On LDPC Decoding with Natural Redundancy

Pulakesh Upadhyaya and Anxiao (Andrew) Jiang
Computer Science and Engineering Department, Texas A&M University
pulakesh@tamu.edu, ajiang@cse.tamu.edu

***Abstract*—Big-data storage has substantial challenges due to accumulative noise in storage media. To ensure its long-term reliability, new techniques for error correction are being explored. This paper studies how to discover natural redundancy in data and use it for error correction. It explores the combination of natural redundancy decoding with low-density parity-check (LDPC) codes for enhanced error-correction performance. It derives analytical equations for the density evolution of LDPC decoding given information from natural-redundancy decoders. It proposes a theoretical model for compressed languages, and studies the performance of iterative decoding between the LDPC decoder and the natural-redundancy decoder. It also presents an upper bound to the code sizes of error-correcting codes given the assistance from natural-redundancy decoders.**

I. Introduction

Big-data storage is having increasingly wide applications. However, it faces a substantial challenge – how to recover data from errors as effectively as possible for reliable long-term storage – due to accumulative noise in storage media. For example, flash memories and other NVMs have noise mechanisms such as charge leakage, read/write disturbs, and cell-quality degradation due to P/E cycling. They make data more and more noisy over time. So there is a strong motivation in exploring new techniques for error correction.

In this paper, we study how to correct errors using *natural redundancy* (NR) in compressed data, and how to combine it with error-correcting codes (ECCs). By natural redundancy, we refer to the redundancy in data that is not artificially added for error correction, such as features in languages/images and structures in databases. In comparison, the redundancy in an ECC (which we shall call *artificial redundancy*) is added in a disciplined way with the specific goal of effective error correction. NR is often a rich resource for error correction for data that are uncompressed or compressed imperfectly. There are various reasons for imperfect compression in practical systems, including high complexity of optimal compression, our limited understanding on the data models (e.g., for languages and images), etc. For data that are encoded as ECCs and later corrupted by errors, as our understanding on the data model improves, we can design better and better NR-decoders to correct the errors.

With NR, a decoding system can be considered as consisting of two decoders: an ECC-Decoder, and an NR-Decoder. They work collaboratively to correct errors or erasures in the ECC codeword. We illustrate it by an example.

Example 1.. *Consider texts compressed by an LZW algorithm that uses a fixed dictionary of size 2^ℓ. The dictionary has 2^ℓ text strings (called patterns) of variable lengths, where every pattern is encoded as an ℓ-bit codeword. Given a text to compress, the LZW algorithm scans it and partitions it into patterns, and maps them to codewords. For instance, if $\ell = 20$ and the text is "Flash memory is an electronic $\cdots$", the partitioning and LZW-codewords can be as illustrated in Fig.* 1 *(a).*

Now suppose some bits in the LZW-codewords are erased. An NR-Decoder can check all the possible solutions, map each solution back to patterns, and use a dictionary of words to eliminate those solutions that contain invalid words. (Such a dictionary of words has been commonly used in spell checkers.) If all the remaining solutions agree on the value of an erased bit, then that erasure is decoded by the NR-Decoder. For instance, suppose each LZW-codeword in Fig. 1 *(a) suffers from two erasures, which lead to four possible solutions/patterns (see Fig.* 1 *(b)). By combining the patterns for each codeword, we can rule out many solutions. For instance, the combination "should becnomially ars an ele" can be eliminated due to the invalid word "becnomially". In fact, the only combination without invalid words (without considering words on the boundary of the string, which might be part of a longer word) is "Flash memory is an ele", so the NR-Decoder can recover all six erasures in the three codewords. (In practice, it is also possible that we get more than one combination that contain only valid words. In that case, an erased bit can be corrected if all such combinations set the same value for that erasure.)*

Suppose that the LZW-codewords, seen as information bits, are protected by a systematic ECC. Then the ECC-Decoder can correct erasures by parity-check constraints, and the NR-Decoder can correct erasures by NR. They can work collaboratively to maximize the number of correctable erasures. □

As this paper is largely motivated by language-based NR, it is worthwhile to note that an LZW algorithm with a dictionary of 2^{20} patterns (as in the above example) can compress the English language to 2.94 bits per character. The UNIX Compress command uses LZW with a smaller dictionary and so achieves a lower compression ratio. There are compression algorithms for languages with higher compression ratios (e.g., syllable-based Burrows-Wheeler Transform achieving 2 bits/character [6]). However, there is still a gap toward Shannon's estimation of 1.34 bits/character for the entropy of English [16], which gives motivation for NR-Decoders. And one may reasonably conjecture that a similar scenario exists for images and videos.

In this work, we study the utilization of NR for erasure correction, including for languages and images. The paper is organized as follows. In Section II, we survey related works. In Section III, we introduce the discovery and utilization of NR in data for erasure correction, including for languages and

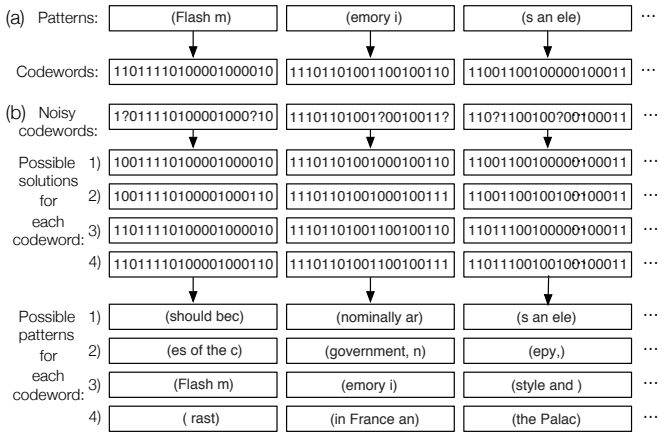

Fig. 1. (a) Compress a text by LZW. (b) NR-decoding for erasures.

images. In Section IV, we study a scheme that combines NR-decoding with low-density parity-check (LDPC) codes, and derive analytical formulas for the density evolution of LDPC decoding given information from the NR-decoder, which are useful for measuring the overall decoding performance. In Section V, we propose a theoretical model for compressed languages, and study the performance of iterative decoding between the LDPC decoder and the NR-decoder. In Section VI, we present further analysis on the performance of NR decoding for general ECCs. In Section VII, we present the conclusions.

II. Related Works

Error-correction with NR is related to joint source-channel coding and denoising. The idea of using the inherent redundancy in a source – or the leftover redundancy at the output of a source encoder – to enhance the performance of the ECC has been studied within the field of joint source-channel coding. In [3], source-controlled channel coding using a soft-output Viterbi algorithm is considered. In [1], a trellis based decoder is used as a source decoder in an iterative decoding scheme. Joint decoding of Huffman and Turbo codes is proposed in [2]. In [4], joint decoding of variable length codes (VLCs) and convolutional/Turbo codes is analyzed. Joint decoding using LDPC codes for VLCs and images are illustrated in [13] and [14], respectively. 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 joint source-channel coding, denoising is also an interesting and well studied technique [8], [11], [12], [15], [20]. A denoiser can use the statistics and features of input data to reduce its noise level for further processing. For discrete memoryless channels with stationary input sequences, a universal algorithm that performs asymptotically as well as optimal denoisers are given in [19]. The algorithm is also universal for a semi-stochastic setting, where the channel input is an individual sequence and the randomness in the channel output is solely due to the channel's noise.

Spell-checking softwares are a typical example of using NR to correct errors in languages. They are widely used in text editors. A spell-checking software usually works at the character level (namely, it does not consider how characters or text strings are encoded by bits), is for uncompressed texts, and uses the validity of words and the correctness of grammar to correct errors that appear in the typing of texts.

Using NR to correct errors at the bit level in compressed texts has been studied in a number of works. In [7], texts compressed by Huffman coding is considered, and a dynamic programming algorithm is used to partition the noisy bit sequence into subsequences that represents words, and to select likely solutions based on the frequencies of words and phrases. In [5], texts that are compressed by Huffman coding and then protected by LDPC codes are studied. An efficient greedy algorithm is used to decompress the noisy bit string, and partition it into stable and unstable regions based on whether each region contains recognizable words and phrases. The stable and unstable regions have polarized RBERs, which are provided as soft information to the LDPC code for better decoding performance. The algorithm is enhanced in [9] by a machine learning method for content recognition, and an iterative decoding algorithm between the NR-Decoder and the ECC-Decoder is used to further improve performance. In [18], texts compressed by Huffman coding and protected by Polar codes are studied. The validity of words is used to prune branches in a list sequential decoding algorithm, and a trie data structure for words is used to make the algorithm more efficient. A concatenated-code model that views the text with NR as the outer code and the Polar code as the inner code is considered, and the rate improvement for the Polar code due to NR is analyzed. That model is further studied in [17], where an optimal algorithm that maximizes the code rate improvement by unfreezing some frozen bits to store information is presented. A model that views NR as the output of a side information channel at the channel decoder is also studied, where NR is shown to improve the random error exponent.

III. NR-Decoding for Languages and Images

In this section, we present techniques for using NR in compressed data, including languages and images, for correcting erasures.

A. NR-Decoding for Language

Consider English texts that are compressed by an LZW algorithm that uses a fixed dictionary of size 2^ℓ. We have introduced a technique that corrects bit erasures based on the validity of words in Example 1. For long compressed texts with erasures, to make the NR-decoding efficient, we use a decoding algorithm based on sliding-windows of variable lengths as follows. Let n_{min} and n_{max} be two integers, where $n_{min} < n_{max}$. We first use a sliding-window of $n_{min}\ell$ bits to scan the compressed text (where every such window

contains exactly n_{min} LZW-codewords), and obtain candidate solutions for each window based on the validity of words (as in Example 1). We then increase the size of the window to $(n_{min}+1)\ell$, $(n_{min}+2)\ell$, $\cdots$, $n_{max}\ell$, and do decoding for each size in the following way: consider a window of $k\ell$ bits that contains k LZW-codewords $C_1, C_2, \cdots, C_k$. Let $S_1 \subseteq \{0,1\}^{(k-1)\ell}$ be the set of candidate solutions for the sub-window that contains the LZW-codewords $C_1, C_2, \cdots, C_{k-1}$; and let $S_2 \subseteq \{0,1\}^{(k-1)\ell}$ be the set of candidate solutions for the sub-window that contains the LZW-codewords $C_2, C_3, \cdots, C_k$. (Both S_1 and S_2 have been obtained in the previous round of decoding.) We now obtain the set of candidate solutions for the current window, which contains C_1, C_2, $\cdots$, C_k, this way. A bit sequence $(b_1, b_2, \cdots, b_{k\ell})$ is in S only if it satisfies two conditions: (1) its first $(k-1)\ell$ bits are a solution in S_1, and its last $(k-1)\ell$ bits are a solution in S_2; (2) the decompressed text corresponding to it contains no invalid words (except on the boundaries). This way, potential solutions filtered by smaller windows will not enter solutions for larger windows, making decoding more efficient. As a final step, an erased bit is decoded this way: if *any* of the windows of size $n_{max}\ell$ containing it (note that there are up to $2n_{max}-1$ such windows) can recover its value (as we did in Example 1), decode it to that value; otherwise it remains as an erasure.

To make the above decoding algorithm more efficient, we also use phrases (such as "information theory", "flash memory") and features such as word/phrase lengths. If a solution for a window contains a valid word or phrase that is particularly long, we may remove other candidate solutions that contain only short words. That is because long words and phrases are very rare: their density among bit sequences of the same length decreases exponentially fast as the length increases. So if they appear, the chance that they are the correct solution is high based on Bayes' rule. The thresholds for such word/phrase lengths can be set sufficiently high such that the probability of making a decoding error is sufficiently small.

We also enhance the decoding performance by using the *co-location* relationship. Co-location means that certain pairs of words/phrases appear unusually frequently in the same context (because they are closely associated), such as "dog" and "bark", or "information theory" and "channel capacity". If two words/phrases with the co-location relationship are detected among candidate solutions for two windows close to each other, we may keep them as candidate solutions and remove other less likely solutions. The reason for this approach is similar to that for long words/phrases. The co-location relationship can appear in multiple places in a text, and therefore help decoding in non-trivial ways. For example, for the text in Fig. 2 (a), the words/phrases that have the co-location relationship with the phrase "flash memory" are shown in Fig. 2 (b). (All of them appear in this text.) How to find words/phrases with the co-location relationship from a corpus of training texts is a well-known technique in Natural Language Processing (NLP) [10]. So we skip its details here.

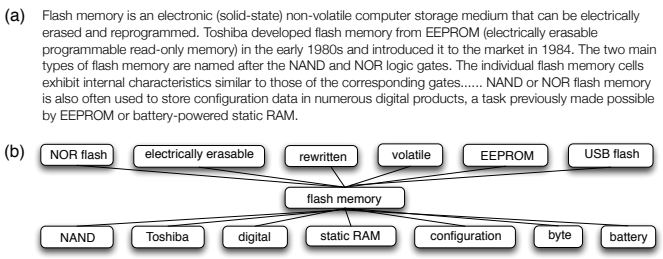

Fig. 2. (a) A sample paragraph from Wikipedia (part of which was omitted to save space). (b) Phrases in it that have the co-location relationship with "flash memory".

B. NR-Decoding for Images

Consider the discovery of NR for images. General images can have global features, and using such redundancy for error correction can be difficult. To gain more insight into the nature of NR in images, we focus in particular on images of handwritten digits, as in Fig. 3 (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.

We now present an NR-decoder for images. It is illustrated in Fig. 3 (b). Assume that a compressed image has λ erasures. Out of the 2^λ possible candidate solutions, usually only a few decompress successfully. (For example, to decompress successfully, the bit sequence needs to end with a valid Huffman codeword. And errors may make it impossible.) To decode noisy images among the successfully decompressed images, we have trained a convolutional neural network for recognizing noisy images, and designed a specialized filter based on features of connected components in decompressed images, as follows:

1) Convolutional Neural Network: The training and test data consist of noisy as well as clean images of handwritten digits. It consists of one input layer, two hidden layers and a output layer. The input layer consists of a 28×28 bilevel image, and the 2×1 output layer classifies the input images as "clean" or "noisy". The size of the convolution window is 5×5. The number of feature maps used in the first and second hidden layers are 5 and 15 respectively.

2) Filter Based on Connected Components: We count the number of components in an image, but without counting those components that have at most two pixels or components that are vertical lines (which may be caused by human or scanning errors). The images that have the fewest components are accepted as candidate images by this filter.

3) Joint Decoder: The final step of decoding is: if all candidate solutions agree on the value an erased bit, set the bit to that value; otherwise, keep it as an erasure.

Example 2. *Suppose that the compressed image with erasures*

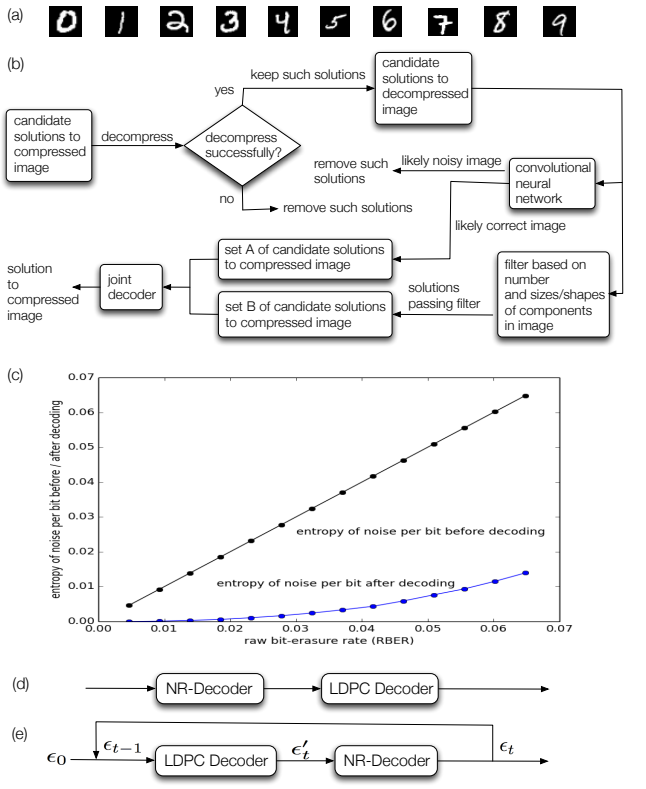

Fig. 3. (a) Examples of handwritten digits. (b) NR-decoder for images. (c) Performance of NR-decoder. (d) A concatenated decoding scheme. (e) An iterative decoding scheme.

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. □

C. Decoding Performance of NR decoders

The decoding performance for NR decoders can be measured as follows. Let $\epsilon \in [0,1]$ be the erasure probability before decoding. After the decoding by natural redundancy, let $\delta \in [0,1]$ be the probability that an originally erased bit remains as an erasure, and let $\rho \in [0, 1-\delta]$ be the probability that an originally erased bit is decoded to 0 or 1 incorrectly. The amount of noise after NR-decoding can be measured by the entropy of the noise (erasures and errors) per bit: $E_{NR} \triangleq \epsilon(\delta + (1-\delta)H(\frac{\rho}{1-\delta}))$, where $H(p) = -p\log p - (1-p)\log(1-p)$ is the entropy function.

We show E_{NR} for the NR-decoder for images in Fig. 3 (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%.

The performance of the NR-decoder introduced above for LZW-compressed English texts, experimented on a large corpus of Wikipedia articles, is shown in the table below. It also reduces noise effectively (between 88.0% and 91.6%) for raw bit-erasure rate from 5% to 30%.

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}	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}	2.04×10^{-2}	2.76×10^{-2}	3.60×10^{-2}
Noise reduction	89.8%	89.0%	88.0%

IV. Combine NR-decoding with LDPC Codes

This section discusses the combination of NR-decoders described in the previous section 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. 3 (d) for an illustration.) 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. 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 $p_0(\epsilon_0) \in [0,1]$, becomes an error (0 or 1) with probability $(1-p_0(\epsilon_0))\gamma_0(\epsilon_0) \in [0, 1-p_0(\epsilon_0)]$, and is decoded correctly (as 0 or 1) with probability $(1-p_0(\epsilon_0))(1-\gamma_0(\epsilon_0))$. (In general, $p_0(\epsilon_0)$ and $\gamma_0(\epsilon_0)$ may be functions of ϵ_0. Note that if the NR-decoder decodes only information bits, and an erasure in the information bits remains as an erasure with probability $p_0(\epsilon_0)'$, then $p_0(\epsilon_0) = Rp_0(\epsilon_0)' + (1-R)$. Also note that the LDPC decoder needs to decode all bits with both errors and erasures.)

A. Decoding Algorithm

We design the following iterative LDPC decoding algorithm, which generalizes both the peeling decoder for BEC and the Gallager B decoder for BSC:

Algorithm 3. *Generalized LDPC decoding algorithm.*

(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.)

B. Density Evolution Analysis

We now analyze the density evolution for the decoding algorithm, for an infinitely long and randomly constructed LDPC code of regular degrees.

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 $\alpha_0 = \epsilon_0(1 - p_0(\epsilon_0))\gamma_0(\epsilon_0)$ and $\beta_0 = \epsilon_0 p_0(\epsilon_0)$. Let $\kappa_0 = \epsilon_0(1 - p_0(\epsilon_0))(1 - \gamma_0(\epsilon_0))$.

Theorem 4. *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 $\nu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m}B_t^m(1-A_t-B_t)^{d_v-1-m}$.*

Proof: Consider the root variable node of a computation tree. After t iterations, let A_t denote the probability that an incoming message to the root node from a neighboring check node is an error, and let B_t denote the probability that the message is correct. Then $1 - A_t - B_t$ is the probability that the message is an erasure. Let μ_t (respectively, ν_t) be the probability that among the $d_v - 1$ incoming messages from neighboring check nodes to the root node, π or more messages are errors (respectively, correct) and the remaining messages are all erasures.

In the $(t+1)$-th iteration, we can have an error in the root node in one of the following cases:

1) The root node was initially (namely, before decoding begins) an error (which has probability α_0), and either of the two disjoint events happens: 1) fewer than τ check-node messages are correct and the remaining messages are all erasures, which happens with probability $\sum_{i=0}^{\tau-1}\binom{d_v-1}{i}B_t^i(1-A_t-B_t)^{d_v-i-1}$; 2) at least one check-node message is an error, which happens with probability $1-(1-A_t)^{d_v-1}$. The probability that either of the two events occurs is $C_t = 1-(1-A_t)^{d_v-1} + \sum_{i=0}^{\tau-1}\binom{d_v-1}{i}B_t^i(1-A_t-B_t)^{d_v-i-1}$.
2) The root node was initially correct (which has probability κ_0), but τ or more check-node messages are errors and the rest are all erasures (which happens with probability $D_t = \sum_{j=\tau}^{d_v-1}\binom{d_v-1}{j}A_t^j(1-A_t-B_t)^{d_v-1-j}$).
3) The root node was initially an erasure (which has probability β_0), and π or more check-node messages are errors and the rest are all erasures (which happens with probability μ_t).

Therefore the error rate after $t+1$ iterations will be $\alpha_{t+1} = \alpha_0 C_t + \kappa_0 D_t + \beta_0 \mu_t$.

In the $(t+1)$-th iteration, we can correct an erasure at a root node correctly if the root node was initially an erasure, and π or more check-node messages are correct and the rest are all erasures. This happens with probability $\beta_0\nu_t$. The root node will remain as an erasure if it is neither corrected mistakenly nor corrected correctly. So the erasure rate after $t+1$ iterations will be $\beta_{t+1} = \beta_0(1-\mu_t-\nu_t)$.

Now we need to find the values of A_t, B_t, μ_t and ν_t. The incoming message from a check node to the root node is correct if out of the $d_c - 1$ non-root variable nodes connected to the check node, an even number of nodes are errors and the rest are all correct (i.e., neither errors nor erasures). That probability is $B_t = \sum_{k=0}^{\lfloor\frac{d_c-1}{2}\rfloor}\binom{d_c-1}{2k}\alpha_t^{2k}(1-\alpha_t-\beta_t)^{d_c-1-2k} = \frac{(1-\beta_t)^{d_c-1}+(1-\beta_t-2\alpha_t)^{d_c-1}}{2}$. The incoming message from a check node to the root node is an error if out of the $d_c - 1$ non-root variable nodes connected to the check node, an odd number of nodes are errors and the rest are all correct. That probability is $A_t = \sum_{k=1}^{\lfloor\frac{d_c}{2}\rfloor}\binom{d_c-1}{2k-1}\alpha_t^{2k-1}(1-\alpha_t-\beta_t)^{d_c-2k} = \frac{(1-\beta_t)^{d_c-1}-(1-\beta_t-2\alpha_t)^{d_c-1}}{2}$. The probability that π or more neighboring check-node messages are errors and the rest are all erasures can be simplified as $\mu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m}A_t^m(1-A_t-B_t)^{d_v-1-m}$. The probability that π or more neighboring check-node messages are correct and the rest are all erasures can be simplified as $\nu_t = \sum_{m=\pi}^{d_v-1}\binom{d_v-1}{m}B_t^m(1-A_t-B_t)^{d_v-1-m}$. This completes the proof. ■

C. Erasure Threshold

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 $\bar{\epsilon}^* = 0.036$. 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%.

V. 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. 3 (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.

A. NR Decoder For Compressed Languages

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. This 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.

B. Iteration with LDPC Decoder

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. 3 (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 in any of the previous iterations). 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*. Lucky and unlucky segments will have different erasure rates during decoding, too.

Every l-information-bit segment is *lucky* with probability p, and *unlucky* with probability $1-p$. A lucky segment is guaranteed to be decoded successfully by the NR-decoder once the number of erasures in it becomes less than or equal to l_θ; and an unlucky segment can be considered as *never* to be decoded by the NR-decoder (because such decoding will not succeed). Since whether a segment is lucky or not is independent of the party-check constraints and the LDPC-decoder, for analysis we can consider it as an inherent property of the segment (which exists even before the decoding begins).

C. Density Evolution Analysis

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.

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.

Lemma 5.

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

Proof: Consider the LDPC-decoding and the NR-decoding in the first iteration. Since the initial erasure rate is ϵ_0, the erasure rate after LDPC decoding will now be $\epsilon'_1 = q_0 \epsilon_0 (1 - (1-\epsilon_0)^{d_c - 1})^{d_v - 1}$ where $q_0 = 1$ by definition. The probability that an l-information-bit segment contains exactly i erasures is given by $\binom{l}{i} (\epsilon'_1)^i (1-\epsilon'_1)^{l-i}$, which is independent of whether the segment is lucky or unlucky. Thus the probability that a *lucky* segment contains up to l_θ erasures is given by $\sum_{i=0}^{l_\theta} \binom{l}{i} (\epsilon'_1)^i (1-\epsilon'_1)^{l-i}$. All such segments are decoded by the NR-decoder successfully, while the remaining segments are not. That leads to the conclusion. ∎

Lemma 6. *The erasure rate after the first iteration of NR-decoding is*

$$\epsilon_1 = \epsilon_0 d_1 ((1-R) + R(1-p)) + (\sum_{k=l_\theta+1}^{l} \frac{k}{l} f_k(1)) Rp$$

Proof: After NR-decoding, the erasure rate of a lucky segment with k erasures is $\frac{k}{l}$, and the erasure rate for unlucky segments and parity-check bits is still ϵ'_1. We have $d_1 = \epsilon'_1 / \epsilon_0$. Hence the overall erasure rate after the 1st iteration of NR-decoding is $\epsilon_1 = \epsilon_0 d_1 ((1-R) + R(1-p)) + (\sum_{k=l_\theta+1}^{l} \frac{k}{l} f_k(1)) Rp$. (See Fig. 4 (b) for an illustration of the computation tree for density evolution. For comparison, we show the tree for classic BP decoding for BEC in Fig. 4 (a).) ∎

Lemma 7. *The erasure rate after the second iteration of LDPC-decoding is*

$$\epsilon'_2 = q_0 q_1 \epsilon_0 (1 - (1-\epsilon_1)^{d_c - 1})^{d_v - 1}$$

.

Proof: We have $q_1 = \frac{\epsilon_1}{\epsilon_1'}$. Since the NR-decoding of the 1st iteration reduces the *overall* erasure probability by a factor of q_1 (from ϵ_1' to ϵ_1), and the root variable node of a computation tree is chosen uniformly at random from the infinitely long and randomly constructed LDPC code, the root node in the tree for the 2nd iteration of LDPC decoding now has the erasure probability $q_1\epsilon_0$. (See Fig. 4 (b).) Hence the equation for the LDPC-decoder for the 2nd iteration will be given by $\epsilon_2' = q_0 q_1 \epsilon_0 (1-(1-\epsilon_1)^{d_c-1})^{d_v-1}$. Note that LDPC decoding is independent of NR-decoding because the parity-check constraints are independent of the bits being lucky-segment bits, unlucky-segment bits or parity-check bits. And note that $d_2 = \frac{\epsilon_2'}{\epsilon_1}$ is the probability that an erasure *remains as an erasure* after the LDPC decoding. If $d_2 = 1$, no change was made by the LDPC-decoder; if $d_2 = 0$, all erasures have been corrected. In both cases, the decoding will end. ■

Lemma 8. *For $t \geq 2$,*

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

Proof: Now consider the second iteration of NR-decoding. We only consider the case when $0 < d_2 < 1$. A lucky segment has zero errors after the second iteration if an only if either one of the two cases happen : a) that the segment already has zero errors after the first iteration, or b) the segment had $l_\theta + 1$ or more errors after the first iteration and it has at most l_θ erasures after second iteration of the LDPC-decoding. Thus if $k = 0$,

$$f_k(2) = f_k(1) + \sum_{i=l_\theta+1}^{l} \sum_{j=0}^{l_\theta} f_i(1)\binom{i}{j}(d_2)^j(1-d_2)^{i-j}$$

A lucky segment cannot have $k \leq l_\theta$ erasures (with $k \geq 1$) after the second iteration of NR-decoding (because if so, it would have corrected those erasures). So we have $f_k(2) = 0$ for that case. Finally, a lucky segment has $l_\theta + 1 \leq k \leq l$ erasures if and only if it had k or more erasures after the first iteration of NR-decoding and it has k erasures after the second iteration of LDPC-decoding. Thus

$$f_k(2) = \sum_{i=k}^{l} f_i(1)\binom{i}{k}(d_2)^k(1-d_2)^{i-k} \text{ if } l_\theta+1 \leq k \leq l$$

The remaining cases can be analyzed similarly. That leads to the conclusion. ■

We now present the analytical formulas for the density evolution of the iterative LDPC-NR decoding scheme. Its proof follows the previous lemmas.

Theorem 9. *For $t \geq 1$,*

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

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

Proof: The decoding performance for the 2nd iteration of the LDPC-decoding has been analyzed in Lemma 7. The erasure rate in unlucky-segment bits and parity-check bits was decreased from ϵ_1' to $\epsilon_1' d_2 = \epsilon_0 d_1 d_2$ by the LDPC-decoding. Now the NR-decoder corrects those lucky segments that had more than l_θ erasures before the LDPC-decoding but now has at most l_θ erasures after the LDPC-decoding. So $\epsilon_2 = \epsilon_0 d_1 d_2((1-R)+R(1-p)) + (\sum_{k=l_\theta+1}^{l} \frac{k}{l} f_k(2))Rp$.

The analysis for the following iterations is similar to the 2nd iteration. In general, since in the i-th iteration the NR-decoder reduces the overall erasure rate by a factor of q_i, the root variable node in the computation tree for the t-th iteration of LDPC decoding has the erasure probability $(\prod_{i=0}^{t-1} q_i)\epsilon_0$. That leads to the conclusion. ■

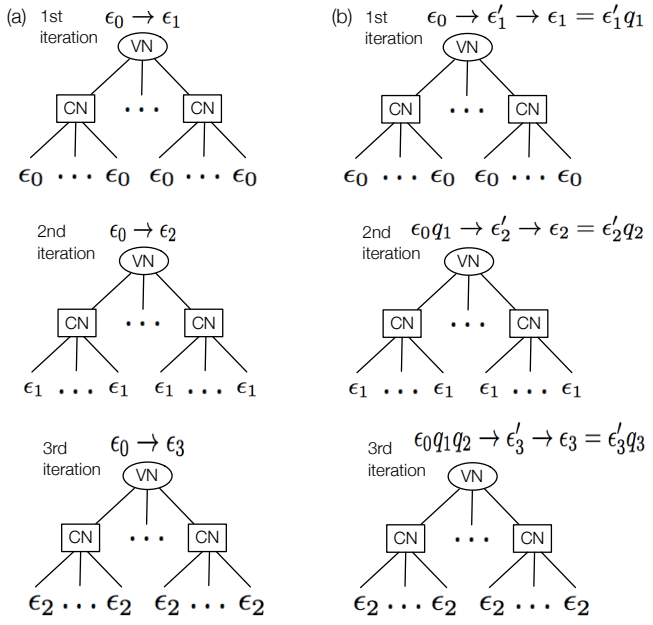

Fig. 4. (a) First three iterations of classic BP decoding (alone) for BEC. (b) First three iterations of BP-decoding and NR decoding.

VI. Upper Bound to ECC Sizes with NR

The previous analysis has been specifically for LDPC codes with belief-propagation decoding algorithms. Let us now consider more general ECCs and their capacity. The NR-decoders for images and languages presented in Section III 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 (similar to Fig. 3 (d)). 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 10. *For an (r, t)-ECC $\mathcal{C}$ with code length n, alphabet size q and $r + t \leq n$, the code's size*

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

Proof: Define $\mathcal{P}_r \triangleq \{P \mid P \subseteq \{1, 2, \cdots, n\},\ |P| = r\}$, and define $S_{t,r}(\mathbf{x}) = \cup_{P \in \mathcal{P}_r} S_{t,P}(\mathbf{x})$. It is not hard to see $|S_{t,P}(\mathbf{x})| = \sum_{i=0}^{t} \binom{n-|P|}{i} (q-1)^i$. Since $\forall\ P_1 \neq P_2$, $S_{t,P_1}(\mathbf{x}) \cap S_{t,P_2}(\mathbf{x}) = \emptyset$, we get $|S_{t,r}(\mathbf{x})| = \binom{n}{r} \sum_{i=0}^{t} \binom{n-r}{i} (q-1)^i$. We now show that $\mathcal{C}$ is an (r, t)-ECC if and only if for any two codewords $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}$, we have $S_{t,r}(\mathbf{x}_1) \cap S_{t,r}(\mathbf{x}_2) = \emptyset$: (1) If $S_{t,r}(\mathbf{x}_1) \cap S_{t,r}(\mathbf{x}_2) = \emptyset$, then for any $P \in \mathcal{P}_r$, since $S_{t,P}(\mathbf{x}_1) \subseteq S_{t,r}(\mathbf{x}_1)$ and $S_{t,P}(\mathbf{x}_2) \subseteq S_{t,r}(\mathbf{x}_2)$, we have $S_{t,P}(\mathbf{x}_1) \cap S_{t,P}(\mathbf{x}_2) = \emptyset$. So $\mathcal{C}$ is an (r, t)-ECC; (2) If $S_{t,r}(\mathbf{x}_1) \cap S_{t,r}(\mathbf{x}_2) \neq \emptyset$, then there exists some $P_1, P_2 \in \mathcal{P}_r$ such that $S_{t,P_1}(\mathbf{x}_1) \cap S_{t,P_2}(\mathbf{x}_2) \neq \emptyset$; and we must have $P_1 = P_2$ (otherwise the two sets are disjoint). So $\mathcal{C}$ is not an (r, t)-ECC. So by the sphere-packing bound, since there are totally $q^n \binom{n}{r}$ pairs of the form $(\mathbf{x}, P)$ where $\mathbf{x} \in \mathcal{A}^n$ and $P \in \mathcal{P}_r$, we get $|\mathcal{C}| \leq \frac{q^n \binom{n}{r}}{|S_{t,r}(\mathbf{x})|} = \frac{q^n \binom{n}{r}}{\binom{n}{r} \sum_{i=0}^{t} \binom{n-r}{i} (q-1)^i} = \frac{q^n}{\sum_{i=0}^{t} \binom{n-r}{i} (q-1)^i}$. ∎

VII. Conclusions

This paper studies the discovery and utilization of NR in data for error correction, including for languages and images. It proposes non-iterative and iterative coding schemes that combine NR-decoding with LDPC-decoding, and analyzes their performance. A sphere-packing upper bound is also shown for general ECCs that receive assistance from NR-decoders.

References

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

[2] 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.

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

[4] 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.

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

[6] J. Lansky, K. Chernik and Z. Vlckova. "Syllable-Based Burrows-Wheeler Transform," 2007.

[7] 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.

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

[9] 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.

[10] C. D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing*, MIT Press, 1999.

[11] E. Ordentlich, G. Seroussi, S. Verdu, and K. Viswanathan, "Universal Algorithms for Channel Decoding of Uncompressed Sources," *IEEE Trans. Information Theory*, vol.

[12] 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.

[13] 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.

[14] L. Pu, 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.

[15] 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.

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

[17] Y. Wang, K. R. Narayanan and A. Jiang, "Exploiting Source Redundancy to Improve the Rate of Polar Codes," in *IEEE International Symposium on Information Theory (ISIT)*, Aachen, Germany, June 2017.

[18] 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.

[19] 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.

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