To the Chairman of Examiners for Part III Mathematics.

Dear Sir,

I enclose the Part III essay of .....

Signed .....

(Director of Studies)

University Cambridge
Centre for Mathematical Sciences

Part III Essay
Block-Sorting Data Compression

Contents

1. Introduction

The following essay introduces basic concepts of information and coding theory with a view to optimal coding schemes. This leads to an optimal code in terms of length, the so-called Huffman Code, presented in chapter 2 including its proof of optimality. The shortest length code is also the optimal code for compressing single characters. This part of the essay is based on chapter 5 of [8]. While the chapter on Huffman coding only focusses on single character compression, significantly higher compression rates can be achieved by compressing larger substrings. The main algorithm in this area is the one developed by Lempel and Ziv ([4] and [5]), presented in chapter 3. It will be shown that Lempel-Ziv coding approaches the entropy of a given data stream and is therefore optimal in the limit. The part on Lempel-Ziv coding will follow chapter 12 of [8] and chapter 2 of [9]. A recent approach to lossless data compression is the so-called Burrows-Wheeler Transformation, presented in chapter 4. This reversible data transformation is not a compression scheme itself, but a pre-stage used to increase the redundancy of arbitrary data by block-sorting. The part on the Burrows-Wheeler Transformation is based on the original paper [1] by M. Burrows and D.J. Wheeler. In chapter 4 the transformation will be introduced and it will be shown that it is always reversible. Next, the essay focusses on the notion of entropy by proving a Central Limit Theorem for return times of blocks in a random process. One application of the theorem is its restatement as an asymptotic normal estimator of the entropy. Chapter 5 on the Central Limit Theorem is based on the original paper [10] by O. Johnson. To observe the effect of the Burrows-Wheeler Transformation, the compression rates of all three algorithms (Huffman compression, the Lempel-Ziv variant Lempel-Ziv-Welch and the Burrows-Wheeler Transformation) are compared with a popular compression scheme like the ZIP format. This is done in the last chapter 6 using own implementations of all algorithms and the standard compression benchmark [6].

2. Huffman Coding

2.1. Basic definitions

To introduce basic concepts of coding, the notion of a code and its main properties are stated first. This chapter follows chapter 5 in [8].

2.1 Definition:

1. *An alphabet A is a (finite or infinite) set of symbols.*
2. *The set of all non-empty words which can be built out of symbols from an alphabet A by concatenation is denoted by A^*.*
3. *A code C for a random variable X with range R over an alphabet A is a mapping which assigns a codeword $w \in A^*$ to every value of X, $C(X) : R \to A^*$.*
4. *The extension of a code is the mapping from R^* to A^*, defined as $C(x_1 \cdots x_n) := C(x_1) \cdots C(x_n)$.*
5. *The expected length L of a code C is defined to be the weighted average of the codeword lengths $l(\cdot)$ with respect to the probability $p(X = x)$ of a symbol $x \in R$: $L(C) := \sum_{x \in R} p(X = x) l(x)$, where $l(x) := l(C(x))$ is the length of the codeword assigned to x.*

So far, a code is just any mapping from a random variable to any set of words over an alphabet. In order to distinguish between codes that might be more useful than others for encoding information, additional notions are defined next.

2.2 Definition:

1. *A code is called non-singular if it maps injectively into the set A^*, i.e. $x_i \neq x_j \Rightarrow C(x_i) \neq C(x_j)$.*
2. *A code is called uniquely decodable if its extension is non-singular, i.e. if every concatenation of codewords $C(s) \in A^*$ corresponds to exactly one string $s = x_1 \cdots x_n \in R^*$.*
3. *A string $s \in A^*$ is called a prefix of a longer string $t \in A^*$ if s is the stem of t.*
4. *A code is called a prefix code (or an instantaneous code) if no codeword is the prefix of a longer codeword.*

The prefix condition makes it possible to decode a message in linear time: by starting from the first symbol in the encoded message and regarding all symbols sequentially, a substring s_0 is decoded as soon as it matches a codeword. Note that, if such a substring s_0 is not decoded immediately, no other codeword will be found afterwards, as no other codeword has s_0 as prefix by definition. This makes any prefix code uniquely decodable.

In order to encode or compress data reversibly, only uniquely decodable codes are of interest from now on. To illustrate those notions, a few examples follow.

2.3 Example:

1. *Over the alphabet* $A = \{0, 1\}$*, define a code* C *with codewords* $C(a) = 01$ *and* $C(b) = 111$*. Here,* $R = \{a, b\}$ *and if* a *occurs with probability* $\mathbb{P}(a) = 0.8$ *and* b *with* $\mathbb{P}(b) = 0.2$*, then* C *has an expected length of* $L(C) = 0.8 * 2 + 0.2 * 3 = 2.2$*.*

2. *Over the alphabet* $A = \{0, 1\}$ *consider the following codes for the three symbols* $\{x, y, z\} = R$*:*

Symbol	C_1	C_2	C_3	C_4
x	*1*	*0*	*00*	*0*
y	*1*	*1*	*11*	*10*
z	*1*	*01*	*001*	*11*

   Code C_1 *is clearly singular and* C_2 *is not uniquely decodable as 01 could be xy or z.* C_3 *is an example of a uniquely decodable code, which can be proven by induction. The main idea is that if a substring 001 appears in the message, the number of 1's until the next 0 appears decides on how to decode 001: if the number is even, then 001...1 has been x and several y's, if the number is odd, then 001...1 has been z and several y's. Code* C_3 *is not prefix since 00 appears as stem of 001.* C_4 *is clearly a prefix code and therefore uniquely decodable, e.g. 11100 can only be decoded as z (11 is the first substring that matches a symbol), y (10) and x (0).*

The second example shows that prefix codes are a real subset of uniquely decodable codes, i.e. there are more codes which are uniquely decodable than there are prefix codes. However, the relationship between uniquely decodable and prefix codes is reviewed in greater detail in the next section.

2.2. The Kraft Inequality

An important property of prefix codes was discovered by Leon Kraft in 1949. It gives a necessary condition for the existence of prefix codes and shows how to construct prefix codewords of given lengths.

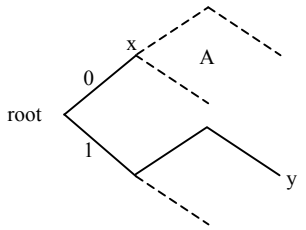
2.4 Theorem: *(Kraft Inequality) Let* C *be any code over an alphabet* A *of size* $S = |A|$ *consisting of* n *codewords of lengths* $l_1, \ldots, l_n$*. Then,*

$$C \text{ is a prefix code} \Rightarrow \sum_{i=1}^{n} S^{-l_i} \leq 1.$$

Conversely, given any numbers $l_1, \ldots, l_n$ *satisfying the Kraft Inequality* $\sum_{i=1}^{n} S^{-l_i} \leq 1$*, there exists a prefix code* $C = \{c_1, \ldots, c_n\}$ *with lengths* $l(c_i) = l_i$*.*

Proof. Let C be any given code with codeword lengths $l_1, \ldots, l_n$ and let l_{max} be the length of the longest codeword. Insert all codewords in a S-ary tree by starting from the root and following the branches according to the digits in the codeword. As C is a prefix code, once a codeword x is inserted, the whole subtree underneath it (subtree A for word x in the following figure) is ruled out for further codewords as they all have x as prefix. Consider now the number of leaves the full tree would have at level l_{max}. This is $S^{l_{max}}$. On the other hand, the subtree A of codeword x has depth $l_{max} - l(x)$ and hence $S^{l_{max}-l(x)}$ leaves at the bottom level.

All leaves at the bottom level of the subtrees for each symbol also belong to the set of leaves the full tree has at level l_{max}. As every subtree underneath a codeword is ruled out for further codewords, all those subsets of leaves are disjoint, and its sum can't be greater than the number of leaves at level l_{max} in the full tree: $\sum_{k=1}^{n} S^{l_{max}-l_k} \leq S^{l_{max}}$. Dividing by $S^{l_{max}}$ gives the Kraft Inequality $\sum_{k=1}^{n} S^{-l_k} \leq 1$.



Conversely, start with a full S-ary tree of depth l_{max}. By assumption, $\sum_{i=1}^{n} S^{-l_i} \leq 1$, so $\sum_{i=1}^{n} S^{l_{max}-l_i} \leq S^{l_{max}}$ after multiplying with $S^{l_{max}}$. As before, this means that the full tree is big enough to contain disjoint subtrees of depth $l_{max} - l_i$ for every codeword of length l_i and therefore contains n nodes at levels $l_1, \ldots, l_n$. To obtain the actual codewords, simply assign $c_1, \ldots, c_n$ with lengths $l_1, \ldots, l_n$ to the corresponding nodes at levels $l_1, \ldots, l_n$ in any order and interpret the paths from the root to the nodes as codewords. This code is prefix free since all subtrees of depth $l_{max} - l_i$ are disjoint. □

Surprisingly, exactly the same statement is true not only for prefix codes, but also for the larger set of uniquely decodable codes. This was proved by McMillan [11] in 1956.

2.5 Theorem: *(McMillan) Any uniquely decodable code C over an alphabet of size S satisfies the Kraft Inequality, i.e. $\sum_{i=1}^{n} S^{-l_i} \leq 1$, where l_i are the lengths of the codewords. Conversely, if numbers $l_1, \ldots, l_n$ are given with $\sum_{i=1}^{n} S^{-l_i} \leq 1$ there exists a uniquely decodable code $C = \{c_1, \ldots, c_n\}$ with lengths $l(c_i) = l_i$.*

Proof. Let C be a given code with codeword lengths $l_1, \ldots, l_n$ and l_{max} be the length of the longest codeword. Note that it is possible to rewrite the sum $\sum_{i=1}^{n} S^{-l_i}$ as $\sum_{l=1}^{l_{max}} n_l S^{-l}$, where n_l denotes the number of codewords of length l.

For any $m \geq 1$, consider C^m, the code which consists of all concatenations of m words of C. As C is uniquely decodable, every codeword in C^m corresponds to exactly one concatenation of m words $c_1 \cdots c_m$ (and vice versa), where $c_i \in C$. It follows that

$$\left(\sum_{i=1}^{n} S^{-l_i} \right)^m = \sum_{i_1=1}^{n} \cdots \sum_{i_m=1}^{n} S^{-(l_{i_1}+\cdots+l_{i_m})} = \sum_{l=1}^{m \cdot l_{max}} \tilde{n}_l S^{-l}, \tag{1}$$

where $\tilde{n}_l$ is the number of codewords of length l in C^m. The first equality here is just the expanded product of sums. The second equality is true since all codewords in C^m correspond 1:1 to concatenations of m words of C, so the sum over all possible $l_{i_1}+\cdots+l_{i_m}$ contains exactly the same codelengths as the sum over all lengths of codewords in C^m. The maximal codelength of C^m is m times the maximal length of C, hence the sum is indexed by $l = 1, \ldots, m \cdot l_{max}$.

Clearly, the number of codewords $\tilde{n}_l$ of length l cannot be greater than the number of all strings which consist of l symbols out of alphabet A, so $\tilde{n}_l \leq S^l$. Substituting this into (1) and taking the mth root yields

$$\sum_{i=1}^{n} S^{-l_i} \leq \left(\sum_{l=1}^{m \cdot l_{max}} S^l S^{-l} \right)^{1/m} = (m \cdot l_{max})^{1/m}.$$

Since m was arbitrary, the inequality also holds in the limit $m \to \infty$. As l_{max} is constant, $(m \cdot l_{max})^{1/m} \to 1$ as $m \to \infty$ and therefore $\sum_{i=1}^{n} S^{-l_i} \leq 1$, which proves the Kraft Inequality for uniquely decodable codes.

Conversely, given numbers $l_1, \ldots, l_n$ with $\sum_{i=1}^{n} S^{-l_i} \leq 1$, use the known Kraft Inequality to even get a prefix code $C = \{c_1, \ldots, c_n\}$ with lengths $l(c_i) = l_i$, which is also uniquely decodable. □

Both theorems together establish a surprising result: Given any prefix code C, C is trivially also a uniquely decodable code. Conversely, given any uniquely decodable code C' with codeword lengths $l_1, \ldots, l_n$, the l_i satisfy $\sum_{i=1}^{n} S^{-l_i} \leq 1$ by McMillan. Using the converse of Kraft there exists a prefix code C'' with exactly the same codeword lengths as the codewords in C' and hence the same expected length, i.e. $L(C') = L(C'')$. This means that the set of prefix codes is as powerful for encoding as the larger set of uniquely decodable codes.

2.3. The construction of an optimal code

Finally, an optimal code C in terms of its expected length shall be constructed. This is done with a view to an actual application on a computer, thus the binary case is considered in this section. The optimal code can be used to encode data with minimal expected length and hence achieves compression. Note that, by the theorems of Kraft and McMillan, it can be assumed without loss of generality that the optimal code is a prefix code. The following theorem states properties of an optimal code.

2.6 Theorem: *Let C be an optimal code over a binary alphabet to encode symbols $\{b_1, \ldots, b_n\}$, where symbol b_i occurs with probability p_i. Then the lengths l_i of the codewords $C = \{c_1, \ldots, c_n\}$, $c_i \in \{0,1\}^*$ must satisfy the following properties.*

1. *If $p_r > p_s$ for any $r, s \in \{1, \ldots, n\}$, then $l_r \leq l_s$.*

2. *The two longest codewords have equal length.*

3. *There is a pair of two longest codewords which are siblings in a tree representation.*

Proof.

1. Let C be a code of minimal length $L(C) = \sum p_i l_i$ for encoding the symbols b_i. Define another code C' to be code C, except that the symbol indices r and s are switched. As C is optimal, $L(C) \leq L(C')$ and therefore $0 \leq L(C') - L(C) = p_r l_s + p_s l_r - p_r l_r - p_s l_s = (l_s - l_r)(p_r - p_s)$. If $p_r > p_s$ it follows that $l_s - l_r \geq 0$ and hence $l_r \leq l_s$.

2. Suppose that the two longest codewords don't have equal length. Let c be the longest codeword. As C is a prefix code, there is no shorter codeword on the path from the root to c in the tree representation of C. Therefore the last branch (=last digit) of the codeword c can be deleted without destroying the prefix property which gives a shorter code. Contradiction to the optimality of C.

3. If no pair of two longest codewords are siblings in the tree, then there exists a maximal length codeword c without a sibling (if it had one, its sibling would have maximal length as well). Therefore the last branch (=last digit) of c can be deleted again without destroying the prefix property which gives a shorter code. Contradiction to the optimality of C.

□

This theorem can directly be applied to inductively construct an optimal code over a binary alphabet, the so-called Huffman code:

Construction of the Huffman code

1. The two least likely symbols will have the longest code lengths by property (1) and are siblings by (3), so they're joined into a supernode with added probability. This ensures that they will have equal length at the end as required by (2).

2. The tree now consists of one node less, so repeat the first step until only one node remains and the tree (called Huffman tree) is constructed.

3. Once the tree is constructed, label the two branches going out of every node with the symbols 0 and 1 and read off the codewords by following the paths from the root to the leaves. This is the Huffman code.

4. Replacing every symbol of the input by its Huffman code then yields a message length which is equal or shorter than the original length when compared in a joint alphabet. No special symbol for separating the codes is needed as the Huffman code is a prefix code.

Empirical results on compression rates using Huffman codes are shown in chapter 6.

3. Lempel-Ziv Coding

3.1. Description of the coding scheme

The Huffman code was proven to be optimal for single characters, but even higher compression rates can be achieved when encoding substrings. This is done by the so-called Lempel-Ziv coding. This chapter follows chapter 12 in [8] and chapter 2 in [9].

3.1 Definition: *A parsing of a string s is a decomposition of s into n substrings s_i such that $s = s_1 \cdots s_n$. A parsing is called distinct if all s_i are distinct.*

Like in the last section on the construction of the Huffman code, inputs over a binary alphabet are considered in this chapter. All symbols are denoted again by $a_i \in \{0, 1\}$. The Lempel-Ziv coding combines string parsing with a prefix property:

Lempel-Ziv coding

1. Given a binary input string $s = a_1 \cdots a_n$ of length n, s is parsed sequentially into substrings s_i as follows: if $s_1 \cdots s_j$ have already been processed, the next substring s_{j+1} is the shortest string which hasn't occured so far. As an example, 1011111 is parsed as $1, 0, 11, 111$. Note that the shortest new string s_{j+1} always extends exactly one previous substring, say s_k, by just one bit b. Therefore, s_k is a prefix of s_{j+1} and s_{j+1} can be expressed as (s_k, b). By construction no string equals a previous one, hence the Lempel-Ziv parsing is distinct.

2. Let the parsing of the message consist of $\kappa(n)$ substrings. To encode a tupel (s_j, b), it suffices to write the index of the prefix s_j and the new bit b. This takes $\log \kappa(n) + 1$ bits.

3. To encode the whole string, all $\kappa(n)$ parsed substrings have to be encoded, which takes $\kappa(n) \cdot (\log \kappa(n) + 1)$ bits.

3.2. Statement and proof of the main theorem

It will be shown in this section that the code length per symbol $\frac{\kappa(n) \cdot (\log \kappa(n) + 1)}{n}$ converges to the entropy of the input sequence $a_1 \cdots a_n$ as $n \to \infty$. As a first preparation, the Lemma by Lempel and Ziv bounds the number of substrings $\kappa(n)$ in a distinct parsing.

3.2 Lemma: *(Lempel and Ziv) The number of substrings $\kappa(n)$ in a distinct parsing of a binary string $s = a_1 \cdots a_n$ satisfies*

$$\kappa(n) \leq \frac{n}{(1 - \epsilon_n) \log n},$$

where $\epsilon_n \to 0$ as $n \to \infty$.

Proof. Suppose that for $m \geq 1$, a string s_m is constructed by concatenating all binary strings of lengths $i = 1, \ldots, m$ in any order. There are 2^i binary strings of length i, so s_m has length $l_m := \sum_{i=1}^{m} i2^i = (m-1)2^{m+1} + 2$ (using a standard formula which is easily proven by induction). The number of substrings $\kappa(n)$ in a distinct parsing of a string of length n is clearly maximized if all parsed substrings are as short as possible. In the case of length $n = l_m$, a binary parsing with a maximal number of substrings would start with the two length 1 substrings, then four length 2 substrings and so on. Therefore,

$$\kappa(l_m) \leq \sum_{i=1}^{m} 2^i = 2^{m+1} - 2 < 2^{m+1} < \frac{l_m}{m-1}, \tag{2}$$

where it was used that $l_m > l_m - 2 = (m-1)2^{m+1}$, so $2^{m+1} < \frac{l_m}{m-1}$. This means that a string of length l_m can be parsed into at most $\frac{l_m}{m-1}$ substrings.

For an arbitrary string s of length n given in the statement of the lemma, choose m such that $l_m \leq n < l_{m+1}$ and devide s into two parts $s = s_1 s_2$, where s_1 has length l_m. By (2) the maximal number of substrings in a parsing of s_1 is bounded by $\frac{l_m}{m-1}$. The remaining part s_2 of length $r = n - l_m$ is then parsed into a maximal number of $\frac{r}{m+1}$ substrings by using the minimal available substring length $m+1$. This is possible since $n < l_{m+1}$ and l_{m+1} is the length of a string consisting of all substrings up to length $m+1$. Hence,

$$\kappa(n) \leq \frac{l_m}{m-1} + \frac{r}{m+1} \leq \frac{l_m}{m-1} + \frac{n - l_m}{m-1} = \frac{n}{m-1}. \tag{3}$$

Finally, m is bounded in (3) in order to obtain the required bound for $\kappa(n)$. Firstly, by definition of l_m,

$$n \geq l_m = \sum_{i=1}^{m} i2^i \geq m2^m \geq 2^m,$$

so $m \leq \log n$, where $\log = \log_2$. Secondly, using $m \leq \log n$,

$$n < l_{m+1} = m2^{m+2} + 2 \leq (m+2)2^{m+2} \leq (\log n + 2)2^{m+2}.$$

Dividing by $(\log n + 2)$, taking the logarithm and subtracting 3 gives

$$m - 1 \geq \log \frac{n}{\log n + 2} - 3. \tag{4}$$

Using the fact that $\log n + 2 \leq 2 \log n \ \forall n \geq 4$ (proof by induction as $n = 4 \Rightarrow 4 \leq 4$ and $\log(n+1) + 2 = \log \frac{n+1}{n} + \log n + 2 \leq 2 \log \frac{n+1}{n} + 2 \log n = 2 \log(n+1)$), inequation (4) can be extended as follows:

$$\begin{aligned} m - 1 &\geq \log \frac{n}{\log n + 2} - 3 = \log n - \log(\log n + 2) - 3 \\ &= \log n \left(1 - \frac{\log(\log n + 2) + 3}{\log n}\right) \geq \log n \left(1 - \frac{\log(2 \log n) + 3}{\log n}\right) \\ &= \log n \left(1 - \frac{\log(\log n) + 4}{\log n}\right) = (1 - \epsilon_n) \log n, \end{aligned} \tag{5}$$

where $\epsilon_n = \frac{\log(\log n)+4}{\log n} \to 0$ as $n \to \infty$. Inserting (5) into (3) proves the claimed bound $\kappa(n) \leq \frac{n}{(1-\epsilon_n)\log n}$. □

Before proving Ziv's Inequality, which is the starting point for the proof of the main theorem, a couple of facts about stochastic processes are introduced next. Stochastic processes are considered in this section as an arbitrary input source of characters is modelled by a stationary ergodic process in the following proofs.

3.3 Definition:

1. *Over a probability space* $(\Omega, S, \mathbb{P})$ *with sample space* Ω*,* σ*-algebra* S *and probability measure* $\mathbb{P}$*, a stochastic process* Z *is a set of random variables* $\{Z_i\}_{i\in I} = \{Z_i : \Omega \to \Omega' | i \in I\}$ *indexed by* I*. In this section, it is assumed that* $I = \mathbb{Z}$*. The joint distribution function of the random variables* $Z_{t_1}, \ldots, Z_{t_k}$ *is denoted by* $\mathbb{P}_{Z_{t_1},\ldots,Z_{t_k}}(z_{t_1}, \ldots, z_{t_k})$*.*
2. *A process* $\{Z_i\}_{i\in\mathbb{Z}}$ *is called stationary if its joint distribution function is invariant when shiftet in time, i.e.* $\mathbb{P}_{Z_{t_1+s},\ldots,Z_{t_k+s}}(z_{t_1}, \ldots, z_{t_k}) = \mathbb{P}_{Z_{t_1},\ldots,Z_{t_k}}(z_{t_1}, \ldots, z_{t_k})\ \forall s$*.*
3. *A process* $\{Z_i\}_{i\in\mathbb{Z}}$ *is called ergodic if a sufficiently long sample (realization) reflects its statistical properties, i.e. mean and variance.*
4. *The entropy* H *of a discrete random variable* X *is* $H(X) := -\sum_x p(x)\log p(x)$*, where* X *takes its values* x_i *with probabilities* $p(x_i)$*. For a continuous* X *with density* $f(x)$*, the entropy is defined as* $H(X) := -\int f(x)\log f(x)dx$*.*
5. *The relative entropy is defined as* $D(p||q) := \sum_x p(x)\log\frac{p(x)}{q(x)}$ *in the discrete case and as* $D(f||g) := \int f(x)\log\frac{f(x)}{g(x)}dx$ *in the continuous case, where* p, q *are probability densities and* f, g *are density functions.*
6. *The entropy rate of a stochastic process* $Z = \{Z_i\}_{i\in\mathbb{Z}}$ *is defined as the limit* $H(Z) := \lim_{n\to\infty}\frac{1}{n}H(Z_1, \ldots, Z_n)$*, where* $(Z_1, \ldots, Z_n)$ *denotes a* n*-dimensional random variable.*
7. *For a process* $\{Z_i\}_{i\in\mathbb{Z}}$*, the* l*th order Markov approximation* M_l *of* $\mathbb{P}$ *is defined as* $M_l(Z_{-(l-1)}, \ldots, Z_0, \ldots, Z_n) := \mathbb{P}(Z^0_{-(l-1)})\prod_{i=1}^n \mathbb{P}(Z_i|Z_{i-l}^{i-1})$ *for all* n*, where* Z_a^b *denotes the vector* $(Z_a, \ldots, Z_b)$ *and* $Z^0_{-(l-1)}$ *is an initial configuration.*

3.4 Lemma: *(Properties)*

1. *The relative entropy is non-negative.*
2. *The density* $\tilde{f}$ *which maximizes the entropy* H *subject to the constraints* $f(x) \geq 0$*,* $\int f(x)dx = 1$ *and* $\mathbb{E} = \int xf(x)dx = \mu$ *has the form* $\tilde{f}(x) = e^{a+bx}$*, where* a *and* b *are chosen to satisfy the three constraints.*

3. *Let X be a random variable taking values in $\mathbb{N}_0$ with mean μ. Then its entropy is bounded by $H(X) \leq (\mu+1)\log(\mu+1) - \mu\log\mu$.*

4. *For random variables X and Y, the entropy of (X,Y) is bounded by the sum of $H(X)$ and $H(Y)$, i.e. $H(X,Y) \leq H(X) + H(Y)$.*

5. *Let $H(Z)$ denote the entropy of the process $Z = \{Z_i\}_{i\in I}$. The Markov approximation satisfies $\lim_{n\to\infty} -\frac{1}{n}\log M_l(Z_1,\ldots,Z_n|Z_{-(l-1)}^0) = H(Z_0|Z_{-l}^{-1})$ and additionally, $\lim_{l\to\infty} H(Z_0|Z_{-l}^{-1}) = H(Z)$.*

Proof.

1. Using Jensen's Inequality for the convex function $-\log$,

$$\begin{aligned} D(p||q) &= \sum_x p(x)\log\left(\frac{p(x)}{q(x)}\right) = \sum_x p(x)\left[-\log\left(\frac{q(x)}{p(x)}\right)\right] \\ &\geq -\log\left(\sum_x p(x)\frac{q(x)}{p(x)}\right) = -\log\left(\sum_x q(x)\right) = -\log 1 = 0, \end{aligned}$$

as $q(x)$ is a probability density. The same statement for $D(f||g)$ is proven similarly:

$$D(f||g) = \int f\left[-\log\left(\frac{g}{f}\right)\right] \geq -\log\left(\int f\frac{g}{f}\right) = -\log\left(\int g\right) = -\log 1 = 0.$$

2. Let f be any other density which satisfies all three constraints. Then

$$H(f) = -\int f\log f = -\int f\left(\log\frac{f}{\tilde{f}} + \log\tilde{f}\right) = -D(f||\tilde{f}) - \int f\log\tilde{f}.$$

Using the definition of $\tilde{f}(x) = e^{a+bx}$ and $D(f||\tilde{f}) \geq 0$ yields

$$\begin{aligned} H(f) &\leq -\int f(x)(a+bx)dx = -a\int f(x)dx - b\int xf(x)dx = -a - b\mu \\ &= -a\int \tilde{f}(x)dx - b\int x\tilde{f}(x)dx = -\int \tilde{f}\log\tilde{f} = H(\tilde{f}), \end{aligned}$$

where it was used that f and $\tilde{f}$ satisfy the constraints.

3. By (2) it is known that the density of a random variable with given mean μ which maximizes the entropy is of the form $f(x) = e^{a+bx}$. Both constraints $\sum_{x=0}^{\infty} f(x) = 1$ and $\sum_{x=0}^{\infty} xf(x) = \mu$ have to be satisfied, hence

$$1 = e^a\sum_{x=0}^{\infty} e^{bx} = \frac{e^a}{1-e^b},$$

$$\mu = e^a e^b\sum_{x=1}^{\infty} x(e^b)^{x-1} = e^a e^b\frac{d}{d(e^b)}\sum_{x=1}^{\infty}(e^b)^x = e^a e^b\frac{d}{d(e^b)}\left(\frac{1}{1-e^b} - 1\right) = \frac{e^a e^b}{(1-e^b)^2}.$$

Substituting $1 = \frac{e^a}{1-e^b}$ yields $\mu = \frac{e^b}{1-e^b}$, so $e^b = \frac{\mu}{1+\mu}$ and $e^a = 1 - e^b = \frac{1}{1+\mu}$. Therefore, the distribution with maximal entropy subject to a given mean μ must be of the form $f(x) = \frac{1}{1+\mu}\left(\frac{\mu}{1+\mu}\right)^x$, $x = 0, 1, \ldots$. The only distribution of this form is the geometric distribution $\gamma(x) = p(1-p)^x$, $x = 0, 1, \ldots$ with mean $\mu = \frac{1}{p} - 1$, for which the entropy can be computed directly:

$$\begin{aligned} H(X_\gamma) &= -\sum_{x=0}^{\infty} p(1-p)^x(\log p + x\log(1-p)) \\ &= -p\log p\sum_{x=0}^{\infty}(1-p)^x - p(1-p)\log(1-p)\sum_{x=1}^{\infty} x(1-p)^{x-1} \\ &= -\log p + p(1-p)\log(1-p)\frac{d}{dp}\sum_{x=1}^{\infty}(1-p)^x \\ &= -\log p + p(1-p)\log(1-p)\frac{d}{dp}\left(\frac{1}{p} - 1\right) \\ &= -\log p - \frac{1-p}{p}\log(1-p). \end{aligned}$$

This shows that the entropy of an arbitrary random variable X is bounded by $H(X) \leq H(X_\gamma) = -\log p - \frac{1-p}{p}\log(1-p) = (\mu+1)\log(\mu+1) - \mu\log\mu$, where $\mu = \frac{1}{p} - 1 \Leftrightarrow p = \frac{1}{1+\mu}$ was substituted.

4. Let $X \sim p_1(x)$ and $Y \sim p_2(y)$. Then, (X, Y) is distributed according to the joint distribution $(X, Y) \sim p_3(x, y)$ and note that $p_1(x) = \sum_y p_3(x, y)$, $p_2(y) = \sum_x p_3(x, y)$. By (1) it is known that $0 \leq D(p||q) = -\sum_x p(x)\log\left(\frac{q(x)}{p(x)}\right)$, which is equivalent to $-\sum_x p(x)\log p(x) \leq -\sum_x p(x)\log q(x)$ for any distributions p, q. Applying this to the distributions $p_3(x, y)$ and $p_1(x)p_2(y)$ gives

$$\begin{aligned} H(X,Y) &= -\sum_{x,y} p_3(x,y)\log p_3(x,y) \leq -\sum_{x,y} p_3(x,y)\log(p_1(x)p_2(y)) \\ &= -\sum_{x,y} p_3(x,y)\log p_1(x) - \sum_{x,y} p_3(x,y)\log p_2(y) = H(X) + H(Y). \end{aligned}$$

5. Those two properties follow by the Shannon-McMillan-Breiman Theorem, see Theorem 15.7.1 and Lemma 15.7.1 of chapter 15.7 in ([8]).

□

After having introduced those basic definitions and its properties, the outline of the further proof is as follows. Instead of $\mathbb{P}(Z_1, \ldots, Z_n)$ for an input process Z, its Markov approximation is considered. Then, an upper bound of the entropy of the Markov approximation is derived in Ziv's Inequality. This result is used in the main theorem and

a resulting corollary, where it is shown that the length per symbol of the Lempel-Ziv coding $\frac{\kappa(n)(\log \kappa(n)+1)}{n}$ (see subsection 3.1) converges to the entropy.

Suppose now that a realization $z_{-(l-1)}^n = (z_{-(l-1)}^0, z_1^n)$ of the binary input process $Z_{-(l-1)}^n$ is given, where $l = 1, 2, \ldots$ is arbitrary and $z_{-(l-1)}^0$ is an initial configuration. Let z_1^n have a distinct parsing $\omega_1 \cdots \omega_\kappa$. For all substrings ω_i in the parsing, denote the preceding substring (history) of length l by π_i, i.e. the preceding substring of $\omega_i = (z_a, \ldots, z_b)$ is $\pi_i = z_{a-l}^{a-1}$. The history of the first parsed substring ω_1 is the initial configuration $\pi_1 = z_{-(l-1)}^0$. Moreover, denote the number of substrings in the parsing with length λ and history π (all π have length l) by $\kappa_{\lambda\pi}$. As the parsing is distinct, every ω_i is uniquely determined by its length λ and its history π, hence $\sum_{\lambda,\pi} \kappa_{\lambda\pi} = \kappa$. Similarly, every ω_i is counted only once in one of the $\kappa_{\lambda\pi}$ and all substrings belonging to $\kappa_{\lambda\pi}$ have equal length λ, so $\sum_{\lambda,\pi} \lambda\kappa_{\lambda\pi} = n$.

3.5 Lemma: *(Ziv's Inequality) Let a realization $z_{-(l-1)}^n = (z_{-(l-1)}^0, z_1^n)$ of the binary input process $Z_{-(l-1)}^n$ be given. For all $l \geq 1$ and for all distinct parsings of z_1^n, the lth order Markov approximation is bounded independently of M_l by*

$$\log M_l(z_1, \ldots, z_n | z_{-(l-1)}^0) \leq -\sum_{\lambda,\pi} \kappa_{\lambda\pi} \log \kappa_{\lambda\pi}.$$

Proof. Let $z_1 \cdots z_n = \omega_1 \cdots \omega_\kappa$ be any distinct parsing. As the parsing is distinct,

$$M_l(z_1, \ldots, z_n | z_{-(l-1)}^0) = M_l(\omega_1, \ldots, \omega_\kappa | \pi_1) = \prod_{i=1}^{\kappa} \mathbb{P}(\omega_i | \pi_i),$$

where the last equality follows by definition of the Markov approximation. Taking the logarithm yields

$$\log M_l(z_1, \ldots, z_n | z_{-(l-1)}^0) = \sum_{i=1}^{\kappa} \log \mathbb{P}(\omega_i | \pi_i) = \sum_{\lambda,\pi} \sum_{\omega_i : l(\omega_i)=\lambda, \pi_i=\pi} \log \mathbb{P}(\omega_i | \pi_i),$$

as every ω_i is determined by its length and pre-state. Inserting $\kappa_{\lambda\pi}$, $\frac{1}{\kappa_{\lambda\pi}}$ and using Jensen's Inequality for the concave log function gives

$$\begin{aligned} \log M_l(z_1, \ldots, z_n | z_{-(l-1)}^0) &= \sum_{\lambda,\pi} \kappa_{\lambda\pi} \sum_{\omega_i : l(\omega_i)=\lambda, \pi_i=\pi} \frac{1}{\kappa_{\lambda\pi}} \log \mathbb{P}(\omega_i | \pi_i) \\ &\leq \sum_{\lambda,\pi} \kappa_{\lambda\pi} \log \left(\frac{1}{\kappa_{\lambda\pi}} \sum_{\omega_i : l(\omega_i)=\lambda, \pi_i=\pi} \mathbb{P}(\omega_i | \pi_i) \right). \end{aligned} \tag{6}$$

Now, for every given λ and π in the first sum,

$$\sum_{\omega_i : l(\omega_i)=\lambda, \pi_i=\pi} \mathbb{P}(\omega_i | \pi_i) \leq \sum_i \mathbb{P}(\omega_i | \pi) = \mathbb{P}(\text{any } \omega | \pi) \leq 1, \tag{7}$$

where the inequality holds true as the constraint $l(\omega_i) = \lambda, \pi_i = \pi$ was removed and the first equality is true as all ω_i are distinct. Substituting (7) into (6) finally shows

$$\log M_l(z_1, \ldots, z_n | z^0_{-(l-1)}) \leq -\sum_{\lambda,\pi} \kappa_{\lambda\pi} \log \kappa_{\lambda\pi}.$$

□

Using Ziv's Inequality, the main theorem can now be proven. It shows that the expression $\frac{\kappa(n) \log \kappa(n)}{n}$ for any distinct parsing in $\kappa(n)$ substrings, not only the one of Lempel-Ziv coding, is bounded above by the entropy.

3.6 Theorem: *(Main theorem) For any stationary ergodic process $Z = \{Z_i\}_{i \in I}$ with entropy $H(Z)$ and any realization of length n of Z, a distinct parsing into $\kappa = \kappa(n)$ substrings satisfies*

$$\limsup_{n \to \infty} \frac{\kappa(n) \log \kappa(n)}{n} \leq H(Z).$$

Proof. By Ziv's Inequality (Lemma 3.5), it is already known that

$$\log M_l(z_1, \ldots, z_n | z^0_{-(l-1)}) \leq -\sum_{\lambda,\pi} \kappa_{\lambda\pi} \log \kappa_{\lambda\pi} = -\sum_{\lambda,\pi} \kappa_{\lambda\pi} \left(\log \kappa + \log \frac{\kappa_{\lambda\pi}}{\kappa} \right),$$

where $\pm \log \kappa$ was inserted. As stated in the introduction to Ziv's Inequality, $\sum_{\lambda,\pi} \kappa_{\lambda\pi} = \kappa$, therefore

$$\log M_l(z_1, \ldots, z_n | z^0_{-(l-1)}) \leq -\kappa \log \kappa - \kappa \sum_{\lambda,\pi} \frac{\kappa_{\lambda\pi}}{\kappa} \log \frac{\kappa_{\lambda\pi}}{\kappa}. \tag{8}$$

Also, as $\sum_{\lambda,\pi} \kappa_{\lambda\pi} = \kappa$, the fractions $\frac{\kappa_{\lambda\pi}}{\kappa} =: p_{\lambda\pi}$ can be interpreted as point masses $p_{\lambda\pi}$ of a probability distribution, where $\sum_{\lambda,\pi} p_{\lambda\pi} = 1$. Similarly, $\frac{n}{\kappa} = \sum_{\lambda,\pi} \lambda p_{\lambda\pi}$ follows directly from $n = \sum_{\lambda,\pi} \lambda \kappa_{\lambda\pi}$ after dividing by κ.

Now, the known results from Lemma 3.4 shall be used to derive the bound stated in the main theorem. In order to do this, define random variables X, Y with $\mathbb{P}(X = \lambda, Y = \pi) = p_{\lambda\pi}$. This is possible by simply defining X and Y discretely on all finitely many tupels (λ, π) as the range for the length is $\lambda = 1, \ldots, n$ and the are 2^l possible strings for the history π of length l over a binary alphabet. Inserting $\mathbb{P}(X = \lambda, Y = \pi) = p_{\lambda\pi}$ into the sum in (8) yields

$$-\sum_{\lambda,\pi} \frac{\kappa_{\lambda\pi}}{\kappa} \log \frac{\kappa_{\lambda\pi}}{\kappa} = -\sum_{\lambda,\pi} p_{\lambda\pi} \log p_{\lambda\pi} = H(X, Y),$$

and thus

$$\log M_l(z_1, \ldots, z_n | z^0_{-(l-1)}) \leq -\kappa \log \kappa + \kappa H(X, Y).$$

After multiplying by $-\frac{1}{n}$, this becomes

$$-\frac{1}{n}\log M_l(z_1,\dots,z_n|z_{-(l-1)}^0) \geq \frac{\kappa}{n}\log\kappa - \frac{\kappa}{n}H(X,Y). \tag{9}$$

Next, the entropy $H(X,Y)$ is bounded by $H(X,Y) \leq H(X)+H(Y)$ using Lemma 3.4. $H(X)$ can be bounded by using the mean only as proven in Lemma 3.4 as well. Note that

$$\mathbb{E}(X) = \sum_{\lambda}\lambda\mathbb{P}(X=\lambda) = \sum_{\lambda,\pi}\lambda\mathbb{P}(X=\lambda,Y=\pi) = \sum_{\lambda,\pi}\lambda p_{\lambda\pi} = \frac{n}{\kappa},$$

so by substituting the mean $\mu = \frac{n}{\kappa}$ into $H(X) \leq (\mu+1)\log(\mu+1) - \mu\log\mu$, it follows that

$$H(X) \leq \frac{n}{\kappa}\log\left(\frac{\kappa+n}{n}\right) + \log\left(\frac{\kappa+n}{\kappa}\right). \tag{10}$$

On the other hand, $\mathbb{P}(Y=\pi) = 2^{-l}$, as any of the 2^l possible binary substrings π of length l is equally likely. By definition of the entropy,

$$H(Y) = -\sum_{\pi}\mathbb{P}(Y=\pi)\log\mathbb{P}(Y=\pi) = -2^l 2^{-l}(-l)\log 2 = l, \tag{11}$$

where $\log = \log_2$. Together, (10) and (11) show that

$$H(X,Y) \leq H(X)+H(Y) \leq \frac{n}{\kappa}\log\left(\frac{\kappa+n}{n}\right) + \log\left(\frac{\kappa+n}{\kappa}\right) + l.$$

Multiplied by $\frac{\kappa}{n}$, this is equivalent to

$$\frac{\kappa}{n}H(X,Y) \leq \log\left(\frac{\kappa}{n}+1\right) + \frac{\kappa}{n}\log\left(\frac{n}{\kappa}+1\right) + \frac{\kappa}{n}l. \tag{12}$$

Consider (12) now in the limit $n\to\infty$ and note that $\kappa = \kappa(n)$. To derive the limit behaviour of all terms in (12), Lemma 3.2 by Lempel and Ziv is used. It stated that $\kappa(n) \leq \frac{n}{(1-\epsilon_n)\log n}$ for any distinct parsing, where $\epsilon_n \to 0$ as $n\to\infty$.

Using the Lemma of Lempel and Ziv and dividing by n, it follows that $0 \leq \frac{\kappa}{n} \leq \frac{1}{(1-\epsilon_n)\log n} \to 0$ as $n\to\infty$. Therefore, $\frac{\kappa}{n}l \to 0$ as l is independent of n and $\log\left(\frac{\kappa}{n}+1\right) \to \log 1 = 0$ as $n\to\infty$. In the middle term of (12), the expression $\frac{n}{\kappa}+1$ in the logarithm tends to infinity but is dominated by the coefficient $\frac{\kappa}{n} \to 0$. Summing up, $\delta_l(n) := \frac{\kappa}{n}H(X,Y) \to 0$ as $n\to\infty$.

Rewriting (9) with δ_l gives

$$\frac{\kappa\log\kappa}{n} = \frac{\kappa(n)\log\kappa(n)}{n} \leq -\frac{1}{n}\log M_l(z_1,\dots,z_n|z_{-(l-1)}^0) + \delta_l(n).$$

In the limit, this leads to

$$\limsup_{n\to\infty}\frac{\kappa\log\kappa}{n} \leq \limsup_{n\to\infty} -\frac{1}{n}\log M_l(z_1,\dots,z_n|z_{-(l-1)}^0) + \delta_l(n) = H(z_0|z_{-l}^{-1}) \to H(Z)$$

as $l\to\infty$, where it was used that $\delta_l(n)\to 0$ as $n\to\infty$ and the two entropy limits for $n\to\infty$ and $l\to\infty$ follow by Lemma 3.4. □

The fact that Lempel-Ziv coding approaches the entropy now follows as a corollary of the main theorem. As derived in subsection 3.1, the length of the Lempel-Ziv code is $\kappa(n)(\log \kappa(n) + 1)$. The length per symbol $\frac{\kappa(n)(\log \kappa(n)+1)}{n}$ is easily bounded by the entropy.

3.7 Corollary: *For any stationary ergodic process $Z = \{Z_i\}_{i \in I}$ with entropy $H(Z)$ and any realization of length n of Z, the length per symbol $\frac{\kappa(n)(\log \kappa(n)+1)}{n}$ of the Lempel-Ziv coding satisfies*

$$\limsup_{n \to \infty} \frac{\kappa(n)(\log \kappa(n) + 1)}{n} \leq H(Z).$$

Proof. By Lemma 3.2 of Lempel and Ziv, $\kappa(n) \leq \frac{n}{(1-\epsilon_n)\log n}$ for any distinct parsing, where $\epsilon_n \to 0$ as $n \to \infty$. As used in the proof of the main theorem before, this means that $0 \leq \frac{\kappa(n)}{n} \leq \frac{1}{(1-\epsilon_n)\log n} \to 0$ as $n \to \infty$. It follows that

$$\limsup_{n \to \infty} \frac{\kappa(n)(\log \kappa(n) + 1)}{n} = \limsup_{n \to \infty} \left(\frac{\kappa(n) \log \kappa(n)}{n} + \frac{\kappa(n)}{n} \right) \leq H(Z),$$

where it was used that $\frac{\kappa(n)}{n} \to 0$ as $n \to \infty$ and that $\limsup_{n \to \infty} \frac{\kappa(n) \log \kappa(n)}{n} \leq H(Z)$ by the main theorem. Clearly, Lempel-Ziv coding cannot beat the entropy, hence it is optimal in the limit. □

Empirical results on compression rates using a Lempel-Ziv variant are presented in chapter 6. The results presented there were obtained with an implementation of the Lempel-Ziv-Welch algorithm, which uses amongst others a variable bitrate encoding. In subsection 3.1, it was derived that every tupel consisting of prefix and appended bit needs $\log \kappa(n) + 1$ bits for storage, because a pointer to any of the $\kappa(n)$ prefixes uses at most $\log \kappa(n)$ bits. The variable bitrate encoder increases the bitrate over time, using fewer bits for the first tupels as those pointers are more likely to point to lower indices. Hence, every tupel consisting of prefix and appended bit needs at most $\log \kappa(n) + 1$ bits (the first tupels need less, the last ones need exactly $\log \kappa(n) + 1$ bits). As a consequence, this technique approaches the entropy even faster than the method analysed in the theorems above. Note that the method by Lempel and Ziv doesn't require any information on the distribution of the input sequence. Such a code is called a universal code.

4. A Block-Sorting Data Transformation

After the universal compression algorithm by Lempel and Ziv has been introduced in chapter 3, a recent approach on data compression is described in this chapter. The following section is based on the original paper ([1]) by M. Burrows and D.J. Wheeler, who describe a block-sorting coding scheme to increase the redundancy of a given input source. The scheme itself doesn't compress the data. The algorithm works as follows:

Burrows-Wheeler Transformation

1. Given any finite input string $s = s_0 \cdots s_{n-1}$ of length n over an arbitrary finite alphabet, let $s^{(k)} = s_k \cdots s_{n-1} s_0 \cdots s_{k-1}$ denote the cyclic left shift of s by k symbols. Note that $s = s^{(0)} = s^{(n)}$.
2. Insert all n shifts $s^{(0)}, \ldots, s^{(n-1)}$ of length n as rows into an $n \times n$ matrix M in any order and sort all rows of M lexicographically, giving a sorted matrix M'.
3. Locate the first cyclic left shift $s^{(1)}$ in the output M' and call its row number r. Output the last column of M' including the index r.

4.1 Example: *As an example, the string $s = bacba$ over the alphabet $\{a, b, c\}$ shall be encoded. Firstly, all cyclic left shifts of s are inserted as rows in a matrix M and sorted lexicographically, giving a matrix M',*

$$M = \begin{pmatrix} b & a & c & b & a \\ a & c & b & a & b \\ c & b & a & b & a \\ b & a & b & a & c \\ a & b & a & c & b \end{pmatrix} \Rightarrow M' = \begin{pmatrix} a & b & a & c & b \\ a & c & b & a & b \\ b & a & b & a & c \\ b & a & c & b & a \\ c & b & a & b & a \end{pmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

Let the rows be indexed by $0, 1, \ldots, (n-1)$. In this example, the first shift $s^{(1)} = acbab$ is located in row $r = 1$ in the matrix M'. The output of the Burrows-Wheeler Transformation is the last column of M' and r, so $(bbcaa, 1)$.

The effect of the Burrows-Wheeler transformation (abbreviated by BWT) can be observed immediately in the example above, where an almost perfect sorting was achieved. For every character a in a cyclic shift of length n, call its successive $n-1$ characters the context of a. The BWT sorts all shifts lexicographically and outputs the last symbol of every shift, which means that all output symbols are grouped together according to their context. In the example, the second b of the input *bacba* appears as first output character because its context *abac* is the lowest in the lexicographical sorting. Hence, for any input string, the BWT groups all symbols together according to their context independently of the actual position of a symbol in the input. This turns out to be especially useful for data compression. Using Lempel-Ziv coding, those repetitions $z \ldots z$

of an equal subsequence z will yield prefixes $z,\ zz, \ldots$ of growing length and thus higher compression, while in the original file, z might always appear isolated.

4.2 Remark: *Consider any shift $s^{(k)} = s_k s_{k+1} \cdots s_{n-1} s_0 \cdots s_{k-1}$, which starts with a symbol $s_k = c$. After having sorted the shift $s^{(k)}$ lexicographically together with all other shifts $\{s^{(j_1)}, \ldots, s^{(j_n)}\}$ starting with symbol c, call its position in the sorting m. This means that $s^{(k)}$ is the mth shift in the sorting of all those starting with c. Note that the symbol c itself is not important for the lexicographical sorting as all shifts $\{s^{(k)}, s^{(j_1)}, \ldots, s^{(j_n)}\}$ begin with c, hence only the context $s_{k+1}, \ldots, s_{k-1}$ of symbol s_k is important for sorting. Likewise, only the context of the first symbol of the other shifts $\{s^{(j_1)}, \ldots, s^{(j_n)}\}$ is of interest for sorting. Now, the position of the symbol s_k shall be determined when all strings are rotated left by one symbol and c will be last character. Observe that if all shifts are rotated left by one, all shifts ending with c are precisely $\{s^{(k+1)}, s^{(j_1+1)}, \ldots, s^{(j_n+1)}\}$. Symbol s_k appears at the end of shift $s^{(k+1)} = s_{k+1} \cdots s_{n-1} s_0 \cdots s_k$ and thus has the same context $s_{k+1}, \ldots, s_{k-1}$ again. Likewise, the last symbols in the other shifts $\{s^{(j_1+1)}, \ldots, s^{(j_n+1)}\}$ have unchanged context. This means that they will have exactly the same order after lexicographical sorting as before and the shift $s^{(k+1)}$, which has $s_k = c$ as last symbol, will again have position m in the sorting.*

Surprisingly, the input string can be uniquely recovered using only the BWT output string s' and the index r as shown in the next constructive proof.

4.3 Theorem: *(Burrows-Wheeler reverse transformation) There exists a reverse transformation to obtain the original string s out of the BWT output (s', r).*

Proof. Let a BWT output (s', r) be given. By construction of the BWT output, s' corresponds to the last column of the sorted matrix M', while all other columns are yet unknown.

Suppose that the original string s was known and let $s^{(0)}, \ldots, s^{(n-1)}$ be the cyclic left shifts as before. As every symbol s_i of s is rotated n times in the shifts, it appears as the last symbol in exactly one shift, the shift $s^{(i+1)}$. The matrix M' contains all shifts as rows and hence all symbols of the input are contained in its last column, which is the BWT output. Therefore, ordinary sortings of s and s' yield the same sorted string t, which corresponds to the first column of M':

$$M' = \begin{pmatrix} \vdots & & \vdots \\ t & \cdots & s' \\ \vdots & & \vdots \end{pmatrix}.$$

The index r marks the row of the first shift $s^{(1)}$ of s, in which the first symbol s_0 of s is rotated to the last position. Hence, s_0 appears as last symbol in row r of M'. By construction, all shifts are cyclic, which means that the following symbol s_1 appears in

row r of the first column, which is the known column t. Note that all other columns of M' are still unknown, so in order to get symbol s_2, a jump from the first to the last column is needed. This is done by locating symbol s_1 in the last column of M' again, using Remark 4.2: If $s_1 = c$ appears as first symbol in the mth of all rows starting with c, it also has to correspond to the mth of all rows with last symbol c, which shall have row number r'. Then, the next symbol s_2 must be the first symbol in row r'. Similarly, s_2 is located in the last column of M' again. Inductively, the whole string can be recovered. □

Finally, the construction for decoding is applied in the continued example:

4.4 Example: *In the last example, the BWT ouput was* $(bbcaa, 1)$. *Sorting the string bbcaa (the last column of* M'*) gives its first column aabbc:*

$$M' = \begin{pmatrix} a & \cdots & b \\ a & \cdots & b \\ b & \cdots & c \\ b & \cdots & a \\ c & \cdots & a \end{pmatrix} \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix}$$

The first symbol s_0 *of* s *is the last symbol in shift* $s^{(1)}$*, which has row number* $r = 1$*, so* $s_0 = b$*. The next symbol* s_1 *is the next one in the cyclic shift, located in the first column of row* 1*, so* $s_1 = a$*. This is the second* a *in the sorted sequence, so its corresponding shift is the second one with an* a *in the last column, which has row number* 4*. From there* s_3 *is found again by looking at the next symbol in the cyclic shift, which is the first symbol in row* 4*, so* $s_3 = c$*. Inductively,* $s = bacba$ *is recovered.*

From an empirical point of view it turns out that the Burrows-Wheeler Transformation is an easy and effective scheme to increase the compression rate. Note that the transformation is computationally intensive because of the lexicographical matrix sorting, while the reverse transformation just requires an ordinary sorting of a string. Burrows and Wheeler suggest an additional simple encoding technique in their paper ([1]) on the BWT, which they call Move-To-Front (MTF) encoding. The MTF makes use of repeated patterns in the BWT output in order to reversibly convert repetitions of different symbols into repetitions of the same low byte values. This increases the effect of the BWT when compressing data. Empirical results are again presented in chapter 6.

5. A Central Limit Theorem for return times in a random process

In chapter 3, it was proven that the universal Lempel-Ziv coding approaches the entropy of an arbitrary stationary ergodic random process. In the following sections, a central limit theorem on return times of a random process is stated and proven, which can be restated to give an asymptotic normal estimator of the entropy. This chapter is based on the original paper ([10]) by O. Johnson.

5.1. Statement of the problem and main theorem

Let a sample $(Z_1, \ldots, Z_n)$ of a stationary random process Z be given, where all Z_i take values in a finite alphabet A. The main aim is to estimate the entropy H of Z. As before, let Z_a^b denote the vector $Z_a^b = (Z_a, Z_{a+1}, \ldots, Z_b)$. Partition a sample Z_i into blocks of length l in the following definition.

5.1 Definition: *Define random variables X_i to be the block of length l up to Z_{il}, so $X_i := Z_{(i-1)l+1}^{il} \in A^l$.*

The main theorem is a statement on the return times of the first k blocks X_i, i.e. the time t it takes for the first k blocks $X_1, \ldots, X_k$ to reappear. The return time is formally defined in the next definition.

5.2 Definition: *The return time S_j of the jth block is defined as the random variable*

$$S_j := \min\{t \geq 1 : \ X_{j+t} = X_j\} \tag{13}$$

for all $j = 1, \ldots, k$.

The main theorem now states that the return times S_j satisfy a central limit theorem, given that the number $k(l)$ and size l of the blocks grow adequately.

5.3 Theorem [10]: *(Main theorem) Suppose that (Z_i) is an independent identically distributed finite alphabet process with entropy H. Let $q_{max} < 1$ be the maximum probability of any symbol. If, as the block length $l \to \infty$, the number of blocks $k(l) \to \infty$ in such a way that $\lim_{l\to\infty} k(l)^{3/2} l q_{max}^l = 0$, then*

$$\frac{\sum_{i=1}^{k(l)} (\log S_i - lH \log 2 + \gamma)}{\sqrt{k(l)\pi^2/6}} \xrightarrow{d} N(0,1),$$

where γ is the Euler constant.

As an important application, the main theorem can be restated to give an asymptotically normal estimator of the entropy H, as stated in the following corollary.

5.4 Corollary [10]: *Under the conditions of the main theorem, the estimator*

$$\hat{H} := \frac{\sum_{i=1}^{k(l)}(\log S_i + \gamma)}{l \log 2 \sqrt{k(l)\pi^2/6}}$$

is asymptotically normal and estimates the entropy.

Proof. By definition of asymptotic normality, a sequence $(T_n(Y_1, \ldots, Y_n))_{n\in\mathbb{N}}$ of estimators is asymptotically normal for a quantity θ if and only if there exist sequences $(\mu_n(\theta))_{n\in\mathbb{N}}$ and $(\sigma_n(\theta))_{n\in\mathbb{N}}$ such that

$$\frac{T_n(Y_1, \ldots, Y_n) - \mu_n(\theta)}{\sigma_n(\theta)} \xrightarrow{d} N(0,1),$$

where the estimator $(T_n(Y_1, \ldots, Y_n))_{n\in\mathbb{N}}$ depends on empirical input values Y_i.

For the claimed estimator $\hat{H}$ in the corollary, let $\sigma_n = \frac{1}{l \log 2}$ be independent of H and $\mu_n(H) = \frac{\sqrt{k(l)}H}{\sqrt{\pi^2/6}}$. Substituting into the definition of asymptotic normality shows that

$$\begin{aligned} \frac{\hat{H}(\log S_1, \ldots, \log S_{k(l)}) - \mu_n(H)}{\sigma_n} &= \frac{1}{\frac{1}{l\log 2}} \left(\frac{\sum_{i=1}^{k(l)}(\log S_i + \gamma)}{l \log 2 \sqrt{k(l)\pi^2/6}} - \frac{\sqrt{k(l)}H}{\sqrt{\pi^2/6}} \right) \\ &= \frac{\sum_{i=1}^{k(l)}(\log S_i - lH \log 2 + \gamma)}{\sqrt{k(l)\pi^2/6}} \xrightarrow{d} N(0,1) \end{aligned}$$

by the main theorem. This proves that $\hat{H}$ is an asymptotically normal estimator. □

5.2. Avoiding early matches

Let the process Z_i be independent and equidistributed over a finite alphabet A. There are $|A|^l$ possibilities for each block X_i of length l, hence each block occurs with probability

$$p = |A|^{-l}. \tag{14}$$

The return time S_j then corresponds to the waiting time until a block reappears and thus is a geometric variable. As the process is assumed to be independent and identically distributed with entropy H, the Asymptotic Equipartition Property (AEP, see chapter 3 in [8]) is satisfied, so that

$$p \cong 2^{-lH}. \tag{15}$$

The problem with the S_j in Definition 5.2 is that early matches may occur in the sequence $X_1, \ldots, X_k$, i.e. $S_i \leq k - i$. As an example, it could happen that $X_2 = X_3$, while for the main theorem the return times of $X_1, \ldots, X_k$ as a whole and not early return times within the block are of interest. To bypass this problem, new random variables R_j are defined, which take the values of S_j if no early match has occured and are set to a new value for early matches.

5.5 Definition:

1. *For a realization $X_1, \ldots, X_n$ of the process, define*

$$D := \{i : \ X_i \neq X_j \ \forall j = (i+1), \ldots, k\}$$

to be the set of all indices that do not see an early match.

2. *For all $i \in D$, set new values $b_i = X_i$ and for all $i \notin D$, define b_i to be a random element chosen uniformly from the set of blocks not yet assigned, i.e. $b_i \in A^l \backslash \bigcup_j b_j$.*

3. *For the new values b_j, define random variables R_j to be the waiting time from the end of every block until b_j is seen again, i.e.*

$$R_j := \min\{t \geq 1 : \ X_{k+t} = b_j\}. \tag{16}$$

5.6 Remark:

1. *If $i \in D$ it follows that $b_i = X_i$. By definition, the numbers S_i and R_i satisfy $X_{k+R_i} = b_i = X_i = X_{i+S_i}$. As both indices are minimal, $k + R_i = i + S_i$ and hence $S_i = R_i + (k - i)$.*

2. *If $i \in D^c$, then an early match has occurred and b_i is equal to a X_j, $j = (i+1), \ldots, k$, between the two b_i in the sequence $X_1, \ldots, b_i, \ldots, X_j, \ldots, X_k, \ldots, b_i$. Therefore, $S_i \leq k - i + R_i$.*

3. *For a random variable $R \sim Geom(p)$, $P(R = k) = p(1-p)^{k-1} \ \forall k \in \mathbb{N}$, the expected value $\mathbb{E}(1/R)$ is*

$$\begin{aligned}
\mathbb{E}(1/R) &= p\sum_{k=1}^{\infty} \frac{1}{k}(1-p)^{k-1} = p\sum_{k=0}^{\infty} \frac{1}{k+1}(1-p)^k \\
&= \frac{p}{1-p}\sum_{k=0}^{\infty} \frac{1}{k+1}(1-p)^{k+1} = \frac{p}{1-p}\sum_{k=0}^{\infty} \int_1^p -(1-x)^k dx \\
&= \frac{-p}{1-p}\int_1^p \sum_{k=0}^{\infty} (1-x)^k dx = \frac{-p}{1-p}\int_1^p \frac{1}{x} dx = \frac{-p}{1-p}\log p.
\end{aligned}$$

By (14) it is known that $p = |A|^{-l}$, while on the other hand the probability p of any block of length l can be bounded by the probability of the most likely symbol occuring l times, $p \leq q_{max}^l$. The value $\mathbb{E}(1/R)$ can then be bounded by

$$\mathbb{E}(1/R) = -\log p \frac{p}{1-p} = l \log |A| \sum_{k=1}^{\infty} p^k \leq l \log |A| \sum_{k=1}^{\infty} (q_{max}^l)^k = O(l q_{max}^l),$$

because $q_{max}^l \to 0$ as $l \to \infty$.

4. *All $|x| \leq 1$ satisfy $1-(1-x)^k \leq kx$ for $k \in \mathbb{N}$: Define $f_k(x) = kx - 1 + (1-x)^k$ and show that $f_k(x) \geq 0 \ \forall x \in [0,1]$. Firstly, compute $f_k'(x) = k - k(1-x)^{k-1} = k(1-(1-x)^{k-1})$. As $x \in [0,1]$, $(1-(1-x)^{k-1}) \in [0,1]$ and thus $f_k'(x) \geq 0$. As $f_k(0) = 0$ it follows that $f_k(x) \geq 0$.*

5. *Let $i \in D$. The probability of two blocks X_i and X_j, $i \neq j$, of length l being equal is the probability that all l symbols are equal, which is at most q_{max}^l. So $\mathbb{P}(X_i = X_j) \leq q_{max}^l < 1$. By the definition of the set D and independence of $\{X_j\}$ it follows that*

$$\begin{aligned}
\mathbb{P}(i \in D) &= \mathbb{P}(X_i \neq X_j \ \forall j = i+1, \ldots, k) = \prod_{j=i+1}^{k} \mathbb{P}(X_i \neq X_j) \\
&= \prod_{j=i+1}^{k} (1 - \mathbb{P}(X_i = X_j)) \geq \prod_{j=i+1}^{k} (1 - q_{max}^l) \\
&\geq \prod_{j=1}^{k} (1 - q_{max}^l) = (1 - q_{max}^l)^k.
\end{aligned}$$

The new random variables R_j will later be used to restate the main theorem in exchange for the S_j. This is possible as in the limit $l \to \infty$, a scaled version of $\sum_i (\log R_i - \log S_i)$ tends to zero in probability, as proven in the next Lemma 5.7. This means that early matches can be neglected in the limit and as shown later, a central limit theorem for the R_i extends to a limit for the S_i.

5.7 Lemma [10]: *Suppose that (Z_i) is an independent identically distributed finite alphabet process with entropy H. Let $q_{max} < 1$ be the maximum probability of any symbol. If, as the block length $l \to \infty$, the number of blocks $k = k(l) \to \infty$ in such a way that $\lim_{l\to\infty} k(l)^{3/2} l q_{max}^l = 0$, then*

$$\frac{\sum_{i=1}^{k} (\log R_i - \log S_i)}{\sqrt{k}} \xrightarrow{p} 0,$$

as $l \to 0$.

Proof. By definition of convergence in probability, it must be shown that

$$\lim_{l\to\infty} \mathbb{P}\left(\left| \frac{\sum_{i=1}^{k(l)} (\log R_i - \log S_i)}{\sqrt{k(l)}} - 0 \right| \geq \delta \right) = 0 \ \forall \delta > 0.$$

This is done by considering $\pm \frac{\sum_{i=1}^{k} (\log R_i - \log S_i)}{\sqrt{k}}$ seperately, starting with the proof that $\mathbb{P}\left(\frac{\sum_{i=1}^{k} (\log S_i - \log R_i)}{\sqrt{k}} \geq \delta \right) \to 0$.

Firstly, $1 \leq S_i \leq R_i + (k-i)$ by Remark 5.6, so

$$\delta \leq \frac{\sum_{i=1}^{k}(\log S_i - \log R_i)}{\sqrt{k}} \leq \frac{\sum_{i=1}^{k}(\log(R_i + (k-i)) - \log R_i)}{\sqrt{k}} = \frac{\sum_{i=1}^{k}\log(1+\frac{k-i}{R_i})}{\sqrt{k}}.$$

It is more likely that the bigger sum is larger than δ, hence

$$\mathbb{P}\left(\frac{\sum_{i=1}^{k}(\log S_i - \log R_i)}{\sqrt{k}} \geq \delta\right) \leq \mathbb{P}\left(\frac{\sum_{i=1}^{k}\log(1+\frac{k-i}{R_i})}{\sqrt{k}} \geq \delta\right) \leq \mathbb{P}\left(\sum_{i=1}^{k}\frac{\sqrt{k}}{R_i} \geq \delta\right),$$

where $\log(1+x) \leq x$ was used for the second inequality, so $\frac{\sum_{i=1}^{k}(\log(1+(k-i)/R_i)}{\sqrt{k}} \leq \sum_{i=1}^{k}\frac{(k-i)}{\sqrt{k}R_i} \leq \sum_{i=1}^{k}\frac{k}{\sqrt{k}R_i} = \sum_{i=1}^{k}\frac{\sqrt{k}}{R_i}$. Using Chebyshev's Inequality $\mathbb{P}(X \geq \epsilon) \leq \frac{1}{\epsilon}\mathbb{E}(X)$ for $\mathbb{P}\left(\sum_{i=1}^{k}\frac{\sqrt{k}}{R_i} \geq \delta\right) = \mathbb{P}\left(\sum_{i=1}^{k}\frac{1}{R_i} \geq \frac{\delta}{\sqrt{k}}\right)$ gives

$$\mathbb{P}\left(\sum_{i=1}^{k}\frac{\sqrt{k}}{R_i} \geq \delta\right) \leq \frac{\sqrt{k}}{\delta}\sum_{i=1}^{k}\mathbb{E}\frac{1}{R_i} \leq \frac{\sqrt{k}}{\delta}kO(q_{max}^{l}l) = O(k(l)^{3/2}lq_{max}^{l}),$$

where $\mathbb{E}(1/R) = O(q_{max}^{l}l)$ is known by Remark 5.6.

In order to proof the same result for $\frac{\sum_{i=1}^{k}(\log R_i - \log S_i)}{\sqrt{k}}$, consider the cases $i \in D$ and $i \in D^c$ separately. If $i \in D$, then $S_i = R_i + (k-i)$ by Remark 5.6, so $R_i \leq S_i$ and hence $\log R_i - \log S_i \leq 0$. For all $i \in D^c$, bound $\log R_i - \log S_i \leq \log R_i$ as $S_i \geq 1$, $\log S_i \geq 0$. Therefore,

$$\begin{aligned}
\mathbb{P}\left(\frac{\sum_{i=1}^{k}(\log R_i - \log S_i)}{\sqrt{k}} \geq \delta\right) &\leq \mathbb{P}\left(\frac{\sum_{i=1}^{k}\log R_i \mathbb{I}(i \in D^c)}{\sqrt{k}} \geq \delta\right) \\
&\leq \frac{1}{\delta\sqrt{k}}\sum_{i=1}^{k}\mathbb{E}(\log R_i \cdot \mathbb{I}(i \in D^c)) \\
&= \frac{1}{\delta\sqrt{k}}\sum_{i=1}^{k}\mathbb{E}(\log R_i)\mathbb{P}(i \in D^c), \qquad (17)
\end{aligned}$$

where the second inequality follows again by Chebyshev. Also, $\mathbb{E}(\log R_i \cdot \mathbb{I}(i \in D^c)) = \mathbb{E}(\log R_i)\mathbb{E}(\mathbb{I}(i \in D^c))$ as $\log R_i$ and $i \in D^c$ are independent and by definition $\mathbb{E}(\mathbb{I}_A) = \mathbb{P}(A)$ for all sets A.

Using that $p = |A|^{-l}$ by (14), $p \leq 1$ and $\mathbb{E}(\log R_i) = -\gamma - \log p + O(p)$ by the following Lemma 5.10, the expected value of $\log R_i$ can be bounded by $\mathbb{E}(\log R_i) = -\gamma - \log p + O(p) \leq -\gamma + l\log|A| + const \cdot p \leq -\gamma + l\log|A| + const = O(l)$.

Also, $\mathbb{P}(i \in D^c) = 1 - \mathbb{P}(i \in D) \leq 1 - (1-q_{max}^{l})^k \leq kq_{max}^{l}$, where the two facts $\mathbb{P}(i \in D) \geq (1-q_{max}^{l})^k$ and $1-(1-x)^k \leq kx\ \forall x \in [0,1]\ \forall k \in \mathbb{N}$ are known by Remark 5.6.

Inserting $\mathbb{E}(\log R_i) = O(l)$ and $\mathbb{P}(i \in D^c) \leq kq_{max}^l$ into (17) gives the bound

$$\frac{1}{\delta\sqrt{k}} \sum_{i=1}^{k} \mathbb{E}(\log R_i)\mathbb{P}(i \in D^c) \leq \frac{1}{\delta\sqrt{k}} kO(l)kq_{max}^l = O(k(l)^{3/2} l q_{max}^l).$$

Summarized, if $\lim_{l\to\infty} k(l)^{3/2} l q_{max}^l = 0$ by assumption then

$$\mathbb{P}\left(\left| \frac{\sum_{i=1}^{k(l)} (\log R_i - \log S_i)}{\sqrt{k(l)}} - 0 \right| \geq \delta \right) = O(k(l)^{3/2} l q_{max}^l) \to 0$$

as $l \to \infty$, which proves the Lemma. □

5.3. Mean and variance of $\log R_i$

The proof of the last Lemma 5.7 used the yet unproven fact $\mathbb{E}(\log R_i) = -\gamma - \log p + O(p)$. The leading order terms of mean and variance of $\log R$ will therefore be calculated in the following two lemmas.

5.8 Lemma [10]: *Any differentiable function f such that $f(x) \to 0$ as $x \to \infty$ satisfies*

$$\left| \sum_{i=1}^{\infty} f(i) - \int_1^{\infty} f(x)dx \right| \leq \frac{1}{2}|f(1)| + \frac{1}{2}\int_1^{\infty} |f'(x)|dx.$$

Proof. Integration by parts on the interval $[a, a+1]$ gives

$$\begin{aligned} -\int_a^{a+1} f'(x)\left(x - a - \frac{1}{2}\right)dx &= -\left[f(x)\left(x - a - \frac{1}{2}\right)\right]_{x=a}^{a+1} + \int_a^{a+1} f(x)dx \\ &= -\frac{1}{2}(f(a) + f(a+1)) + \int_a^{a+1} f(x)dx, \end{aligned}$$

so

$$\int_a^{a+1} f(x)dx - \frac{1}{2}(f(a)+f(a+1)) \leq \left| -\int_a^{a+1} f'(x)\left(x - a - \frac{1}{2}\right)dx \right| \leq \frac{1}{2}\int_a^{a+1} |f'(x)|dx,$$

where the absolute value was moved under the integral and it was used that $|x-a-1/2| \leq 1/2$ for all $x \in [a, a+1]$. Similarly,

$$\frac{1}{2}(f(a) + f(a+1)) - \int_a^{a+1} f(x)dx \leq \left| \int_a^{a+1} f'(x)\left(x - a - \frac{1}{2}\right)dx \right| \leq \frac{1}{2}\int_a^{a+1} |f'(x)|dx.$$

This means that $\left|\frac{1}{2}(f(a)+f(a+1)) - \int_a^{a+1} f(x)dx\right| \leq \frac{1}{2}\int_a^{a+1} |f'(x)|dx$, which gives the statement of the Lemma:

$$\begin{aligned}
\left|\sum_{i=1}^{\infty} f(i) - \int_1^{\infty} f(x)dx\right| &= \left|\frac{1}{2}f(1) + \sum_{a=1}^{\infty}\frac{1}{2}(f(a)+f(a+1)) - \sum_{a=1}^{\infty}\int_a^{a+1} f(x)dx\right| \\
&\leq \frac{1}{2}|f(1)| + \sum_{a=1}^{\infty}\left|\frac{1}{2}(f(a)+f(a+1)) - \int_a^{a+1} f(x)dx\right| \\
&\leq \frac{1}{2}|f(1)| + \frac{1}{2}\sum_{a=1}^{\infty}\int_a^{a+1} |f'(x)|dx \\
&= \frac{1}{2}|f(1)| + \frac{1}{2}\int_1^{\infty} |f'(x)|dx.
\end{aligned}$$

□

Lemma 5.8 and the following Remark 5.9 are used as a preparation for the proof of mean and variance of $\log R_i$ stated in the later Lemma 5.10.

5.9 Remark:

1. *For any $c \geq 0$, compute the derivative $\frac{d}{dx}\left(e^{-cx}\log x\right) = e^{-cx}\frac{1}{x} - ce^{-cx}\log x$. Integrating the derivative from 1 to infinity shows that*

$$0 = \left[e^{-cx}\log x\right]_{x=1}^{\infty} = \int_1^{\infty} e^{-cx}\frac{1}{x}dx - \int_1^{\infty} ce^{-cx}\log x dx.$$

Using the substitution $t = cx$, $x = \frac{t}{c}$, $dx = \frac{1}{c}dt$ for the second equality gives

$$\int_1^{\infty} e^{-cx}\log x dx = \frac{1}{c}\int_1^{\infty} e^{-cx}\frac{1}{x}dx = \frac{1}{c}\int_c^{\infty} e^{-t}\frac{c}{t}\frac{1}{c}dt = \frac{1}{c}\int_c^{\infty} t^{-1}e^{-t}dt.$$

2. *The incomplete gamma function is defined as $\Gamma(s,x) = \int_x^{\infty} t^{s-1}e^{-t}dt$. By definition, $\frac{1}{c}\Gamma(0,c) = \frac{1}{c}\int_c^{\infty} t^{-1}e^{-t}dt = \int_1^{\infty} e^{-cx}\log x dx$, where the second equality follows by the previous calculation. Moreover, using integral tables for the Gamma function it can be derived that $\Gamma(0,c) = -\gamma - \log c + c + O(c^2)$, where γ is Euler's constant.*

3. *Let $X \sim f(x)$ be a random variable defined on $[a,\ldots,\infty)$ with cumulative distribution function $F(x) = \mathbb{P}(X < x) = 1 - \mathbb{P}(X \geq x)$, $F'(x) = f(x)$. The expectation $\mathbb{E}(g(X))$ for a differentiable function g can then be expressed using integration by*

parts and $F(a) = 0$, $F(\infty) = 1$,

$$\begin{aligned}
\mathbb{E}\left(g(X)\right) &= \int_a^\infty g(x)f(x)dx = \left[g(x)F(x)\right]_{x=a}^\infty - \int_a^\infty g'(x)(1-\mathbb{P}\left(X \geq x\right))dx \\
&= \left[g(x)F(x)\right]_{x=a}^\infty - \int_a^\infty g'(x)dx + \int_a^\infty g'(x)\mathbb{P}\left(X \geq x\right)dx \\
&= \left[g(x)(F(x)-1)\right]_{x=a}^\infty + \int_a^\infty g'(x)\mathbb{P}\left(X \geq x\right)dx \\
&= g(a) + \int_a^\infty g'(x)\mathbb{P}\left(X \geq x\right)dx.
\end{aligned}$$

5.10 Lemma [10]: *Let* $R \sim Geom(p)$*, then*

1. $\mu(p) := \mathbb{E}\left(\log R\right) = -\gamma - \log p + O(p)$,
2. $\sigma^2(p) := \mathrm{Var}\left(\log R\right) = \frac{\pi^2}{6} + O(p\log p)$,
3. $\mathbb{E}\left(\left|\log R - \mu(p)\right|^3\right) \leq K = const < \infty$.

Proof.

1. Define $c := -\log(1-p) \geq 0$ and $f(x) := e^{-cx}\log x$. Computing the derivate shows that $|f'(x)| = |e^{-cx}/x - ce^{-cx}\log x| \leq |e^{-cx}/x| + |ce^{-cx}\log x|$. Denote the sum in Lemma 5.8 by $S := \sum_{i=1}^\infty f(i)$ and the integral by $I := \int_1^\infty f(x)dx$ and note that $f(1) = 0$, so that

$$\begin{aligned}
|S-I| &= \left|\sum_{i=1}^\infty e^{-ci}\log i - \int_1^\infty e^{-cx}\log x dx\right| \\
&\leq \frac{1}{2}|0| + \frac{1}{2}\int_1^\infty |f'(x)|dx \\
&\leq \frac{1}{2}\int_1^\infty \frac{e^{-cx}}{x}dx + \frac{c}{2}\int_1^\infty e^{-cx}\log x dx \\
&= \frac{1}{2}\left[e^{-cx}\log x\right]_{x=1}^\infty - \frac{1}{2}\int_1^\infty (-c)e^{-cx}\log x dx + \frac{c}{2}\int_1^\infty e^{-cx}\log x dx \\
&= cI,
\end{aligned}$$

   where $|f'(x)| \leq |e^{-cx}/x| + |ce^{-cx}\log x| = e^{-cx}/x + ce^{-cx}\log x$ was used as both terms are non-negative for $x \geq 1$ and $\frac{1}{2}\int_1^\infty \frac{e^{-cx}}{x}dx$ was integrated by parts.

   For $R \sim Geom(p)$, $P(R=k) = p(1-p)^{k-1}\ \forall k \in \mathbb{N}$, the first expectation can now be calculated using $c = -\log(1-p)$,

$$\mathbb{E}\left(\log R\right) = \sum_{x=1}^\infty p(1-p)^{x-1}\log x = p\sum_{x=1}^\infty e^{-c(x-1)}\log x = pe^c\sum_{x=1}^\infty e^{-cx}\log x.$$

The previous estimation showed that $|S|-|I| \leq |S-I| \leq cI$, so $S \leq |S| \leq I(1+c)$ as $I \geq 0$ for $x \geq 1$. Therefore,

$$pe^c \sum_{x=1}^{\infty} e^{-cx} \log x \leq \frac{p}{1-p} \int_1^{\infty} e^{-cx} \log x dx (1+c) = \frac{p}{1-p} \frac{\Gamma(0,c)}{c}(1+c),$$

where $I = \int_1^{\infty} e^{-cx} \log x dx = \frac{1}{c}\Gamma(0,c)$ follows by Remark 5.9. A Taylor expansion gives $c = p + \frac{1}{2}p^2 + \ldots = O(p)$ and $\frac{p}{(1-p)c} = 1 + \frac{1}{2}p + \frac{5}{12}p^2 + \ldots = 1 + O(p)$. Further, $p \leq c$ as $\log(1-p) \leq -p$, so $-\log c \leq -\log p$ and $\Gamma(0,c) = -\gamma - \log c + c + O(c^2)$ by Remark 5.9, which shows that

$$\begin{aligned} \frac{p}{1-p}\frac{\Gamma(0,c)}{c}(1+c) &\leq (1+O(p))^2(-\gamma - \log c + c + O(c^2)) \\ &\leq (1+O(p))(-\gamma - \log p + O(p) + O(p^2)) \\ &= -\gamma - \log p + O(p). \end{aligned}$$

This shows that $\mu(p) = \mathbb{E}(\log R) = -\gamma - \log p + O(p)$.

2. Similarly, define a new function $f(x) := e^{-cx}(\log x)^2$ with derivative $|f'(x)| = |2e^{-cx}\log x/x - ce^{-cx}(\log x)^2| \leq |2e^{-cx}\log x/x| + |ce^{-cx}(\log x)^2|$. As before it can be shown that

$$|S-I| = \left| \sum_{i=1}^{\infty} e^{-ci}(\log i)^2 - \int_1^{\infty} e^{-cx}(\log x)^2 dx \right| \leq cI.$$

With the help of the fact that $I = \pi^2/(6c) + (-\gamma + \log c)^2/c + 1 - O(c)$, it follows that $\mathrm{Var}(\log R) = \mathbb{E}((\log R)^2) - \mu(p)^2 = \pi^2/6 + O(p\log p)$.

3. In order to prove the bound on $\mathbb{E}\left(|\log R - \mu(p)|^3\right)$, partition the range of integration $\mathbb{R}$ into three sets $A_1 = \{x : |\log x - \mu(p)| \leq 1\}$, $A_2 = \{x : \log x - \mu(p) \geq 1\}$ and $A_3 = \{x : \log x - \mu(p) \leq -1\}$. This results in the partition $\mathbb{E}\left(|\log R - \mu(p)|^3\right) = K_1 + K_2 + K_3$, where $K_i = \mathbb{E}\left(|\log R - \mu(p)|^3 \mathbb{I}_{A_i}\right)$, $i \in \{1,2,3\}$. All values K_i are now calculated explicitly.

   Firstly, all $x \in A_1$ satisfy $|\log x - \mu(p)| \leq 1$ by definition, so $|\log x - \mu(p)|^3 \leq 1$ and hence $K_1 = \mathbb{E}\left(|\log R - \mu(p)|^3 \mathbb{I}_{x \in A_1}\right) \leq \mathbb{E}(1) = 1$.

   Secondly, K_2 is calculated using Chernoff's bound $\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}(e^{sX})}{e^{sa}}$ $\forall s > 0$ and the second moment $\mathbb{E}(R^2) = \frac{2-p}{p^2}$ of the geometric distribution. Applying Chernoff's bound on $X = \log R$ with $s = 2$ yields

$$\begin{aligned} \mathbb{P}(\log R - \mu(p) \geq t) &= \mathbb{P}(\log R \geq t + \mu(p)) \\ &\leq \frac{\mathbb{E}(R^2)}{\exp(2(t+\mu(p)))} \\ &\leq \frac{2-p}{p^2} e^{-2t} e^{-2(-\gamma - \log p + O(p))} \\ &\leq 2e^{2\gamma} e^{-2t}, \end{aligned}$$

where $2-p \leq 2$ was used, the denominator $\frac{1}{p^2}$ was cancelled by $e^{-2(-\log p)} = p^2$ and $e^{-2O(p)} \leq 1$. Also, observe that $K_2 = \mathbb{E}\left(g(\log R - \mu(p))\right)$ with $g(t) = t^3$. K_2 can now be bounded with the help of $\mathbb{P}\left(\log R - \mu(p) \geq t\right) \leq 2e^{2\gamma}e^{-2t}$ and Remark 5.9 as follows

$$\begin{aligned} K_2 &= \int_1^\infty 3t^2 \mathbb{P}\left(\log R - \mu(p) \geq t\right)dt \leq \int_1^\infty 6t^2 e^{2\gamma}e^{-2t}dt \\ &= \left[-\frac{3}{2}e^{2(\gamma-t)}(2t^2+2t+1)\right]_{t=1}^{\infty} = \frac{15}{2}e^{2(\gamma-1)} < 4. \end{aligned}$$

Thirdly, note that

$$\mathbb{P}\left(\log R - \mu(p) \leq -t\right) = \mathbb{P}\left(R \leq e^{-t+\mu(p)}\right) = \mathbb{P}\left(R \leq e^{-t-\gamma-\log p+O(p)}\right) \leq 1-\exp(-e^{-\gamma-t}),$$

so by integrating over the set $A_3 = \{x : \ \log x - \mu(p) \leq -1\}$,

$$K_3 = \int_{-\infty}^{-1} 3t^2 \mathbb{P}\left(\log R - \mu(p) \leq t\right)dt \leq \int_{-\infty}^{-1} 3t^2(1-\exp(-e^{-\gamma+t}))dt \approx 3.02 < 4.$$

Together, the values $K_1 \leq 1$, $K_2 \leq 4$ and $K_3 \leq 4$ establish the claimed bound $\mathbb{E}\left(|\log R - \mu(p)|^3\right) = K_1 + K_2 + K_3 \leq 9 =: K$.

□

5.4. Asymptotic independence

From the section 'Avoiding early matches' it is known that the R_i are geometrically distributed with parameter p_i, so $R_i \sim Geom(p_i)$. After mean and variance of $\log R_i$ were deduced in the previous section, explicit bounds on the difference between the joint probability distribution and the product of the marginal distributions are derived in the following Lemma 5.11. This result will be needed in the next Proposition 5.13, where it will be shown that the $\log R_i$ are asymptotically independent in the limit $l \to \infty$ and hence satisfy a Central Limit Theorem.

5.11 Lemma [10]: *Let (Z_i) be an independent identically distributed finite alphabet process with entropy H and (R_i) be the random variables defined in Definition 5.5. Then for any s, m, $\boldsymbol{a} = (a_1, \ldots, a_{m-1})$,*

$$\left(1 - \frac{p_m}{1-S^*}\right)^{s-1} \leq \mathbb{P}\left(R_m \geq s | \boldsymbol{R} = \boldsymbol{a}\right) \leq (1-p_m)^{s-m},$$

where $\boldsymbol{R} = (R_1, \ldots, R_{m-1})$ and $S^* = p_1 + \ldots + p_{m-1}$.

Proof. By definition of the random variable R_m in 5.5, observe that

$$R_m \geq s \Leftrightarrow \min\{t \geq 1 : \ X_{k+t} = b_m\} \geq s \Leftrightarrow X_{k+i} \neq b_m \ \forall i = 1, \ldots, (s-1).$$

Therefore, the distribution of R_m given $\boldsymbol{R} = \boldsymbol{a}$ can be made explicit by

$$\mathbb{P}\left(R_m \geq s | \boldsymbol{R} = \boldsymbol{a}\right) = \prod_{i=1}^{s-1} \mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right). \tag{18}$$

Now consider each term in the product (18) separately for all i. If for some $j \leq m-1$, the $a_j = i$, then $R_j = a_j = i$ as $\boldsymbol{R} = \boldsymbol{a}$. By definition of $R_j := \min\{t \geq 1 : \ X_{k+t} = b_j\}$ it follows that $X_{k+i} = b_j$, hence $X_{k+i} \neq b_m$ and the contribution of that i to the product (18) is 1.

In the case $a_j \neq i$ for a fixed i and all j, the probability $\mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right)$ will now be computed. Firstly, suppose that $X_{k+i} \in \{b_l : \ l \in I\}$, where $I \subseteq \mathbb{N}$ denotes the set of all possible indices.

Secondly, observe that if $a_j > i$, then as before $R_j = \min\{t \geq 1 : \ X_{k+t} = b_j\} > i$, hence $X_{k+l} \neq b_j$ for all $l \leq i$. In particular, $X_{k+i} \neq b_j$. This means that the index j is not needed in the set I of all possible indices. Define the set of all excluded indices $J = \{j : \ a_j > i\}$. Thus $X_{k+i} \in \{b_l : \ l \in I\}$ can now be refined as $X_{k+i} \in \{b_l : \ l \in I \setminus J\}$. If $a_j < i$, then $X_{k+a_j} = b_j$ and no information for X_{k+i} is obtained.

Thirdly, if $a_j \neq i$ for all $j = 1, \ldots, (m-1)$ is known only, the index m still belongs to I, i.e. $b_m \in \{b_l : \ l \in I \setminus J\}$,

$$\mathbb{P}\left(X_{k+i} = b_m, X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right) = \mathbb{P}\left(X_{k+i} = b_m\right) = p_m. \tag{19}$$

Knowing the conditional information $\boldsymbol{R} = \boldsymbol{a}$ on the a_j for $j = 1, \ldots, (m-1)$ is equivalent to refining the possible values for X_{k+i} to the set $\{b_l : \ l \in I \setminus J\}$ as every given $R_j = a_j$ might exclude an index in I,

$$\mathbb{P}\left(X_{k+i} = b_m | \boldsymbol{R} = \boldsymbol{a}\right) = \mathbb{P}\left(X_{k+i} = b_m | X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right). \tag{20}$$

The probability $\mathbb{P}\left(X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right)$ can now be calculated directly. Using $J = \{j_1, \ldots, j_m\}$,

$$\begin{aligned}
\mathbb{P}\left(X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right) &= 1 - \mathbb{P}\left(X_{k+i} \notin \{b_l : \ l \in I \setminus J\}\right) \\
&= 1 - \mathbb{P}\left(X_{k+i} \in \{b_l : \ l \in J\}\right) \\
&= 1 - \mathbb{P}\left(X_{k+i} = b_{j_1} \vee \ldots \vee X_{k+i} = b_{j_m}\right) \\
&= 1 - \sum_{j=1}^{m-1} p_j \mathbb{I}_{\{j \in J\}} \\
&= 1 - \sum_{j=1}^{m-1} p_j \mathbb{I}_{(a_j > i)}.
\end{aligned} \tag{21}$$

Equations (19), (20) and (21) together give the explicit expression

$$\begin{aligned}
\mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right) &= 1 - \mathbb{P}\left(X_{k+i} = b_m | \boldsymbol{R} = \boldsymbol{a}\right) \\
&= 1 - \mathbb{P}\left(X_{k+i} = b_m | X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right) \\
&= 1 - \frac{\mathbb{P}\left(X_{k+i} = b_m, X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right)}{\mathbb{P}\left(X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right)} \\
&= 1 - \frac{\mathbb{P}\left(X_{k+i} = b_m\right)}{\mathbb{P}\left(X_{k+i} \in \{b_l : \ l \in I \setminus J\}\right)} \\
&= 1 - \frac{p_m}{1 - \sum_{j=1}^{m-1} p_j \mathbb{I}_{(a_j > i)}} \\
&= 1 - \frac{p_m}{1 - S_i}, \qquad (22)
\end{aligned}$$

where $S_i = \sum_{j=1}^{m-1} p_j \mathbb{I}_{(a_j > i)}$. It is immediate to see that (22) is a decreasing function in S_i, which has range $0 \leq S_i \leq S^*$. By bounding expression (22), the product (18) will also be bounded.

Firstly, as $S_i \leq S^* \ \forall i$ and (22) is decreasing in S_i, it follows that $\mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right) \geq 1 - \frac{p_m}{1 - S^*}$. Bounding all $s-1$ terms gives the first claimed inequality in the Lemma,

$$\mathbb{P}\left(R_m \geq s | \boldsymbol{R} = \boldsymbol{a}\right) = \prod_{i=1}^{s-1} \mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right) \geq \left(1 - \frac{p_m}{1 - S^*}\right)^{s-1}.$$

Secondly, observe that the product (18) is maximized when the first $(m-1)$ values of a_i appear in the first $(m-1)$ places, i.e. $\{a_1, \ldots, a_{m-1}\} = \{1, \ldots, m-1\}$. As stated in the beginning of the proof, in this case the contribution to the product (18) is 1. It is clear that the remaining terms $i = m, \ldots, (s-1)$ satisfy $a_j \neq i$ and thus are bounded by $\mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right) \leq 1 - p_m$, where $0 \leq S_i$ was used in (22). In this way, the second claimed inequalitity is obtained,

$$\mathbb{P}\left(R_m \geq s | \boldsymbol{R} = \boldsymbol{a}\right) = \prod_{i=1}^{s-1} \mathbb{P}\left(X_{k+i} \neq b_m | \boldsymbol{R} = \boldsymbol{a}\right) \leq \prod_{i=1}^{m-1} 1 \prod_{i=m}^{s-1} (1 - p_m) = (1 - p_m)^{s-1-m+1}.$$

□

The last Lemma 5.11 will now be used to establish a Central Limit Theorem for the random variables $\log R_i$ in the next proposition. It shows that scaled versions of the $\log R_i$ have the property that its limit distribution coincides with the product of its marginal distributions if $\lim_{l \to \infty} lk(l)^{1/2} q_{max}^l = 0$. This in turn implies by definition that the $\log R_i$ are asymptotically independent and hence satisfy a Central Limit Theorem.

5.12 Remark:

1. *For any non-negative random variable X, the expected value can be expressed as $\mathbb{E}(X) = \int_0^\infty x\mathbb{P}(X \leq x)dx = \int_0^\infty \mathbb{P}(X \geq x)dx$ by Remark 5.9, where $a = 0$ and $g(X) = X$ was used.*

2. *Define the function*

$$H_{X,Y}(x,y) := \mathbb{P}(X \geq x, Y \geq y) - \mathbb{P}(X \geq x)\mathbb{P}(Y \geq y).$$

Then

$$\begin{aligned}
&\int_0^\infty \int_0^\infty H_{X,Y}(x,y)dxdy \\
&= \int_0^\infty \int_0^\infty \mathbb{P}(X \geq x, Y \geq y)dxdy - \int_0^\infty \mathbb{P}(X \geq x)dx \int_0^\infty Prob(Y \geq y)dy \\
&= \mathbb{E}(XY) - \mathbb{E}(X)\mathbb{E}(Y) \\
&= \mathrm{Cov}(X,Y).
\end{aligned}$$

Let f and g be any complex continuously differentiable (and invertible) functions. Using the substitutions $x = f^{-1}(a)$, $f(x) = a$, $f'(x)dx = da$ and $y = g^{-1}(b)$, $g(y) = b$, $g'(y)dy = db$ yields

$$\begin{aligned}
&\mathrm{Cov}(f(X), g(Y)) \\
&= \int_0^\infty \int_0^\infty \mathbb{P}(f(X) \geq a, g(Y) \geq b) - \mathbb{P}(f(X) \geq a)\mathbb{P}(g(Y) \geq b)dadb \\
&= \int_0^\infty \int_0^\infty \mathbb{P}(X \geq f^{-1}(a), Y \geq g^{-1}(b)) - \mathbb{P}(X \geq f^{-1}(a))\mathbb{P}(Y \geq g^{-1}(b))dadb \\
&= \int_0^\infty \int_0^\infty H_{X,Y}(f^{-1}(a), g^{-1}(b))dadb \\
&= \int_0^\infty \int_0^\infty f'(x)g'(y)H_{X,Y}(x,y)dxdy.
\end{aligned}$$

5.13 Proposition [10]: *Let (Z_i) be an independent identically distributed finite alphabet process with entropy H and (R_i) be the random variables defined in Definition 5.5. Then*

$$\alpha := \left| \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \sum_{j=1}^{m} \log R_j\right) - \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \sum_{j=1}^{m-1} \log R_j\right) \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_m\right) \right|$$

is $O(lk(l)^{1/2}q_{max}^l)$.

Proof. Define random variables $U = \log R_m \geq 0$, $V = \sum_{j=1}^{m-1} \log R_j \geq 0$ and apply the

formula from Remark 5.12, where $f(t) = g(t) = \exp(it/\sqrt{k})$.

$$\begin{aligned}
\alpha &= \left| \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \sum_{j=1}^{m} \log R_j \right) - \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \sum_{j=1}^{m-1} \log R_j \right) \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_m \right) \right| \\
&= \left| \mathrm{Cov}\left(\exp\left(\frac{iU}{\sqrt{k}} \right), \exp\left(\frac{iV}{\sqrt{k}} \right) \right) \right| \\
&= \left| \frac{-1}{k} \int_0^\infty \int_0^\infty \exp\left(\frac{iu}{\sqrt{k}} \right) \exp\left(\frac{iv}{\sqrt{k}} \right) H_{U,V}(u,v) du dv \right| \\
&\leq \left| \frac{-1}{k} \right| \int_0^\infty \int_0^\infty |H_{U,V}(u,v)| du dv. \qquad (23)
\end{aligned}$$

Next, define a new function

$$f(p) := \int_0^\infty (1-p)^{e^u - 1} du,$$

which has derivative $f'(p) = \frac{1}{(1-p)\log(1-p)} + \frac{1}{(1-p)^2} \int_0^\infty (1-p)^{e^u} du$. As $p \in (0,1)$, observe that $\frac{1}{(1-p)^2} \int_0^\infty (1-p)^{e^u} du \geq 0$, hence

$$-f'(p) = \frac{-1}{(1-p)\log(1-p)} - \frac{1}{(1-p)^2} \int_0^\infty (1-p)^{e^u} du \leq \frac{1}{p(1-p)},$$

where it was used that $\log(1-p) \leq -p \ \forall |p| < 1$ (define $g(p) = p + \log(1-p)$, then $g(0) = 0$, $g'(p) = 1 - 1/(1-p) \leq 0$ for $|p| < 1$, thus $g(p) \leq 0$), so $-\log(1-p) \geq p$ and $-1/\log(1-p) \leq \frac{1}{p}$.

In addition, let $t(x) := f'(p)x + f(p) - f'(p)p$ be the tangent to f at the point $(p, f(p))$. As f has an increasing but negative gradient[1], $t(q) \leq f(q)$ for all $p \leq q$. This means that

$$f(p) - f(q) \leq f(p) - t(q) = f(p) - f'(p)q - f(p) + f'(p)p = -f'(p)(q-p) \leq \frac{q-p}{p(1-p)}.$$

This result is now applied on $p = p_m$ and $q = \frac{p_m}{1-S^*}$, which shows that

$$\begin{aligned}
\int_0^\infty (1-p_m)^{e^u - 1} - \left(1 - \frac{p_m}{1-S^*} \right)^{e^u - 1} du &= f(p_m) - f\left(\frac{p_m}{1-S^*} \right) \\
&\leq \frac{\frac{p_m}{1-S^*} - p_m}{p_m(1-p_m)} \\
&= \frac{S^*}{(1-S^*)(1-p_m)} \\
&\leq \frac{k(l) q_{max}^l}{(1-S^*)(1-p_m)} \\
&= O(k(l) q_{max}^l), \qquad (24)
\end{aligned}$$

[1]Note that this fails if p is in an ϵ-neighbourhood of 1.

where the denominator is swallowed up into the $O(\cdot)$ term and $S^* \leq k(l)q_{max}^l$ can be seen as follows: $m \leq k(l)$ as R_m is the waiting time for block m and there are only $k(l)$ blocks. In all blocks of length l, every symbol occurs with maximal probability q_{max}, hence every block appears with maximal probability q_{max}^l and $S^* = p_1 + \ldots + p_{m-1} \leq (m-1)q_{max}^l \leq k(l)q_{max}^l$.

Now, non-negative functions h_- and h_+ will be found such that $-h_-(u,v) \leq H(u,v) \leq h_+(u,v)$. Use Lemma 5.11 to sum over values $\boldsymbol{a}$ such that $\sum_j \log a_j \geq v$ or $\sum_j \log a_j < v$.

1. For $v \geq \mathbb{E}(V)$,

$$h_-(u,v) \leq \left((1-p_m)^{e^u-1} - \left(1 - \frac{p_m}{1-S^*}\right)^{e^u-1} \right) \mathbb{P}(V \geq v)$$

and analogously $h_+ \leq (1-p_m)^{1-m}\mathbb{P}(U \geq u)\mathbb{P}(V \geq v)$. Thus by (24), integration over $\{v \geq \mathbb{E}(V)\}$ gives $\int h_-(u,v)dudv = \int (1-p_m)^{e^u-1} - (1-\frac{p_m}{1-S^*})^{e^u-1}du \int \mathbb{P}(V \geq v)dv \leq O(k(l)q_{max}^l)\mathbb{E}|V-\mu| = O(k(l)^{3/2}q_{max}^l)$. Also, using that $\mathbb{E}(U) = O(l)$ by Lemma 5.10, $\int h_+ dudv \leq (1-p_m)^{1-m}\mathbb{E}(U)\mathbb{E}|V-\mu| = O(lk(l)^{3/2}q_{max}^l)$.

2. For $v \leq \mathbb{E}(V)$,

$$h_+(u,v) \leq \left((1-p_m)^{e^u-1} - \left(1 - \frac{p_m}{1-S^*}\right)^{e^u-1} \right) \mathbb{P}(V \leq v)$$

and analogously $h_- \leq (1-p_m)^{1-m}\mathbb{P}(U \geq u)\mathbb{P}(V \geq v)$. As in the previous case, $\int h_+(u,v)dudv = O(k(l)^{3/2}q_{max}^l)$ and $\int h_-(u,v)dudv = O(lk(l)^{3/2}q_{max}^l)$.

Together, over the whole interval $[0,\infty)$, the integrals satisfy $\int h_- dudv = \int h_+ dudv = O(k(l)q_{max}^l) + O(lk(l)^{3/2}q_{max}^l) = O(lk(l)^{3/2}q_{max}^l)$. By choice of h_- and h_+, the function $H(u,v)$ is bounded by $-h_-(u,v) \leq H(u,v) \leq h_+(u,v)$, which means that $|H(u,v)| \leq h_-(u,v) + h_+(u,v)$. Substituting into (23) yields

$$\begin{aligned}
\alpha &\leq \left|\frac{-1}{k}\right| \int_0^\infty \int_0^\infty |H_{U,V}(u,v)| dudv \\
&\leq \frac{1}{k} \int_0^\infty \int_0^\infty h_-(u,v) + h_+(u,v) dudv \\
&\leq \frac{1}{k} O(lk(l)^{3/2}q_{max}^l) \\
&\leq O(lk(l)^{1/2}q_{max}^l),
\end{aligned}$$

where it was used that the number of blocks $k(l)$ is non-negative. □

5.5. Completing the proof of the main theorem

Proposition 5.13 was the last preparation for proving the main theorem. Recall that the Lyapunov Central Limit Theorem (see reference [18] in [10]) states that for independent random variables $Y_1, \ldots, Y_k$ with $\mathbb{E}(Y_i) = \mu_i$, $\mathrm{Var}(Y_i) = \sigma_i^2$ and finite centered absolute third moment $m_i = \mathbb{E}|Y_i - \mu_i|^3$, if

$$\frac{\sum_{i=1}^k m_i}{\left(\sum_{i=1}^k \sigma_i^2\right)^{3/2}} \to 0,$$

as $k \to \infty$, then

$$\frac{\sum_{i=1}^k (Y_i - \mu_i)}{\sqrt{\mathrm{Var}\left(\sum_{i=1}^k Y_i\right)}} \xrightarrow{d} N(0,1)$$

for $k \to \infty$. The main theorem is finally repeated and proven.

Theorem [10]: *(Main theorem) Suppose that (Z_i) is an independent identically distributed finite alphabet process with entropy H. Let $q_{max} < 1$ be the maximum probability of any symbol. If, as the block length $l \to \infty$, the number of blocks $k(l) \to \infty$ in such a way that $\lim_{l\to\infty} k(l)^{3/2} l q_{max}^l = 0$, then*

$$\frac{\sum_{l=1}^{k(l)} (\log S_i - lH \log 2 + \gamma)}{\sqrt{k(l)\pi^2/6}} \xrightarrow{d} N(0,1),$$

where γ is the Euler constant.

Proof. Given the number of blocks $k(l) = k$, let $T_1, \ldots, T_k$ be a sequence of independent random variables with $\mathbb{E}(\log T_i) = \mu(p_i) = \mathbb{E}(\log R_i)$, $\mathrm{Var}(\log T_i) = \pi^2/6 + O(p \log p) = \mathrm{Var}(\log R_i)$ and third moment $\mathbb{E}|\log T_i - \mu(p)|^3 = \mathbb{E}|\log R_i - \mu(p)|^3 \leq K = 9$. (This is possible since stochastic independence doesn't depend on the expectation or the variance).

The $\log T_1, \ldots, \log T_k$ now satisfy the Lyapunov Central Limit Theorem. Using that $\sqrt{\mathrm{Var}\left(\sum_{i=1}^k \log T_i\right)} = \sqrt{k\mathrm{Var}(\log T_i)} = \sqrt{k\pi^2/6}$, the Lyapunov Central Limit Theorem yields

$$\frac{\sum_{i=1}^k \log T_i}{\sqrt{k\pi^2/6}} - \frac{\sum_{i=1}^k \mu(p_i)}{\sqrt{k\pi^2/6}} = \frac{\sum_{i=1}^k (\log T_i - \mu(p_i))}{\sqrt{k\pi^2/6}} \xrightarrow{d} N(0,1), \tag{25}$$

where $O(p \log p) \to 0$ as the probability per block $p = |A|^{-l} \to 0$ for $l \to \infty$.

Here, $\mu(p) = -\gamma - \log p + O(p)$ is known by Lemma 5.10. By the Asymptotic Equipartition Property in (15), $p = 2^{-lH}$ as $l, k \to \infty$, independently of the actual p_i, which means that $-\log p = lH \log 2$ and thus

$$\sum_{i=1}^{k} \mu(p_i) \to k(-\gamma + lH \log 2), \tag{26}$$

where it was also used that $O(p) \to 0$. Together with (25), statement (26) shows that

$$\frac{\sum_{i=1}^{k}(\log T_i - lH \log 2 + \gamma)}{\sqrt{k\pi^2/6}} \xrightarrow{d} N(0,1). \tag{27}$$

Next, the Central Limit Theorem will be extended to the $\log R_i$. Note that the terms $\mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_j\right)$ in Proposition 5.13 satisfy

$$\begin{aligned} \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_j\right) &= \mathbb{E}\left(\cos\left(\frac{\log R_j}{\sqrt{k}}\right) + i \sin\left(\frac{\log R_j}{\sqrt{k}}\right)\right) \\ &\leq \mathbb{E}\left|\cos\left(\frac{\log R_j}{\sqrt{k}}\right) + i \sin\left(\frac{\log R_j}{\sqrt{k}}\right)\right| = 1, \end{aligned} \tag{28}$$

as $\mathbb{E}(X) \leq \mathbb{E}(|X|)$ for all random variables. By Proposition 5.13 and (28),

$$\begin{aligned} &\left|\mathbb{E} \exp\left(\sum_{j=1}^{k} \frac{i}{\sqrt{k}} \log R_j\right) - \prod_{j=1}^{k} \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_j\right)\right| \\ &\leq \left|\mathbb{E} \exp\left(\sum_{j=1}^{k} \frac{i}{\sqrt{k}} \log R_j\right) - \mathbb{E} \exp\left(\sum_{j=1}^{k-1} \frac{i}{\sqrt{k}} \log R_j\right) \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_k\right)\right| \\ &+ \left|\mathbb{E} \exp\left(\sum_{j=1}^{k-1} \frac{i}{\sqrt{k}} \log R_j\right) \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_k\right) - \prod_{j=1}^{k} \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_j\right)\right| \\ &\leq O(lk(l)^{1/2} q_{max}^l) + \left|\mathbb{E} \exp\left(\sum_{j=1}^{k-1} \frac{i}{\sqrt{k}} \log R_j\right) - \prod_{j=1}^{k-1} \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_j\right)\right|, \end{aligned} \tag{29}$$

where $\pm \mathbb{E} \exp\left(\sum_{j=1}^{k-1} \frac{i}{\sqrt{k}} \log R_j\right) \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_k\right)$ was inserted.

Bounding the remaining absolute values in (29) $k-1$ times in the same fashion shows that

$$\begin{aligned} &\left|\mathbb{E} \exp\left(\sum_{j=1}^{k} \frac{i}{\sqrt{k}} \log R_j\right) - \prod_{j=1}^{k} \mathbb{E} \exp\left(\frac{i}{\sqrt{k}} \log R_j\right)\right| \\ &\leq k \cdot O(lk(l)^{1/2} q_{max}^l) \\ &= O(lk(l)^{3/2} q_{max}^l). \end{aligned} \tag{30}$$

By assumption, $\lim_{l\to\infty} k(l)^{3/2} l q_{max}^l = 0$, which establishes the asympotic independence of $\log R_j$. In the limit $l \to \infty$, the random variables $\log R_j$ therefore satisfy the same Central Limit Theorem as the $\log T_i$ in (27), hence

$$\frac{\sum_{i=1}^k (\log R_i - lH \log 2 + \gamma)}{\sqrt{k\pi^2/6}} \xrightarrow{d} N(0,1). \tag{31}$$

Statement (31) now directly carries over to the $\log S_i$,

$$\begin{aligned}
&\frac{\sum_{i=1}^k (\log S_i - lH \log 2 + \gamma)}{\sqrt{k\pi^2/6}} \\
&= \frac{\sum_{i=1}^k (\log S_i - lH \log 2 + \gamma)}{\sqrt{k\pi^2/6}} + \frac{\sum_{i=1}^k \log R_i}{\sqrt{k\pi^2/6}} - \frac{\sum_{i=1}^k \log R_i}{\sqrt{k\pi^2/6}} \\
&= \frac{-1}{\sqrt{\pi^2/6}} \cdot \frac{\sum_{i=1}^k (\log R_i - \log S_i)}{\sqrt{k}} + \frac{\sum_{i=1}^k (\log R_i - lH \log 2 + \gamma)}{\sqrt{k\pi^2/6}} \xrightarrow{d} N(0,1),
\end{aligned}$$

where (31) was used and $\frac{\sum_{i=1}^k (\log R_i - \log S_i)}{\sqrt{k}} \to 0$ is known by Lemma 5.7. This proves the main theorem. □

6. Empirical results on compression

In the last chapter, the theoretical results on compression are applied in practise. In order to do this, an own implementation of all presented algorithms was written and is appended to this essay. The program includes an implementation for constructing the Huffman code (Huffman), the Lempel-Ziv variant Lempel-Ziv-Welch (LZW) with variable bitrate encoder and an implementation of the Burrows-Wheeler transformation (BWT) including the Move-To-Front scheme (MTF, see end of chapter 4). The standard Canterbury Corpus Benchmark ([6]) was used to compare the compression rates. The following table shows the best results of all algorithms when compressing the Canterbury Corpus, including the common compression tool WinZip with option 'maximal compression':

algorithm	output filesize in bytes	rate in %
uncompressed	2812342	100.0
Huffman	1672576	59.5
LZW	978765	34.8
WinZip	737450	26.2
BWT+LZW	714181	25.4
BWT+MTF+Huffman	871214	31.0
BWT+MTF+LZW	625541	22.2

The table shows that LZW coding makes considerably better use of the redundancy in an input stream compared to Huffman coding due to compressing substrings. Partial sorting via BWT increases the redundancy significantly, as observed by compressing the BWT output using LZW coding. Here, the benchmark could be compressed more than 250000 bytes more efficiently than without the BWT. Note that there would be no change between Huffman coding and BWT+Huffman, as the BWT output contains the same symbols with same multiplicities and hence with the same probability distribution as the input. Using MTF encoding, repetitions of different symbols in the BWT output are reversibly transformed to repetitions of the same low byte values, which increases the redundancy a second time. Now, there exist far more low byte values in the file than high byte values, hence the probability distribution was shifted to lower symbols and Huffman coding yields better compression. Moreover, there exist substrings of different symbols which had different prefix pointers when Lempel-Ziv coded before and now get same pointers after they had been encoded with the same low byte values. This results in a better Lempel-Ziv compression, which drops far beyond 700000 bytes (from 714181 bytes for BWT+LZW down to 625541 bytes for BWT+MTF+LZW) and thus beats the WinZip compression by more than 100000 bytes.

References

[1] M. Burrows, D.J. Wheeler,
A Block-Sorting Lossless Data Compression Algorithm,
Research Report 124, Digital Systems Research Center, 1994.

[2] T.A. Welch,
A Technique for High Performance Data Compression,
IEEE Computer, Vol. 17, No. 6, June 1984, pp. 8-19.

[3] D.A. Huffman,
A Method for the Construction of Minimum Redundancy Codes,
Proceedings of the IRE, 40, pp. 1098-1101, 1952.

[4] J. Ziv, A. Lempel,
A Universal Algorithm for Sequential Data Compression,
IEEE Transactions on Information Theory. Vol. IT-23, No. 3, May 1977, pp. 337-343.

[5] J. Ziv, A. Lempel,
Compression of Individual Sequences via Variable Rate Coding,
IEEE Transactions on Information Theory. Vol. IT-24, No. 5, September 1978, pp. 530-535.

[6] Canterbury Corpus Compression Benchmark,
http://corpus.canterbury.ac.nz.

[7] J. Kleinberg, E. Tardos,
Algorithm Design,
Addison Wesley, ISBN 0-321-29535-8.

[8] T. Cover, J. Thomas,
Elements of Information Theory,
John Wiley & Sons, ISBN 0-471-06259-6.

[9] P. Shields,
The Ergodic Theory of Discrete Sample Paths,
American Mathematical Society, ISBN 0-821-80477-4.

[10] O. Johnson,
A Central Limit Theorem for Non-Overlapping Return Times,
Journal of Applied Probability, Volume 43, Number 1 (2006), 32-47.

[11] B. McMillan,
Two inequalities implied by unique decipherability,
IEEE Transactions on Information Theory 2 (4), pp. 115-116.

A. Appendix

The compression results presented in chapter 6 were obtained with an own implementation, available on the following CD-Rom. It contains a directory 'BSDC' (Block-Sorting Data Compression) with the main application 'Project1.exe' and its source code.