Analyzing and Visualizing the Effect of Noise on Compressed Textual Data

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Computer Science & Engineering

By:
Mr. Sudesh Kumar
(80732023)

Under the supervision of:
Mrs. Shalini Batra
Sr. Lecturer, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004
JUNE 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Analyzing and Visualizing the Effect of Noise on Compressed Textual Data"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Computer Science submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of Mrs. Shalini Batra and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(Sudesh Kumar)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Mrs. Shalini Batra)
Lecturer (SS),
Computer Science and Engineering Department,
Thapar University,Patiala.

Countersigned by:

(Dr. SEEMA BAWA)
Professor & Head,
Computer Science & Engineering Department,
Thapar University,
Patiala.

(Dr. R.K.SHARMA)
Dean (Academic Affairs),
Thapar University,
Patiala.

Acknowledgement

The real spirit of achieving a goal is through the way of excellence and austerous discipline. I would have never succeeded in completing my task without the cooperation, encouragement and help provided to me by various personalities.

First of all, I render my gratitude to the ALMIGHTY who bestowed self-confidence, ability and strength in me to complete this work. Without his grace this would never come to be today's reality.

With deep sense of gratitude I express my sincere thanks to my esteemed and worthy Supervisor **Mrs. Shalini Batra (Sr. Lecturer),** Department of Computer Science and Engineering for her valuable guidance in carrying out this work under her effective supervision, encouragement, enlightenment and cooperation. Most of the novel ideas and solutions found in this thesis are the result of our numerous stimulating discussions. Her feedback and editorial comments were also invaluable for writing of this thesis.

I shall be failing in my duties if I do not express my deep sense of gratitude towards **Dr. Seema Bawa,** Professor and Head of Computer Science and Engineering Department who has been a constant source of inspiration for me throughout this work.
I am also thankful to all the staff members of the Department for their full cooperation and help.

Most importantly, I would like to thank my parents and the Almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope.

Place: TU, Patiala

Date: **(Sudesh Kumar)**

Abstract

Clustering organizes data into groups such that each group contains similar type of data. The majority of document (contain similar data) clustering algorithms require a vector representation for each document. Consequently, the memory required during clustering can be extremely high when clustering hundreds of thousands of documents. In recent year there is lot of work going on clustering by using compression. Compression reduces the run-time memory requirements for clustering and also does not degrade the final cluster quality.

Compression is one of the techniques for better utilization of storage devices, resulting in saving of storage space. This thesis addresses the compression by using the technique called Normalized Compression Distance (NCD). The Normalized Compression Distance is based on algorithmic complexity developed by Kolmogorov, called Normalized Information Distance. Normalized Compression Distance can be used to cluster objects of any kind, such as music, texts, or gene sequences (microarray classification). The NCD between two binary strings is defined in terms of compressed sizes of the two strings and of their concatenation; it is designed to be an effective approximation of the non computable but universal Kolmogorov distance between two strings.

This correspondence studies the influence of noise on the normalized compression distance, a measure based on the use of compressors to compute the degree of similarity of two files. This influence is approximated by a first order differential equation which gives rise to a complex effect, which explains the fact that the NCD may give values greater than 1. Finally, the influence of noise on the clustering of files of different types is explored and the final analysis is that the NCD performs well even in the presence of quite high noise levels.

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

With the growth of multimedia and internet, compression has become the thrust area in the field of computers. Popularity of multimedia has a led to integration of various types of computer data. Multimedia combines many of data types like text, graphics, still images, animation, audio and video data. Compression requires regeneration for the original data from the compressed from in exactly the same form. Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth.

1.1 Data Compression

Data compression is the art or science of representing information in compact form. We create these compact representations by identifying and using structures that exit in the data. Data can be characters in the text file, number that are samples of speech or image, or sequences of number that are generated by other processes [1].

Compression reduces the quantity of data sent to storage, often doubling the effective capacity of the media (depending on the nature of the data). If the data is later restored/recovered, the system automatically decompresses the data and restores it to its original state. Most computer users realize that there are freely available programs that can compress text files to about one quarter of their original size. Data compression reduces the number of bits used to store or transmit information. Information is store in the form of files. Data in these files consists of text consisting of manuscripts, program memos and other readable files or binary data including database files, executable files and many other type of information [2]. Nearly all data contains some redundant information. The objective of data compression is to transform redundant data in to a form that is smaller but still retains the same information. The less well known aspect of data compression is that combining two or more files together to create a larger single conglomerate archive file prior to compression often yields better compression in aggregate. This has been used to great advantage in widely popular programs like tar or

pkzip, combining archival and compression functionality. Only in recent years have scientists begun to appreciate the fact that compression ratios signify a great deal of important statistical information.

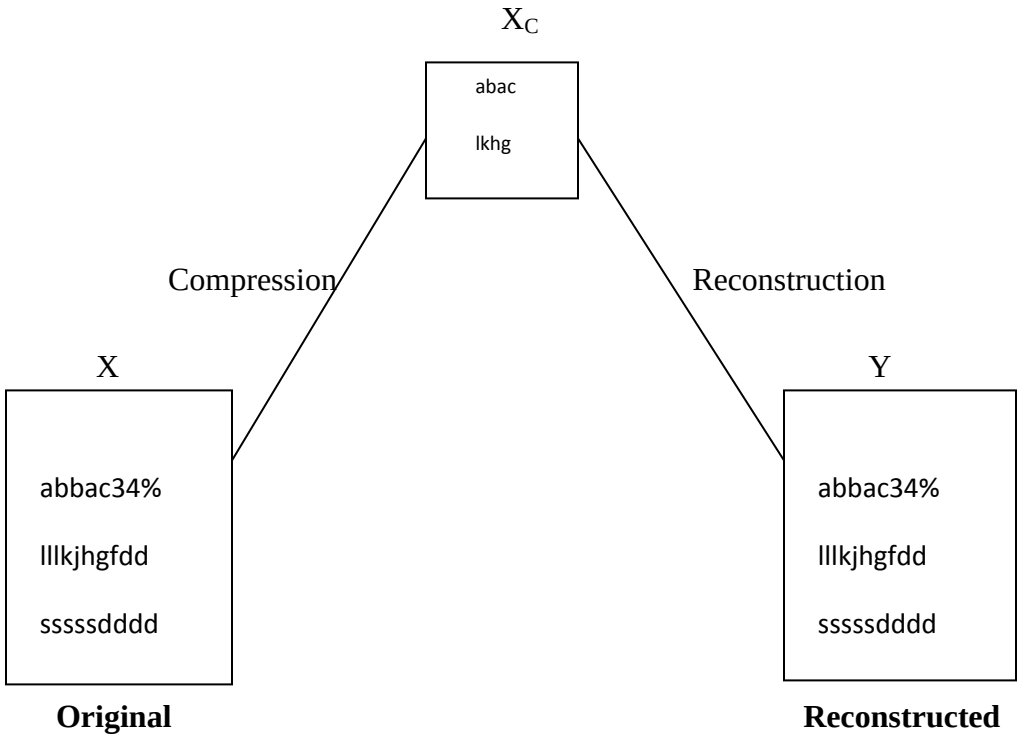

Figure 1.1: Compression and Reconstruction

The key concept is to realize that if two files are very similar in their contents, then they will compress much better when combined together prior to compression, as compared to the sum of the size of each separately compressed file. If two files have little or nothing in common, then combining them together would not yield any benefit over compressing each file separately. By using even the simplest string-matching type compression made in the 1970's it is possible to construct evolutionary trees for animals fully automatically using files containing their mitochondrial gene sequence. The main advantage of this approach is its robustness in the face of strange or erroneous data. Another key advantage is the simplicity and ease of use. Uncompressed graphics, audio and video data require considerable storage capacity and also data transfer of uncompressed data over digital networks requires very high bandwidth. The exploding use of word-wide web on the

internet has increased demand for speedily transfer of data. Compressing files before transmitting them saves telecommunications bandwidth.

1.1.1 Lossless versus lossy compression

- **Lossless Compression:-**Lossless compression techniques, as their name implies, involve no loss of information. If data have been losslessy compressed, the original data can be recovered exactly from the compressed data. Lossless compression is generally used for application that cannot tolerate any difference between the original and reconstructed data [2]. Text compression is an important area for lossless compression. It is very important that the reconstruction is identical to the text original, as very small differences can result in statements with very different meanings. Consider the sentences “Do not send money” and “Do now send money”. A similar argument holds for computer files and for certain types of data such as bank records. Lossless compression algorithms usually exploit statistical redundancy in such a way as to represent the sender's data more concisely without error. Lossless compression is possible because most real-world data has statistical redundancy. For example, in English text, the letter 'e' is much more common than the letter 'z', and the probability that the letter 'q' will be followed by the letter 'z' is very small. Examples of the lossless compression are Run-length, Huffman, LZ77, LZ78, LZW, etc.
- **Lossy Compression:-**Another kind of compression, called lossy data compression or perceptual coding, is possible if some loss of fidelity is acceptable. In lossy compression, the reconstruction differs from the original data [2]. The difference between the original and the reconstruction is often called the distortion. Lossy techniques are generally used for the compression of data that originate at analog signals, such as speech and video. Examples of the lossy compression include JPEG, MPEG, etc.

Lossless compression schemes are reversible so that the original data can be reconstructed, while lossy schemes accept some loss of data in order to achieve higher compression. An example of lossless vs. lossy compression is the following string:

26.888888888

This string can be compressed as:

26.[9]8

Interpreted as, "twenty five point 9 eights", the original string is perfectly recreated, just written in a smaller form. In a lossy system, using

27

Instead, the original data is lost, at the benefit of a smaller file size.

1.1.2 Finite and Infinite Object

In the domain of mathematical objects discussing here, there are two broad categories: finite and infinite. Finite objects are those whose extent is bounded. Infinite objects are those that are “larger” than any given precise bound. For example, if we perform 100 flips of a fair coin in a sequence and retain the results in order, the full record will be easily written upon a single sheet of A4 size paper, or even a business card. Thus, the sequence is finite. But if we instead talk about the list of all prime numbers greater than 5, then the sequence written literally is infinite in extent. There are far too many to write on any given size of paper no matter how big. It is possible, however, to write a computer program that could, in principle, generate every prime number, no matter how large, eventually, given unlimited time and memory. It is important to realize that some objects are infinite in their totality, but can be finite in a potential effective sense by the fact that every finite but a priori unbounded part of them can be obtained from a finite computer program.

1.1.3 Strings and Languages

A bit, or binary digit, is just a single piece of information representing a choice between one of two alternatives, either 0 or 1. A character is a symbol representing an atomic unit of written language that cannot be meaningfully subdivided into smaller parts. An alphabet is a set of symbols used in writing a given language. A string is an ordered list

(normally written sequentially) of 0 or more symbols drawn from a common alphabet. For a given alphabet, different languages deem different strings permissible. In English, 26 letters are used, but also the space and some punctuation should be included for convenience, thus increasing the size of the alphabet. A language (in the formal sense) is a set of permissible strings made from a given alphabet. In computer files, the underlying base is 256 because there are 256 different states possible in each indivisible atomic unit of storage space, the byte. A bit is a symbol from a 2-symbol, or binary, alphabet. In this thesis, there is not usually any need for an alphabet of more than two characters, so the notational convention is to restrict attention to the binary alphabet in the absence of countervailing remarks. A byte is equivalent to 8 bits, so the 256-symbol alphabet is central to real computers. For theoretical purposes however, we can dispense with the complexities of large alphabets by realizing that we can encode large alphabets into smaller ones; indeed, this is how a byte can be encoded as 8 bits. Usually numbers are encoded as a sequence of characters in a fixed radix format at the most basic level, and the space required to encode a number in this format can be calculated with the help of the logarithm function. The logarithm function is always used to determine a coding length for a given number to be encoded, given a probability or integer range. Similarly, it is safe to assume that all logarithmic values are taken on base 2 so that we may interpret the results in bits. $\sum$ is used to represent the alphabet used. We usually work with the binary alphabet, so in that case $\sum = \{0,1\}$. $\sum^*$ is used to represent the space of all possible strings including the empty string. This notation may be a bit unfamiliar at first, but is very convenient and is related to the well-known concept of regular expressions. Regular expressions are a concise way of representing formal languages as sets of strings over an alphabet. The curly braces represent a set (to be used as the alphabet in this case) and the * symbol refers to the closure of the set; by closure we mean that the symbol may be repeated 0, 1, 2, 3, 5, or any number of times. By definition, $\{0,1\}^* = S_{n\geq 0}\{0,1\}^n$. It is important to realize that successive symbols may or may not be the same. It can be seen that the number of possible binary strings is infinite, yet any individual string in this class must itself be finite. For a string x, $|x|$ is used to represent the length, measured in symbols, of that string.

1.1.4 Turing Machines

The Turing machine is an abstract mathematical representation of the idea of a computer. A Turing machine is the simplest form of a computer. It generalizes and simplifies all the specific types of deterministic computing machines into one regularized form. They were described in 1936 by Alan Turing. They are not intended as a practical computing technology, but a thought experiment about the limits of mechanical computation. A Turing machine is defined by a set of rules which describe its behavior. It receives as its input a string of symbols, which may be thought of as a "program", and it outputs the result of running that program, which amounts to transforming the input using the given set of rules. Just as there are universal computer languages, there are also universal Turing machines. A Turing Machine is universal if it can simulate any other Turing Machine. When such a universal Turing machine receives as input a pair <x,y>, where *x* is a formal specification of another Turing machine Tx, it outputs the same result as one would get if one would input the string *y* to the Turing machine Tx [3]. Just as any universal programming language can be used to emulate any other one, any universal Turing machine can be used to emulate any other one.

1.2 Normalized Information Distance

Normalized information distance is a parameter-free similarity measure based on compression, used in pattern recognition, data mining, phylogeny, clustering, and classification. In pattern recognition, learning, and data mining one obtains information from objects containing information. This involves an objective definition of the information in a single object, the information to go from one object to another object in a pair of objects, the information to go from one object to any other object in a multiple of objects, and the shared information between objects, the normalized information distance is a universal distance measure for objects of all kinds. It is based on Kolmogorov complexity [4]. Kolmogorov complexity measures the absolute information content of individual objects for the purpose of data mining, especially for clustering. The information distance between two sequences x and y can be defined as the length of a

shortest binary program that computes x given y, and also computes y given x. However, such a distance does not take the length of the sequence into account. Normalized information distance D(x, y) for measuring the similarity relations between sequences. Such a distance takes values in [0, 1] for all sequence pairs and should satisfy the following relations:

1. D(x, y) = 0 iff x = y

2. D(x, y) = D(y, x) (symmetry)

3. D(x, y) ≤ D(x, z) + D(z, y) (triangle inequality)

The normalized information distance is a universal similarity metric that memorizes every computable metric. For two strings x and y the NID is defined as follows [6]:

$$D(x,y) = \frac{\max \{K(x|y),K(x|y)\}}{\max \{K(x),K(y)\}} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(1.2.1)$$

Where K(x|y) is the conditional Kolmogorov complexity (CKC) of the string x given the string y; K(x) is equivalent to K(x|ε), being ε the empty string. Both CKC and NID are incomputable.

1.2.1 Kolmogorov Complexity

Kolmogorov complexity formalizes the concept of simplicity or complexity. Intuitively, a string is simple if it can be described in a few words, like the string of one million ones, and is complex if there is no such short description, like for a random string whose shortest description is specifying it bit by bit. Typically one is only interested in descriptions or codes that are effective in the sense that decoders are algorithms on some computer. Consider the following n-sequence

0100011011000001010011100101110111

Although it may appear random, it is the enumeration of all binary strings. A natural way to compress it is to represent it by the procedure that generates it: enumerate all strings in binary, and stop at the n-th bit. The compression achieved in bits is

$$|\text{compression length}| \leq 2 \log n + O(1)$$

More generally, with the universal Turing machine, encoding of data of a computer program that generates it, can be done easily. The length of the smallest program that produces the bit string x is called the Kolmogorov complexity of x [5].

The Kolmogorov complexity of a sequence or string is actually a measure of randomness or, when inverted, the regularity of the patterns in that string. Turing machines are used to measure that regularity with the length of the shortest program for that Turing machine that has the sequence as output. Such a program is called a description of the sequence. This description is relative to the Turing machine that has the description as input.

Definition: The Kolmogorov complexity of a string x is the length of the smallest program that outputs x, relative to some model of computation [6]. That is,

$$K_T(x) = \min_p\{\ell(p) : T(p)=x\}$$

or

$$K_T(x) = \infty \text{ if no such p exists.}$$

Where T is Turing machine, ℓ (p) is the length of p measured in bits. This complexity measure depends on T, and one may ask whether there exists a Turing machine which leads to the shortest codes among all Turing machines for all x . T is considered to be a Turing machine that takes as input program a binary string of zeros and ones, so the program is an element of the set {0,1}*, which is the set of all finite binary strings. Binary strings are used because everything that can be decoded, like *e.g.*, scientific data, can be coded by a string of zeros and ones. The length of the program, ℓ(p), is the number of zeros and ones. So the definition takes as the complexity of a string x the length of the program p that consists of the least number of bits and that will generate x when given to T. If no such program exists then the complexity is considered to be infinite.

When x is a finite string then there is always a program that will describe it. Let's take a program that will merely print the number literally. This program will be larger than the string. However, if x is infinite and no finite program exists, and then x is uncomputable by definition.

1.2.2 Conditional Kolmogorov Complexity

There is another form of *K* which is a bit harder to understand but still important for our discussion, called as conditional Kolmogorov Complexity and written as

$$K(z|y).$$

The notation is confusing to some because the function takes two arguments. Its definition requires a slight enhancement of the earlier model of a Turing machine. If *y* gives a lot of information about *z* then $K(z|y) << K(z)$, but if *z* and *y* are completely unrelated, then $K(z \mid y) \approx K(z)$. For example, if $z = y$, then *y* provides a maximal amount of information about *z*. K(z) is equivalent to K(z|ε), being ε the empty string [6].

The rest of the thesis is organized in the following order:

Chapter 2 reviews the topics in detail under the heading of literature survey.

Chapter 3 states the problem considered in the thesis.

Chapter 4 gives a detailed introduction about normalized compression distance (NCD), and talk about some properties. It introduces the idea of a mathematical distance function and discusses the notion of a similarity metric and gives small introduction to open-source software package CompLearn, used for visualization of results.

Chapter 5 explains the experiment performed and the results evaluated. The graphs generated between textual data and clustering experiments performed with the CompLearn Toolkit in the presence of noise are depicted in this section.

Chapter 6 presents the conclusion of this thesis and suggestions for future work.

Thesis concludes with references.

CHAPTER 2

Literature Review

In this section, work done in the area of clustering by compression is reviewed and focus has been made on clustering by using Normalized Compression Distance. The exploding use of World-Wide Web on the Internet has increased demand for speedy transfer of data, graphics and images. Compressing file before transmitting them saves telecommunication bandwidth. The growth of data compression, started around 1950.Compression algorithms is very useful for a variety of application. Originally, their main use was what their name suggests, reducing the size of computer files without losing any information. It was interesting to see that they were equally useful for all type such as text, bit maps, many type of images, sound, scientific experimental data, computer code (source and object code), combined data (text and images, for instance), and so on [2].

Normalized Compression Distance (NCD) metrics represent a feature-free similarity measure for arbitrary binary sequences. As such, they are less subject to issues inherent in many methods of sequence similarity detection (such as local alignment or position-weight matrix motif discovery tools). NCD metrics have been successfully used to construct phylogenetic relationships between viruses and to perform protein classification. Genomic sequences are usually compared using evolutionary distance, a procedure that implies the alignment of the sequences. Alignment of long sequences is a long procedure and the obtained dissimilarity results are not a metric. Recently the normalized compression distance (NCD) was introduced as a method to calculate the distance between two generic digital objects, and it seems a suitable way to compare genomic strings [8].

In 2005, a new method was presented for clustering based on compression. All data are created equal but some data are more alike than others. The method doesn't use subject-specific features or background knowledge, and works as follows: First, universal similarity distance is determined, the normalized compression distance or NCD, computed from the lengths of compressed data files (singly and in pair wise concatenation). Second, hierarchical clustering method is applied [7].

Conventional similarity metrics used to sustain diversity in evolving populations are not well suited to sequential decision tasks. Genotypes and phenotypic structure are poor predictors of how solutions will actually behave in the environment. Faustino J. Gomez [25] proposed measuring similarity directly on the behavioral trajectories of evolving candidate policies using a universal similarity measure based on algorithmic information theory: normalized compression distance (NCD).

Is it possible to model the phenomenon of two images being perceived by humans as visually similar, either mathematically or computationally? Such a model would be useful in many applications such as web search engines and digital watermarking. Nicholas Tran, [10] reviewed the development of modern theories of image similarity, starting with an empirical study in 1936 involving human subjects evaluating sets of variations of a standard image, carefully chosen to reveal the principles behind the observed results and ending with the recent information-theoretic formulation of mathematical distance functions which have been shown to be universal in the sense that they minorize all computable functions measuring similarity between two objects. Nicholas Tran use an information-theoretic distortion measure called the Normalized Compression Distance (NCD), first proposed by M. Li et al. [6], to determine whether two rectangular gray-scale images are visually distinguishable to a human observer. Image distinguish ability is a fundamental constraint on operations carried out by all players in an image watermarking system [10].

In recent years, microarray technology has become an important tool in biomedical research, and in particular, cancer. A common use involves finding genes that have clinical importance for the cancer type under study. Another very important application is classification of microarray samples for clinical diagnosis. The normalized compression distance can be applied to gene expression data analysis. Typically, microarray-based classification involves using a feature subset selection method in connection with a specific distance metric. The performance is dependent on the selection of the methods. With proposed approach there is no need for feature subset or distance metric selection and all the data can be used directly with the universal similarity metric [11].

In 2008 work was done in area of identified Cover Songs using NCD. The goal was to determine whether two songs are different versions of same composition or not. That is not an easy task because the covers are usually intentionally different from the original songs. Successful cover songs identification yields important information on the similarity between musical pieces. They used a common approach that extract the harmonic features (i.e. chords) and compare them, first chord extraction: using a Hidden Markov Model to estimate chord sequence from chromagram and second is comparison: comparing chord sequences using different sequence matching technique. The distance for every pair of songs was calculated by NCD and written in to a distance matrix and then tested on two standard compression algorithms: zip and bzip2 [12].

In present days lot of research is going on compression techniques. CompLearn, a open source visualization tool is a suite of simple-to-use utilities that can be used to apply compression techniques to the process of discovering and learning patterns. The compression-based approach used is powerful because it can mine patterns in completely different domains. It can classify musical styles of pieces of music and identify unknown composers. It can identify the language of bodies of text. It can discover the relationships between species of life and even the origin of new unknown viruses such as SARS [13].

During literature review it was analyzed although a lot of research is going on clustering by compression, very few papers have actually focused on compression of text data and more importantly what will be effect of adding noise on compressed textual data. So, we decided to focus on this area in detail.

CHAPTER 3

Problem Statement

3.1 Problem Definition

The Normalized Compression Distance is an approach that is used for clustering. It is a measure based on the use of compressors to compute the degree of similarity of two files. The massive use of Internet has enormously increased the traffic of files across potentially noisy channels that can change their original contents. NCD is a similarity measure based on the use of compressors, so noise could make NCD get wrong results: a clustering application using NCD as a measure of distance would classify as dissimilar two similar files corrupted by noise. This influence is approximated by a first order differential equation which gives rise to a complex effect, which explains the fact that the NCD may give values greater than 1. Finally, the influence of noise on the clustering of files of different types is explored.

3.2 Methodology

The step-by-step methodology to be followed for how the NCD changes when applied to two text files, one of which is distorted by an increasing noise ratio.

(a) Perform clustering on simple text data.
(b) Noise is applied with certain probability to individual words in files.
(c) Now performed clustering in the presence of noise and visualize the data with the CompLearn Toolkit and analyze how the NCD changes when applied to two files, one of which is distorted by an increasing noise ratio.

CHAPTER 4

Normalized Compression Distance

Normalized Compression Distance is a similarity measure based on a data compressor. Normalized Information Distance is simply the instantiation of NCD using the theoretical (and uncomputable) Kolmogorov compressor. We first review the definition of a metric and the in Section 4.2 we explain precisely what is meant by universality in the case of NID. We discuss compressor axioms in Section 4.3, and properties of NCD in Section 4.4.

4.1 Similarity Metric

In mathematics, different distances arise in all sorts of contexts, and one usually requires these to be a "metric". A precise formal meaning to the loose distance notion of "degree of similarity" used in the pattern recognition literature is given here.

Metric: Let W be a nonempty set and R^+ be the set of nonnegative real numbers. A metric on W is a function $D : W \times W \rightarrow R^+$ satisfying the metric (in)equalities [14]:

- $D(x, y) = 0$ iff $x = y$,
- $D(x, y) = D(y, x)$ (symmetry), and
- $D(x, y) \leq D(x, z) + D(z, y)$ (triangle inequality).

The value $D(x, y)$ is called the distance between $x, y \in W$. A familiar example of a metric is the Euclidean metric, the everyday distance $e(a,b)$ between two objects a,b expressed in, say, meters. Clearly, this distance satisfies the properties

$e(a,a) = 0$, $e(a,b) = e(b,a)$,

and

$e(a,b) \leq e(a, c)+e(c,b)$

(for instance, a = Amsterdam, b = Brussels, and c = Chicago.) We are interested in "similarity metrics". For example, if the objects are classical music pieces then the

function $D(a,b) = 0$ if a and b are by the same composer and $D(a,b) = 1$ otherwise, is a similarity metric. This metric captures only one similarity aspect (feature) of music pieces, presumably an important one because it subsumes a conglomerate of more elementary features.

4.2 Normal Compressor

Axioms determine a large family of compressors that include both most (if not all) real-world compressors and ensure the desired properties of the NCD [6].

4.2.1. Definition: A compressor C is normal if it satisfies, up to an additive $O(\log n)$ term, with n the maximal binary length of an element of W involved in the (in)equality concerned, the following:

1. *Idempotency*: $C(xx) = C(x)$, and $C(\lambda) = 0$, whereλ □ is the empty string.
2. *Monotonicity*: $C(xy) \geq C(x)$.
3. *Symmetry*: $C(xy) = C(yx)$.
4. *Distributivity*: $C(xy)+C(z) \leq C(xz)+C(yz)$.

Idempotency: A reasonable compressor will see exact repetitions and obey idempotency up to the required precision. It will also compress the empty string to the empty string.

Monotonicity: A real compressor must have the monotonicity property, at least up to the required precision. The property is evident for stream-based compressors, and only slightly less evident for block-coding compressors.

Symmetry: This is related to the stream-based property: the initial file x may have regularities to which the compressor adapts; after crossing the border to y it must unlearn those regularities and adapt to the ones of y. This process may cause some imprecision in symmetry that vanishes asymptotically with the length of x, y. Stream-based compressors of the Lempel-Ziv family, like gzip and pkzip, and the predictive PPM family, like PPMZ, are possibly not precisely symmetric. A compressor must be poor indeed (and will certainly not be used to any extent) if it doesn't satisfy symmetry up to the required precision. Apart from stream-based, the other major family of compressors is block-

coding based, like bzip2. They essentially analyze the full input block by considering all rotations in obtaining the compressed version. It is to a great extent symmetrical, and real experiments show no departure from symmetry.

Distributivity: The distributivity property is not immediately intuitive. In Kolmogorov complexity theory the stronger distributivity property

$$C(xyz)+C(z) \leq C(xz)+C(yz) \quad \text{.................................} \quad (4.2.1)$$

holds (with $K = C$). However, to prove the desired properties of NCD below, only the weaker distributivity property

$$C(xy)+C(z) \leq C(xz)+C(yz) \quad \text{..................................} \quad (4.2.2)$$

above is required, also for the boundary case were $C = K$. In practice, real-world compressors appear to satisfy this weaker distributivity property up to the required precision[15].

4.2.2 Definition: Define

$$C(y|x) = C(xy) - C(x) \quad \text{...} \quad (4.2.3)$$

This number $C(y|x)$ of bits of information in y, relative to x, can be viewed as the excess number of bits in the compressed version of xy compared to the compressed version of x, and is called the amount of conditional compressed information. In the definition of compressor the decompression algorithm is not included (unlike the case of Kolmorogov complexity, where the decompressing algorithmic given by definition), but it is easy to construct one: Given the compressed version of x in $C(x)$ bits, the compressor can be run on all candidate strings z—for example, in length-increasing lexicographical order, until we find the compressed string $z0 = x$. Since this string decompresses to x we have found

$x = z0$. Given the compressed version of xy in $C(xy)$ bits, this process is repeated using strings xz until the string $xz1$ is found, of which the compressed version equals the compressed version of xy. Since the former compressed version decompresses to xy, $y = z1$is found. By the unique decompression property it has been found that $C(y|x)$ is the extra number of bits required to describe y apart from describing x. It is intuitively

acceptable that the conditional compressed information $C(x|y)$ satisfies the triangle inequality

$$C(x|y) \leq C(x|z)+C(z|y).$$

4.2.3. Lemma. A normal compressor satisfies additionally subadditivity*:*

$$C(xy) \leq C(x)+C(y).$$

PROOF. Consider the special case of distributivity with z the empty word so that $xz = x$, $yz = y$, and $C(z) = 0$.

4.3 Background in Kolmogorov complexity

Technically, the Kolmogorov complexity of x given y is the length of the shortest binary program, for the reference universal prefix Turing machine, that on input y outputs x; it is denoted as $K(x|y)$. The Kolmogorov complexity of x is the length of the shortest binary program with no input that outputs x; it is denoted as $K(x) = K(x|\lambda)$ where λ denotes the empty input. Essentially, the Kolmogorov complexity of a file is the length of the ultimate compressed version of the file. The information distance $E(x, y)$ is defined as the length of the shortest binary program for the reference universal prefix.

The information distance E(x, y) was introduced, defined as the length of the shortest binary program for the reference universal prefix Turing machine that, with input x computes y, and with input y computes x. It was shown there that, up to an additive logarithmic term, $E(x, y) = \max\{K(x|y),K(y|x)\}$. It was shown also that $E(x, y)$ is a metric, up to negligible violations of the metric inequalties. The normalized version of $E(x, y)$, called the normalized information distance, is defined as [6]

$$\text{NID(x,y)} = \frac{.\max\{K(x|y),K(y|x)\}}{\max\{K(x),K(y)\}} \quad \text{.....(4.3.1)}$$

It is also a metric. Thus, if two files (of whatever type) are similar (that is, close) according to the particular feature described by a particular normalized admissible distance (not necessarily metric), then they are also similar (that is, close) in the sense of

the normalized information metric. This justifies calling the latter the similarity metric. It is well known fact that different pairs of objects may have different dominating features, yet every such dominant similarity is detected by the NID. However, this metric is based on the notion of Kolmogorov complexity. Unfortunately, the Kolmogorov complexity is non-computable in the Turing sense. Approximation of the denominator of (4.3.1) by a given compressor C is straightforward: it is $\max\{C(x),C(y)\}$. The numerator is more tricky. It can be rewritten as

$$\max\{K(x, y)-K(x),K(x, y)-K(y)\} \quad \text{.....(4.3.2)}$$

within logarithmic additive precision, by the additive property of Kolmogorov complexity. The term $K(x, y)$ represents the length of the shortest program for the pair (x, y). In compression practice it is easier to deal with the concatenation xy or yx. Again, within logarithmic precision $K(x, y) = K(xy) = K(yx)$. Following a suggestion by Steven de Rooij, one can approximate (4.3.2) best by $\min\{C(xy),C(yx)\}-\min\{C(x),C(y)\}$. Here, and in the later experiments using the CompLearn Toolkit, we simply use $C(xy)$ rather than $\min\{C(xy),C(yx)\}$. The result of approximating the NID using a real compressor C is called the normalized compression distance (NCD). The theory as developed for the Kolmogorov-complexity based NID, may not hold for the (possibly poorly) approximating NCD [16] .

4.4 Normalized Compression Distance

The normalized compression distance is an approach that is used for clustering. It's based on algorithmic complexity developed by Kolmogorov, called Normalized Information Distance. The normalized information distance is a universal similarity metric that memorizes every computable metric. For two strings x and y the NID is defined as follows:

$$D(m,n) = \frac{\max\{K(m|n),K(n|m)\}}{\max\{K(m),K(n)\}} \quad \text{.....(4.4.1)}$$

Where K(x|y) is the conditional Kolmogorov complexity (CKC) of the string x given the string y; K(x) is equivalent to K(x|ε), being ε the empty string. Both CKC and NID are incomputable [6].

A natural measure of similarity assumes that two objects x and y are similar if the basic blocks of x are in y and vice versa. If this happens object x can be described by making reference to the blocks belonging to y, thus the description of x will be very simple using the description of y. This is what a compressor does to code the concatenated xy sequence: a search for information shared by both sequences in order to reduce the redundancy of the whole sequence. If the result is small, it means that a lot of information contained in x can be used to code y, following the similarity conditions described in the previous paragraph. This was formalized by Cilibrasi and Vit´anyi [4], giving rise to the concept of Normalized Compression Distance (NCD), which is based on the use of compressors to provide a measure of the similarity between the objects. This distance may then be used to cluster those objects. This idea is very powerful, because it can be applied in the same way to all kind of objects, such as music, texts or gene sequences. There is no need to use specific features of the objects to cluster. The only thing needed to compute the distance from one object *x* to another object *y*, is to measure the ability of *x* to turn the description of *y* simple and vice versa. The normalized version of the admissible distance $E_C(x, y)$, the compressor *C* based approximation of the normalized information distance (4.3.1), is called the normalized compression distance or NCD [18].

$$NCD(x,y) = \frac{C(xy) - \min\{C(x),C(y)\}}{\text{Max}\{C(x),(y)\}} \quad \text{.....(4.4.2)}$$

- C(x) - compressed size of x.
- C(xy) – compressed size of the concatenation of x and y.

C(x) denotes the length of the text x compressed using some compression algorithm which asymptotically reaches the entropy of x, when the length of x tends to infinity.

$$0 <= NCD(x, y) <= 1.$$

When NCD(x, y) = 0, then x and y are similar, if NCD(x,y) = 1, they are dissimilar.

Figure 4.1: X, Y and Z are three documents without compression

Figure 4.2: X, Y and Z are documents after compression

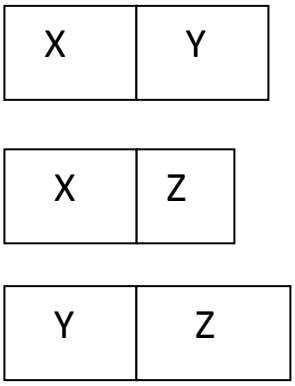

Figure 4.3: X, Y and Z are three compressed concatenated documents

This NCD is the main concept of this work. It is the real-world version of the ideal notion of normalized information distance NID in (4.3.1).

4.4.1. Remark : In practice, the NCD is a non-negative number $0 \leq r \leq 1+e$ representing how different the two files are. Smaller numbers represent more similar files. The e in the upper bound is due to imperfections in our compression techniques, but for most standard compression algorithms one is unlikely to see an e above 0.1

There is a natural interpretation to NCD(x, y): If, say, $C(y) \geq C(x)$ then we can rewrite

$$\text{NCD} = \frac{C(xy)-C(x)}{C(y)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.4.3)$$

That is, the distance NCD(x, y) between x and y is the improvement due to compressing y using x as previously compressed "data base," and compressing y from scratch, expressed as the ratio between the bit-wise length of the two compressed versions. Relative to the reference compressor we can define the information in x about y as $C(y)-C(y|x)$. Then, using (4.2.3),

$$\text{NCD}(x, y) = 1 - \frac{C(y) - C(y|x)}{C(y)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4.4.4)$$

That is, the NCD between x and y is 1 minus the ratio of the information x about y and the Information in y.

4.5.2. Theorem: If the compressor is normal, then the NCD is satisfying the metric (in)equalities, that is, a similarity metric[7].

Proof: If the compressor is normal, then NCD is a normalized admissible distance. It remains to show it satisfies the three metric (in)equalities [14].

(1) By idempotency for $y \neq x$.

(2) NCD(x, y) = NCD(y, x). The NCD is unchanged by interchanging x and y in (4.4.2).

(3) The difficult property is the triangle inequality. Without loss of generality we assume

$C(x) \leq C(y) \leq C(z)$. Since the NCD is symmetrical, there are only three triangle inequalities that can be expressed by NCD(x, y), NCD(x, z), NCD(y, z). We verify them in turn:

(a) NCD(x, y) $\leq$ NCD(x, z)+NCD(z, y): By distributivity, the compressor itself satisfies $C(xy)+C(z) \leq C(xz)+C(zy)$. Subtracting $C(x)$ from both sides and rewriting,

$$C(xy)-C(x) \leq C(xz)-C(x)+C(zy)-C(z).$$

Dividing by $C(y)$ on both sides we find

$$\frac{C(xy)-C(x)}{C(y)} \leq \frac{C(xz)-C(x)+C(zy)-C(z)}{C(y)}$$

The left-hand side is ≤ 1.

(i) Assume the right-hand side is ≤ 1. Setting $C(z) = C(y)+D$, and adding D to both the numerator and denominator of the right-hand side, it can only increase and draw closer to 1. Therefore,

$$\frac{C(xy)-C(x)}{C(y)} \leq \frac{C(xz)-C(x)+C(zy)-C(z)+\Delta}{C(y)+\Delta}$$

$$= \frac{C(zx)-C(x)}{C(z)} + \frac{C(zy)-C(y)}{C(z)} \quad \text{.....(4.4.5)}$$

which was what we had to prove.

(ii) Assume the right-hand side is > 1. We proceed like in the previous case, and add D to both numerator and denominator. Although now the right-hand side decreases, it must still be greater than 1, and therefore the right-hand side remains at least as large as the left-hand side.

(b) NCD(x, z) ≤ NCD(x, y) + NCD(y, z): By distributivity we have $C(xz) + C(y) \leq C(xy)+C(yz)$. Subtracting $C(x)$ from both sides, rearranging, and dividing both sides by $C(z)$ we obtain

$$\frac{C(xz)-C(x)}{C(z)} \leq \frac{C(xy)-C(x)}{C(z)} \leq \frac{C(yz)-C(y)}{C(z)}$$

The right-hand side doesn't decrease when $C(y)$ is substituted for the denominator $C(z)$ of the first term, since $C(y) \leq C(z)$. Therefore, the inequality stays valid under this substitution, which was what we had to prove.

(c) NCD(y, z) ≤ NCD(y, x) + NCD(x, z): By distributivity $C(yz)+C(x) \leq C(yx)+C(xz)$. Subtracting $C(y)$ from both sides, using symmetry, rearranging, and dividing both sides by $C(z)$ we obtain

$$\frac{C(yz)-C(y)}{C(z)} \leq \frac{C(xy)-C(x)}{C(z)} \leq \frac{C(yz)-C(y)}{C(z)}$$

The right-hand side doesn't decrease when we substitute $C(y)$ for the denominator $C(z)$ of the first term, since $C(y) \leq C(z)$. Therefore, the inequality stays valid under this substitution, which was what we had to prove [19].

We see C(x) denotes the length of the text x compressed using some compression algorithm which asymptotically reaches the entropy of x, when the length of x tends to infinity

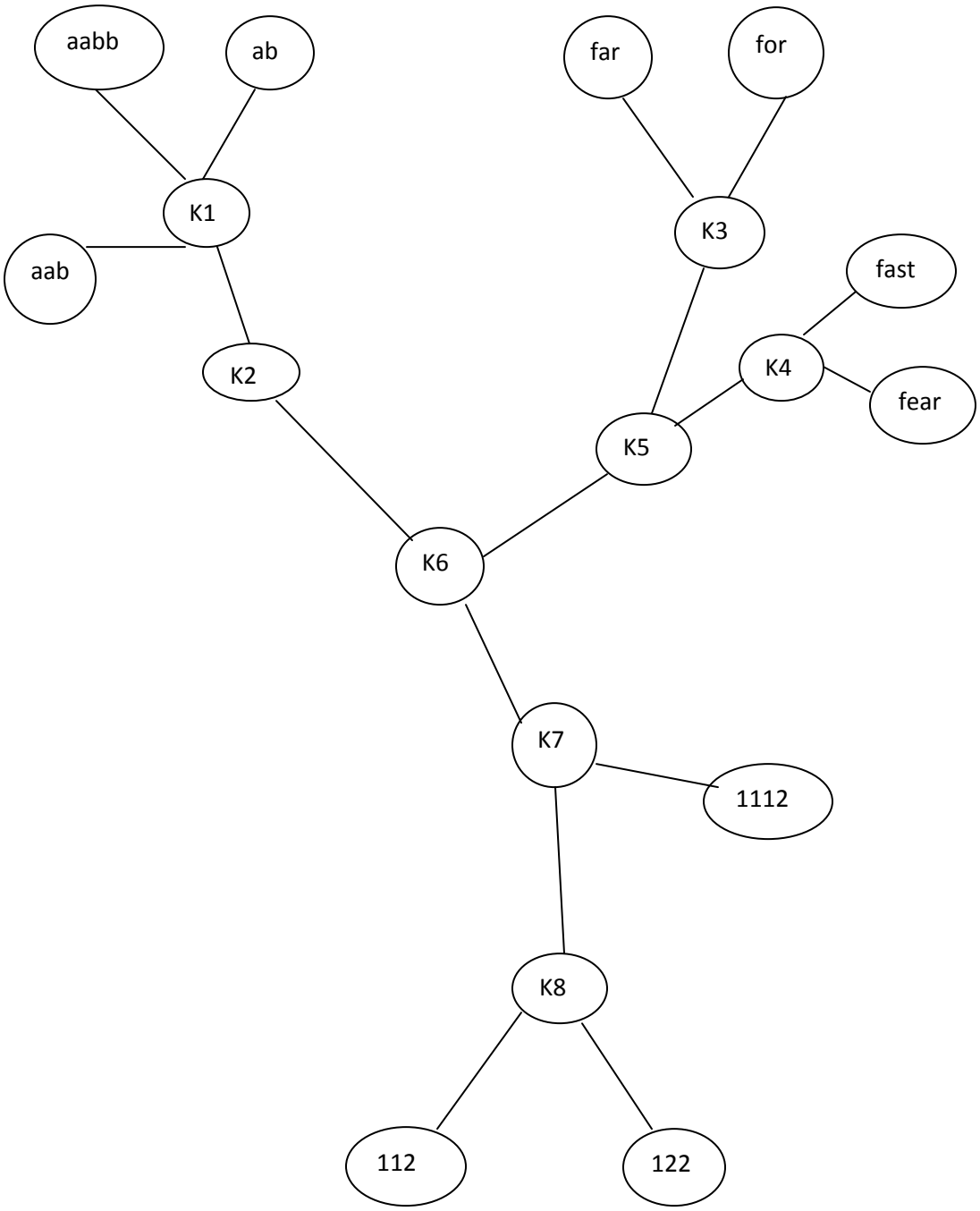

Figure 4.4: A heterogeneous dataset containing same type of data

4.5 Huffman coding Compression Technique

Text compression is a technique used on computer systems to compress data, such that the data occupies less storage space, or can be transmitted in shorter time. The compressor has to compact the data in such a way that a decompressed can restore the compacted data to its original form, without any distortion. When we compute normalization compressor distance then required compressed data so we use here Hoffman coding technique for data compression.

In computer science and information theory**,** Huffman coding is an entropy encoding algorithm used for lossless data compression. The term refers to the use of a variable-length code table for encoding a source symbol (such as a character in a file) where the variable-length code table has been derived in a particular way based on the estimated probability of occurrence for each possible value of the source symbol. It was developed by David A. Huffman, and published in the 1952 paper "A Method for the Construction of Minimum-Redundancy Codes"[20]. Huffman coding uses a specific method for choosing the representation for each symbol, resulting in a prefix code(sometimes called "prefix-free codes") (that is, the bit string representing some particular symbol is never a prefix of the bit string representing any other symbol) that expresses the most common characters using shorter strings of bits than are used for less common source symbols. Huffman was able to design the most efficient compression method of this type: no other mapping of individual source symbols to unique strings of bits will produce a smaller average output size when the actual symbol frequencies agree with those used to create the code. A method was later found to do this in linear time if input probabilities (also known as weights) are sorted. For a set of symbols with a uniform probability distribution and a number of members which is a power of two, Huffman coding is equivalent to simple binary block encoding, e.g., ASCII coding [21].

Suppose we have a simple alphabet with the following *n* characters and the assigned frequency counts, and we use a fixed length code requiring bits for each character.

Character	Code	Freq	Bits
A	000	10	30
E	001	15	45
I	010	12	36
S	011	3	9
T	100	4	12
Sp	101	13	39
Nl	110	1	3
Total			174

Table 4.1: Fixed Length Coding Scheme Ignoring Frequency

The algorithm works by constructing a binary tree from the bottom up, using the frequency counts of the symbols to repeatedly merge sub trees together. Intuitively, the symbols that are more frequent should occur higher in the tree and the symbols that are less frequent should be lower in the tree

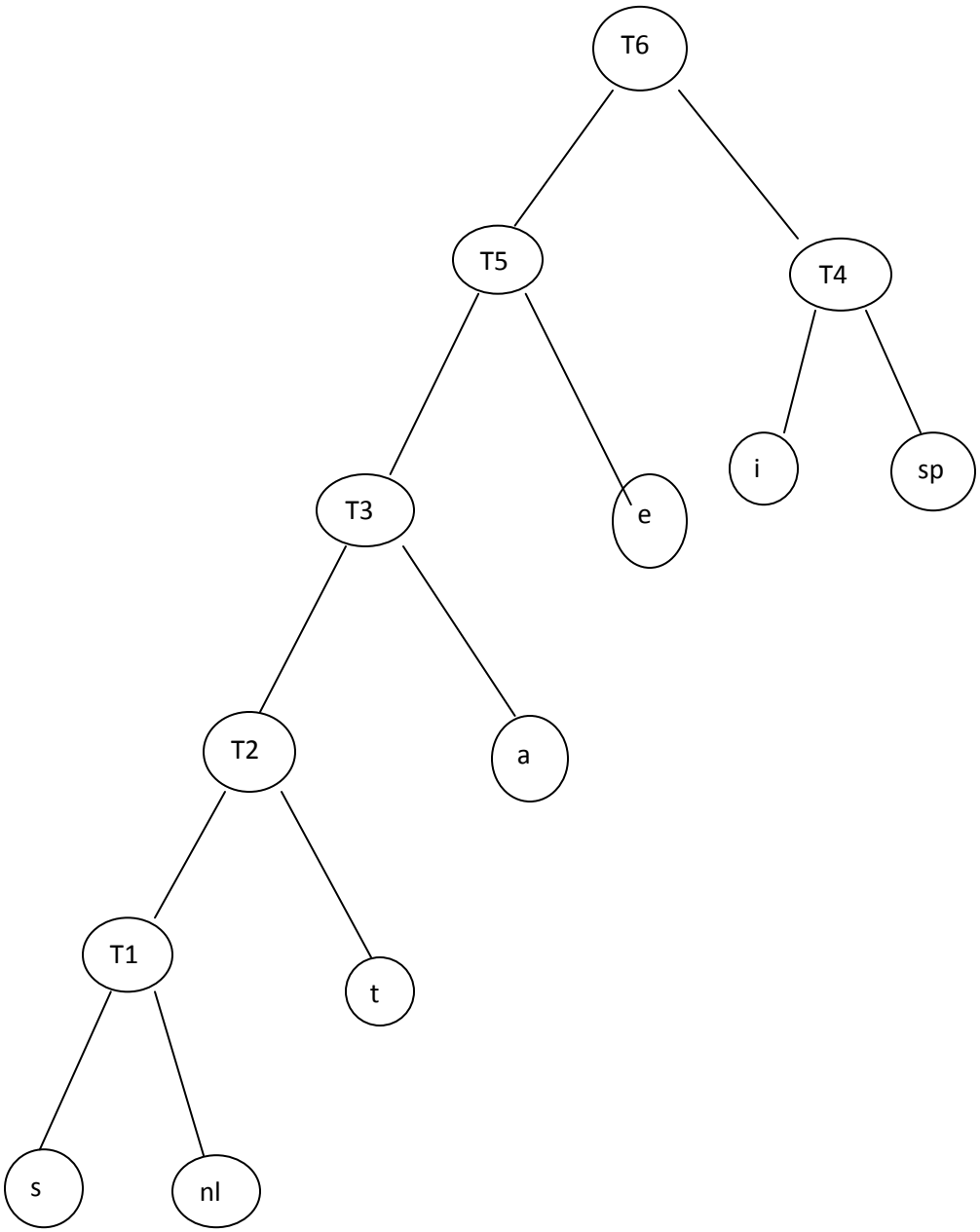

Figure 4.5: Binary tree using Huffman's algorithm

Character	Code	Freq	Bits
A	001	10	30
E	01	15	30
I	10	12	24
S	00000	3	15
T	0001	4	16
Sp	11	13	26
Nl	00001	1	5
Total			146

Table: 4.2 Optimal Variable Length Code

So that the before compression file size 174 bits but after using Huffman technique it is reduced to 146 bits. We use this technique in NCD and find difference between two files.

4.6 CompLearn

A generic open-source software package (CompLearn) for building tree structured representations from normalized compression distance (NCD). CompLearn was developed at the National Research Institute CWI in Amsterdam. It was created to support the research paper Algorithmic Clustering of Music. CompLearn is a software system built to support compression-based learning in a wide variety of applications. It provides this support in the form of a library written in highly portable ANSI C that runs in most modern computer environments with minimal confusion. It also supplies a small suite of simple, composable command-line utilities as simple applications that use this library. Together with other commonly used machine-learning tools such as LibSVM and GraphViz, CompLearn forms an attractive offering in machine-learning frameworks and

toolkits. It is designed to be extensible in a variety of ways including modular dynamic-linking plugging and a language-neutral SOAP interface to supply instant access to core functionality in every major language [13]. There are no domain-specific parameters to set and only a handful of general settings. It runs under Windows and UNIX and requires very small amounts of disk space to install and run. you can drag-and-drop files and folders to generate a best-fitting unrooted binary tree in 3D. Now automatically decompresses files compressed by bzip2 and gzip.

CHAPTER 5

Testing & Results

Let us consider a file of size ‘a’ which is compressed by a given compressor into another file of size ‘b’. If we add noise to the original file and compress it, as the amount of noise increases, the compressor will be able to reduce less and less the file size, until it will be unable to reduce it at all, once the contents of the file become fully random. Therefore the size of the compressed file will start at ‘b’, when no noise is added, and will increase steadily to ‘a’, which will be reached when the whole initial file has been replaced by random noise [22].

At a given point in this procedure, if we add Δx noise to the file (i.e. change randomly the values of Δx bytes in the file), we may expect the size increase (the compression loss) proportional to the amount of the file which has not yet been replaced by noise. Therefore, the evolution of the compressed size y will be defined by $\Delta y = \gamma(a - y)\Delta x$. When $\Delta x \rightarrow 0$, this equation becomes the first-order differential equation $dy/dx = \gamma(a - y)$. The solution of this equation, taking into account the indicated initial conditions, is

$$y = a - (a - b)e^{-\gamma x} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5.1)$$

where x is the amount of noise added and the value of y (the size of the compressed file) is b for $x = 0$; a for $x \rightarrow \infty$.

Consider the definition of NCD and assume that we want to study the variation of the distance NCD(p, q) between a fixed file p, and another file q which is being contaminated by growing amounts of noise. Without loss of generality, we may assume that $C(p) \leq C(q)$. Therefore, the above distance becomes

$$NCD(p,q) = \frac{C(pq) - C(p)}{C(q)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5.2)$$

We have seen that, as the amount of noise x introduced in file q grows by

$$C(q) = a - (a - b)e^{-\gamma x}.$$

It is easy to see that C(pq) will evolve in a similar way, although with different constants, because the noise introduced in the second part of the file not only destroys redundancies in that section of the file, but also prevents possible cross-compressions with the first part, which does not receive noise. So, $C(pq) = c\text{-}d.e^{-\varphi x}$. Finally, C(p) is a constant. Replacing these values, under certain conditions the NCD formula can be approximated by

$$NCD(p,q) = \alpha+\beta e^{-\gamma x}-\delta e^{-\varphi x} \quad \text{.....(5.3)}$$

where the values of the constants depend on the actual files p and q compressed; x is again the amount of noise added. If we compute the average distortion introduced by a noise level ℓ, $0 \leq \ell \leq 1$, we come to a similar equation, since NCD(p; q) is a constant:

$$\Delta_\ell(p,q) \equiv E[NCD\ (p,q+n_\ell \bmod r) - NCD(p,q)]$$

$$\approx \alpha + \beta e^{-\gamma \ell} - \delta e^{-\varphi \ell} \quad \text{..... (5.4)}$$

where n_ℓ is a random string whose length is equal to the length of q and whose ith character is nonzero with probability 1; r is the size of the file type range (i.e. 4 for DNA, 93 for ASCII, and 256 for the rest) [22]. Modulo operation is performed to maintain each value in its proper range. For each files p and q we estimate $\Delta_\ell(x,y)$ for $\ell \in \{0.05,0.10,0.15.....\ldots 1\}$.An interesting result is that, for some data types, the average distortion increases until it reaches a maximum at some $l < 1$, and then decays steadily converging toward a smaller value when the level of noise is increased; texts follow this behavior (Fig.5 1). This phenomenon explains the already mentioned fact that the NCD can sometimes reach values greater than 1 when comparing files that share very little information: the region in which $\Delta_\ell(x; y)$ has a value greater than $\Delta_1(x; y)$.

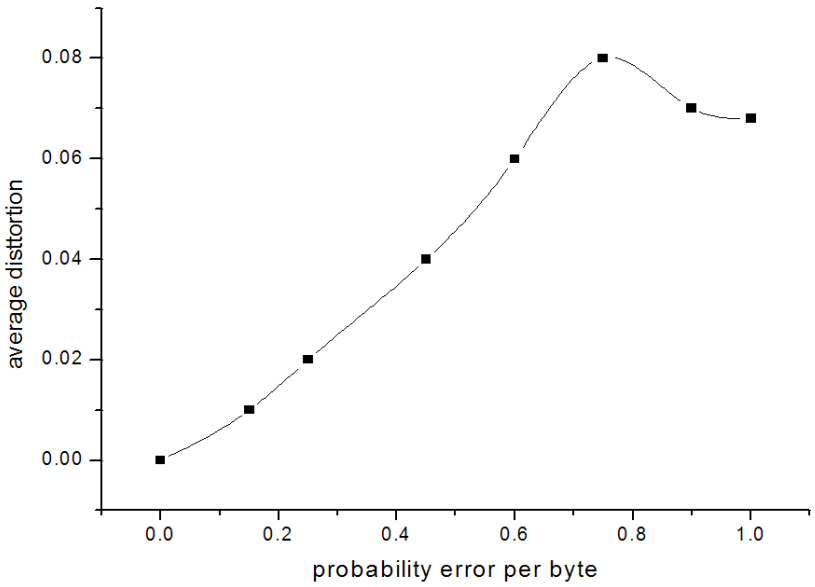

Figure 5.1: An example of the typical decay behavior of the average distortion with text file.

P	the raven
Q	the art of the life
NCD without noise	1.021
NCD + 15% noise	1.032
NCD + 25% noise	1.051
NCD + 50% noise	1.062
NCD + 65% noise	1.08
NCD + 75% noise	1.10
NCD + 90% noise	1.09
NCD + 100% noise	1.084

Table 5.1: Different NCD values on different noise level between the P and Q

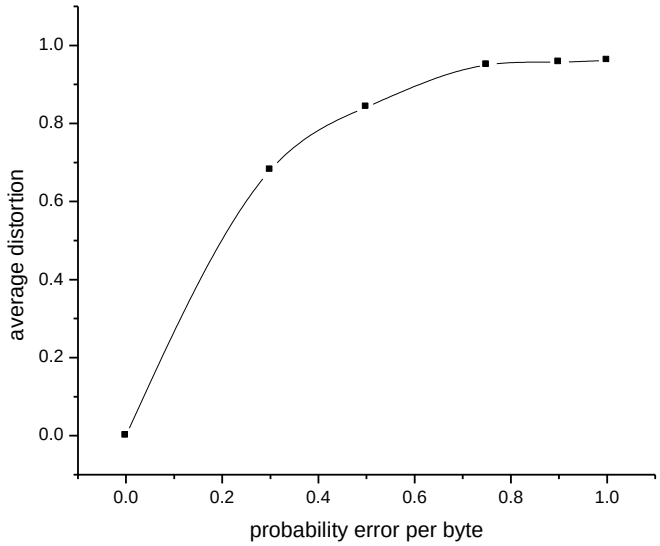

Figure 5.2: An example of the typical non decay behavior of the average distortion with text file.

R	vision is good
S	vision is poor
NCD + without noise	.086
NCD + 25% noise	.771
NCD + 50% noise	.932
NCD + 75%	1.04
NCD + 90%	1.05
NCD + 100%	1.055

Table 5.2: Different NCD values on different noise level between R and S

Finally, show two real clustering experiments performed with the CompLearn Toolkit in the presence of noise. In Fig. 5, several text file result from clustering texts in which several levels of noise have been added to each text.

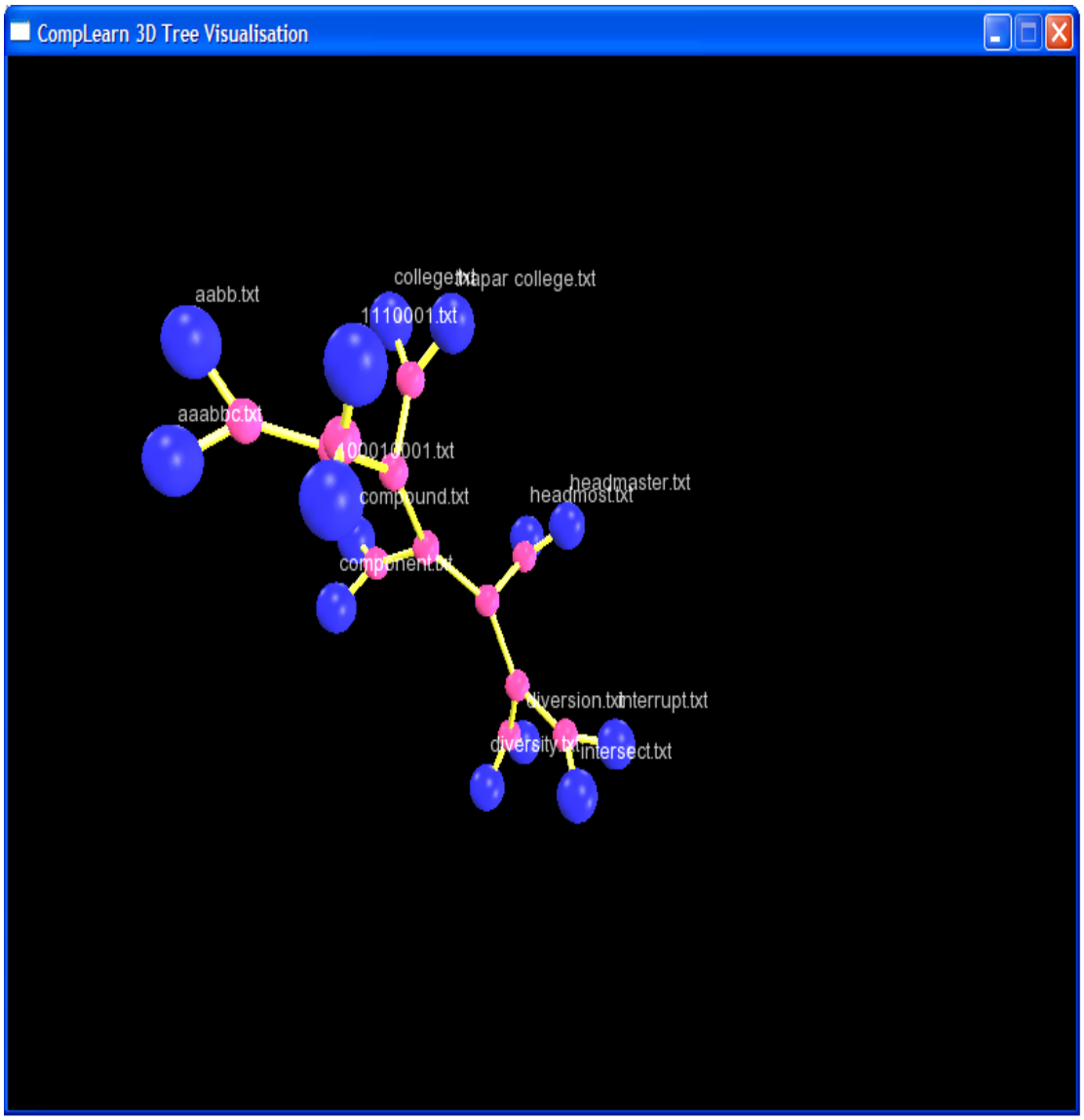

Figure 5.3: NCD-driven clustering of texts without adding any noise.

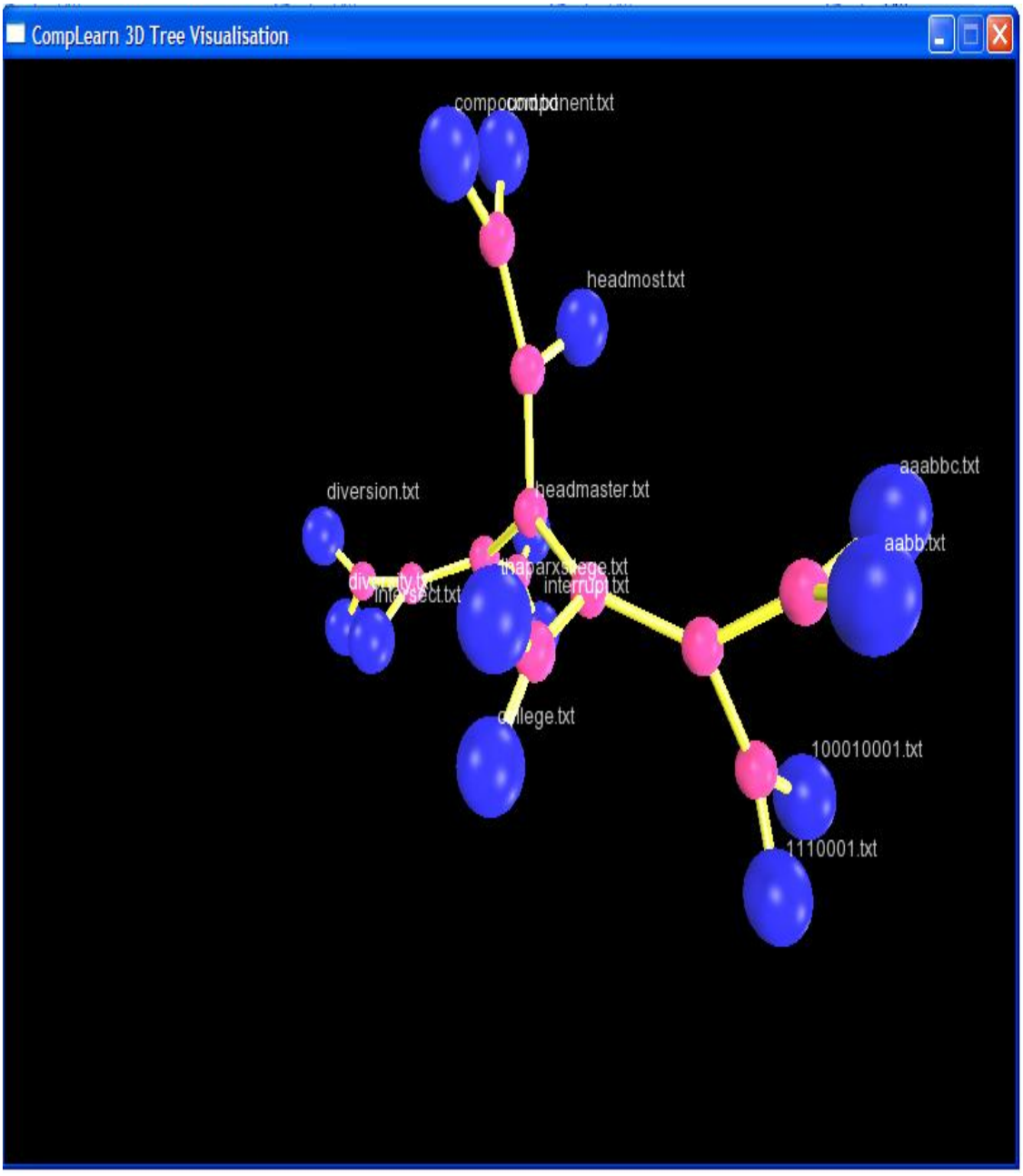

Figure 5.4: NCD-driven clustering of texts in which 25% noise has been added in P and Q

It can be clearly seen from the above picture that the quality of the clustering degrades slowly due to the linear growth of the average distortion.

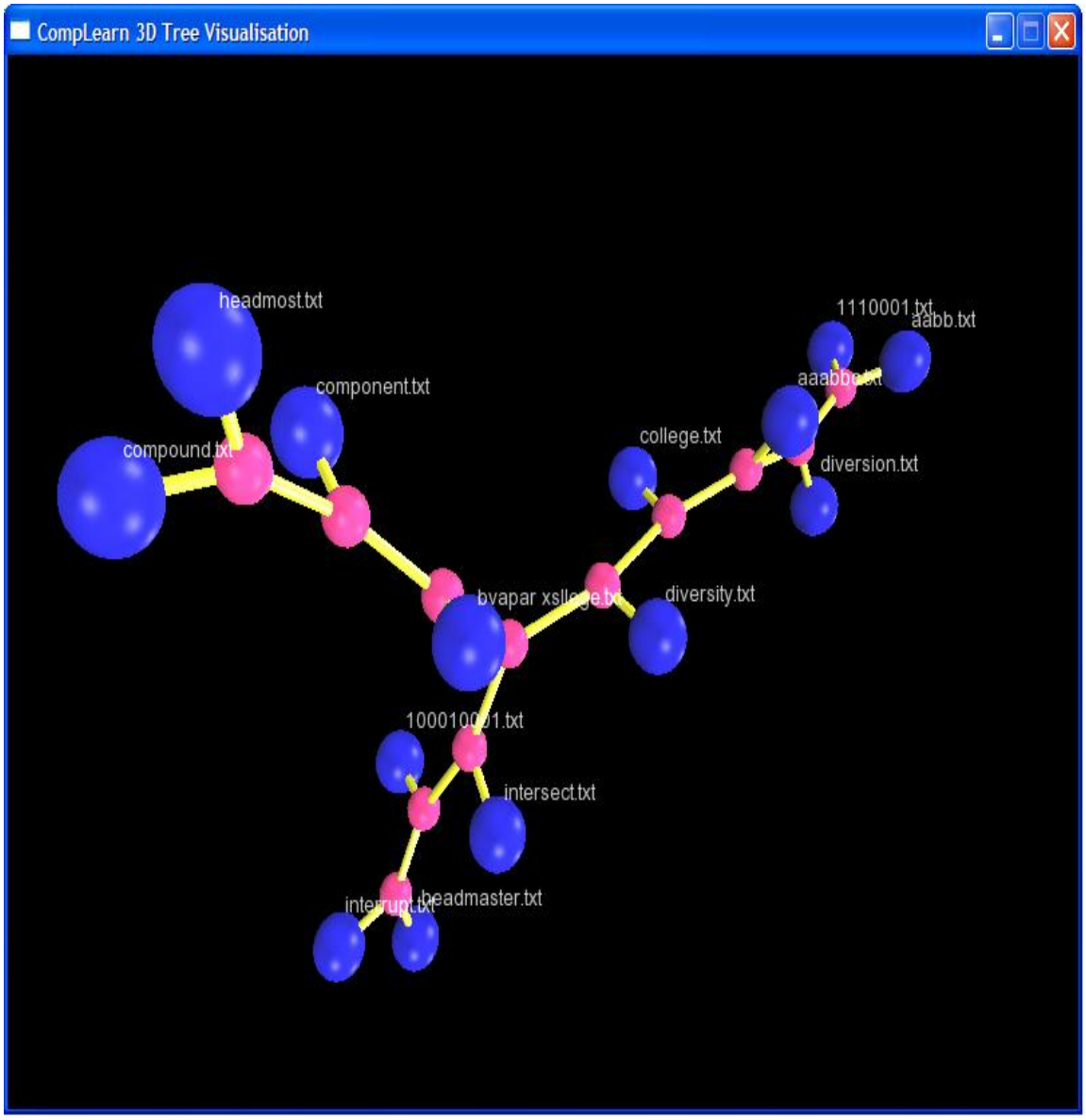

Figure 5.5: NCD-driven clustering of texts in which 50% noise has been added in P and Q

The above picture clearly indicates that the quality of the clustering degrades slowly due to the linear growth of the average distortion.

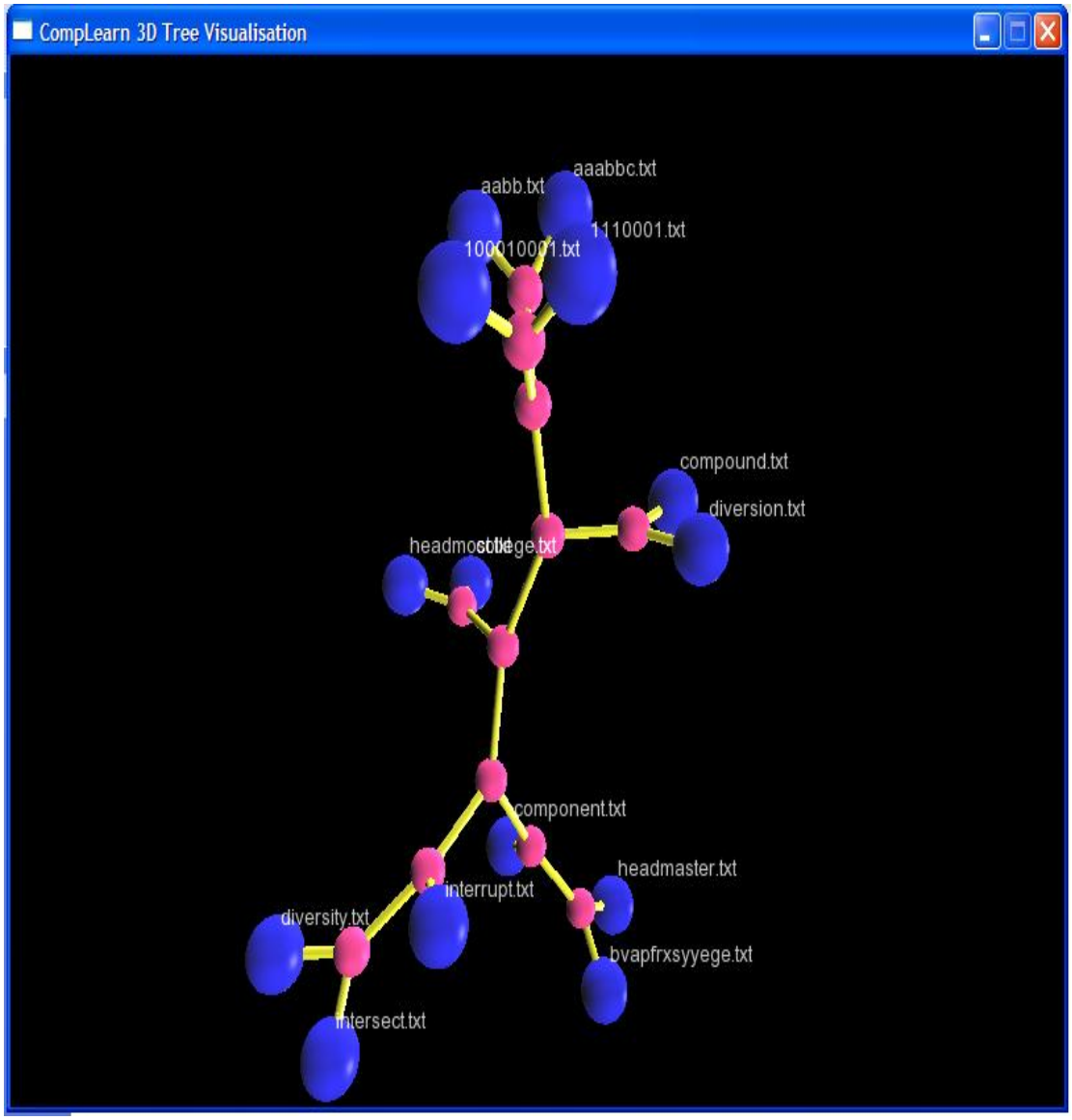

Figure 5.6: NCD-driven clustering of texts in which 75% noise has been added in P and Q

Figure 5.6 shows that the quality of the clustering degrades slowly due to the linear growth of the average distortion.

Similar experiments are repeated with different text in Fig 5.7 by using CompLearn Toolkit.

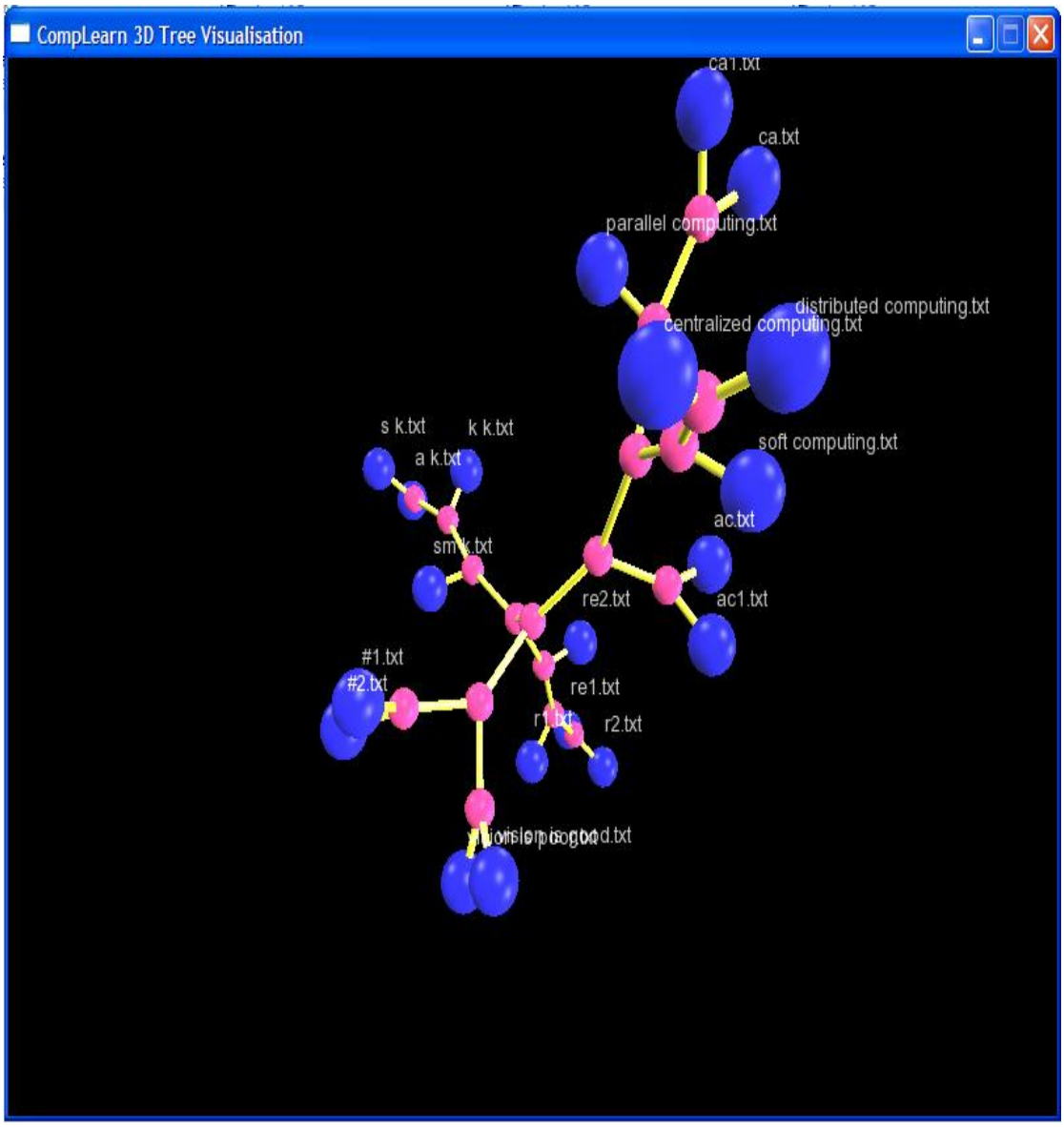

Figure 5.7: NCD-driven clustering of texts without adding any noise.

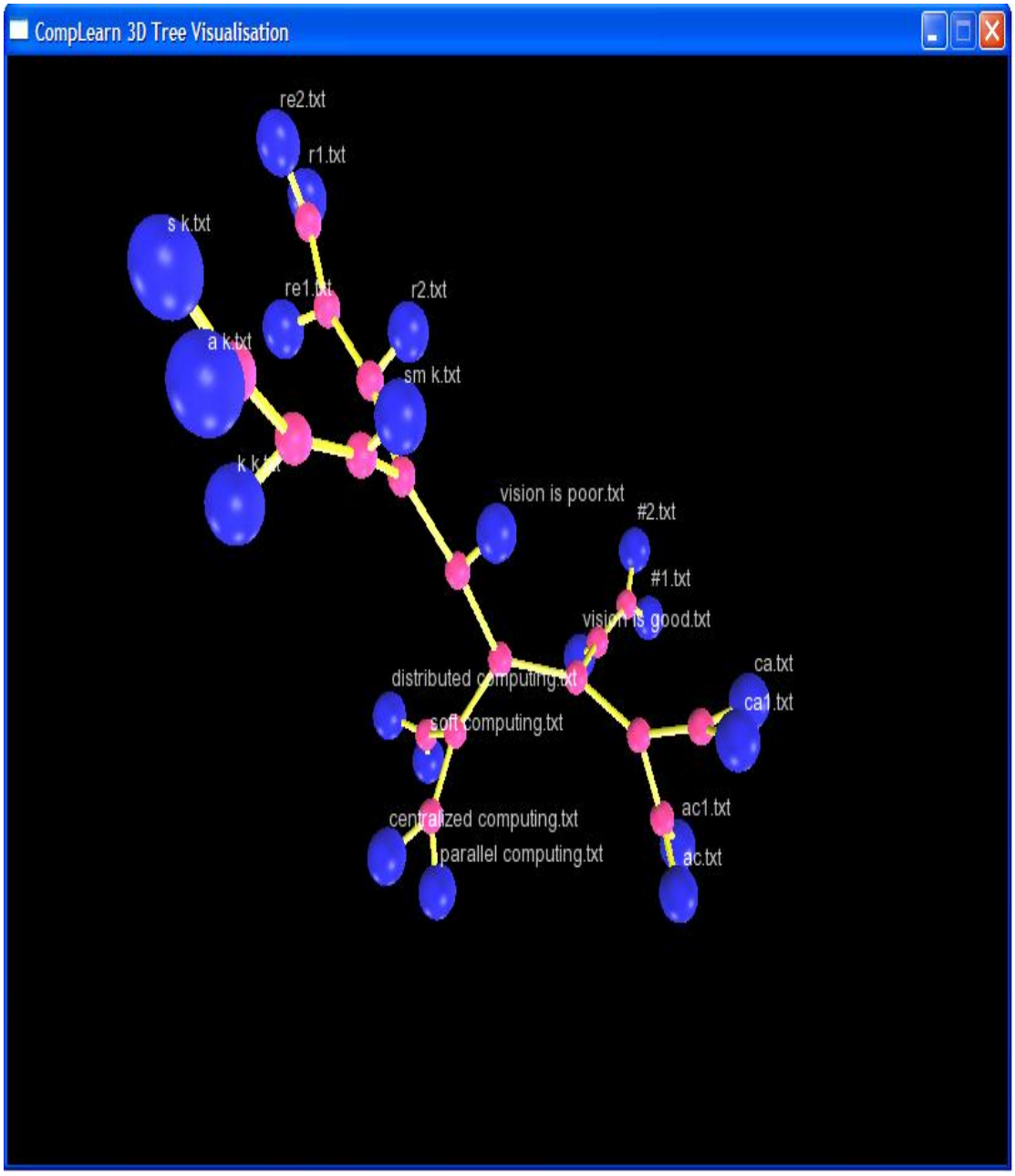

Figure 5.8: NCD-driven clustering of texts in which 25% noise has been added in R and S.

In Figure 5.8 shows that the quality of the clustering degrades slowly due to the linear growth of the average distortion.

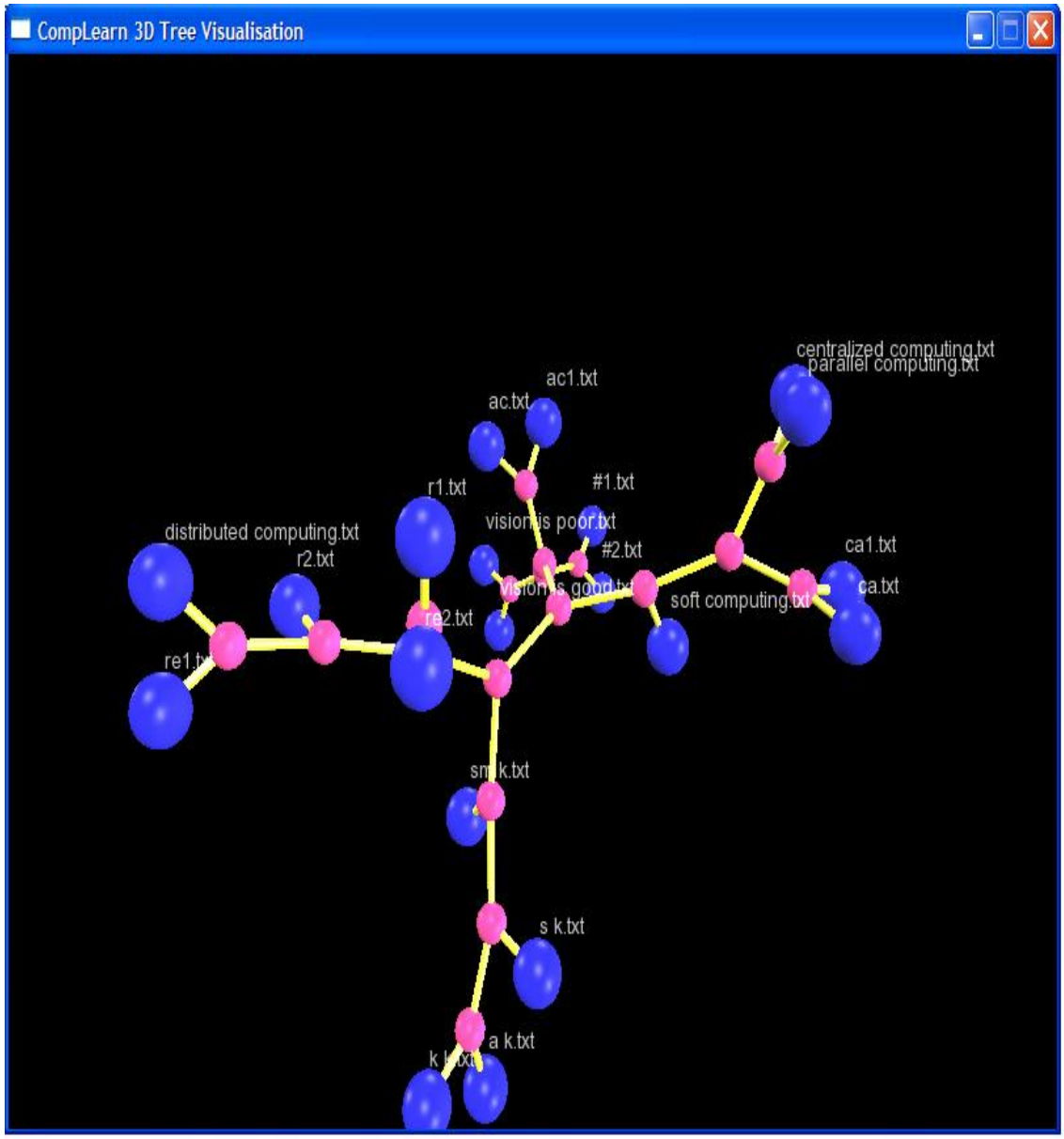

Figure 5.9: NCD-driven clustering of texts in which 50% noise has been added in R and S.

In Figure 5.9 it is clearly evident that the quality of the clustering degrades slowly due to the linear growth of the average distortion

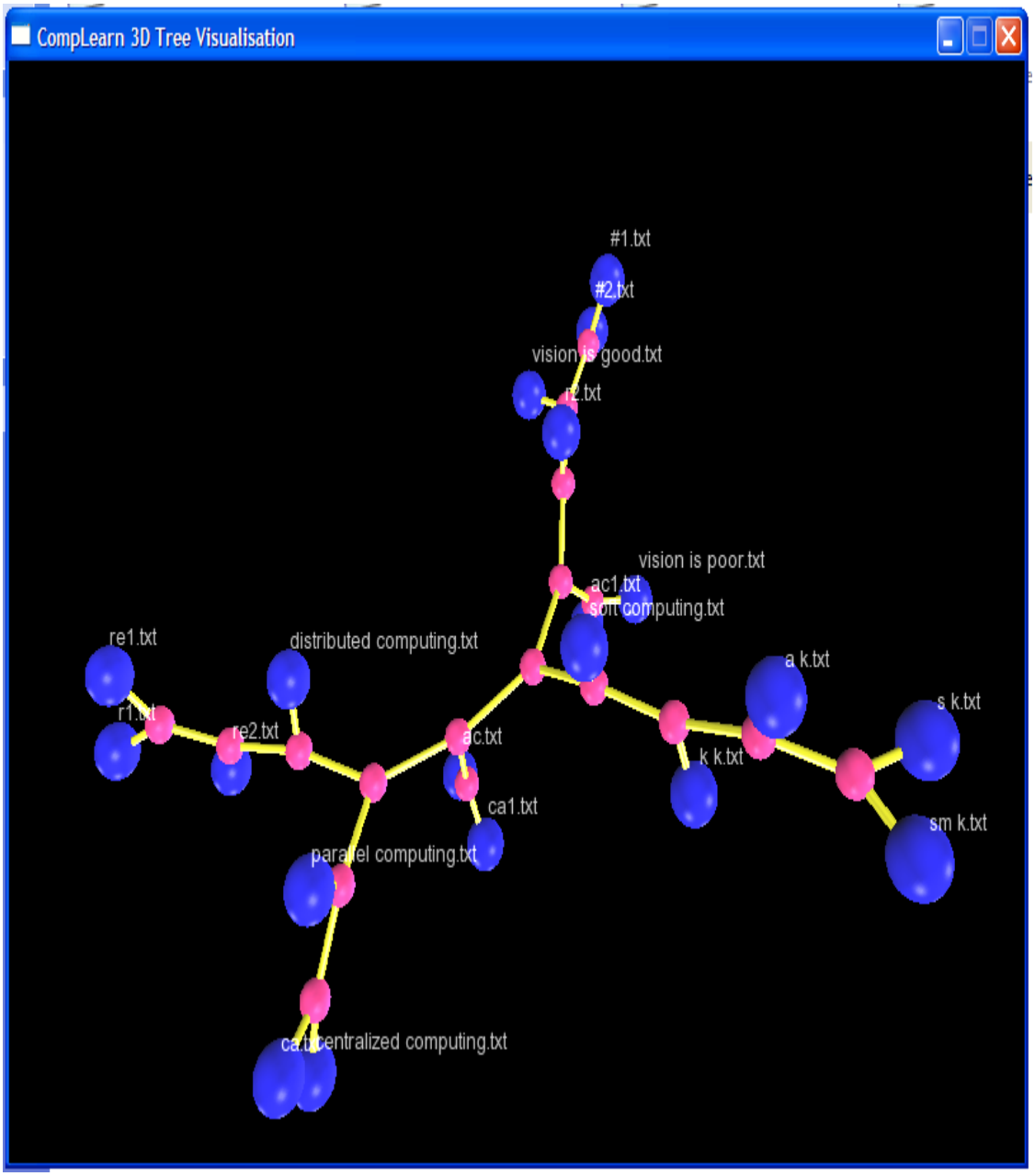

Figure 5.10: NCD-driven clustering of texts in which 75% noise has been added in R and S.

In Figure 5.10 can be visualization that the quality of the clustering degrades slowly due to the linear growth of the average distortion.

Chapter 5

Conclusion and Future scope

When the NCD is used to compute the distance between two different files, the second file can be considered as a noisy version of the first. Therefore, the effect on the NCD of the progressive introduction of noise in a file can provide information about the measure itself. In this correspondence, we forward a theoretical reasoning of the expected effect of noise introduction, which explains why the NCD can get values greater than 1 in some cases. A first batch of our experiments confirms the theoretical model. A second batch explores the effects of noise on the precision of clustering's based on the use of the NCD. It can be noticed that the clustering process is qualitatively resistant to noise, for the results do not change much with quite large amounts of it.

The major conclusion which can be drawn from the entire experimentation is that as noise grows in textual file the clusters are scattered clearly indicating the worsening of clusters due to noise on the other hand there is marginal effect of noise on a compressed textual file.

In the future, we intend to tackle a quantitative demonstration of the NCD resistance to noise. We shall also try other metrics and clustering procedures, appropriate to the different file types, to compare their resistance to noise with our NCD results.

References

[1] Khalid Sayood, "Introduction to Data Compression", Third Edition Morgan Kaufmann Publishers, ISBN 13:978-0-12-620862-7, December 15, 2005.

[2] Data Compression,
URL: http://en.wikipedia.org/wiki/Data_compression.

[3] Turing Machine,
URL: http://plato.stanford.edu/entries/turing-machine.

[4] Paul M. B. Vitanyi, Frank J. Balbach, Rudi L. Cilibrasi, and Ming Li, "Normalized Information Distance", September 2008.

[5] Lance Fortnow, "Kolmogorov Complexity", January 2000.

[6] M. Li, P. M. B. Vitanyi, "An Introduction to Kolmogorov Complexity and Its Applications", 2nd Ed., Springer-Verlag, New York1997.

[7] Cilibrasi, R.L., Vitanyi, P.M.B. "Clustering by compression" IEEE Transactions on Information Theory, Vol. 51, No.4, 1523–1545 2005.

[8] Chen, S. Kwong, and M. Li. "A compression algorithm for DNA sequences and its applications in genome comparison", In Genome Informatics: Proceedings of the 10th Workshop on Genome Informatics, pages 51–61, 1999.

[9] Matti Nykter, Olli Yli-Harja and Ilya Shmulevich, "Normalized Compression Distance for Gene Expression Analysis", 2005.

[10] Nicholas Tran, "The Normalized Compression Distance and Image Distinguishability", 2007

[11] N. Krasnogor and D. Pelta, "Measuring the similarity of protein structures by means of the universal similarity metric", *Bioinformatics*, Vol. 20, No. 7, pp. 1015–1021, 2004.

[12] Cilibrasi R, Vitányi P, De Wolf R, "Algorithmic Clustering of Music Based on String Compression", Computer Music J., Vol. 28, No. 4, pp. 49-67, 2004.

[13] R. Cilibrasi, A. L. Cruz, and S. de Rooijet, The CompLearn Toolkit [Online]. Available: http://www.complearn.org.

[14] M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi, "The similarity metric", IEEE Transactions of Information Theory, Vol. 50, No.12, 3250-3264, 2004.

[15] Bennett, C., Gacs, P., Li, M., Zurek, W., Vitányi, P., "Information Distance", IEEE Transactions on Information Theory, Vol. 44, No. 4, 1407-1423,1998.

[16] R. Cilibrasi, P. Vitanyi. "The Google similarity distance", IEEE/ACM Transactions on Knowledge and Data Engineering,

[17] Manuel Cebrian, Manuel Alfonseca, and Alfonso Ortega "Common Pitalls Using the Normalized Compression Distance: What to Watch Out for in a Compressor" Communication in Information and System, Vol. 5, No. 4, pp. 367-384, 2005.

[18] Kiril Simov and Petya Osenova, "Experiments with Normalized Compression Metric", Linguistic Modelling Laboratory, Bulgarian Academy of Sciences 5-6 December 2005.

[19] Huffman Coding.
URL: http://en.wikipedia.org/wiki/Huffman_coding.

[20] S. Braunstein, C. Fuchs, D. Gottesman, and H. Lo. "A quantum analog of huffman coding". IEEE Transactions of Information Theory, Vol.46,1644–1649, 2000.

[21] Manuel Cebrián, Manuel Alfonseca, Alfonso Ortega, "The Normalized Compression Distance Is Resistant to Noise", 0018-9448, 2007.

[22] Sameh Ghwanmeh, Riyad Al-Shalabi and Ghassan Kanaan , "Efficient Data Compression Scheme using Dynamic Huffman Code Applied on Arabic Language" Journal of Computer Science, Vol. 2, No.12, 885-888, 2006.

[23] Volker Nannen, "A Short Introduction to Kolmogorov Complexity", http://volker.nannen.com/work/mdl/, April, 2003.

[24] Rudi Cilibrasi and Paul Vitanyi, "Similarity of Objects and the Meaning of Words".

[25] Faustino J. Gomez, "Sustaining Diversity using Behavioral Information Distance" GECCO'09, July 8-12, 2009.

Papers Published

1. Sudesh Kumar and Shalini Batra, "Cross Document Co-reference resolution" 3rd National Coreference on Recent Advances and Future Trends in IT, Punjabi University, Patiala, Punjab, April 9-10, pp. 209, 2009.

2. Sudesh Kumar and Shalini Batra, "Analyzing and Visualizing the Effect of Noise on Compressed Textual Data" CiiT International Journal of Data Mining and knowledge Engineering. ISSN 0974-9675, http://ciitresearch.org/ [Communicated].