Bounds for codes correcting insertion, deletion and substitution errors

Philippe van Elderen

Supervisor: Jos Weber
EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft Universtiy of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Applied Mathematics
January 9, 2025

Name of the student: Philippe van Elderen
Final project course: AM3000 Bachelorproject
Thesis committee: Jos Weber, Joost de Groot

Layman's summary

Data is stored using symbols like ones and zeros. In the process of reading and storing data, errors occur which are corrected by adding redundant symbols to the data. The manner in which we do this, is called an error-correcting code. We are interested in some properties of these codes such as size and efficiency. In earlier research, bounds on the size of error-correcting codes that can correct different kind of errors have been established. However, these bounds use the size of an unknown set V. In this thesis, the size of V will be examined, in order to be able to give some insight in earlier found bounds on error-correcting codes.

Summary

When data is stored, transmitted or read, errors occur. In order to be able to correct these errors, error-correcting codes are used. These codes add redundant information, in order to be able to correct errors that might occur. There are three types of errors that might occur, namely substitution, insertion and deletion errors. Substitution errors are errors where the original symbol is wrongly decoded or stored and the receiver reads another symbol. Insertion errors are errors where the receiver reads a symbol at a location where there should not be a symbol and a deletion error is an error where the receiver does not read a symbol while the reader should have. Substitution errors are the most common and therefore, codes which correct these type of errors are more extensively examined. There are however methods of storing data, for example storing data using DNA as medium, for which insertion and deletion or indel errors occur at a significant rate and therefore error-correcting codes should not only be able to correct substitution errors, but also indel errors.[1] [2]We want codes to be as efficient as possible, in other words, to be able to correct as many errors as possible by adding as few redundant symbols as possible. The size of a code is the number of codewords in a code and we would like this number to be as big as possible. Ward Spee has done research into the maximum size of error-correcting codes that can correct exactly t' deletions, exactly t'' insertions and at most s substitutions for his masters thesis at the University of Technology Delft.[3] In his research he has found various bounds on the size of these error-correcting codes. These bounds contain the size of a set V which is not known. This bachelor thesis project revolves around determining bounds on this size.

Later in this thesis, the set V will be explained further. For now it suffices to say it depends on a word $\mathbf{x}$, the maximum number of substitutions s, the number of deletions t' and the number of insertions t''. Firstly, the need and workings of error-correcting codes will be explained. Subsequently, the set V will be defined and explained in more detail and we will show the results obtained by Ward Spee, for which this set is essential. Lastly we will examine the size of V for different parameters and apply our findings to the bounds provided by Spee.

The main findings of this thesis are the construction of an exact expression for the size of V for specific t', t'' and s which is then used in the bounds found by Ward Spee on the maximum size of codes. The behavior of these bounds is discussed and some insight is given into the problem for larger values of t', t'' and s.

Contents

Chapter 1

Introduction

When storing, transmitting or reading data, binary or of another kind, errors can occur. The most well known and studied type of errors are substitution errors. These are errors where a symbol is substituted by a different symbol. Without error-correcting codes, this can lead to errors in the transmission of information. Suppose one wants to communicate the wind direction. Then we first need to assign binary words to each wind direction, say south = 00, west = 01, north = 11 and east = 10. These words are called information words. Assume the wind blows from the north, then in order to communicate this, we need to send the word 11. However, if an error occurs, say in the second position, the receiver may receive and decode the information as follows

$$\textbf{North} \xrightarrow[\text{encodes to}]{} 11 \xrightarrow[\text{error occurs}]{} 10 \xrightarrow[\text{decodes to}]{} \textbf{East}$$

We usually cannot prevent errors from occurring, however we can correct errors by adding extra bits. These bits are called redundant bits, since they do not contain information. The words we obtain by adding these redundant bits are called codewords and all codewords together form the error-correcting code. One of the simplest error-correcting codes is the three repetition code. As the name suggests, this code simply repeats the information words three times. The information words and codewords are shown in Table 1.1 where the redundant bits are coloured blue.

When decoding codewords, one always assumes as few errors as possible. Thus if the received word differs from all codewords, we decode to the closest codewords from our list of codewords. Now if we again want to communicate that the wind direction is north, we send the codeword 111111. If an error occurs at the second position the receiver reads the word 101111 which is not a codeword. The distance between two words $\mathbf{x}, \mathbf{y}$ of equal length is defined as

$$d(\mathbf{x}, \mathbf{y}) = \text{the number of places where } \mathbf{x} \text{ and } \mathbf{y} \text{ differ}$$

Since $d(101111, 111111) = 1$, $d(101111, 000000) = 5$, $d(101111, 101010) = 2$ and $d(101111, 010101) = 4$ we decode as follows

Wind direction	Information word	Codeword
South	00	000000
West	01	010101
North	11	111111
East	10	101010

Table 1.1: Example of three repetition code on wind direction

$$\textbf{North} \xrightarrow[\text{encodes to}]{} 111111 \xrightarrow[\text{error occurs}]{} 101111 \xrightarrow[\text{corrects to}]{} 111111 \xrightarrow[\text{decodes to}]{} \textbf{North}$$

Thus we can accurately correct this error.

1.1 The quality of codes

In the example we were able to correct one error, but how can we be sure this is always possible? Are there instances in which we can correct more than one error? These questions can be answered by looking at the minimal distance between the codewords. Suppose the minimal distance between two codewords is $2s + 1$. We will show that we can correct up to s errors by contradiction. Suppose we cannot correct s errors. Then there exist codewords $\mathbf{c}_1, \mathbf{c}_2$ such that there exists a word $\mathbf{x}$ with $d(\mathbf{c}_1, \mathbf{x}) = s$ and

$$d(\mathbf{c}_2, \mathbf{x}) \leq d(\mathbf{c}_1, \mathbf{x}) = s \tag{1.1}$$

however, by the triangle inequality we have that

$$d(\mathbf{c}_1, \mathbf{x}) + d(\mathbf{x}, \mathbf{c}_2) \geq d(\mathbf{c}_1, \mathbf{c}_2) = 2s + 1 \tag{1.2}$$

But this gives us a contradiction, since by 1.1 we have that

$$d(\mathbf{c}_1, \mathbf{x}) + d(\mathbf{x}, \mathbf{c}_2) \leq 2 \cdot d(\mathbf{c}_1, \mathbf{x}) \leq 2s \tag{1.3}$$

thus we can always correct at least s errors.

A quick example shows us that we cannot always correct $s + 1$ errors. Let

$$\mathbf{c}_1 = \underbrace{11 \ldots 11}_{2s+1 \text{ ones}}, \mathbf{c}_2 = \underbrace{00 \ldots 00}_{2s+1 \text{ zeros}}$$

Then $d(\mathbf{c}_1, \mathbf{c}_2) = 2s+1$. If we send $\mathbf{c}_1$ and $s+1$ errors occur, we receive a word $\mathbf{x}$ with $s+1$ zeros and $2s+1-(s+1) = s$ ones. Since $d(\mathbf{c}_1, \mathbf{x}) = s+1 > s = d(\mathbf{c}_2, \mathbf{x})$, we decode to $\mathbf{c}_2$ and we are not able to decode correctly.

The quality of error-correcting codes depends also on its code rate R. The code rate depends on the number of codewords and their length. The formal definition is as follows

Definition 1. *The code rate R of an error-correcting code C with q-ary codewords of length n, is defined as*

$$R = \frac{\log_q |C|}{n} \tag{1.4}$$

A high code rate implies that a larger portion of all possible words of length n are being used to convey information. Therefore, we are looking to maximize $|C|$ without increasing n, the length of the codewords.

1.2 Indel errors

Before we can define the set in which we are interested, we need to explain indel errors. Indel is short for insertions and deletions and indel errors are combinations of indel- and deletion errors. Suppose we send the word 11 once more and a deletion error occurs at the second position:

$$\textbf{North} \xrightarrow[\text{encodes to}]{} 11 \xrightarrow[\text{error occurs}]{} 1 \xrightarrow[\text{decodes to}]{} \textbf{East/North/West}$$

While an insertion can happen at three positions, namely before the first symbol, between the two symbols or after the second symbol. This might look as follows:

$$\textbf{North} \xrightarrow[\text{encodes to}]{} 11 \xrightarrow[\text{error occurs}]{} 101 \xrightarrow[\text{decodes to}]{} \textbf{East/North/West}$$

Now it is time to give the definition of the set $V(\mathbf{x}, t', t'', s)$.

1.3 Definition of the set $V_{t',t'',s}(\mathbf{x})$

The two types of error-correcting codes, substitution and indel, have been widely studied. However, codes that can correct combinations of substitution and indel errors have not yet been studied as extensively. Bounds on the size of these codes have been studied by Ward Spee in his thesis project [3]. The bounds he has found depend on the size of the set $V_{t',t'',s}(\mathbf{x})$. This set consists of all q-ary words that can be obtained from a word $\mathbf{x}$ of length n by applying t' deletions, t'' insertions and at most s substitutions. If we take $q = 2$, $n = 3$, $\mathbf{x} = 011$, $t' = 1$, $t'' = 0$ and $s = 1$ for example, then the vectors in the set V are of length 2 since we apply 1 deletion and 0 insertions. We can show by hand that the set $V_{1,0,1}(011)$ spans $\mathcal{B}_2(2) = \{00, 01, 10, 11\}$ using Table 1.2. Here $\mathcal{B}_2(2)$ denotes all binary words of length 2. The set $\mathcal{B}_q(n)$ denotes the set of all q-ary words of length n. For example, the set $\mathcal{B}_3(2)$ denotes all ternary words of length 2 thus

$$\mathcal{B}_3(2) = \{00, 01, 02, 10, 11, 12, 20, 21, 22\}$$

	first position sub	second position sub	third position sub
first position del	11	01	10
second position del	11	01	00
third position del	11	00	01

Table 1.2: Possible binary words from 011 by applying one deletion and substitution

1.4 Results found by Ward Spee

Ward Spee has found some bounds on codes that correct combinations of indel and substitution errors. The following is a lower bound on the number of codewords of a code with q-ary codewords of length n that can correct up to t indel errors and at most s substitution errors, where $t' + t'' \leq t$. Here $M_q(n, t, s)$ denotes the maximum number of q-ary codewords of length n of a code that can correct t' deletions, t'' insertions and at most s substitutions with $t' + t'' \leq t$.

Lemma 1. *[3] (Lemma 4.3) Let $n \geq 1, q \geq 2, 0 \leq t \leq n$ and $0 \leq s \leq n$ be integers. The following gives a lower bound on $M_q(n, t, s)$,*

$$M_q(n, t, s) \geq \frac{q^n}{V^{avr}_{t,t,2s}} \tag{1.5}$$

where $V^{avr}_{t,t,2s} = q^{-n} \sum_{\boldsymbol{x} \in \mathcal{B}_q(n)} |V_{t,t,2s}(\boldsymbol{x})|$

The following lemma gives another lower bound on $M_q(n, t, s)$

Lemma 2. *[3] (Lemma 4.7) Let $n \geq 1, q \geq 2, 0 \leq t \leq n$ and $0 \leq s \leq n$ be integers. The following gives a lower bound on $M_q(n, t, s)$,*

$$M_q(n, t, s) \geq \sum_{\boldsymbol{x} \in \mathcal{B}_q(n)} \frac{1}{|V_{t,t,2s}(\boldsymbol{x})|} \tag{1.6}$$

Spee has also derived an upper bound on $M_q(n, t, s)$ in the following theorem

Theorem 1. *[3] (Theorem 6.2) Let $n \geq 2, q \geq 2, 0 \leq t < n$ and $0 \leq s \leq n$ be integers. The following gives an upper bound on $M_q(n, t, s)$ for all integers $0 \leq r \leq n$ and $0 \leq t', t'' < n$ such that $t' + t'' = t$,*

$$M_q(n, t, s) \leq \frac{q^{n-t'+t''}}{min_{\boldsymbol{x} \in \mathcal{B}_q(n); r(\boldsymbol{x}) > r} |V_{t',t'',s}(\boldsymbol{x})|} + q \sum_{i=1}^{r} \binom{n-1}{i-1} (q-1)^{i-1} \tag{1.7}$$

Our goal is to find upper and lower bounds on the size of this set $V_{t',t'',s}(\mathbf{x})$ which depend on the word $\mathbf{x}$, the number of deletions and insertions t' and t'' and the number of substitutions s, in order to give an insight in the upper and lower bounds on $M_q(n, t, s)$ found by Ward Spee.

1.5 Organisation of this thesis

In order to find bounds for the size of the set $V_{t',t'',s}(\mathbf{x})$, we first give exact expressions in the case in which either $s = 0$ or $t' = t'' = 0$ in Chapter 2. Then we will try to find exact expressions or bounds for the size of $V_{1,0,1}(\mathbf{x})$ and $V_{0,1,1}(\mathbf{x})$ and look at the behaviour of these bounds for large values of q and n in Chapter 3. Finally we consider the case in which $t' = t'' = s = 1$ and we will use the results to say something useful about the bounds provided by Ward Spee in Chapter 4. Finally in Chapter 5 we will state the conclusions and give some recommendations regarding further research into the size of $V_{t',t'',s}(\mathbf{x})$.

Chapter 2

Exclusively indel or substitution errors

Before we consider the case in which there occur both substitution errors and indel errors, we look at the case in which we either have exclusively substitution errors or exclusively indel errors.

2.1 Exclusively substitution errors

If we look at the case $t' = t'' = 0$ then the size of our set $V(\mathbf{x}, 0, 0, s)$ is simply equal to the number of vectors we can obtain by changing a number of symbols between 0 to s. We define the set of words we can obtain as

$$V(\mathbf{x}, 0, 0, s) = \mathcal{S}_s(\mathbf{x})$$

The size of this set is known and straightforward to prove. When we change $0 \leq j \leq s$ symbols in a word, there are $\binom{n}{j}$ choices for the set of symbols we change and every symbol we change can become any of $q-1$ symbols. Therefore, if there occur exactly j substitutions, we can obtain exactly $\binom{n}{j}(q-1)^j$ words. Hence the total number of words we can obtain by changing at most s symbols is equal to

$$|\mathcal{S}_s(\mathbf{x})| = \binom{n}{0}(q-1)^0 + \binom{n}{1}(q-1)^1 + ... + \binom{n}{s}(q-1)^s = \sum_{j=0}^{s} \binom{n}{j}(q-1)^j$$

We conclude that the exact size of $V(\mathbf{x}, 0, 0, s)$ is always known whenever there are no indel errors made.

2.2 Exclusively indel errors

The case in which we only encounter indel errors is more complicated than the case in which we only encounter substitutions for several reasons. First of all, the length of the word will increase or decrease by $|t'' - t'|$ symbols. Secondly, the size of $V(\mathbf{x}, t', t'', s)$ does not only depend on the length of the word $\mathbf{x}$ but also on other properties of $\mathbf{x}$. When we look at binary words of length 2 for example, with 1 deletion and 1 insertion, the sets $V(\mathbf{x}, 1, 1, 0)$ are the following:

$$V(00, 1, 1, 0) = \{00, 01, 10\}$$

$$V(01, 1, 1, 0) = \{00, 01, 10, 11\} = V(10, 1, 1, 0)$$

$$V(11, 1, 1, 0) = \{01, 10, 11\}$$

Hence the type of word $\mathbf{x}$ does influence the size of the set $V(\mathbf{x}, t', t'', s)$.

2.2.1 Insertion only

We first give the following definition:

Definition 2. *The set $\mathcal{I}_{t''}(\boldsymbol{x}) = \mathcal{V}_{0,t'',0}(\boldsymbol{x})$ is the set of words that can be reached by exactly t'' insertions on $\boldsymbol{x}$.*

The words in this set are of length $n + t''$ of which t'' have been inserted. The size of this set has been stated and proven by Ward Spee [3]. The original result has been stated by Levenshtein. [4] We will state the expression for this size and give an alternative proof. Since the proof is quite long and the result is known, the proof can be found in the appendix.

Theorem 2. *[4] Let $\boldsymbol{x}$ be an arbitrary word in $\mathcal{B}_q(n)$. Then the cardinality of the set $\mathcal{I}_{t''}(\boldsymbol{x})$ is given by the following expression:*

$$|\mathcal{I}_{t''}(\boldsymbol{x})| = \sum_{i=0}^{t''} \binom{n + t''}{i} (q-1)^i$$

We conclude that the size of the sets $\mathcal{S}_s(\mathbf{x})$ and $\mathcal{I}_{t''}(\mathbf{x})$ do only depend on the number of substitutions or insertions and the length of the word and do not depend on the structure of the word $\mathbf{x}$.

2.2.2 Deletion only

We need the following definition when we examine the case of only deletions:

Definition 3. *The set $\mathcal{D}_{t'}(\boldsymbol{x}) = \mathcal{V}_{t',0,0}(\boldsymbol{x})$ is the set of words that can be reached by exactly t' deletions on $\boldsymbol{x}$.*

We will use an example to illustrate that the set $\mathcal{D}_{t'}(\mathbf{x})$ does not only depend on the parameters t' and n, but also on the structure of the word $\mathbf{x}$. Consider the words $000, 010 \in \mathcal{B}_2(3)$. Then the following holds with regard to the set $\mathcal{D}_{t'}(\mathbf{x})$;

$$\mathcal{D}_1(000) = \{00\}, \mathcal{D}_1(010) = \{01, 00, 10\}$$

and

$$\mathcal{D}_2(000) = \{0\}, \mathcal{D}_2(010) = \{0, 1\}$$

Thus we can conclude that there cannot exist a simple expression depending on n and t' to describe the size of the set $\mathcal{D}_{t'}(\mathbf{x})$. For binary $\mathbf{x}$ and $t \leq 5$ there exists a formula for $\mathcal{D}_(\mathbf{x})$ provided by Mercier, Khabbazian and Bhargava. [5]
In order to get an insight in the problem, we take n to be small and we work in a binary alphabet. The case $n = 1$ is trivial since deleting one symbol always results in the empty word. For $n = 2$ we have that $|\mathcal{D}_1(00)| = |\mathcal{D}_1(11)| = 1$ and $|\mathcal{D}_1(10)| = |\mathcal{D}_1(01)| = 2$. For $n = 3$ we have that $|\mathcal{D}_1(000)| = |\mathcal{D}_1(111)| = 1, |\mathcal{D}_1(100)| = |\mathcal{D}_1(110)| = |\mathcal{D}_1(011)| = |\mathcal{D}_1(001)| = 2$ and $|\mathcal{D}_1(101)| = |\mathcal{D}_1(010)| = 3$.
We can see a pattern emerge. Namely the size of the set $\mathcal{D}_1(\mathbf{x})$ is equal to the number of runs in $\mathbf{x}$ say $r(x)$. A run is defined as a sequence of identical symbols where the symbols directly before and directly after the run are different from the symbol in the run. Hence the word 00000 consists of only one run of length 5 and not multiple runs of different lengths. The following lemma has been stated first by Levenshtein.

Lemma 3. *[4] Let $n \geq 1$ be an integer and let $\boldsymbol{x} \in \mathcal{B}_2(n)$ be a word with $r(\boldsymbol{x})$ runs. Then the number of words that can be attained by deleting one symbol of $\boldsymbol{x}$ is given by*

$$|\mathcal{D}_1(\boldsymbol{x})| = r(\boldsymbol{x})$$

Proof. We begin by showing that we can obtain at least $r(\mathbf{x})$ words by deleting one symbol. We number the runs $1, 2, 3, ..., r(\mathbf{x})$. We obtain $r(\mathbf{x})$ different words by deleting one symbol in any of the $r(\mathbf{x})$ runs since if we look at two arbitrary words we obtained where we deleted symbols in runs i and j with $i < j$, then the words differ in run i. Since the location of the symbol we delete in a run does not make a difference we conclude that we can obtain exactly $r(\mathbf{x})$ words by deleting one symbol from $\mathbf{x}$. $\square$

We now obtained an expression for the size of $\mathcal{D}_1(\mathbf{x})$ i.e. for one deletion. Now we will try to find an expression for $1 \leq t' < n$. Table 2.1 shows the size of $\mathcal{D}_{t'}(\mathbf{x})$ for $t' = 2, 3$ and different words $\mathbf{x}$ of length 3 and 4;

From this table we can already note that the size of $\mathcal{D}_2(\mathbf{x})$ does not only depend on the number of runs, but also on the number of ones and zeros. We can say the following about the size of $\mathcal{D}_{n-1}(\mathbf{x})$ and $\mathcal{D}_{n-2}(\mathbf{x})$.
We define the lowest number of ones or zeros in a word $\mathbf{x}$ to be $l(\mathbf{x})$. Then the word 00011 has an $l(\mathbf{x})$ of 2. We can express the size of $\mathcal{D}_{n-1}(\mathbf{x})$ and $\mathcal{D}_{n-2}(\mathbf{x})$ as follows:

	r($\mathbf{x}$)	$\lvert\mathcal{D}_2(\mathbf{x})\rvert$	$\lvert\mathcal{D}_3(\mathbf{x})\rvert$
000	1	1	-
001	2	2	-
010	3	2	-
100	2	2	-
011	2	2	-
101	3	2	-
110	2	2	-
111	1	1	-
0000	1	1	1
0001	2	2	2
0010	3	3	2
0100	3	3	2
1000	2	2	2
0011	2	3	2
0101	4	4	2
0110	3	4	2
1001	3	4	2
1010	4	4	2
1100	2	3	2
0111	2	2	2
1011	3	3	2
1101	3	3	2
1110	2	2	2
1111	1	1	1

Table 2.1: The size of $\mathcal{D}_{t'}(\mathbf{x})$ for different t' and $\mathbf{x}$

Lemma 4. *Let $\boldsymbol{x}$ be a word in $\mathcal{B}_2(n)$. The cardinality of the set $\mathcal{D}_{n-1}(\boldsymbol{x})$ can be expressed solely by $r(\boldsymbol{x})$, the number of runs of $\boldsymbol{x}$. This can be expressed as follows:*

$$|\mathcal{D}_{n-1}(\boldsymbol{x})| = \begin{cases} 1 & \textit{if } r(\boldsymbol{x}) = 1 \\ 2 & \textit{if } r(\boldsymbol{x}) > 1 \end{cases} \tag{2.1}$$

Proof. First of all we consider the case $r(\mathbf{x}) = 1$. Then $\mathbf{x}$ consists of either n ones or n zeros. When $n-1$ deletions occur, there is only one possible word we can obtain in both cases, namely a one in the first case and a zero in the second. Thus $|\mathcal{D}_{n-1}(\mathbf{x})| = 1$ whenever $r(\mathbf{x}) = 1$.
Secondly we need to consider the case $r(\mathbf{x}) > 1$. In this case the word $\mathbf{x}$ consists of both ones and zeros. When we delete $n-1$ symbols, we can therefore always end up with either a one or a zero. Thus $|\mathcal{D}_{n-1}(\mathbf{x})| = 2$ whenever $r(\mathbf{x}) > 1$.

□

Lemma 5. *Let $\boldsymbol{x} \in \mathcal{B}_2(n)$, then the cardinality of $\mathcal{D}_{n-2}(\boldsymbol{x})$ can be described by $r(\boldsymbol{x})$ and $l(\boldsymbol{x})$ as follows:*

$$|\mathcal{D}_{n-2}(\boldsymbol{x})| = \begin{cases} 1 & \textit{if } r(\boldsymbol{x}) = 1 \\ 2 & \textit{if } r(\boldsymbol{x}) = 2 \textit{ and } l(\boldsymbol{x}) = 1 \\ 3 & \textit{if } r(\boldsymbol{x}) = 2 \textit{ and } l(\boldsymbol{x}) \geq 2 \\ 3 & \textit{if } r(\boldsymbol{x}) \geq 3 \textit{ and } l(\boldsymbol{x}) = 1 \\ 4 & \textit{if } r(\boldsymbol{x}) \geq 3 \textit{ and } l(\boldsymbol{x}) \geq 2 \end{cases} \tag{2.2}$$

Proof. i) We start with the case $r(\mathbf{x}) = 1$. In this case $\mathbf{x}$ consists of either n ones or n zeros and the deletion of $n-2$ symbols results in the word 11 (when $\mathbf{x} = 11\ldots11$) or 00 (when $\mathbf{x} = 00\ldots00$). Thus $|\mathcal{D}_{n-2}(\mathbf{x})| = 1$ whenever $r(\mathbf{x}) = 1$.

ii) Now suppose that $r(\mathbf{x}) = 2$ and $l(\mathbf{x}) = 1$. Since $l(\mathbf{x}) = 1$ we know that $\mathbf{x}$ has either exactly one *one* or one *zero*. The number of runs is equal to 2 which means that either the first or the last symbol must be equal to the single one or zero. We have four possible words with these properties namely $\mathbf{x} \in \{011\ldots11, 100\ldots00, 00\ldots001, 11\ldots110\}$. In all four cases there are only two words in the set $\mathcal{D}_{n-2}(\mathbf{x})$ namely $\{01, 11\}, \{10, 00\}, \{00, 01\}$ and $\{11, 10\}$ respectively. We conclude that $|\mathcal{D}_{n-1}(\mathbf{x})| = 2$ whenever $r(\mathbf{x}) = 2$ and $l(\mathbf{x}) = 1$.

iii) A word $\mathbf{x}$ with 2 runs and an $l(\mathbf{x})$ greater or equal to 2 is a word of $n-2 \geq i \geq 2$ ones followed by $n-i$ zeros or the other way around. Since $l(\mathbf{x}) \geq 2$ we can always obtain the words 00 and 11 when deleting $n-2$ symbols. We can also obtain the word 01 when $\mathbf{x}$ starts with zeros and 10 when it starts with ones but not the word 10 in the first case and 01 in the second. Hence $|\mathcal{D}_{n-2}(\mathbf{x})| = 3$ whenever $r(\mathbf{x}) = 2$ and $l(\mathbf{x}) \geq 2$.

iv) Suppose $\mathbf{x}$ has at least 3 runs and an $l(\mathbf{x})$ of 1. Then there is always at least one one before a zero, and at least one zero before a one, hence $01, 10 \in \mathcal{D}_{n-2}(\mathbf{x})$. Since $l(\mathbf{x}) = 1$ either $00 \in \mathcal{D}_{n-2}(\mathbf{x})$ or $11 \in \mathcal{D}_{n-2}(\mathbf{x})$ but not both. We conclude that $|\mathcal{D}_{n-2}(\mathbf{x})| = 3$ whenever $r(\mathbf{x}) \geq 3$ and $l(\mathbf{x}) = 1$.

v) Finally suppose that $r(\mathbf{x}) \geq 3$ and $l(\mathbf{x}) \geq 2$. Since $l(\mathbf{x}) \geq 2$ we know that $00, 11 \in \mathcal{D}_{n-2}(\mathbf{x})$ and since $r(\mathbf{x}) \geq 3$ we know that $01, 10 \in \mathcal{D}_{n-2}(\mathbf{x})$ thus $|\mathcal{D}_{n-2}(\mathbf{x})| = 4$ whenever $r(\mathbf{x}) \geq 3$ and $l(\mathbf{x}) \geq 2$.

□

Mercier et al has determined the exact size of these sets for $2 \leq t' \leq 5$ [5], however the analytical expression becomes too complex to include in this thesis.

Chapter 3

Combinations of substitution and indel errors

Now we will be looking at combinations of indel and substiution errors and the size of $V_{t',t'',s}(\mathbf{x})$. There are only three exact results known on explicit values for t', t'' and s, namely the size of $V_{1,1,0}(\mathbf{x})$, which has been derived by Sala and Dolecek [6] and the size of $V_{1,0,s}(\mathbf{x})$ and $V_{0,1,1}(\mathbf{x})$ which have been derived by Abu-Sini and Yaakobi [7]. However, these expressions only hold for binary words. Ward Spee has stated and proven an expression for $|V_{1,0,1}(\mathbf{x})|$ that holds for arbitrary $\mathbf{x} \in \mathcal{B}_q(n)$. We do not expect to find an exact expression for the size of $V_{t',t'',s}(\mathbf{x})$ but we are interested in finding bounds on the size of this set. Ward Spee has examined lower bounds on the size of these sets, however the goal of this rapport is to look for upper bounds.

3.1 Size of $V_{1,0,1}(\mathbf{x})$

The size of $V_{1,0,1}(\mathbf{x})$ is known for arbitrary $\mathbf{x} \in \mathcal{B}_q(n)$. Ward Spee has derived and proven a general expression for $|V_{1,0,1}(\mathbf{x})|$ in his master thesis. We will state the results here, however for the proof we refer the reader to the thesis of Ward Spee.[3]

Lemma 6. *Let $n \geq 1$ and $q \geq 2$ be integers and $\boldsymbol{x} \in \mathcal{B}_q(n)$. Then, the following holds,*

$$|V_{1,0,1}(\boldsymbol{x})| = \begin{cases} (n-1)(q-1)+1 & \text{if } r(\boldsymbol{x}) = 1, \\ r(\boldsymbol{x})((n-2)(q-1)-1)+q+2 & \text{if } r(\boldsymbol{x}) \geq 2. \end{cases} \tag{3.1}$$

n, r($\mathbf{x}$) \ q	2	3	4	5
2, 1	7	19	37	61
2, 2	8	23	46	77
3, 1	11	33	67	113
3, 2	13	41	85	145
3, 3	14	45	94	161
4, 1	16	51	106	181
4, 2	19, 20	63, 67	133, 142	229, 245
4, 3	21	71	151	261
4, 4	22	75	160	277
5, 1	22	73	154	265
5, 2	26, 28	89, 97	190, 208	329, 361
5, 3	29, 30	101, 105	217, 226	377, 393
5, 4	31	109	235	409
5, 5	32	113	244	425

Table 3.1: Possible values of $|V_{0,1,1}(\mathbf{x})|$ for different q, n and $r(\mathbf{x})$

3.2 Size of $V_{0,1,1}(\mathbf{x})$

The size of $V_{0,1,1}(\mathbf{x})$ is known only for binary words and given by Abu-Sini and Yaakobi [7].

Lemma 7. *([7], Theorem 10) Let $n \geq 1$ be an integer and let $\boldsymbol{x} \in \mathcal{B}_2(n)$ be a binary word with $r = r(\boldsymbol{x})$ runs of lengths $l_1, l_2, \ldots, l_r$. The number of words that can be obtained from $\boldsymbol{x}$ by one insertion and at most one substitution is given by*

$$|V_{0,1,1}(\boldsymbol{x})| = (n+2)^2 - 2 - \sum_{i=1}^{r} \frac{l_i(l_i+5)}{2} \tag{3.2}$$

In this section we will try to find an expression for $V_{0,1,1}(\mathbf{x})$ for $x \in \mathcal{B}_q(n)$ with $q > 2$. Since the size of this set is known for binary words and depends only on the length of the word n and the lengths of the runs of the word $l_1, l_2, \ldots, l_r$ it makes sense to investigate whether words with similar runs and length create the same size of $V_{0,1,1}(\mathbf{x})$. Therefore we have used a computer to find the size of this set for different n, q and $r(\mathbf{x})$. The results are shown in Table 3.1.

First of all we note that for the number of runs equal to $2, 3, \ldots, n-2$ there seems to be more than one possible value for the value of $|V_{0,1,1}(\mathbf{x})|$. The difference here lies in the length of the runs. A word with length 5 and 3 runs might have two runs of length 1 and one of length 3 or it might have two runs of length 2 and one of length 1. These words result in a different size of $V_{0,1,1}(\mathbf{x})$. In order to get an idea of a general expression for all $\mathbf{x}$ we start by showing that the following holds for $\mathbf{x}$ with $r(\mathbf{x}) = 1$.

Lemma 8. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ be a word with $r(\boldsymbol{x}) = 1$. Then we have that*

$$|V_{0,1,1}(\boldsymbol{x})| = 1 + (n+1)(q-1) + \binom{n+1}{2}(q-1)^2$$

Proof. Without loss of generality we can assume that

$$\mathbf{x} = \underbrace{00\ldots00}_{\text{n zeros}}$$

Then we can note that by inserting one symbol and substituting at most one symbol, we cannot obtain a word with more than 2 non-zero symbols. Furthermore note that we can obtain all words with at most two non-zero symbols of length $n+1$ from $\mathbf{x}$ by one insertion and at most one substitution. Therefore the size of $V_{0,1,1}(\mathbf{x})$ is equal to the number of words of length $n+1$ with at most two non-zero symbols, hence

$$|V_{0,1,1}(\mathbf{x})| = 1 + (n+1)(q-1) + \binom{n+1}{2}(q-1)^2$$

which is exactly the expression we wanted to prove.

□

Later in this thesis we will write this expression in polynomial form which is given by the following expression:

$$\begin{aligned}|V_{0,1,1}(\mathbf{x})| &= 1 + (n+1)(q-1) + \binom{n+1}{2}(q-1)^2 \\ &= \frac{(n+1)(n(q-1)^2 + 2(q-1))}{2} + 1\end{aligned}$$

Now we will try to find an expression for the maximum cardinality of $V_{0,1,1}(\mathbf{x})$. It appears that the maximum cardinality is attained whenever $r(\mathbf{x}) = n$. We will later prove this. In order to derive an expression for the maximum cardinality of $V_{0,1,1}(\mathbf{x})$ we make use of a computer to determine the maximum values of this set for different q and n. The results are shown in Table 3.2.

From these results, an expression for the maximum cardinality of $V_{0,1,1}(\mathbf{x})$ can be found namely

$$\max_{\mathbf{x} \in \mathcal{B}_q(n)} |V_{0,1,1}(\mathbf{x})| = n^2q^2 - (2n^2 - n - 1)q + n(n-1)$$

If the reader wants to know how we have found this expression, we refer the reader to the appendix where we show in detail how this is done for the expression shown in Lemma 9. The expressions stated in Lemma 12 and in Table 3.4 have been derived in a similar fashion.
In order to show that this is true, we will first show that it is the case whenever $r(\mathbf{x}) = n$ and then we will show that for all n,q and $\mathbf{y}, \mathbf{x} \in \mathcal{B}_q(n)$ it holds that $|V_{0,1,1}(\mathbf{y})| < |V_{0,1,1}(\mathbf{x})|$ for $r(\mathbf{x}) = n$ and $r(\mathbf{y}) < n$.

$q \backslash n$	2	3	4	5
2	8	14	22	32
3	23	45	75	113
4	46	94	160	244
5	77	161	277	425
6	116	246	426	656
7	163	349	607	937
8	218	470	820	1268
9	281	609	1065	1649
10	352	766	1342	

Table 3.2: Maximum values of $|V_{0,1,1}(\mathbf{x})|$ for different n and q

Lemma 9. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ and $r(\boldsymbol{x}) = n$. Then we have that*

$$|V_{0,1,1}(\boldsymbol{x})| = n^2q^2 - (2n^2 - n - 1)q + n(n-1)$$

Proof. First we will show that the expression holds for $n = 2, q \geq 2$ and then we will complete the proof using induction on n to show it holds for all $n \geq 2$. Let $\mathbf{x} \in \mathcal{B}_q(2)$ be arbitrary. Then without loss of generality we can assume that $\mathbf{x} = 01$ since for $a \neq b$ and $c \neq d$ we have that $|V_{0,1,1}(ab)| = |V_{0,1,1}(cd)|$ for $a, b, c, d \in \{0, 1, \ldots, q-1\}$. We want to prove that for $n = 2$:

$$|V_{0,1,1}(\mathbf{x})| = n^2q^2 - (2n^2 - n - 1)q + n(n-1) = 4q^2 - 5q + 2 \qquad (3.3)$$

First we apply at most one substitution. We can obtain all words of length 2 with either a 0 at the first position, or a 1 at the second position. Now by applying an insertion to these words, we can obtain all words of length 3, with either a zero at the first or second position, or a one at the second or third position. Now we still need to count all words with one of these properties. There are q^2 words where the first symbol is a 0. There are $(q-1)q$ words where the second symbol is a 0 and the first symbol is not a 0 (otherwise we count words double). There are $(q-1)q$ words where the second symbol is a 1 and the first symbol is not a 0. Finally there are $(q-1)(q-2)$ words where the third symbol is a 1, the first symbol is not a 0 and the second symbol is not a 0 or 1. Thus the total number of words is

$$|V_{0,1,1}(01)| = q^2 + 2q(q-1) + (q-1)(q-2) = 4q^2 - 5q + 2 \qquad (3.4)$$

which is exactly what we needed to prove.
Now we complete the proof using induction on n. We already showed that the expression holds for $n = 2, q \geq 2$. Now suppose that the expression is true for all $n \leq N$ and $q \geq 2$ for some $N \geq 2$. Then for $n = k \leq N$ we have

$$|V_{0,1,1}(\mathbf{x})| = k^2q^2 - (2k^2 - k - 1)q + k(k-1)$$

Now we consider an arbitrary word $\mathbf{y} \in \mathcal{B}_q(k+1)$ with $r(\mathbf{x}) = k+1$. When we look at the set $|V_{0,1,1}(\mathbf{y})|$ we can distinguish between two cases, namely either the insertion and the substitution occur beore the $k+1$'th symbol or they do not. If they do happen before the last symbol, we know from the induction hypothesis that this results in exactly $|V_{0,1,1}(\mathbf{x})|$ different words. Now we consider the case that either the insertion occurs at the end of the word, or the last symbol is substituted.

Case 1: last symbol is substituted: If the last symbol is substituted, then there are $q-1$ choices for this substitution. There are q choices for an insertion at the first position and in order to avoid double counting (see Theorem 2) we have $q-1$ possible values for insertions at positions 2 to $k+1$. Therefore we count a total of $(q-1)(k(q-1)+q)$ different words when we substitute the last symbol. It is clear that we did not count words double, since the last symbol remained unchanged in the first set of words, while it is changed (by a substitution) in the second set.

Case 2: Insertion after last symbol and one substitution first k symbols: Now we consider the final case, in which there occurs an insertion at the end of the word and at most one substitution at the first k symbols. For the insertion and the substitution there are exactly $q-1$ choices, therefore the total number of words where the insertion happens at the end equals $k(q-1)^2$. We again did not count words double, since the last symbol is different from the $k+2'th$ symbol of $\mathbf{y}$ we did not count words from the set $|V_{0,1,1}(\mathbf{x})|$ and the $k+1$'th symbol is different when compared to the $k+1$'th symbol in the set of word where the $k+1$'th symbol is substituted. We have only missed the case in which there does not occur any substitution and the insertion occurs at the end of the word. However, this is equivalent to substituting the $k+1$'th symbol and inserting a symbol before the $k+1$'th symbol. Thus we conclude that

$$\begin{aligned}
|V_{0,1,1}(\mathbf{y})| &= |V_{0,1,1}(\mathbf{x})| + (q-1)(k(q-1)+q) + k(q-1)^2 \\
&= k^2q^2 - (2k^2-k-1)q + k(k-1) + (q-1)(kq-k+q) + k(q^2-2q+1) \\
&= k^2q^2 - (2k^2-k-1)q + k(k-1) + (k+1)q^2 - (2k+1)q + k + kq^2 - 2kq + k \\
&= (k^2+2k+1)q^2 - (2k^2+3k)q + k(k-1) + 2k \\
&= (k+1)^2q^2 - (2(k+1)^2 - (k+1) - 1)q + (k+1)k
\end{aligned}$$

which is exactly the expression for $n = k+1$. Hence we have proved the induction towards n. Therefore the expression holds for all $n, q \geq 2$.

□

We stated that the cardinality of $V_{0,1,1}(\mathbf{x})$ is maximal for $r(\mathbf{x}) = n$ without proof. In order to prove this, we will first need to proof the following lemma:

Lemma 10. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{B}_q(n)$ such that $q \geq 3$, $d(\boldsymbol{x}, \boldsymbol{y}) = 1$ and $r(\boldsymbol{x}) = r(\boldsymbol{y}) + 1$. Then $|V_{0,1,1}(\boldsymbol{x})| > |V_{0,1,1}(\boldsymbol{y})|$.*

Proof. If $d(\mathbf{x}, \mathbf{y}) = 1$ then $\mathbf{x}$ and $\mathbf{y}$ only differ in one position, say $x_i \neq y_i$. We can write $\mathbf{x}$ and $\mathbf{y}$ as the following:

$$\mathbf{x} = x_1 x_2 \dots x_{i-1} x_i x_{i+1} \dots x_n$$

and

$$\mathbf{y} = x_1 x_2 \dots x_{i-1} y_i x_{i+1} \dots x_n$$

Since $r(\mathbf{x}) > r(\mathbf{y})$ this implies that either $y_i = x_{i-1} \neq x_i$ or $y_i = x_{i+1} \neq x_i$. We can assume without loss of generality that $y_i = x_{i-1}$. Hence

$$\mathbf{x} = x_1 x_2 \dots x_{i-2} x_{i-1} x_i x_{i+1} \dots x_n$$

and

$$\mathbf{y} = x_1 x_2 \dots x_{i-2} x_{i-1} x_{i-1} x_{i+1} \dots x_n$$

Now suppose there would only occur substitutions and insertions before the $(i-2)^{nd}$ symbol or after the $(i+1)^{st}$ symbol. Then the size of the set $V_{0,1,1}(\mathbf{x})$ and $V_{0,1,1}(\mathbf{y})$ would be equal since they are identical where the substituion and insertion take place. Thus there is only a diffence in size of these sets if there are inserions or substitutions at or between the $(i-2)^{nd}$ and $(i+1)^{th}$ symbol. If the insertion happens before x_{i-2} or after x_{i+1} then for both $\mathbf{x}$ and $\mathbf{y}$ there are $4(q-1)$ choices for the substitution between x_{i-2} and x_{i+1}. The only difference in size between $V_{0,1,1}(\mathbf{x})$ and $V_{0,1,1}(\mathbf{y})$ occurs therefore when both the insertion and/or substitution occur between x_{i-2} and x_{i+1}. If we let

$$\mathbf{a} = x_{i-2} x_{i-1} x_i x_{i+1}$$

and

$$\mathbf{b} = x_{i-2} x_{i-1} x_{i-1} x_{i+1}$$

then we can therefore conclude that

$$|V_{0,1,1}(\mathbf{x})| - |V_{0,1,1}(\mathbf{y})| = |V_{0,1,1}(\mathbf{a})| - |V_{0,1,1}(\mathbf{b})|$$

Without loss of generality we can let $x_i = 0$ and $x_{i-1} = 1$. Since we need to ensure that $r(\mathbf{x}) = r(\mathbf{y}) + 1$ we have that $r(\mathbf{a}) \in \{2, 3, 4\}$ and $r(\mathbf{b}) \in \{1, 2, 3\}$. If $r(\mathbf{a}) = 2$ we have $\mathbf{a} = 1100$ and $\mathbf{b} = 1110$ thus $r(\mathbf{a}) = r(\mathbf{b})$ hence we conclude that $r(\mathbf{a}) \neq 2$. Now suppose that $r(\mathbf{a}) = 3$, then $\mathbf{a} = 110c$ and $\mathbf{b} = 111c$ where $c \in \{2, 3, \dots, q-1\}$. Finally suppose that $r(\mathbf{a}) = 4$, then $\mathbf{a} = c10d$ and $\mathbf{b} = c11d$ where $c, d \in \{2, 3, \dots, q-1\}$.

Thus $\mathbf{a} = 110c, \mathbf{b} = 111c$ or $\mathbf{a} = c10d, \mathbf{b} = c11d$ with $c, d \in \{2, 3, \dots, q-1\}$. We can use the same reasoning we used before to conclude that $|V_{0,1,1}(\mathbf{a})| - |V_{0,1,1}(\mathbf{b})| = |V_{0,1,1}(10)| - |V_{0,1,1}(11)|$. We will determine all words that can be reached from both 11 and 10 by applying an insertion and at most one substitution. When applying an insertion and exactly one substitution to the word 11 we can reach all words of length 3 that include at least one 1. The total number of words of length 3 with at least one 1 is equal to the sum of the number of words with exactly one 1, the number of words with exactly two ones and the word with three ones. Thus $|V_{0,1,1}(11)| = 3(q-1)^2 + 3(q-1) + 1 = 3q^2 - 3q + 1$. The set $V_{0,1,1}(10)$ contains all words of length 3 that contain at least one 1 or one 0. Since 000 is in this set and every word in $V_{0,1,1}(11)$ is also in this set we can conclude that $|V_{0,1,1}(10)| > |V_{0,1,1}(11)|$ and therefore we conclude that $|V_{0,1,1}(\mathbf{x})| > |V_{0,1,1}(\mathbf{y})|$.

$\square$

We can use this result to show that $|V_{0,1,1}(\mathbf{x})|$ is maximal for $r(\mathbf{x}) = n$.

Lemma 11. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ with $q \geq 3$ such that $|V_{0,1,1}(\boldsymbol{x})| \geq |V_{0,1,1}(\boldsymbol{y})|$ for all $\boldsymbol{y} \in \mathcal{B}_q(n)$. Then $r(\boldsymbol{x}) = n$.*

Proof. Suppose there exists a word $\mathbf{x} \in \mathcal{B}_q(n)$ such that $|V_{0,1,1}(\mathbf{x})| \geq |V_{0,1,1}(\mathbf{y})|$ for all $\mathbf{y} \in \mathcal{B}_q(n)$ with $r(\mathbf{x}) < n$. Then $\mathbf{x}$ contains a run of length 2 or higher. Suppose this run starts at position i. Then $\mathbf{x} = x_1 x_2 \ldots x_{i-1} x_i x_{i+1} \ldots x_n$ with $x_{i-1} \neq x_i = x_{i+1}$. Let $\mathbf{y} \in \mathcal{B}_q(n)$ such that $y_j = x_j$ for all $j \neq i$ and $y_i \notin \{x_{i-1}, x_i\}$. This is possible since $q \geq 3$. Then $r(\mathbf{y}) = r(\mathbf{x}) + 1$ hence by Lemma 10 we conclude that $|V_{0,1,1}(\mathbf{y})| > |V_{0,1,1}(\mathbf{x})|$ which is a contradiction. Therefore we conclude that $r(\mathbf{x})$ must be equal to n.

$\square$

With the same reasoning we can prove that $|V_{0,1,1}(\mathbf{x})|$ is minimal for $r(\mathbf{x}) = 1$. We therefore have derived a lower and an upper bound for the size of the set $|V_{0,1,1}(\mathbf{x})|$ namely

Theorem 3. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ with $q \geq 3$. Then the following holds,*

$$\frac{(n+1)(n(q-1)^2 + 2(q-1))}{2} + 1 \leq |V_{0,1,1}(\boldsymbol{x})| \leq n^2 q^2 - (2n^2 - n - 1)q + n(n-1) \tag{3.5}$$

with equality for $r(\boldsymbol{x}) = 1$ and $r(\boldsymbol{x}) = n$ respectively.

We needed the condition $q \geq 3$ to hold since we needed to be able to choose a third symbol in order to have both $d(\mathbf{x}, \mathbf{y}) = 1$ and $r(\mathbf{x}) = r(\mathbf{y}) + 1$. We want to extend the upper and lower bound to the binary case and therefore we need to show that $|V_{0,1,1}(\mathbf{x})|$ is maximal for $r(\mathbf{x}) = n$ and minimal for $r(\mathbf{x}) = 1$. Fortunately we already established that there exists an expression for $|V_{0,1,1}(\mathbf{x})|$ for binary words namely

$$|V_{0,1,1}(\mathbf{x})| = (n+2)^2 - 2 - \sum_{i=1}^{r} \frac{l_i(l_i+5)}{2} \tag{3.6}$$

Maximizing or minimizing this expression for set n is equivalent to minimizing or maximizing the sum $\sum_{i=1}^{r} \frac{l_i(l_i+5)}{2}$ under the condition that $l_1 + l_2 + \ldots + l_r = n$.

In the case that $r(\mathbf{x}) = n$ we have $l_1 = l_2 = \ldots = l_n = 1$, thus

$$\sum_{i=1}^{r} \frac{l_i(l_i+5)}{2} = 3n \tag{3.7}$$

and in case $r(\mathbf{x}) = 1$ we have $l_1 = n$ and

$$\sum_{i=1}^{r} \frac{l_i(l_i+5)}{2} = \frac{n(n+5)}{2} \tag{3.8}$$

For $n \geq 2$ we have that $3n < \frac{n(n+5)}{2}$ since the equation

$$\frac{n(n+5)}{2} - 3n = 0 \implies n^2 - n = 0 \implies n = 0, 1$$

has only two roots both less than 2.
Now let $\mathbf{x} \in \mathcal{B}_q(n)$ be arbitrary with $1 < r(\mathbf{x}) < n$. Suppose $\mathbf{x}$ has r runs of length $l_1, l_2, \ldots, l_r$, where $l_i \leq l_{i+1}$. Now let **x'** be a word with runs of length $l_1, l_2, \ldots, l'_r, l'_{r+1}$ where $l'_r + l'_{r+1} = l_r$. Then $r(\mathbf{x'}) = r + 1 > r(\mathbf{x})$ and

$$\sum_{i=1}^{r} \frac{l_i(l_i+5)}{2} > \sum_{i=1}^{r-1} \frac{l_i(l_i+5)}{2} + \frac{l'_r(l'_r+5) + l'_{r+1}(l'_{r+1}+5)}{2}$$

since

$$l_r(l_r+5) = (l'_r + l'_{r+1})(l'_r + l'_{r+1} + 5) > l'_r(l'_r+5) + l'_{r+1}(l'_{r+1}+5)$$

Thus for all $\mathbf{x}$ with $r(\mathbf{x}) < n$ there exists a word **x'** with $r(\mathbf{x'}) > r(\mathbf{x})$ such that

$$|V_{0,1,1}(\mathbf{x})| < |V_{0,1,1}(\mathbf{x'})|$$

hence $|V_{0,1,1}(\mathbf{x})|$ is maximal for $r(\mathbf{x}) = n$.

We again take $\mathbf{x}$ with runs $l_1, l_2, \ldots, l_r$. Let $\mathbf{y}$ be a word with $r-1$ runs of length $l_1, l_2, \ldots, l'_{r-1}$ where $l'_{r-1} = l_{r-1} + l_r$. Then by the same reasoning as above we find

$$\sum_{i=1}^{r-2} \frac{l_i(l_i+5)}{2} + \frac{l'_{r-1}(l'_{r-1}+5)}{2} > \sum_{i=1}^{r} \frac{l_i(l_i+5)}{2}$$

and

$$|V_{0,1,1}(\mathbf{x})| > |V_{0,1,1}(\mathbf{y})|$$

whenever $r(\mathbf{y}) < r(\mathbf{x})$ hence $|V_{0,1,1}(\mathbf{x})|$ is minimal for $r(\mathbf{x}) = 1$.

3.3 Behaviour of lower and upper bound $|V_{0,1,1}(\mathbf{x})|$

It might be interesting to investigate the behaviour of the lower and upper bound of $|V_{0,1,1}(\mathbf{x})|$ for large values of n and q. For large values of q the higher order terms of q will dominate hence

$$\frac{(n+1)(n(q-1)^2+2(q-1))}{2}+1 \approx \frac{n(n+1)q^2}{2} \tag{3.9}$$

$$n^2q^2-(2n^2-n-1)q+n(n-1) \approx n^2q^2 \tag{3.10}$$

and therefore for large q the inequality becomes approximately

$$\frac{n(n+1)q^2}{2} < |V_{0,1,1}(\mathbf{x})| < n^2q^2 \tag{3.11}$$

Now suppose n is very large while q is not. Then the following holds,

$$\frac{(n+1)(n(q-1)^2+2(q-1))}{2}+1 \approx \frac{(q-1)^2n^2}{2} \tag{3.12}$$

$$n^2q^2-(2n^2-n-1)q+n(n-1) \approx (q^2-2q+1)n^2 = (q-1)^2n^2 \tag{3.13}$$

Thus for large n the inequality becomes approximately

$$\frac{n^2(q-1)^2}{2} < |V_{0,1,1}(\mathbf{x})| < n^2(q-1)^2 \tag{3.14}$$

Finally we consider the case in which both n and q are large. Then $n^2q^2 \gg n^2q$ and $n^2q^2 \gg nq^2$ hence the inequality becomes

$$\frac{n^2q^2}{2} < |V_{0,1,1}(\mathbf{x})| < n^2q^2 \tag{3.15}$$

We conclude that for large values of n and q, the cardinality of the set $V_{0,1,1}(\mathbf{x})$ lies in the interval $[\frac{m}{2}, m]$ where $m = n^2q^2$.

3.4 The size of $V_{1,1,1}(\mathbf{x})$

In order to get an insight in the size of $V_{1,1,1}(\mathbf{x})$ we again look at different choices for $\mathbf{x}$, n and q. Table 3.3 shows the size of the set $V_{1,1,1}(\mathbf{x})$ for different $r(\mathbf{x}), n$ and q found by a computer algorithm.

We can see that the number of different possible sizes of $V_{1,1,1}(\mathbf{x})$ increases very rapidly, when q and n increase. Therefore it might be more useful to look at the maximum size for different $r(\mathbf{x})$. We will first focus our attention on the words for which $|V_{1,1,1}(\mathbf{x})|$ is maximal. We expect this to be the case for

n, r($\mathbf{x}$) \ q	2	3	4	5
2,1	4	9	16	25
2,2	4	9	16	25
3,1	7	19	37	61
3,2	8	25	52	89
3,3	8	26,27	56,60	98,107
4,1	11	33	67	113
4,2	14,15	49, 53	106,115	185,201
4,3	15,16	58,63,66	131,144,152	234,259,274
4,4	16	69,72,75	164,174,183	301,322,339
5,1	16	51	106	181
5,2	22,24	81,89	178,196	313,345
5,3	25,26,28,29	102,106,111,118,119,122	233,242,255,272,273,281	418,434,459,490,491,506
5,4	28,30	129,136,140,143,145,151	308,327,336,342,348,360	565,602,618,627,639,659
5,5	31	156, 164,165,172,179	385,407,409,425,427,441	718,759,766,792,796,819

Table 3.3: Possible values for $|V_{1,1,1}(\mathbf{x})|$ for different $\mathbf{x}$ and q

n \ r($\mathbf{x}$)	1	2	3	4	5
2	q^2	q^2	-	-	-
3	$3q^2 - 3q + 1$	$5q^2 - 8q + 4$	$7q^2 - 16q + 12$	-	-
4	$6q^2 - 8q + 3$	$12q^2 - 22q + 11$	$18q^2 - 40q + 24$	$24q^2 - 60q + 39$	-
5	$10q^2 - 15q + 6$	$21q^2 - 40q + 20$	$33q^2 - 72q + 41$	$45q^2 - 106q + 64$	$58q^2 - 144q + 89$

Table 3.4: Expression for $|V_{1,1,1}(\mathbf{x})|$ for different choices of n and $r(\mathbf{x})$

$\mathbf{x}$ such that $r(\mathbf{x}) = n$. We can again derive expressions for the maximal size of $|V_{1,1,1}(\mathbf{x})|$ for different values of n, q and $r(\mathbf{x})$ based on numerical results as shown in the appendix. These expressions are shown in Table 3.4 without proof for $n \geq 2$. The expressions in red only hold for $q \geq 3$.

We start again by considering words $\mathbf{x}$ with $r(\mathbf{x}) = 1$. We want to prove the following,

Lemma 12. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ be arbitrary such that $r(\boldsymbol{x}) = 1$, $n \geq 2$ and $q \geq 2$. Then the follwing holds,*

$$|V_{1,1,1}(\boldsymbol{x})| = \frac{n(n-1)}{2}q^2 - n(n-2)q + \frac{n^2 - 3n + 2}{2} \tag{3.16}$$

Proof. We will prove this by remarking that the order of deletions, insertions and substitution is irrelevant for the cardinality of $V_{1,1,1}(\mathbf{x})$ hence we can apply a deletion first and conclude that $|V_{1,1,1}(\mathbf{x})| = |V_{0,1,1}(\mathbf{y})|$ where $r(\mathbf{y}) = 1$ and $\mathbf{y} \in \mathcal{B}_q(n-1)$. We again take $\mathbf{x} = 000\ldots000$ (n zeros). Then $\mathbf{y} = 000\ldots00$

($n-1$ zeros) and using Lemma 8 we find

$$\begin{aligned}|V_{1,1,1}(\mathbf{x})| = |V_{0,1,1}(\mathbf{y})| &= \frac{n((n-1)(q-1)^2 + 2(q-1))}{2} + 1 \\ &= \frac{n(n-1)(q^2-2q+1) + 2nq - 2n}{2} + 1 \\ &= \frac{n(n-1)}{2}q^2 - n(n-2)q + \frac{n^2-3n+2}{2}\end{aligned}$$

which is the expression we wanted to prove. □

We expect the size of $V_{1,1,1}(\mathbf{x})$ to attain its minimum whenever $r(\mathbf{x}) = 1$. The following lemma states this and we will prove this.

Lemma 13. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ be arbitrary such that $r(\boldsymbol{x}) = 1$, $n \geq 2$ and $q \geq 2$. Then for all $\boldsymbol{x}$' $\in \mathcal{B}_q(n)$ we have*

$$|V_{1,1,1}(\boldsymbol{x})| \leq |V_{1,1,1}(\boldsymbol{x}\text{'})| \tag{3.17}$$

Proof. Let $\mathbf{x}$' $\in \mathcal{B}_q(n)$ be arbitrary. Let $\mathbf{y}$ be the word we obtain by deleting the first symbol of $\mathbf{x}$ and let $\mathbf{y}$' be the word we obtain by deleting the first symbol of $\mathbf{x}$' Then we note that

$$|V_{1,1,1}(\mathbf{x}\text{'})| \geq |V_{0,1,1}(\mathbf{y}\text{'})| \geq |V_{0,1,1}(\mathbf{y})| = |V_{1,1,1}(\mathbf{x})| \tag{3.18}$$

since $r(\mathbf{x}) = 1$ and since the order of deletion, insertion and substitution is irrelevant for the size of $|V_{1,1,1}(\mathbf{x})|$. Hence we conclude that $|V_{1,1,1}(\mathbf{x})|$ does indeed attain its minimum whenever $r(\mathbf{x}) = 1$.

□

3.5 Upper bound for $|V_{1,1,1}(\mathbf{x})|$

Now we look at $|V_{1,1,1}(\mathbf{x})|$. Suppose $\mathbf{x} \in \mathcal{B}_q(n)$ with $r(\mathbf{x}) = n$ and suppose that every three consecutive symbols are distinct. When applying a deletion to $\mathbf{x}$, we obtain a word of length $n-1$ with $n-1$ runs. Using Lemma 9 we can determine the following upper bound for $|V_{1,1,1}(\mathbf{x})|$ for these specific $\mathbf{x}$, namely:

$$|V_{1,1,1}(\mathbf{x})| \leq n \cdot |V_{0,1,1}(\mathbf{y})| = n((n-1)^2q^2 - (2(n-1)^2 - n)q + (n-1)(n-2))$$

3.6 Bounds on $|V_{t',t'',s}(\mathbf{x})|$

Thus far we have only examined the case in which there occurs at most one insertion, deletion or substitution. We would like to determine bounds for a general number of insertions, deletions and substitutions in order to say more about the bound provided by Ward Spee.

3.6.1 An upper bound

In order to find an upper bound on the size of $|V_{t',t'',s}(\mathbf{x})|$ we first examine the set of words we can obtain by applying t' deletions and t'' insertions and then we apply the substitutions.

Let $\mathbf{x} \in \mathcal{B}_q(n)$ be arbitrary. Then by applying t' deletions, we obtain the set $\mathcal{D}_{t'}(\mathbf{x})$, i.e. we obtain all words of length $n - t'$ that are contained in $\mathbf{x}$. For example, the word 102 is contained in the word 12202 and can be obtained by deleting the second and third symbol. Now when we apply t'' insertions to the set $\mathcal{D}_{t'}(\mathbf{x})$ to obtain the set $V_{t',t'',0}(\mathbf{x})$. Since we do not remove or substitute any symbols, we note that for every word $\mathbf{y} \in V_{t',t'',0}(\mathbf{x})$ there exists a word $\mathbf{y}' \in \mathcal{D}_{t'}(\mathbf{x})$ such that $\mathbf{y}'$ is contained in $\mathbf{y}$.

Furthermore, suppose that $\mathbf{y} \in \mathcal{B}_q(n - t' + t'')$ such that there exists a $\mathbf{y}' \in \mathcal{D}_{t'}(\mathbf{x})$ that is contained in $\mathbf{y}$. Then we can apply t'' insertions to $\mathbf{y}'$ in order to obtain $\mathbf{y}$. Hence $\mathbf{y} \in V_{t',t'',0}(\mathbf{x})$. We conclude that for every word $\mathbf{y}$ in $V_{t',t'',0}(\mathbf{x})$ there exists a word $\mathbf{y}' \in \mathcal{D}_{t'}(\mathbf{x})$ that is contained in $\mathbf{y}$ and that every word $\mathbf{y} \in \mathcal{B}_q(n - t' + t'')$ for which there exists a word $\mathbf{y}' \in \mathcal{D}_{t'}(\mathbf{x})$ that is contained in $\mathbf{y}$ is in the set $V_{t',t'',0}(\mathbf{x})$. Thus the set $V_{t',t'',0}(\mathbf{x})$ is exactly the set of words of length $n-t'+t''$ that contain at least one word from the set $\mathcal{D}_{t'}(\mathbf{x})$.

Now we can use this set to determine the size of $V_{t',t'',s}(\mathbf{x})$. We state and prove the following regarding the set of words in $V_{t',t'',s}(\mathbf{x})$.

Lemma 14. *Let $\boldsymbol{x} \in \mathcal{B}_q(n)$ be arbitrary. Let $D_{t',t'',s}(\boldsymbol{x})$ be the set of words that we can obtain by applying $t'' + s$ insertions to the words in the set $\mathcal{D}_{t'+s}(\boldsymbol{x})$. Then the following holds*

$$|V_{t',t'',s}(\boldsymbol{x})| \leq |D_{t',t'',s}(\boldsymbol{x})|$$

Proof. Let $\mathbf{y} \in V_{t',t'',s}(\mathbf{x})$. Then we can obtain $\mathbf{y}$ by applying t' deletions, t'' insertions and $0 \leq s' \leq s$ substitutions. Since substituting a symbol x_i into another symbol $x_j \neq x_i$ is equivalent to deleting the symbol and inserting the symbol x_j at the same position, we can also obtain $\mathbf{y}$ by applying $t' + s'$ deletions and $t'' + s'$ insertions to $\mathbf{x}$. Evidently, this also means we can obtain $\mathbf{y}$ by applying $t' + s$ deletions and $t'' + s$ insertions to $\mathbf{x}$ since we can always delete $s - s'$ symbols and insert the same $s - s'$ symbols to obtain the original word. Therefore we conclude that $\mathbf{y} \in D_{t',t'',s}(\mathbf{x})$ and since $\mathbf{y}$ is arbitrary also

$$|V_{t',t'',s}(\mathbf{x})| \leq |D_{t',t'',s}(\mathbf{x})|$$

□

Now we need to find the value of $|D_{t',t'',s}(\mathbf{x})|$ for different t', t'', s and $\mathbf{x}$. We start by considering the case $n = 2$. Since we are only interested in the case $t', t'', s \geq 1$, applying $t' + s$ deletions results in the empty word and therefore we can obtain any word of length $t'' + s$.

A good approach in finding an expression for $|D_{t',t'',s}(\mathbf{x})|$ seems to be looking for expressions for different types of $\mathbf{x}$. We could for example look for an expression for $\mathbf{x}$ with $r(\mathbf{x}) = r$ and $l(\mathbf{x}) = l$. However this does not seem to lead to any usable results, since the number of different possible values of $|D_{t',t'',s}(\mathbf{x})|$ with a fixed number of runs r, is already 8 for $n = 4, q \geq 3$ and over 20 for $n = 5, q \geq 3$. If there were to be an expression for $|D_{t',t'',s}(\mathbf{x})|$ depending only on t', t'', s and r, we would expect to find only one possible value for all $\mathbf{x}$ with the same parameters. We encountered this problem when we tried to find an expression for $|D_{t',t'',s}(\mathbf{x})|$ by first determining what all the possible values were. One might therefore conclude that it is impossible to find an exact expression for $|D_{t',t'',s}(\mathbf{x})|$ for arbitrary $\mathbf{x}$, however we can find exact expressions for specific type of $\mathbf{x}$. This is exactly what we will do in the next section.

3.6.2 Finding an expression for $|D_{t',t'',s}(\mathbf{x})|$ for specific $\mathbf{x}$

First we consider the case in which $r(\mathbf{x}) = 1$. First of all we note that in this case $V_{t',t'',s}(\mathbf{x}) = D_{t',t'',s}(\mathbf{x})$. Suppose $\mathbf{x} = 000\ldots000$ and $\mathbf{x} \in \mathcal{B}_q(n)$. Then the set $V_{t',t'',s}(\mathbf{x})$ is the set of words in $\mathcal{B}_q(n - t' + t'')$ that contain at least $n - t' - s$ zeros. We know how many words of length n contain exactly k zeros, namely

$$|\{\mathbf{x} \in \mathcal{B}_q(n) : \mathbf{x} \text{ contains exactly } k \text{ zeros}\}| = \binom{n}{k}(q-1)^{n-k}$$

since we have $\binom{n}{k}$ choices for the arrangement of the k zeros and other $n - k$ symbols can be any apart from 0.
Therefore, the total number of words in $\mathcal{B}_q(n)$ which contain at least $n - t' - s$ zeros is equal to

$$|V_{t',t'',s}(000\ldots000)| = \sum_{k=n-t'-s}^{n} \binom{n}{k}(q-1)^{n-k} = \sum_{k=0}^{t'+s} \binom{n}{k}(q-1)^{k} \qquad (3.19)$$

3.6.3 Arbitrary $\mathbf{x}$

Now it is time to consider arbitrary $\mathbf{x} \in \mathcal{B}_q(n)$. We look at the set $\mathcal{D}_{t'+s}(\mathbf{x})$. We count all words in the set $D_{t',t'',s}(\mathbf{x})$ by adding up the number of words in the sets $\mathcal{I}_{t''+s}(\mathbf{y})$ for all $\mathbf{y} \in \mathcal{D}_{t'+s}(\mathbf{x})$. Determining the size of $D_{t',t'',s}(\mathbf{x})$ using this method will result in double counting many words, namely all words that we can obtain by inserting $t'' + s$ symbols to more than one $\mathbf{y} \in \mathcal{D}_{t'+s}(\mathbf{x})$. For every instance of a word that is counted double, we need to subtract our total by one. Now suppose a word can be obtained by inserting $t'' + s$ symbols to three words in $\mathcal{D}_{t'+s}(\mathbf{x})$. In this case we first added this words three times to our set and then removed it three more times, thus we need to add it one

more time. In order to determine the exact size of $D_{t',t'',s}(\mathbf{x})$ we would need to continue this process until we look at the words that can be obtained from all words in $\mathcal{D}_{t'+s}(\mathbf{x})$, however this results in too many calculations and is therefore unfeasible.
For every $\mathbf{y} \in \mathcal{D}_{t'+s}(\mathbf{x})$ we can determine the value of $\mathcal{I}_{t''+s}(\mathbf{y})$ since this expression is known and does not depend on $\mathbf{y}$. For the number of words we count double we will state and prove the following lemma:

Lemma 15. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{B}_q(n)$ such that $d(\boldsymbol{x}, \boldsymbol{y}) = d$. Then the number of words that are both in $\mathcal{I}_{t''}(\boldsymbol{x})$ and $\mathcal{I}_{t''}(\boldsymbol{y})$ say A, is greater or equal to $|\mathcal{I}_{t''-d}(\boldsymbol{z})|$ where $\boldsymbol{z} \in \mathcal{B}_q(n+d)$.*

Proof. Suppose $\mathbf{x}, \mathbf{y} \in \mathcal{B}_q(n)$ such that $d(\mathbf{x}, \mathbf{y}) \leq t''$. Let $\mathbf{z}$ be a word of length $n+d$ that contains both $\mathbf{x}$ and $\mathbf{y}$. This is possible since $d(\mathbf{x}, \mathbf{y}) = d$. Then any word in $\mathcal{I}_{t''-d}(\mathbf{z})$ is both contained in $\mathcal{I}_{t''}(\mathbf{x})$ and $\mathcal{I}_{t''}(\mathbf{y})$ hence

$$A \geq |\mathcal{I}_{t''-d}(\mathbf{z})|$$

□

Now all we need to do is determine the distance between any two words in $\mathcal{D}_{t'}(\mathbf{x})$ and determine the instances where we obtained words by inserting $t'' + s$ symbols to three words in $\mathcal{D}_{t'+s}(\mathbf{x})$ and we can state an upper bound for $|D_{t',t'',s}(\mathbf{x})|$.
We will illustrate how this process works by showing an example.
Suppose we want to determine the size of $\mathcal{D}_{1,1,1}(\mathbf{x})$. Let $\mathbf{x} = 1022 \in \mathcal{B}_3(4)$. Then we have $\mathcal{D}_{t'+s}(\mathbf{x}) = \{10, 12, 02, 22\}$. Now we know that

$$|\mathcal{D}_2 1022| \leq \mathcal{I}_2(10) + \mathcal{I}_2(12) + \mathcal{I}_2(02) + \mathcal{I}_2(22) = 132$$

by Theorem 2. Now we consider the words we counted multiple times. The distances between the words in $\{10, 12, 02, 22\}$ are $1, 2, 2, 1, 1$ and 1. Since in this case $t'' = 1$ we subtract by 4 and obtain

$$|\mathcal{D}_2(1022)| \leq 128$$

while the exact value is equal to 72, hence the bound is not very strong.

Chapter 4

Application to bounds on codes

Now we have found bounds on the size of the set $V_{t',t'',s}(\mathbf{x})$, we can use these to improve the bounds found by Ward Spee. Spee has given three bounds using the size of $V_{t',t'',s}(\mathbf{x})$ which we have given in Chapter 1.

4.1 Bounds using $|V_{t,t,2s}(\mathbf{x})|$

The first and second bounds given, use the size of $V_{t,t,2s}(\mathbf{x})$ and since we have only considered the case regarding $V_{1,1,1}(\mathbf{x})$ and below, we can not say anything useful regarding these bounds. Therefore, more research should be done into the size of $V_{t,t,2s}(\mathbf{x})$ and more general $V_{t,t,2s}(\mathbf{x})$.

4.2 Upper bound using $|V_{1,1,1}(\mathbf{x})|$

Spee has given an upper bound on $M_q(n,t,s)$, where $t \geq t'+t''$, using $|V_{t',t'',s}(\mathbf{x})|$ namely

$$M_q(n,t,s) \leq \frac{q^{n-t'+t''}}{min_{\mathbf{x}\in\mathcal{B}_q(n);r(\mathbf{x})>r}|V_{t',t'',s}(\mathbf{x})|} + q\sum_{i=1}^{r}\binom{n-1}{i-1}(q-1)^{i-1} \tag{4.1}$$

Now by Lemma 12

$$min_{\mathbf{x}\in\mathcal{B}_q(n);r(\mathbf{x})>r}|V_{1,1,1}(\mathbf{x})| \geq \frac{n(n-1)}{2}q^2 - n(n-2)q + \frac{n^2-3n+2}{2} \tag{4.2}$$

we can rewrite Equation 4.1 for $t', t'' = 1, s = 1$ as

$$M_q(n, 2, 1) \leq \frac{2q^n}{n(n-1)q^2 - 2n(n-2)q + n^2 - 3n + 2} + q \sum_{i=1}^{r} \binom{n-1}{i-1} (q-1)^{i-1} \tag{4.3}$$

Since the upper bound holds for all r, it also holds for $r = 0$ hence the second addend is 0 and we obtain the following

$$M_q(n, 2, 1) \leq \frac{2q^n}{n(n-1)q^2 - 2n(n-2)q + n^2 - 3n + 2} \tag{4.4}$$

It might be interesting to study the behaviour of this bound for large values of n and q.

4.3 Behaviour of an upper bound on $M_q(n, 2, 1)$

For large values of $q \gg n$, the lower powers of q do not contribute and we obtain the following bound

$$M_q(n, 2, 1) \leq \frac{2q^n}{n(n-1)q^2 - 2n(n-2)q + n^2 - 3n + 2} \approx \frac{2q^n}{n(n-1)q^2} = \frac{2q^{n-2}}{n(n-1)} \tag{4.5}$$

A more realistic scenario is one with large values for $n \gg q$. In this case we obtain the next bound

$$M_q(n, 2, 1) \leq \frac{2q^n}{(q^2 - 2q + 1)n^2} \tag{4.6}$$

Finally we consider the scenario in which both n and q are sufficiently large. Then the lower terms will disappear and we obtain the following bound

$$M_q(n, 2, 1) \leq \frac{2q^n}{q^2 n^2} = \frac{2q^{n-2}}{n^2} \tag{4.7}$$

Since the total number of words in $\mathcal{B}_q(n)$ is equal to q^n, we deduce that for every codeword, there must be at least $\frac{q^2 n^2}{2}$ words that are not in the code.
Since the numerical results in the thesis of Ward Spee only consider $n = 20$ and $t = 3$, we cannot directly compare these results with exact results from Spee.

Chapter 5

Conclusions & Recommendations

We can conclude that the size of the set $|V_{t',t'',s}(\mathbf{x})|$ is very useful for determining upper and lower bounds on the size of codes that can correct combinations of indel and substitution errors. In Section 4.2 we have used our results from Section 3.4 to obtain insight into an upper bound on $M_q(n, 2, 1)$ stated by Ward Spee namely the following:

$$M_q(n, 2, 1) \leq \frac{2q^n}{n(n-1)q^2 - 2n(n-2)q + n^2 - 3n + 2}$$

where $M_q(n, 2, 1)$ denotes the maximal size of a code that can correct exactly $t' + t'' \leq 2$ indel errors and at most 1 substitution error.

Some of the expressions found and lemmas stated in this thesis might give an insight in the set of words that can be reached from any word $\mathbf{x}$ depending on the structure of $\mathbf{x}$ and the number of substitutions, insertions and deletions. We conclude that we can in general, reach more words from words with a higher amount of runs, than from words with very few runs.

Next we give a couple of suggestions for further research.

- One could further investigate bounds on the size of $D_{t',t'',s}(\mathbf{x})$ by finding better approximations for the number of words that are counted multiple times in Section 3.6.3. Using these bounds an expression for a bound on $V_{t',t'',s}(\mathbf{x})$ can be determined and used in the bounds provided by Ward Spee.
- Another approach could be the use of graphs to find bounds on $|V_{t',t'',s}(\mathbf{x})|$ since it might not be reasonable to use the approach used in this thesis for larger values of t', t'' and s. For example, Spee has used the Caro-Wei theorem to improve lower bounds on $M_q(n, t, s)$.

- Finally, since Lemma 1 uses the average size of $|V_{t,t,2s}(\mathbf{x})|$ it is reasonable to investigate not only the maximum or minimum size of $|V_{t',t'',s}(\mathbf{x})|$, but also the average. In case this is too complicated we suggest trying to find upper and lower bounds on the average. One way this could be achieved is by determining the median value of $|V_{t',t'',s}(\mathbf{x})|$ and by noting that the average cannot lie closer to the minimum or the maximum than to the median.

Bibliography

[1] S. Kosuri and G. M. Church, ”Large-scale de novo DNA synthesis: Technologies and applications,”Nature Methods, vol. 11, no. 5, p. 499, 2014.

[2] T. Xue and F. C. M. Lau, "Notice of Violation of IEEE Publication Principles: Construction of GC-Balanced DNA With Deletion/Insertion/Mutation Error Correction for DNA Storage System," in IEEE Access, vol. 8, pp. 140972-140980, 2020, doi: 10.1109/ACCESS.2020.3012688.

[3] W.J.P. Spee, ”Bounds on the maximum size of deletion, insertion and substitution correcting codes”, https://repository.tudelft.nl/record/uuid:5996e986-167f-4094-9ccd-a17d5f10a702, pp. 15-33, May. 2023.

[4] V. I. Levenshtein, ”Elements of the coding theory (in Russian),”Discrete mathematics and mathematics problems of cybernetics Nauka, Moscow, pp. 207-235, 1974.

[5] H. Mercier, M. Khabbazian, and V. K. Bhargava, ”On the number of subsequences when deleting symbols from a string,” IEEE Transactions on Information Theory, vol. 54, no. 7, pp. 3279-3285, Jun. 2008

[6] F. Sala and L. Dolecek, ”Counting sequences obtained from the synchronization channel,” 2013 IEEE International Symposium on Information Theory, pp 2925-2929, Jul. 2013.

[7] M. Abu-Sini and E. Yaakobi, ”On Levenshtein’s reconstruction problem under insertions, deletions, and substitutions,” IEEE Transactions on Information Theory, vol. 67, no. 11, pp 7132-7158, Sep. 2021.

Appendices

Appendix A

Proof of Theorem 2

Proof. In order to determine the size of $\mathcal{I}_{t''}(\mathbf{x})$, it might seem reasonable to count all possible words we can create by inserting t'' symbols into a q-ary word $\mathbf{x}$ of length n. However, this way we will be double counting words and therefore overestimating the size of $\mathcal{I}_{t''}(\mathbf{x})$. If we take the binary word 01 for example, and insert one symbol we have that

$$\mathcal{I}_1(01) = \{001, 101, 011, 010\}$$

while we have $2 \cdot 3 = 6$ different ways of inserting one symbol, namely two choices for the symbol we insert (1 or 0) and three choices for the place of insertion (first, second or third position). We have counted the words 001 and 011 double, since there are two ways of creating the same word. To obtain 001 we can either insert a 0 at the first or second position and to create 011 we can either insert a 1 at the second or third position. In order to avoid double counting words, we disregard the possibility of an insertion being equal to the symbol directly before it. For example, if we consider the binary word $\mathbf{x} = 01$ and we apply one insertion, we only consider $2 - 1 = 1$ possible values for the insertion at the second or third position. If the insertion occurs at the second position, it cannot be equal to 0, hence it must be 1 and if the insertion happens at the third position it cannot be 1 and must therefore be 0. This way we can still obtain all words in $\mathcal{I}_1(01)$ and we correctly count $2 \cdot (2-1) + 2 = 4$ ways of inserting one symbol with the restriction we imposed.

If there occur more insertions, we make a distinction between the symbols of the original word and the newly inserted symbols. We define $\mathbf{x}$ as

$$\mathbf{x} = x_1 x_2 \dots x_n$$

and we denote all insertions by $i_1, i_2, \dots, i_{t''} \in \{0, 1, \dots, q-1\}$. For every insertion i_j, $1 \leq j \leq t''$ there either exists a $k \in \{1, 2, \dots, n-1\}$ such that i_j happens after x_k and before x_{k+1}, or i_j happens before x_1 or after x_n. For every insertion i_j that occurs after x_1 we impose the following restriction: if i_j

occurs after x_k and before x_{k+1} we have that $i_j \neq x_k$ and if i_j occurs after x_n we have $x_j \neq x_n$. If we take $n = 5$ and $t'' = 3$ for example, then the following may occur

$$\mathbf{x} = x_1x_2x_3x_4x_5 \xrightarrow[\text{insertions occur}]{} i_3x_1x_2x_3i_1x_4x_5i_2$$

where $i_3 \in \{0, 1, \ldots, q-1\}$, $i_1 \in \{0, 1, \ldots, q-1\}\backslash x_3$ and $i_2 \in \{0, 1, \ldots, q-1\}\backslash x_5$.

We want to make sure we do not count words double or miss any possible words using this restriction.
i) Let $\mathbf{x} \in \mathcal{B}_q(n)$ be arbitrary. Suppose there occur t'' insertions on $\mathbf{x}$ and we obtain a word $\mathbf{y} \in \mathcal{B}_q(n)$ that cannot be obtained by imposing the restriction on the insertions $i_1, \ldots, i_{t''}$. We call the insertions $y_1, \ldots, y_{t''}$. Now we will show that this word can in fact be obtained by t'' insertions following the restriction from $\mathbf{x}$. Suppose x_l is the largest l such that there is at least one y_j in between x_l and x_{l+1} (or if $l = n$ after x_n) such that $y_j = x_l$. Now we let the last of these, say $y_m = x_l$, be redefined to be x_l and let x_l become an insertion. Now we note that there are no longer any insertions between x_l and x_{l+1} that are equal to x_l and therefore they satisfy the restriction. We continue to apply this process to the largest x_i for which there is an insertion y_j between x_i and x_{i+1} with $y_j = x_i$ until there are no such x_i left. When this is the case, the restriction has been met without changing the word. Therefore we conclude that we can indeed obtain every word by applying the restriction to all insertions.

In the example below we show how this process works in practice with the insertions in red

$$\mathbf{x} = x_1x_2x_3x_4x_5 = 01210 \xrightarrow[\text{insertions occur}]{} 012\textcolor{red}{02}10\textcolor{red}{0} \to 012\textcolor{red}{02}1\textcolor{red}{0}0 \to 01\textcolor{red}{20}21\textcolor{red}{0}0$$

Now all we need to do is counting all words we can obtain using the restriction on the insertions without counting words double. Let $\mathbf{x} = x_1x_2 \ldots x_n \in \mathcal{B}_q(n)$ be arbitrary. Suppose there are exactly $i \leq t''$ insertions which occur before x_1 that are equal to x_1. Then there are $t'' - i$ insertions that have exactly $q - 1$ possible values that can occur in any of the $n + t''$ places of the word we obtain when applying t'' insertions. Since the order of insertions do not change the word, the total number of words we can obtain with exactly i insertions before and equal to x_1 is $\binom{n+t''}{t''-i}(q-1)^{t''-i}$. Since the number of insertions that occur before x_1 and are equal to x_1 ranges from 0 to t'' we need to sum all different cases to obtain the total number of words

$$|\mathcal{I}_{t''}(\mathbf{x})| = \sum_{i=0}^{t''} \binom{n+t''}{t''-i} (q-1)^{t''-i} = \sum_{i=0}^{t''} \binom{n+t''}{i} (q-1)^{i}$$

which finishes the proof. □

Appendix B

Derivation of expressions for $|V_{t',t'',s}(\mathbf{x})|$

The reader may ask themself how we found the expression in Lemma 9 and 12 and the expressions in Table 3.4. This has been done by using the values found by a computer search (as shown in various tables) to find patterns depending on n and q. The expression corresponding to Lemma 9 was found using values from Table 3.2 where $r(\mathbf{x}) = n$. First we find an expression using only the variable q for constant n. When we do this for various n, we can determine a pattern in the variable n to find an expression for n and q. This is the same strategy we have used for the expressions in Lemma 12 and in Table 3.4. We will now show the process for the expressions used in Lemma 9 and 12 and the expressions in Table 3.4.

B.1 Derivation of Lemma 9

In order to derive an expression for $|V_{0,1,1}(\mathbf{x})|$ with $r(\mathbf{x}) = n$ we look at the values for different q and $n = 2$ first. We find the values $8, 23, 46$ and 77 for $n = 2$ and $q \in \{2, 3, 4, 5\}$ using a computer. We note that the differences between consecutive values are $15, 23$ and 31, hence they increase by 8 whenever q increases by 1. Thus we can express $|V_{0,1,1}(\mathbf{x})|$ as the following sum when $n = 2, r(\mathbf{x}) = n$:

$$\begin{aligned}|V_{0,1,1}(\mathbf{x})| &= (1+7) + 15 + 23 + \ldots + 8(q-1) - 1\\ &= \frac{(q-1)(7 + 8(q-1) - 1)}{2} + 1\\ &= 4q^2 - 5q + 2\end{aligned}$$

We do the same for $n \in \{3, 4, 5\}$ in order to find out how the coefficients change and therefore what the expression dependent on both n and q looks like.
For $n = 3$ we find the values $14, 45, 94$ and 161 for $q \in \{2, 3, 4, 5\}$. The differences between consecutive values are $31, 49$ and 67 and increase by 18 whenever q increases by 1. Thus we can derive the following:

$$\begin{aligned}|V_{0,1,1}(\mathbf{x})| &= (1 + 13) + 31 + 49 + \ldots + 18(q-1) - 5\\ &= \frac{(q-1)(13 + 18(q-1) - 5)}{2} + 1\\ &= 9q^2 - 14q + 6\end{aligned}$$

For $n = 4$ we find the values $22, 75, 160$ and 277 for $q \in \{2, 3, 4, 5\}$. The differences between consecutive values are $53, 85$ and 117 hence they increase by 32 whenever q increases by 1. Thus for $n = 4$ we derive:

$$\begin{aligned}|V_{0,1,1}(\mathbf{x})| &= (1 + 21) + 53 + 85 + \ldots + 32(q-1) - 11\\ &= \frac{(q-1)(21 + 32(q-1) - 11)}{2} + 1\\ &= 16q^2 - 27q + 12\end{aligned}$$

And finally for $n = 5$ we found the values $32, 113, 244$ and 425. The differences between consecutive values are $81, 131$ and 181 hence they increase by 50 whenever q increases by 1. Thus for $n = 5$ we derive:

$$\begin{aligned}|V_{0,1,1}(\mathbf{x})| &= (1 + 31) + 81 + 131 + \ldots + 50(q-1) - 19\\ &= \frac{(q-1)(31 + 50(q-1) - 19)}{2} + 1\\ &= 25q^2 - 44q + 20\end{aligned}$$

We assume that the coefficients can be expressed by a polynomial function $p(n)$ of n. For the first coefficient we have $p_1(2) = 4, p_1(3) = 9, p_1(4) = 16$ and $p_1(5) = 25$ hence $p_1(n) = n^2$. For the second coefficient we have $p_2(2) = -5, p_2(3) = -14, p_2(4) = -27$ and $p_2(5) = -44$. Thus

$$\begin{aligned}p_2(n) &= -(5 + 9 + 13 + 17 + \ldots + 4(n-1) + 1)\\ &= -\frac{(n-1)(5 + 4(n-1) + 1)}{2}\\ &= -(2n^2 - n - 1)\end{aligned}$$

And for the third coefficient we have $p_3(2) = 2, p_3(3) = 6, p_3(4) = 12$ and $p_3(5) = 20$ hence we derive that $p_3(n) = n^2 - n = n(n-1)$. Substituting these polynomials into the expression yields an expression for general $q, n \geq 2$:

$$|V_{0,1,1}(\mathbf{x})| = n^2q^2 - (2n^2 - n - 1)q + n(n-1)$$

Which is exactly the expression shown in Lemma 9.

Since we have used the same process to derive the expressions in Lemma 12 and Table 3.4 we omit these derivations.

Appendix C

Python code

In order to get an insight into the behaviour of the size of $V_{t',t'',s}(\mathbf{x})$, I have used Python to find these values for small n and q. The Python code I have used in this thesis is partly copied from the master thesis of Ward Spee. The rest of the code was created by myself to find more values, or look for all words that match certain properties. For example, I have created code that returns all words $\mathbf{x}$ for which $|V_{0,1,1}(\mathbf{x})| = a$ for some value a. In the code below

```python
### Creating the set V_{t',t'',s}(x) for x in B_q(n). x is a string
                                            .
import math

def insert(x, k, i):
  #insert symbol k on position i into x where 0 <= i <= n
  return x[0:max(0,i)] + str(k) + x[i:len(x)]
def substitute(x, k, i):
  #substitute symbol k on position i into x where 1 <= i <= n
  return x[0:max(0,i-1)] + str(k) + x[i:len(x)]
def delete(x,i):
  #delete position i from x
  return x[0:max(0,i-1)] + x[i:len(x)]

def insertion_set(x,q):
  #returns the set of words that can be reached
  #from x by 1 insertion
  n = len(x)
  symbols = range(0,q)
  x_ins = []
  for i in range(0, n+1):
    for k in symbols:
      y = insert(x,k,i)
      x_ins.append(y)
  return set(x_ins)

def substitution_set(x,q):
  #returns the set of words that can be reached
  #from x by at most 1 substitution
  n = len(x)
  symbols = range(0,q)
  x_sub = []
  for i in range(1, n+1):
    for k in symbols:
      y = substitute(x, k, i)
      x_sub.append(y)
  return set(x_sub)

def deletion_set(x):
  # returns the set of words that can be reached from x by 1
                                  deletion
  n = len(x)
  x_del = []
  for i in range(1, n+1):
    y = delete(x, i)
    x_del.append(y)
  return set(x_del)

def V(x, q, td, ti, ss):
  # returns the set of words that can be reached from x by exactly
  # td deletions, ti insertions and at most ss substitutions.
  # The 'set' function ensures that no words are counted double.
  V_list = [x]
  while td > 0:
    y_list = []
    for x in V_list:
      y_list += list(deletion_set(x))
    V_list = list(set(y_list))
    ss -= 1

  while ti > 0:
    y_list = []
    for x in V_list:
      y_list += list(insertion_set(x,q))
    V_list = list(set(y_list))
    ti -= 1
  return V_list
```

This was all the code used from the master thesis of Ward Spee. [3] The following code is my own and it uses the functions defined by Spee.

```
def NV(x,q,td,ti,ss):
  # returns the number of words in V(x,td,ti,ss)
  l = len(V(x,q,td,ti,ss))
  return l

def Setq(n,q):
  # returns a list of all q-ary words of length n
  a = 1
  m = list(range(0,q))
  while a < n:
    for i in m:
      l = insertion_set(str(i),q)
      p = list(l)
      m = m+p
    a = a +1
  mylist = list(dict.fromkeys(m))
  b = (q**n-q)/(q-1)
  b = int(b)
  newlist = mylist[b:]
  return newlist

def NS(n,q,td,ti,ss):
  # returns a list of all values of |V(x,td,ti,ss)|
  # for all q-ary words x of length n
  s = Setq(n,q)
  for i in range(0,len(s)):
    s[i] = NV(s[i],q,td,ti,ss)
  return s

def MAX(n,q,td,ti,ss):
  # returns the maximum value of |V(x,td,ti,ss)|
  # for all q-ary words of length n
  m = max(NS(n,q,td,ti,ss))
  return m

def NRUN(y):
  # returns the number of runs of a word x
  r = 1
  for i in range(0,len(y)-1):
    if y[i] == y[i+1]:
      r = r
    else r = r+1
  return r

def MV(n,q,td,ti,ss):
  # returns the number of runs of the words
  # for which |V(x,td,ti,ss)| is maximal
  s = Setq(n,q)
  r = Setq(n,q)
  for i in range(0,len(s)):
    s[i] = NV(s[i],q,td,ti,ss)
  m = MAX(n,q,td,ti,ss)
  for j in range(0,len(s)):
    if s[j] == m:
      print(NRUN(str(r[j])))

def RSET(n,q,r):
  # returns the set of words for which
  # the number of runs is equal to r
  s = Setq(n,q)
  l = []
  for i in s:
    if NRUN(i) == r:
      l.append(i)
  return l

def RNS(n,q,r,td,ti,ss):
  # returns all possible values of |V(x,td,ti,ss)|
  # for q-ary words x with r runs
  # this is the function used to determine
  $ the values shown in the tables
  s = RSET(n,q,r)
  for i in range(0,len(s)):
    s[i] = NV(s[i],q,td,ti,ss)
  return list(set(s))

def SFW(n,q,r,w,td,ti,ss):
  # returns the set of q-ary words x for which
  # |V(x,td,ti,ss)| is equal to w
  s = RSET(n,q,r)
  l = []
  for i in range(0,len(s)):
    if NV(s[i],q,td,ti,ss) == w:
      l.append(s[i])
  return l

def MRNS(n,q,r,td,ti,ss):
  # returns the maximum value of
  # |V(x,td,ti,ss)| for all q-ary words x with
  # r runs
  m = max(RNS(n,q,r,td,ti,ss))
  return m

def com(x,y,q,td,ti,ss):
  # returns the difference between |V(x,td,ti,ss)|
  # and |V(y,td,ti,ss)| where x and y are q-ary words
  x_l = V(x,q,td,ti,ss)
  y_l = V(y,q,td,ti,ss)
  main = len(x_l)-len(y_l)
  return main

def Sp(x,q,td,ti,ss):
  # returns the set D(t',t'',s,x)
  l = V(x,q,td+ss,0,0)
  s = []
  for i in range(0,len(l)):
    s = s+(V(l[i],q,0,ti+ss,0))
  s = list(set(s))
  return s

def LSP(x,q,td,ti,ss):
  # returns the size of D(t',t'',s,x)
  r = len(Sp(x,q,td,ti,ss))
  return r

def SD(n,q,td,ti,ss):
  # returns all possible values for D(t',t'',s,x) for given n and q
  l = []
  for x in Setq(n,q):
    l.append(LSP(x,q,td,ti,ss))
  return list(set(l))

def FSD(w,n,q,td,ti,ss):
  # returns all words x for which D(t',t'',s,x) = w
  s = Setq(n,q)
  l = []
  for i in range(0,len(s)):
    if LSP(s[i],q,td,ti,ss) == w:
      l.append(s[i])
  return l
```

```
def    NSP(x,q,td,ti,ss):
  # returns all words that are not in D(t',t'',s,x)
  s = Setq(len(x)+ti-td,q_
  m = Sp(x,q,td,ti,ss)
  main = list(set(s).difference(m))
  return main

def DIF(x,y):
  # returns the difference between x and y
  d = 0
  for i in range(0,len(x)):
    if x[i] == y[i]:
      d = d
    else:
      d = d+1
  return d
```