let us call the non-erased bits *fixed bits*. Assume that after NR-decoding, a non-fixed bit (i.e., erasure) remains as an erasure with probability $p_0 \in [0,1]$, becomes an error (0 or 1) with probability $(1-p_0)\gamma_0 \in [0, 1-p_0]$, and is decoded correctly (as 0 or 1) with probability $(1-p_0)(1-\gamma_0)$. (Note that if the NR-decoder decodes only information bits, and an erasure in the information bits remains as an erasure with probability p'_0, then $p_0 = Rp'_0 + (1-R)$. Also note that the LDPC decoder needs to decode all bits with both errors and erasures.)

We design the following iterative LDPC decoding algorithm, which generalizes both the peeling decoder for BEC and the Gallager B decoder for BSC: (1) let $\pi \in [1, d_v - 1]$ and $\tau \in [1, d_v - 1]$ be two integer parameters; (2) in each iteration, for a variable node v that is an erasure, if π or more non-erased message bits come from $d_v - 1$ check nodes and they all have the same value, set v to that bit value; (3) if v is not a fixed bit and not an erasure (but possibly an error) in this iteration, change v to the opposite bit value if τ or more non-erased message bits come from $d_v - 1$ check nodes and they all have that opposite value. (The updated value of v will be sent to the remaining check node in the next iteration.) We now analyze the density evolution for the decoding algorithm, for an infinitely long and randomly constructed LDPC code. (For $t = 0, 1, 2 \cdots$, let α_t and β_t be the fraction of codeword bits that are errors or erasures, respectively, after t iterations of LDPC decoding. We have $\alpha_0 = \epsilon_0(1-p_0)\gamma_0$ and $\beta_0 = \epsilon_0 p_0$. Let $\kappa_0 = \epsilon_0(1-p_0)(1-\gamma_0)$.)

Theorem 2. *For a regular (d_v, d_c) LDPC code with variable-node degree d_v and check-node degree d_c, we have $\alpha_{t+1} = \alpha_0 C_t + \kappa_0 D_t + \beta_0 \mu_t$, where $C_t = 1-(1-A_t)^{d_v-1} + \sum_{i=0}^{\tau-1}\binom{d_v-1}{i} B_t^i (1-A_t-B_t)^{d_v-i-1}$, $D_t = \sum_{j=\tau}^{d_v-1}\binom{d_v-1}{j} A_t^j (1-A_t-B_t)^{d_v-1-j}$, $\mu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m} A_t^m (1-A_t-B_t)^{d_v-1-m}$ with $A_t = \frac{(1-\beta_t)^{d_c-1}-(1-\beta_t-2\alpha_t)^{d_c-1}}{2}$ and $B_t = \frac{(1-\beta_t)^{d_c-1}+(1-\beta_t-2\alpha_t)^{d_c-1}}{2}$. And $\beta_{t+1} = \beta_0(1-\mu_t-\nu_t)$, where $\nu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m} B_t^m (1-A_t-B_t)^{d_v-1-m}$.*

Define *erasure threshold* ϵ^* as the maximum erasure probability (for ϵ_0) for which the LDPC code can decode successfully (which means the error/erasure probabilities α_t and β_t both approach 0 as $t \to \infty$). Let us show how the NR decoder can substantially improve ϵ^*. Consider a regular LDPC code with $d_v = 5$ and $d_c = 100$, which has rate 0.95 (a typical code rate for storage systems). Without NR-decoding, the erasure threshold is $\tilde{\epsilon}^* = 0.036$. Now let $\pi = 1$ and $\tau = 4$. For compressed images, when $\epsilon_0 = 0.065$, the NR-decoder gives $p_0 = 0.247$ and $\gamma_0 = 0.0008$, for which the LDPC decoder has $\lim_{t\to\infty} \alpha_t = 0$ and $\lim_{t\to\infty} \beta_t = 0$. (The same happens for $\epsilon_0 < 0.065$.) So with NR-decoding, $\epsilon^* \geq 0.065$, which means the improvement in erasure threshold is more than 80.5%. For LZW-compressed texts, when $\epsilon_0 = 0.3$, the NR-decoder gives $p_0 = 0.156$ and $\gamma_0 = 0.0008$, for which the LDPC decoder has $\lim_{t\to\infty} \alpha_t = 0$ and $\lim_{t\to\infty} \beta_t = 0$. (The same happens for $\epsilon_0 < 0.3$.) So with NR-decoding, $\epsilon^* \geq 0.3$, which means the improvement in erasure threshold is more than 733.3%.

IV. Iterative LDPC Decoding with NR

In this section, we study the decoding performance when we use *iterative decoding* between the LDPC decoder and NR-decoder, as shown in Fig. 2 (e). (In last section's study, the NR-decoder is followed by the LDPC decoder, without iterations between them.) We focus on languages, and present a theoretical model for compressed languages as follows.

Let $T = (b_0, b_1, b_2, \cdots)$ be a compressed text. Partition T into segments $S_0, S_1, S_2 \cdots$, where each segment $S_i = (b_{il}, b_{il+1}, \cdots, b_{il+l-1})$ has l bits. Consider erasures.

Let $\theta \in [0,1]$, $l_\theta \triangleq \lfloor l\theta \rfloor$ and $p \in [0,1]$ be parameters. We assume that when a segment S_i has at most l_θ erasures, the NR-decoder can decode it by checking the validity of the up to 2^{l_θ} candidate solutions (based on the validity of their corresponding words/phrases, grammar, etc.), and either determines (independently) the correct solution with probability p or makes no decision with probability $1-p$. And this NR-decoding operation can be performed only *once* for each segment. Here l_θ models the limit on time complexity (because the decoder needs to check 2^{l_θ} solutions), and p models the probability of making an error-free decision (which is a simplification of the practical NR-decoders shown in the last section that make very high-confidence, although not totally error-free, decisions). The model is suitable for compression algorithms such as LZW coding with a fixed dictionary, Huffman coding, etc., where each segment can be decompressed to a piece of text. The greater l is, the better the model is.

The compressed text T is protected as information bits by a systematic LDPC code. The LDPC code uses the peeling decoder for BEC (where $d_c - 1$ incoming messages of known values at a check node determine the value of the outgoing message on the remaining edge) to correct erasures. See the decoding model in Fig. 2 (e). In each iteration, the LDPC decoder runs *one iteration* of BP decoding, then the NR-decoder tries to correct those l-information-bit segments that contain at most l_θ erasures (if those segments were never decoded by the NR-decoder before). Let $\epsilon_0 < 1$ be the BEC's erasure rate. Let ϵ'_t and ϵ_t be the LDPC codeword's erasure rate after the t-th iteration of the LDPC decoder and the NR-decoder, respectively. Next, we analyze the density evolution for regular (d_v, d_c) LDPC codes of rate $R = 1 - \frac{d_v}{d_c}$.

Note that since the NR-decoder decodes only information bits, for the LDPC decoder, the information bits and parity-check bits will have different erasure rates during decoding. Furthermore, information bits consist of l-bit segments, while parity-check bits do not. For such an l-bit segment, if the NR-decoder can decode it successfully when it has no more than l_θ erasures, let us call the segment *lucky*; otherwise, call it *unlucky*. Luck and unlucky segments will have different erasure rates during decoding, too.

For $t = 1, 2, 3 \cdots$ and $k = 0, 1, \cdots, l$, let $f_k(t)$ denote the probability that a lucky segment contains k erasures after t iterations of decoding by the NR-decoder. Define $q_0 = 1$, $q_t \triangleq \frac{\epsilon_t}{\epsilon'_t}$ and $d_t \triangleq \frac{\epsilon'_t}{\epsilon_{t-1}}$ for $t \geq 1$. Note that decoding will end after t iterations if one of these conditions occurs: (1) $\epsilon'_t = 0$, because all erasures are corrected by the t-th iteration; (2) $d_t = 1$, because the LDPC decoder corrects no erasure in the t-th iteration, and nor will the NR-decoder since the input codeword is identical to its previous output. We now study density evolution before those boundary cases occur.

Theorem 3. *For $t \geq 1$,*

$$\epsilon_t = ((1-R) + R(1-p))\epsilon_0 (\prod_{i=1}^{t} d_t) + Rp \sum_{k=l_\theta+1}^{l} \frac{k}{l} f_k(t),$$

$$\epsilon'_t = (\prod_{m=0}^{t-1} q_m)\epsilon_0 (1 - (1-\epsilon_{t-1})^{d_c-1})^{d_v-1}.$$

$$f_k(1) = \begin{cases} \sum_{i=0}^{l_\theta} \binom{l}{i} (\epsilon'_1)^i (1-\epsilon'_1)^{l-i} & \text{if } k = 0 \\ 0 & \text{if } 1 \leq k \leq l_\theta \\ \binom{l}{k} (\epsilon'_1)^k (1-\epsilon'_1)^{l-k} & \text{if } l_\theta + 1 \leq k \leq l \end{cases}$$

For $t \geq 2$,

$$f_k(t) = \begin{cases} f_k(t-1) + \sum_{i=l_\theta+1}^{l} \sum_{j=0}^{l_\theta} f_i(t-1) \binom{i}{j} (d_t)^j (1-d_t)^{i-j} & \\ & \text{if } k = 0 \\ 0 & \text{if } 1 \leq k \leq l_\theta \\ \sum_{i=k}^{l} f_i(t-1) \binom{i}{k} (d_t)^k (1-d_t)^{i-k} & \text{if } l_\theta + 1 \leq k \leq l \end{cases}$$

The iterative decoding scheme can improve the erasure threshold ϵ^* for LDPC codes substantially. For example, for a $(d_v = 5, d_c = 100)$ LDPC code, when $l = 15$, $l_\theta = 9$ and $p = 0.1$, 0.5 and 0.9, ϵ^* is increased from $\tilde{\epsilon}^* = 0.036$ to 0.039, 0.067 and 0.242, respectively.

V. Sampling-based Decoding for Random Codes

A. Sliding-Window Decoder for Prefix-free Codes

Beside LZW coding, prefix-free codes are another important choice for compression. In [13], [15], [19], [31], Huffman codes for English-text characters were used for the study of NR, where every character (letter or punctuation mark) is represented by a variable-length Huffman codeword. A significant challenge for NR-decoding is that the compressed file does not specify the boundaries of Huffman codewords, making it difficult to recognize words/phrases, especially for high BER. To address the problem, we propose a *sliding-window decoding* technique: use a sliding window (of a variable size) to check different segments of the noisy compressed file; and if by flipping at most a few bits, the bits in the window can be decompressed as a long yet relatively common word/phrase (such as "information"), then this solution is highly likely to be correct, because long words/phrases are extremely sparse. The latter point is shown in the following example.

Example 4. *Consider lower-case words. Assume there are M_n words whose Huffman codewords have n bits. Then the* density *of such words is $D_n = \frac{M_n}{2^n} = 10^{-x_n}$. We show M_n and x_n (collected from Wikipedia, a very large text corpus) in Fig.* 3. *It can be seen that the word density decreases exponentially fast for large n. So long words are very sparse.* □

The sliding-window technique can enhance existing NR-decoders, such as the Word-Recognition NR-decoding algorithm in [13]. The key is how to correct errors inside a window both efficiently and reliably, without exhaustive search. That leads to the random-code decoding algorithm below.

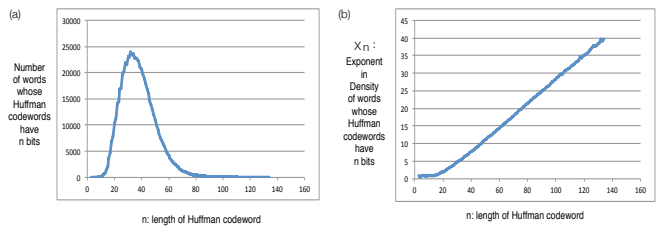

Fig. 3. (a) The number of words M_n whose Huffman codewords have n bits. (b) The exponent x_n in the density 10^{-x_n} for words whose Huffman codewords have n bits.

B. Sampling-based Decoder for Random Codes

Consider a window of n bits. It can have 2^n possible values; however, only a small subset of them $\mathcal{C} \subset \{0,1\}^n$ correspond to valid words/phrases. Let $\mathbf{y} = (y_1, y_2, \cdots, y_n) \in \{0,1\}^n$ be the bit-string we receive in the window, and let $t < n$ be an integer parameter. Assuming that there exists a codeword $\mathbf{x} \in \mathcal{C}$ with Hamming distance $d_H(\mathbf{x}, \mathbf{y}) = t$, we would like to find $\mathbf{x}$ with high probability and low time complexity. Note that an exhaustive search will have complexity $\binom{n}{t} = O(n^t)$, which is exponential in t and inefficient in practice.

We start by defining a sampling function. Given a set $P \subseteq \{1, 2, \cdots, n\}$, the *sampling function* $f_{n,P} : \{0,1\}^n \to \{0,1,?\}^n$ is: $\forall\ \mathbf{b} = (b_1, b_2, \cdots, b_n) \in \{0,1\}^n$, we have $f_{n,P}(\mathbf{b}) = (b'_1, b'_2, \cdots, b'_n)$ where each $b'_i = b_i$ if $i \in P$ and $b'_i =?$ otherwise. (Here "?" represents an un-sampled bit.) Define the *sample-value set* as $V_{n,P} \triangleq \{(b_1, b_2, \cdots, b_n) \in \{0,1,?\}^n | \text{for } 1 \le i \le n, b_i =? \text{ if } i \notin P\}$. We have $|V_{n,P}| = 2^{|P|}$. For any sample value $\mathbf{s} \in V_{n,P}$, let $E_{n,P}(\mathcal{C}, \mathbf{s}) \subseteq \mathcal{C}$ be the subset of codewords that, when sampled by $f_{n,P}$, have the sample value $\mathbf{s}$; that is, $E_{n,P}(\mathcal{C}, \mathbf{s}) \triangleq \{\mathbf{b} \in \mathcal{C} \mid f_{n,P}(\mathbf{b}) = \mathbf{s}\}$. $E_{n,P}(\mathcal{C}, \mathbf{s})$ may contain zero, one or more codewords of $\mathcal{C}$.

We build a data structure called *sample dictionary* $Dic_{n,P}(\mathcal{C}) \triangleq \{(\mathbf{s}, E_{n,P}(\mathcal{C}, \mathbf{s})) \mid \mathbf{s} \in V_{n,P}\}$. It is a set of $(key, value)$ pairs, where each key is a sample value $\mathbf{s} \in V_{n,P}$, and its corresponding value is the set $E_{n,P}(\mathcal{C}, \mathbf{s})$. Given $\mathbf{y}$, if its sample value $f_{n,P}(\mathbf{y})$ matches that of $\mathbf{x}$, we can use it to look up the dictionary, and find $\mathbf{x}$ in the set $E_{n,P}(\mathcal{C}, f_{n,P}(\mathbf{y}))$.

We now generalize the above discussion to k sampling functions f_{n,P_1}, f_{n,P_2}, $\cdots$, f_{n,P_k}. By choosing a suitable k, a good balance between decoding complexity and decoding-success probability can be achieved. Let P_1, P_2, $\cdots$, P_k be k subsets of $\{1, 2, \cdots, n\}$. We can use the k corresponding sampling functions $f_{n,P_1}, f_{n,P_2}, \cdots, f_{n,P_k}$ to sample the code $\mathcal{C}$, and get an *aggregated sample dictionary* $Dic_{n,P_1,\cdots,P_k}(\mathcal{C}) \triangleq \{(i, \mathbf{s}, E_{n,P_i}(\mathcal{C}, \mathbf{s})) \mid 1 \le i \le k, \mathbf{s} \in V_{n,P_i}\}$. We build the dictionary before decoding as *preprocessing*.

Define the *candidate codewords* for $\mathbf{y}$ as $Cand_{n,P_1,\cdots,P_k}(\mathcal{C}, \mathbf{y}) \triangleq \bigcup_{i=1}^{k} E_{n,P_i}(\mathcal{C}, f_{n,P_i}(\mathbf{y}))$. They are codewords that match $\mathbf{y}$ for at least one of the k sampling functions. So we can also define it as $Cand_{n,P_1,\cdots,P_k}(\mathcal{C}, \mathbf{y}) \triangleq \{\mathbf{b} \in \mathcal{C} \mid \exists 1 \le i \le k \text{ such that } f_{n,P_i}(\mathbf{b}) = f_{n,P_i}(\mathbf{y})\}$.

We now describe our decoding strategy: given a received string $\mathbf{y} \in \{0,1\}^n$, use the k sample values $f_{n,P_1}(\mathbf{y})$, $f_{n,P_2}(\mathbf{y})$, $\cdots$, $f_{n,P_k}(\mathbf{y})$ as keys to look up values in the dictionary $Dic_{n,P_1,\cdots,P_k}(\mathcal{C})$, and return a set of candidate codewords $Cand_{n,P_1,\cdots,P_k}(\mathcal{C}, \mathbf{y})$. (The candidate codewords will be filtered further based on their Hamming distance to $\mathbf{y}$, the frequencies of its words/phrases in training texts, co-location relationship with phrases elsewhere, etc., and be combined with an existing NR-decoder such as [13]. Our objective here is to include $\mathbf{x}$ as a candidate codeword.)

We now analyze two aspects of the decoding strategy's performance: its probability of including $\mathbf{x}$ as a candidate codeword, and its expected time complexity. Given any $\mathbf{a} = (a_1, \cdots, a_n) \in \{0,1\}^n$ and $\mathbf{b} = (b_1, \cdots, b_n) \in \{0,1\}^n$, define $M(\mathbf{a}, \mathbf{b}) = \{i \mid 1 \le i \le n,\ a_i = b_i\}$. We have $|M(\mathbf{a}, \mathbf{b})| = n - d_H(\mathbf{a}, \mathbf{b})$. Let $P_{IN}(t)$ denote the probability that the target codeword $\mathbf{x}$ is among the candidate codewords, namely, $P_{IN}(t) \triangleq Pr\{\mathbf{x} \in Cand_{n,P_1,\cdots,P_k}(\mathcal{C}, \mathbf{y}) \mid \mathbf{x} \in \mathcal{C},\ d_H(\mathbf{x}, \mathbf{y}) = t\}$.

Lemma 5. *$\forall \mathbf{b} \in \mathcal{C}$, $\mathbf{b} \in Cand_{n,P_1,\cdots,P_k}(\mathcal{C}, \mathbf{y})$ if and only if there exists $i \in \{1, 2, \cdots, k\}$ such that $P_i \subseteq M(\mathbf{b}, \mathbf{y})$.*

$\mathcal{C}$ is an unstructured code. To facilitate analysis, we assume the following random model: Let $\mathcal{C}$ be a random code, whose codewords are chosen independently and uniformly at random from the vector space $\{0,1\}^n$. In addition, let $|P_1| = |P_2| = \cdots = |P_k| = m$ for some $m \le n - t$, and let each P_i independently choose its m elements uniformly at random from $\{1, 2, \cdots, n\}$ without replacement.

Theorem 6. $P_{IN}(t) = 1 - \left(1 - \frac{\binom{n-t}{m}}{\binom{n}{m}}\right)^k$.

The time complexity of decoding is determined by the number of candidate codewords we need to examine. For $i = 1, 2, \cdots, k$, the i-th sampling function f_{n,P_i} samples m bits of the received string $\mathbf{y}$, and uses the sample $f_{n,P_i}(\mathbf{y})$ to look up the set of candidate codewords $E_{n,P_i}(\mathcal{C}, f_{n,P_i}(\mathbf{y}))$ in the dictionary. So the total number of candidate codewords to examine (with possible overlapping for different sampling functions) is $\sum_{i=1}^{k} |E_{n,P_i}(\mathcal{C}, f_{n,P_i}(\mathbf{y}))|$. Let $\mu(t)$ denote the *expected* number of candidate codewords to examine given that there exists a codeword $\mathbf{x} \in \mathcal{C}$ with $d_H(\mathbf{x}, \mathbf{y}) = t$.

Theorem 7. $\mu(t) = k\left(\frac{\binom{n-t}{m}}{\binom{n}{m}} + (|\mathcal{C}| - 1)\left(\frac{1}{2}\right)^m\right)$.

Since long words/phrases are very sparse, $\mu(t)$ can be quite small for large n. Given n, $|\mathcal{C}|$ and t, we choose the parameters k and m to achieve a good balance between decoding complexity and success rate. For example, when $n = 48$ and $t = 10$, for n-bit English words compressed by Huffman coding (for example, "information" is such a 48-bit word, and there are $|\mathcal{C}| = 12,895$ such words), we can choose parameters m and k such that only less than 250 (instead of $\binom{48}{10} \gg 250$) candidate words need to be checked on average, and the correct word is included in the checked words with

probability $P_{IN}(t) \geq 0.99$. We show more results in Fig. 4, where we let $n = 40$ to 60, $t = 6$, 8 and 10, and show the minimum value of $\mu(t)$ for which there exist values of k and m that make $P_{IN}(t) \geq 0.99$. We see that $\mu(t)$ varies between 1 and 1,617, which is much less than $\binom{n}{t}$. (The curve for $t = 6$ is between 1 and 4, so it looks almost flat.) Note that here the corresponding values of $\binom{n}{t}$ is approximately between 10^7 and 10^{12}. It can be seen that the decoding algorithm reduces the number of candidate codewords substantially.

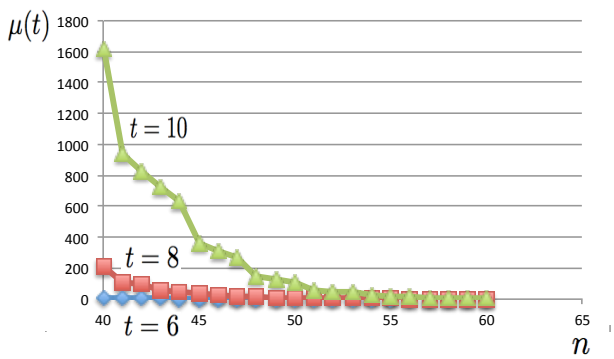

Fig. 4. Performance of sampling-based decoder for random codes. Here the x-axis is n, and the y-axis is the minimum value of $\mu(t)$ for which there exists a feasible solution to k and m given the condition that $P_{IN}(t) \geq 0.99$ for $t = 6$, 8 and 10.

We have combined the word-recognition algorithm in [13] with the sliding-window decoding technique here, for suitably chosen n and t. The new algorithm improves the error-correction performance substantially. Consider compressed texts protected by an $(4376, 4095)$ LDPC code designed by MacKay [20], which has rate 0.936 and is designed for BSC of error probability 0.2% (a typical parameter setting in storage systems). We compare the new algorithm with two known algorithms: using the BP decoding of the LDPC code alone, and the word-recognition algorithm in [13]. The results are shown in Fig. 5, where *success rate* is defined as the probability that an LDPC codeword is decoded correctly.

BER	0.2%	0.3%	0.4%	0.5%	0.6%	0.7%
P_{ldpc}	100%	98.2%	77.5%	27.4%	2.9%	0
P_{soft}	100%	99.9%	99.5%	97.9%	94.2%	84.6%
P_{slid}	100%	100%	99.9%	99.2%	98.4%	94.5%
BER	0.8%	0.9%	1.0%	1.1%	1.2%	1.3%
P_{ldpc}	0	0	0	0	0	0
P_{soft}	67.1%	47.8%	26.7%	12.4%	3.9%	1.4%
P_{slid}	84.8%	68.3%	47.9%	28.7%	14.2%	5.8%

Fig. 5. The success rate of decoding with LDPC code alone (P_{ldpc}), the word-recognition algorithm (P_{soft}) [13], and the enhanced algorithm using sliding-window decoding (P_{slid}), when the bit error probability (BER of a binary-symmetric channel) increases from 0.2% to 1.3%.

VI. Capacity of ECC with Natural Redundancy

In this section, we study two closely related theoretical models for ECCs with NR. We first study the capacity for information transmission when NR-decoding is present before ECC-decoding. We then study ECCs with finite length, and present an upper bound to the code sizes given that the ECCs receive assistance from NR-decoding.

A. Channel Capacity with Natural Redundancy

Consider compressed data with NR protected as information bits by a systematic ECC. A decoding scheme is shown in Fig. 6, where an NR-decoder is followed by an ECC-decoder. For the ECC, the channel together with the NR-decoder can be seen as a *compound channel*, where the channel adds noise and the NR-decoder reduces noise. The *compound-channel capacity* is defined in the conventional way, namely, as the maximum rate at which the channel input $X_1^n = (x_1, x_2, \cdots, x_n)$ (with $n \to \infty$) can be transmitted reliably.

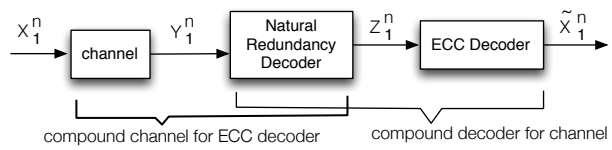

Fig. 6. A decoding scheme that combines NR-decoding with ECC-decoding.

We have studied NR-decoding for both BEC and BSC in previous sections. Note that in principle, the NR-decoder can decode not only information bits, but also parity-check bits by using parity-check constraints. That motivates us to study the following two compound-channel models: (1) in the Compound-BEC model, the BEC erases each bit $x_i \in \{0, 1\}$ independently with probability p; then for each bit y_i, if it is an erasure, the NR-decoder marks it as "original erasure" (so that this marked information is known to the ECC decoder), then independently recovers its value (correctly) as x_i with probability $(1-\delta)(1-\epsilon)$, recovers its value (incorrectly) as $1-x_i$ with probability $(1-\delta)\epsilon$, and let it remain as an erasure with probability δ; (2) in the Compound-BSC Model, the BSC flips each bit $x_i \in \{0, 1\}$ independently with probability p; then for each bit y_i, the NR-decoder marks it as "NR-decoded bit" independently with probability r. If y_i is marked as an "NR-decoded bit", the NR-decoder independently sets its value (correctly) to x_i with probability $1-q$, and sets its value (incorrectly) to $1-x_i$ with probability $q < p$.

Theorem 8. *The capacity of Compound-BEC is* $C_{c-BEC} = 1 - p + p(1-\delta)(1-H(\epsilon))$. *And the capacity of Compound-BSC is* $C_{c-BSC} = (1-r)(1-H(p)) + r(1-H(q))$.

B. Upper Bound to ECC Sizes with NR

The previous sections have presented analysis specifically for LDPC codes with belief-propagation decoding algorithms. Let us now consider general finite-length ECCs and their sizes. The NR-decoders for images and languages presented in Section II have a common feature: they both have very low error probabilities introduced by NR-decoding, namely, the

corrections are made with high confidence by NR-decoders. That motivates us to study the following theoretical model for error correction.

Let $\mathcal{A} = \{0, 1, \cdots, q-1\}$ be an alphabet, where $q \geq 2$. Let $\mathcal{C} \subseteq \mathcal{A}^n$ be a code of length n. Let r and t be integer parameters with $r+t \leq n$. Let the decoding process be an NR-decoder followed by an ECC-decoder, as shown in Fig. 6. Given a noisy word $\mathbf{y} = (y_1, y_2, \cdots, y_n) \in \mathcal{A}^n$, assume that the NR-decoder can determine the correct values of at least r symbols with certainty, without introducing additional errors. (Note that in practice, the errors corrected by the NR-decoder are only a small portion of such bits (symbols with $q = 2$). Many more such bits are non-errors, and the NR-decoder can determine that they are error-free because they belong to highly likely patterns, such as long and common phrases. Also note that in general, the NR-decoder can decode both information bits and parity-check bits.) Let $P \subseteq \{1, 2, \cdots, n\}$ denote the indexes of such determined symbols (where $|P| \geq r$), and without loss of generality (WLOG), we may assume $|P| = r$ for code analysis (because having larger $|P|$ only helps more). WLOG, we may also assume that the symbols of $\mathbf{y}$ with indexes in P are already correct symbols (because the NR-decoder determines their values anyway). After the NR-decoding, the ECC-decoder takes the pair $(\mathbf{y}, P)$ as input, and decodes it using maximum-likelihood (ML) decoding: the output is a codeword $\mathbf{x} = (x_1, x_2, \cdots, x_n) \in \mathcal{C}$ such that: (1) $\forall\ i \in P$, $x_i = y_i$; (2) the Hamming distance $d_H(\mathbf{x}, \mathbf{y}) \triangleq |\{i \mid 1 \leq i \leq n, x_i \neq y_i\}| = |\{i \mid 1 \leq i \leq n, i \notin P, x_i \neq y_i\}|$ is minimized.

$\forall\ \mathbf{x}, \mathbf{y} \in \mathcal{A}^n$ and $P \subseteq \{1, 2, \cdots, n\}$, if $x_i = y_i$ for every $i \in P$, we say $\mathbf{x} =_P \mathbf{y}$. We define $S_{t,P}(\mathbf{x}) \triangleq \{(\mathbf{y}, P) \mid \mathbf{x} =_P \mathbf{y},\ d_H(\mathbf{x}, \mathbf{y}) \leq t\}$. If $\forall\ \mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}$ and $P \subseteq \{1, 2, \cdots, n\}$ with $|P| = r$, we have $S_{t,P}(\mathbf{x}_1) \cap S_{t,P}(\mathbf{x}_2) = \emptyset$, we call $\mathcal{C}$ an (r, t)-ECC. An (r, t)-ECC is an error-correcting code that can correct t Hamming errors when the NR-decoder determines the values of any r symbols. It is an extension of t-error correcting codes. We have the following sphere packing bound.

Theorem 9 *For an (r, t)-ECC $\mathcal{C}$ with code length n, alphabet size q and $r + t \leq n$, the code's size*

$$|\mathcal{C}| \leq \frac{q^n}{\sum_{i=0}^{t} \binom{n-r}{i} (q-1)^i}.$$

VII. Computational-Complexity Tradeoff

NR can be used for both compression and error correction. How to use it suitably depends on many factors, such as available coding techniques, hardware design, etc. In this chapter, we discuss one such tradeoff: the computational complexity of using NR for compression or error correction. Real NR is hard to model precisely, so we explore this topic from a theoretical point of view, and consider NR in general forms. We show that certain types of redundancy are computationally efficient for compression, while others are so for error correction. Note that there exist works on analyzing the hardness of certain types of source coding schemes [16], [17], [27] and channel coding schemes [3], [7], [8], [29], [30]. In contrast, here we focus on the tradeoff between the two.

Let $B = (b_1, b_2, \cdots, b_n) \in \{0, 1\}^n$ be an n-bit message with NR. Define $\mathcal{V} : \{0, 1\}^n \to \{0, 1\}$ as a *validity function*: B is a valid message if and only if $\mathcal{V}(B) = 1$. The set of all valid messages of n bits is $\mathcal{M} \triangleq \{B \in \{0, 1\}^n \mid \mathcal{V}(B) = 1\}$. For simplicity, for both source and channel coding, assume that the valid messages in $\mathcal{M}$ are equally likely.

First, consider source coding. Let $k = \lceil \log_2 |\mathcal{M}| \rceil$. Define an *optimal lossless compression scheme* to be an injective function $C_{opt} : \mathcal{M} \to \{0, 1\}^k$ that compresses any valid message $B \in \mathcal{M}$ to a distinct k-bit vector $C_{opt}(B)$. Define the *Data Compression Problem* as follows: Given a validity function $\mathcal{V}$, find an injective function $C_{opt} : \mathcal{M} \to \{0, 1\}^k$.

Next, consider channel coding. Assume that a valid message $X = (x_1, x_2, \cdots, x_n) \in \mathcal{M}$ is transmitted through a binary-symmetric channel (BSC), and is received as a noisy message $Y = (y_1, y_2, \cdots, y_n) \in \{0, 1\}^n$. Maximum likelihood (ML) decoding requires us to find a message $Z = (z_1, z_2, \cdots, z_n) \in \mathcal{M}$ that minimizes the Hamming distance $d_H(Y, Z)$. Define the *Error Correction Problem* as follows: Given a validity function $\mathcal{V}$ and a message $Y \in \{0, 1\}^n$, find a valid message $Z \in \mathcal{M}$ that minimizes the Hamming distance $d_H(Y, Z)$.

Let $\mathcal{F}$ be the set of all functions from the domain $\{0, 1\}^n$ to the codomain $\{0, 1\}$. (We have $|\mathcal{F}| = 2^{2^n}$.) The function $\mathcal{V}$ represents NR in data. In practice, different types of data have different *types* of NR. Let us define the latter concept formally. For any subset $\mathcal{T} \subseteq \mathcal{F}$, let $\mathcal{T}$ be called a *type* of validity functions (which represents a type of NR). When $\mathcal{V}$ can only be a function in $\mathcal{T}$ (instead of $\mathcal{F}$), we denote the Data Compression Problem and the Error Correction Problem by $\mathcal{P}_{dc}^{\mathcal{T}}$ and $\mathcal{P}_{ec}^{\mathcal{T}}$, respectively. The hardness of the problems $\mathcal{P}_{dc}^{\mathcal{T}}$ and $\mathcal{P}_{ec}^{\mathcal{T}}$ depends on $\mathcal{T}$. Let $S_{dc=NP,ec=P}$ denote the set of types $\mathcal{T}$ (where each type is a subset of $\mathcal{F}$) for which the data compression problem $\mathcal{P}_{dc}^{\mathcal{T}}$ is NP-hard while the error correction problem $\mathcal{P}_{ec}^{\mathcal{T}}$ is polynomial-time solvable. Similarly, let $S_{dc=P,ec=NP}$ (or $S_{dc=P,ec=P}$, $S_{dc=NP,ec=NP}$, respectively) denote the set of types $\mathcal{T}$ for which $\mathcal{P}_{dc}^{\mathcal{T}}$ is polynomial-time solvable while $\mathcal{P}_{ec}^{\mathcal{T}}$ is NP-hard (or $\mathcal{P}_{dc}^{\mathcal{T}}$ and $\mathcal{P}_{ec}^{\mathcal{T}}$ are both polynomial-time solvable, or both NP-hard, respectively). The following theorem shows that there exist validity-function types for each of those four possible cases.

Theorem 10. *The four sets $S_{dc=NP,ec=P}$, $S_{dc=P,ec=NP}$, $S_{dc=P,ec=P}$ and $S_{dc=NP,ec=NP}$ are all non-empty.*

The above result shows a wide range of possibilities for the computational-complexity tradeoff between source and channel coding. In practice, it is worthwhile to study the properties of natural redundancy (e.g., whether the redundancy is mainly local or global, which differs for different types of data), and choose appropriate coding schemes based on computational complexity along with other important factors.

VIII. Concluding Remarks

This paper has shown that the natural redundancy in compressed data can be used to significantly improve the error correction performance. Both theoretical analysis and practical

algorithms are presented. However, there are still many open problems, including how to discover NR in complex images, and how to integrate error correction with more deep learning techniques. They remain as our future research directions.

REFERENCES

[1] L. Alvarez, P. Lions and J. Morel, "Image Selective Smoothing and Edge Detection by Nonlinear Diffusion. II," in *SIAM Journal on numerical analysis*, vol. 29, no. 3, pp. 845–866, 1992.

[2] R. Bauer and J. Hagenauer, "On Variable Length Codes for Iterative Source/Channel Decoding," in *Proceedings of Data Compression Conference*, pp. 273–282, 2001.

[3] J. Bruck and M. Naor, "The Hardness of Decoding Linear Codes with Preprocessing," in *IEEE Transactions on Information Theory*, vol. 36, no. 2, pp. 381–385, March 1990.

[4] A. Buades, B. Coll and J. Morel, "A Review of Image Denoising Algorithms, with a New One," in *Multiscale Modeling & Simulation*, vol. 4, no. 2, pp. 490530, 2005.

[5] P. Chatterjee and P. Milanfar, "Is denoising dead?" in *IEEE Transactions on Image Processing*, vol. 19, no. 4, pp. 895911, 2010.

[6] R. Coifman and D. Donoho, *Translation-invariant De-noising*, Springer, 1995.

[7] I. Dumer, D. Micciancio, and M. Sudan, "Hardness of Approximating the Minimum Distance of a Linear Code," in *IEEE Transactions on Information Theory*, vol. 49, no. 1, pp. 22–37, January 2003.

[8] U. Feige and D. Micciancio, "The Inapproximability of Lattice and Coding Problems with Preprocessing," in *Journal of Computer and Systems Sciences*, vol. 69, no. 1, pp. 45–67, August 2004.

[9] M. Fresia and G. Caire, "Combined Error Protection and Compression with Turbo Codes for Image Transmission Using a JPEG2000-like Architecture," in *Proc. International Conference on Image Processing*, pp. 821–824, 2006.

[10] L. Guivarch, J. Carlach and P. Siohan, "Joint Source-channel Soft Decoding of Huffman Codes with Turbo-codes," in *Proceedings of Data Compression Conference (DCC)*, pp. 83-92, 2000.

[11] J. Hagenauer, "Source-controlled Channel Decoding," in *IEEE Transactions on Communications*, vol. 43, no. 9, pp. 2449–2457, 1995.

[12] M. Jeanne, J. Carlach and P. Siohan, "Joint Source-channel Decoding of Variable-length Codes for Convolutional Codes and Turbo Codes," in *IEEE Transactions on Communications*, vol. 53, no. 1, pp. 10-15, 2005.

[13] A. Jiang, Y. Li and J. Bruck, "Error Correction through Language Processing," in *Proc. IEEE Information Theory Workshop (ITW)*, Jerusalem, Israel, April 2015.

[14] A. N. Kim, S. Sesia, T. Ramstad, and G. Caire, "Combined Error Protection and Compression using Turbo Codes for Error Resilient Image Transmission," in *Proceedings of International Conference on Image Processing (ICIP)*, vol. 3, pp. III-912-15, 2005.

[15] Y. Li, Y. Wang, A. Jiang and J. Bruck, "Content-assisted File Decoding for Nonvolatile Memories," in *Proc. 46th Asilomar Conference on Signals, Systems and Computers*, pp. 937–941, Pacific Grove, CA, 2012.

[16] J. Lin, "Vector Quantization for Image Compression: Algorithms and Performance," Doctoral Dissertation, Brandeis University, 1992.

[17] J. Lin, J. Storer and M. Cohn, "Optimal Pruning for Tree-structured Vector Quantization," in *Information Processing and Management*, vol. 28, no. 6, 1992.

[18] M. Lindenbaum, M. Fischer, and A. Bruckstein, "On Gabor's Contribution to Image Enhancement," in *Pattern Recognition*, vol. 27, no. 1, pp. 18, 1994.

[19] J. Luo, Q. Huang, S. Wang and Z. Wang, "Error Control Coding Combined with Content Recognition," in *Proc. 8th International Conference on Wireless Communications and Signal Processing*, pp. 1–5, 2016.

[20] D. MacKay, *Encyclopedia of Sparse Graph Codes*, $http://www.inference.phy.cam.ac.uk/mackay/codes/data.html\#l132$.

[21] E. Ordentlich, G. Seroussi, S. Verdu, and K. Viswanathan, "Universal Algorithms for Channel Decoding of Uncompressed Sources," *IEEE Trans. Information Theory*, vol. 54, no. 5, pp. 2243–2262, May 2008.

[22] E. Ordentlich, G. Seroussi, S. Verdu, M. Weinberger and T. Weissman, "A Discrete Universal Denoiser and Its Application to Binary Images," in *Proc. International Conference on Image Processing*, vol. 1, pp. 117, 2003.

[23] Z. Peng, Y. Huang and D. Costello, "Turbo codes for Image Transmission – A Joint Channel and Source Decoding Approach," in *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 18, no. 6, pp. 868–879, 2000.

[24] C. Poulliat, D. Declercq, C. Lamy-Bergot, and I. Fijalkow, "Analysis and Optimization of Irregular LDPC Codes for Joint Source-channel Decoding," *in IEEE Communications Letter*, vol. 9, no. 12, pp. 1064–1066, 2005.

[25] L. Pu, and Z. Wu, A. Bilgin, M. Marcellin, and B. Vasic, "LDPC-based Iterative Joint Source-channel Decoding for JPEG2000," in *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 577–581, 2007.

[26] L. Rudin, S. Osher and E. Fatemi, "Nonlinear Total Variation based Noise Removal Algorithms," in *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.

[27] M. Ruhl and H. Hartenstein, "Optimal Fractal Coding is NP-hard," in *Proc. Data Compression Conference (DCC)*, April 1997.

[28] C. E Shannon, "Prediction and Entropy of Printed English," in *Bell System Technical Journal*, vol. 30, no. 1, pp. 50–64, 1951.

[29] A. Vardy, "Algorithmic Complexity in Coding Theory and the Minimum Distance Problem,"in *Proc. 29th ACM Symposium on Theory of Computing (STOC)*, pp. 92–109, El Paso, Texas, May 1997.

[30] A. Vardy, "The Intractability of Computing the Minimum Distance of a Code," in *IEEE Transactions on Information Theory*, pp. 43, no. 6, pp. 1757–1766, November 1997.

[31] Y. Wang, M. Qin, K. R. Narayanan, A. Jiang and Z. Bandic, "Joint Source-channel Decoding of Polar Codes for Language-based Sources," in *Proc. IEEE Global Communications Conference (Globecom)*, Washington D.C., December 2016.

[32] L. Yaroslavsky and M. Eden, *Fundamentals of Digital Optics: Digital Signal Processing in Optics and Holography*, Springer-Verlag New York, Inc., 1996.

[33] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu and M. Weinberger, "Universal Discrete Denoising: Known Channel," in *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 5–28, 2005.