Authorship Attribution Using Entropy Encoding

Brandy Poag-Dorado

May 13, 2013

Contents

Abstract

There has been a new development in the field of data-mining; now data-compression methods are being used to classify data. Data mining provides methods for searching for previously unknown but meaningful data patterns in fully or semi-automated ways. The main purpose of the data mining process is to extract information from a data set and transform it into an understandable structure that is useful. Where as, the main purpose of data compression is to reduce the size of data while maintaining the integrity of the data this is known as source coding.

Data compression software is traditionally used to economize disk storage and data transmission cost. Now the same data compression software is being used in other areas including machine learning and data-classification. Exploring the Shannon's Source Coding Theorem and the entropy encoding process we can understand how data compression software can be used to mine data.

I will use two entropy encoding methods, Huffman Coding and Arithmetic Coding, to compress some books from various authors. Then I will use a metric of similarity, the Normalized Compression Distance (NCD), to classify textual data. Then we will be able to detect the similarities within the data well enough to indicate the author of a given text. The NCD proves to be a fairly accurate classifier and requires a relatively small amount of textual data. This experiment has some surprising results.

1 **Information Theory**

One of the many branches of Computer Science is Information Theory. This branch is shared by many other fields including, bio-informatics, electrical engineering, and applied mathematics. Information Theory is concerned quantification data and provides methods for measuring data. Data is the foundation of Computer Science so its no surprise that Information Theory has become very useful in other areas of computer science as well including natural language processing, data-mining, quantum computing, plagiarism detection, data analysis, and cryptography. One of the major areas in Information Theory is data compression.

2 **Data Compression**

Ashtonishing amounts of data are created, used, stored, and transmitted on a daily basis. For example, EROS in South Dakota is the largest data land mass in the world and contains 130 terabytes of data. There is a project underway involving the aeronautical programs from several countries that will generate a terabytes of data everyday. Data comes in many forms text, music, videos, internet, databases, and a number of other sources.

The more data that exist the more resources are taxed and the ability to process, store, and transmit data becomes strained. Storage capabilities and the speed at which data is transmitted have improved greatly. However acquiring more storage can be expensive and keeping up with the ever present need for space and speed is difficult.

Data compression alleviates the strain caused by large amounts of data by improving the speed at which data is transmitted and the reducing the space required to store the data. Data compression decreases the size of the data therefore the amount of data being stored is reduced. The idea behind data compression is similar to deflating a balloon. When a balloon is deflated it can not serve its purpose but the balloon is easy to transport and store in this state. So the ballon should stay in this deflated state until it is needed then inflate the balloon as needed.

For example, let there be some text ab aa bb aa b aa this text contains 16 characters in its inflated state. It would nice if we could some how reduce the number of characters need to represent the same message, or in other words deflate or compress this data. So a code can be created let 0-ab, 1-'aa', 2-'bb', 3-'b', and 4-' '. Then the original text is replaced with the newly created codewords and now the text becomes 04142414341 this text contains only 11 characters. This is the basic idea of data compression to reduce the size of data. From 16 characters to 11 characters this is a significant improvement.

Data Compression allows the data to be represented in another form that requires fewer bits. Data Compression relies on some pattern or redundancy in the data in order to be compressed. Once the this pattern is recognized then it maybe exploited.

Data compression algorithms consist of two parts. The first part is the encoder method, this method takes an input X, which is the data that needs to be compressed. The encoder method outputs Xc, which is the compressed representation of the original data that requires fewer bits to represent than the original data. This compressed representation of the data, Xc, is a binary string. The second method is the decoder method that takes the input Xc and outputs Y, which is the reconstructed data. The decoding method is the inverse of the encoding method. Depending on the type of data compression algorithm used the reconstructed data may or may not be identical to the original data. Determining which type of algorithm to use will depend on the application of the reconstructed data. Data compression is very much an experimental science, there is no one all-purpose algorithm that can be used for all situations. Types of data compression are divided into two main categories lossless compression and lossy compression.

2.1 **Lossless Compression**

The earliest form of data compression was introduced by Samuale Morse. Morse devised a system for sending letters by telegraph by encoding letters as dots and dashes. He noticed that some of the letters appeared with greater frequencies than other letters and he encoded the letters with the greatest frequency with the shortest codewords. Morse had devised a lossless compression scheme.

Lossless compression ensures that the original data can be retrieved in its entirety. Some data must retain its integrity in order to be useful, because a small change in the data can change the data's meaning significantly. For example, if a message is sent saying Do not send the money and one letter is altered the message's meaning can be completely changed. What if the letter t became a w the result would be Do now send the money". This can cause some major problems. As a rule of thumb any textual data or data that will be enhanced later on to yield more information, such as x-rays, must have the integrity of the data preserved. The focus of this paper will be centered on the application of lossless data compression schemes, in particular entropy encoding compression schemes.

2.2 **Lossy Compression**

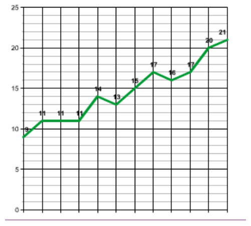
Lossy Compression schemes have a less constraints than lossless compression schemes. The reconstruction of the compressed data does not need to be identical to the original data. When compressing data there are some cases where an insignificant amount of data can be lost and yet the meaning of the data can still be preserved. For instance, a picture of a blue sky maybe able to lose the data of a few pixels and the naked eye will not notice a difference. Another instance where this compression scheme is useful is when processing audio data. Often this data can be compressed in this manner because a recorder may record sounds that a human can not recognize so those sounds can be removed and a human will not be able to notice the difference. In these situations, lossy compression schemes maybe used to compress a message even further than a lossless compression scheme. Lossy compression schemes do not guarantee that all of the orignal data is recoverable this sacrifice makes it possible to gain significantly greater compression ratios than lossless compression techniques.

3 **Modeling**

The process of developing a compression algorithm can be divided into two phases. The first phase is called modeling, here information about any redundancy that exist in the data is extracted. Then the redundancy with in the data can be described in the form of a model. Modeling can be done in a number of ways lets look at some examples.

3.1 **Structure Model**

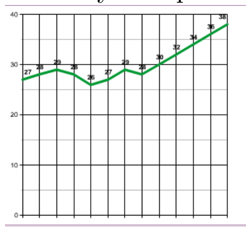
Let there be a sequence of numbers $F(n) = x_1, x_2, \ldots, x_12$ with the values $9, 11, 11, 11, 14, 13, 15, 17, 16, 17, 20, 21$, in order to transmit or store these numbers a minum of $5 bits/sample$ would be needed. If this sequence of numbers where to be ploted on a graph we could see that the points on the graph almost form a line.



We then could exploit the structure of this data by using a model, the equation $F'(n) = n + 8$ for $n = 1, 2, , 12$, formed by the line. Then the structure could be descibed by the difference in the data and the model, known as the residual. This residual is given by the equation $E(n) = F(n) - F'(n) = 0, 1, 0, -1, 1, -1, 0, 1, -1, -1, 1, 1$. Notice that the residual consists of only three numbers $-1, 0, 1$ and would only require $2bits/element$ to represent the residual sequence data if we assigned a code of $00 - -1$, a code of $01 - 0$, and a code of $10 - 1$. Because we were able to recognize the structure of the data we could use that structure to predict the values of the data. All we need to do is to store or transmit the parameters of the model and the residual sequence to compress the original data and we are able to reduce the number of bits required from $5bits$ to $2bits$. This scheme requires that both the encoder and decoder be aware of the scheme that is being used.

3.2 **Predictive Model**

There are other ways to model data lets look at another example of a model. Consider the sequence $F(n) = x_1, x_2, \ldots, x_13$ with the values $27, 28, 29, 28, 26, 27, 29, 28, 30, 32, 34, 36, 38$ this data can not easily be described by an equation of a line.



Notice that all the values are close to one another in proximity are close to each other in value. Another way to model this data is to send the initial value followed by the values that are the differences in the preceding value. So that the sequence to be transmitted would be $E(n) = 27, 1, 1, -1, -2, 1, 2, -1, 2, 2, 2, 2, 2$. This technique of using the past values to predict the next values is called predictive coding.

3.3 **Statistical Model**

There is another approach to finding the redundancy in the data, this is a statistical approach. Many types of data especially textual data contain skewed frequencies of the occurrences of symbols and can be modeled by a statistical model. For instance,the letter e occurs considerably more often than the letter z. For this reason one may take advantage of this knowledge by assigning binary codes of different lengths to different symbols, the most commonly occurring symbols will get assigned the shortest codes. Assume there is a sequence

$F(n) = (a, c, b, a, r, a, y, a, r, a, n, c, a, r, r, a, y, c, r, a, n, c, f, a, r, c, f, a, a, r, c, f, a, a, a, r, c, a, w, a, y)$ which consist of 8 different symbols would require 3 bits per symbol. Noticing that some symbols appear more frequently than others one could assign different length codewords to the symbols. Using this coding scheme $a - 1, b - 01100, c - 001, f - 0100, n - 0111, r - 000, w - 01101, y - 0101$ each symbol would not require the same number of bits to encode. Now replacing the symbols with the codewords, or binary sequences that represent the symbol, we will use $106bits$ to encode opposed to $123bits$. Because there are 41 symbols in the sequence this works out to be $2.58bits/symbol$, and compression ratio of $1.16 : 1$ can be achieved.Throughout the remained of this paper this is the model that will be used. Entropy coding schemes use statistical or probablistic models. Compression schemes of this type depend heavily on the model, if the model is not an accurate representation of the source then reguardless of the algorithm used it is impossible to achieve optimal compression. Thus it is very important to have a model that is a reflection of the source.

4 Shannon

Data compression was revolutionized by Claude E. Shannon. in 1948 when he wrote "A Mathematical Theory of Communication". Later he went on to teach the first ever Information Theory class at MIT. Shannon was interested in signal processing and reliably storing and communicating data, he also wanted to find out how small data could be compressed. Shannon revolutionized data compression with his theories and mathematical interpretations of coding. Before Shannon no one else had been able to mathematically quantify information.

5 Self-informaition

Shannon defined a quantity called self-informaition, which is the information content associated with the outcome of a random variable. Self-information is also referred to as the surprisal because there is an element of surprise when a low probability outcome occurs. By definition, the amount of self-information contained in a probabilistic event depends only on the probability of that event: the smaller its probability, the larger the self-information associated with receiving the information that the event indeed occurred. Intuitively this makes sense, how boring would the news be if the broad cast consisted of news about events with high probabilities.

Self-information of the event $a_i$can be defined as

$$i(a_i) = \log_b \frac{1}{P(a_i)} = \log_b P(a_i)^{-1} = -\log_b P(a_i)$$

Where $P(a_i)$ is the probability that the outcome a_i will occur. When the base on the log is not specified then the base is assumed to be 2 and the units of the self-information is given in bits.

A symbol of an alphabet can be thought of as an outcome of a random variable. If there is some alphabet A that is a set of symbols called letters, then we can define the probability of a symbol and the self-information associated with that symbol. Suppose there are m symbols $a_1, a_2, \ldots, a_m$ from the alphabet A and the ith symbol occurs with the frequency p_i, a fraction between 0 and 1. So $p_1 + p_2 + \ + p_m = \sum_{i=1}^{m} P(a_i) = 1$.

The self-information of the ith symbol is defined as

$$i(a_i) = \log_2 \frac{1}{p_i}$$

For example, let $m = 4$, and let $A = a, b, c, d$, and the probability of the symbol $p(a) = p(b) = p(c) = p(d) = .25$. Then the $I(a) = I(b) = I(c) = I(d) = \log_2 \frac{1}{.25} = \log_2 4 = 2 bits/symbol$. In this example the symbols share equal probabilities and have the same amounts of information content.

When the probabilities of the symbols are more skewed the information content or amount of surprisal changes considerably.

For example, let $m = 4$, and let $A = a, b, c, d$, and the probability of the symbol $p(a) = .7$ and $p(b) = p(c) = p(d) = .1$. Then the $I(a) = \log_2 \frac{1}{.7} = .51 bits/symbol$ and $I(b) = I(c) = I(d) = \log_2 \frac{1}{.1} = 3.32 bits/symbol$. Here we can see that the less frequent symbols contain more information.

6 **Entropy**

A message is a string of letters, so we can compute the quantity of information not only the individual symbol but for the entire message. This is known as the weighted average self-information. Shannon defined this quantity to be Entropy or Shannon Entropy of the message. The first-order entropy of the entire system is the average information content of the symbols as they are actually seen. The entropy of the source S with an alphabet $A = a_1, a_2, \ldots, a_m$ that generates a sequence $s_1, s_2, \ldots, s_n$ is given by

$$H(S) = \sum_{i=1}^{n} P(a_i) \log_2 \frac{1}{P(a_i)} = -\sum_{i=1}^{n} P(a_i) \log_2 P(a_i)$$

where a_i is the ith symbol of the source and $P(a_i)$ is the probability of the ith symbol of the source. The entropy of the source will be some value greater than zero. Here we are assuming that the messages are independent of one another. This formula is only valid when the source is an iid or independently and identically distributed model and the probability model matches the source.

Shannon said that if "we have a set of possible events whose probabilities of occurrence are $p_1, p_2, \ldots, p_n$. These probabilities are known but that is all we know concerning which event will occur. Can we nd a measure of how much choice is involved in the selection of the event or of how uncertain we are of the outcome?

If there is such a measure, say $H(p_1, p_2, \ldots, p_n)$, it is reasonable to require of it the following properties:

1. H should be continuous in the probabilities of i or p_i.

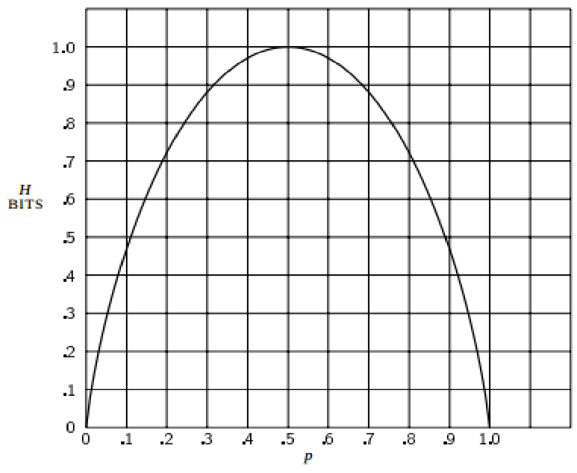
2. If all the p_i are equal, $p_i = \frac{1}{n}$, then H should be a monotonic increasing function of n. With equally likely events there is more choice, or uncertainty, when there are more possible events.

3. If a choice be broken down into two successive choices, the original H should be the weighted sum of the individual values of H."

So Entropy really quantifies the unevenness of the probability distribution. Look at the case when the entropy of an outcome is zero $H(X) = 0$, this is the lowest possible entropy. Here there is no uncertainty, and the probability distribution is fully localized and exactly one outcome has the probability of one and all other outcomes have the probability of zero. The complete opposite situation would occur when all of the outcomes have exactly the same probability then the entropy reaches a maximum $H(X) = \log_2 |X|$, for a uniform distribution. In this case the source is truly random and therefore can not be compressed.

Now we will consider the case when then are only two outcomes with the probabilities p and $q = 1 - p$, we will graph the entropy as a function of p.

$$H = -(p \log_2 + q \log_2 q)$$



Lets revisit the previous example, let $m = 4$, and let $A = a, b, c, d$, and the probability of the symbol $p(a) = p(b) = p(c) = p(d) = .25$, and let $n = 8$, we can define the source S to be the sequence *bdaabdcc*. Therefore the entropy

$$H(S) = -\sum_{i=1}^{8} .25 \log_2(.25) = 4bits$$

In this example the symbols share equal probabilities and the entropy is quite large, this the maximum entropy.When the probabilities of the symbols are more skewed the entropy can be lowered. For example, let $m = 4$, and let $A = a, b, c, d$, and the probability of the symbol $p(a) = .7$ and $p(b) = p(c) = p(d) = .1$, and let $n = 10$, we can define the source S to be the sequence *bdaaaaacaa*. Therefore the entropy is

$$H(S) = .1\log_2(.1)+.1\log_2(.1)+.7\log_2(.7)+.7\log_2(.7)+.7\log_2(.7)+.7\log_2(.7)+.7\log_2(.7)+.1\log_2(.1)+.7\log_2(.7)+.7\log_2(.7) =$$

In this example the entropy is considerably lower. entropy.

7 **Source Coding Theorem**

Shannon's Source Coding Theorem, also known as the noiseless coding theorem, says that given a discrete memoryless source with the entropy $H(S)$, the average code-word length L for any distortionless source coding is bounded as $L \geq H(S)$. This proves that the best any lossless compression scheme can do is to encode the source with the same number of bits as the entropy.

7.1 **Coding**

Source coding, or coding is the second phase of the compression process. Coding, is the assignment of sequences (usually binary) to elements of an alphabet which is a collection of symbols, otherwise known as letters. The set of binary sequences created in this process is called a code. Some codes uses the same number of bits to represent each symbol this is called a fixedlength code. There are different types of codes block-block, block-variable, variable-variable, and variable-block codes.

The first type block-block code, or some call this a fixed-length code, this is the oldest and most widely used type of code, one example of this type is ASCII code. ASCII code takes an alphabet of 64 characters or symbols and maps them onto 6-bit or 8-bit binary codewords. Ex. $'A' - 1000001$ and $'a' - 1000011$. These type of codes do not provide compression.

The next type block-variable code here fixed-length source messages are mapped onto variable-length codewords. Huffman codes are an example of this type of code. For example, $'a' - 01$ and $'b' - 0001$. Variable-variable code maps one or more symbols from the source messages onto variable-length codewords, dictionary techniques uses this approach. For example, $'th' - 01$ and $'rat' - 001$. Finally, as you may have guessed, variable-block codes maps one or more symbols from the source messages onto fixed-length codewords. For example $'and' - 111$, $'th' - 000$, and $'e' - 010$.

Variable length codes map source symbols to codes of variable lengths. Both block-variable and variable-variable codes are considered variable length codes.

7.2 **Symbol Codes**

Lets consider some of the appealing properties of the symbol codes. First we define a code $C(X)$ to be the function of X, where X is a symbol from the alphabet A and we define the extension of the code using concatenation to be $C(X^n)$ and

$$C(X^n) = C(X_1)C(X_n)$$

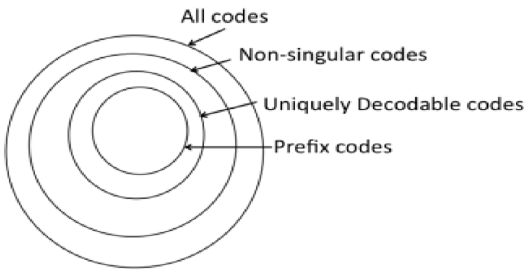
We would like the codes to satisfy the following three properties.

1. Non-singularity or unique so that for all symbols from the alphabet A, if two symbols are not the same the codes of the symbols are not the same.
2. Unique-decodability or biunique means that the code of a source is unique and no other source will share the same code.
3. Self-punctuating or instantaneous or prefix code means that no code of a symbol will be the prefix of another symbol.

This property is nice because as soon as the codeword has been seen it may be decoded rather than needing to see an entire sequence. Notice that if a code is biunique then the code must also be unique and if a code is a prefix then it is biunique. Therefore prefix code contain all three properties.

7.3 **Prefix Codes**

Lets look at some examples of possible codes to determine if the codes are prefix or not. The codewords $010, 0101, 110$ could not all be a part of a prefix code because the codeword 010 is a prefix of 0101 and this violates the prefix property. The codewords $001, 1, 01$ could all be part of a prefix code. Prefix codes are usually variable length codes.



7.4 **Kraft-McMillan Theorem**

The Kraft-McMillan theorem contains two parts but first we must define the term b-ary code. A b-ary code for some positive integer B, is a code C that maps symbols from the source alphabet to B number of symbols. We will only consider this theorem for the case when the code is binary in other words B is 2.

The first part of the theorem is McMillan's part, he says that for any uniquely-decodable binary code C with K codewords the following inequality must be satisfied,

$$\sum_{i=1}^{K} \frac{1}{2^{L(i)}} \leq 1$$

where $L(i) = |C(i)|$.

The second part of the theorem is Kraft's part, he says that if a code with lengths that satisfy McMillan's inequality than there exist a binary prefix code C with the same lengths.

Therefore, if a code is uniquely decodable the codeword lengths have to satisfy the McMillan inequality and if the given codeword lengths satisfy the Kraft-McMillan inequality we can always find a prefix code within the given codeword lengths. By restricting ourselves to prefix codes there are no non-prefix codes that are uniquely decodable with shorter average codeword lengths.

8 **Measures of Performance**

8.1 **Compression Ratio**

Being a prefix is not the only quality that makes a good code there areother qualities that help determine if a code is optimal. Another way to measure the performance of a code is to look at the ratio of the number of bits required to represent the data before compression to the number of bits required to represent the data after compression. This ratio is called the compression ratio. If $65,536 bits$ are needed before compression and $16,384 bits$ are needed afterward then the compression ratio is $4:1$. The compression ratio can also be given as a percentage in this senario the compression ratio is 75%.

8.2 **Compression Rate**

Another measure of performance is to consider the average number of $bits/symbol$ required to represent a single sample, this is called the rate. The rate is also called to as the compression rate. For example, assume we are compressing an image that contains 8 bits per byte (pixel) then the average number of bits per pixel in the compressed representation is 2 thus the rate is 2 bits per byte (pixel).

8.3 **Entropy**

There are many ways to measure performance but arguably the best method is to consider the entropy of the source message. Shannon's source coding theorem proves that it is impossible to to compress data beyond the entropy of the source without resulting in the loss of information. Therefore the best that any lossless compression scheme can do is to encode the source with an average number of bits equal to the entropy of the source, this is called entropy coding.

8.4 **Metric Entropy**

The final measure of performance we will consider is the metric entropy. The metric entropy is the entropy of the source message divided by the length of the source message. This metric allows randomness of the message to be evaluated. Metric entropy can take on values from 0 to 1. A one meaning that the source is equally distributed and is very random and a zero meaning that the source is not very random.

9 History of Entropy Coding

The remainder of the paper will be concerned with entropy coding techniques. We will take a closer look at two entropy coding methods. First lets take a brief look back at the history of entropy coding.

9.1 Shannon-Fano Coding

In 1949, Claude Shannon and Robert Fano create similar methods for assinging prefix codes to symbols from an alphabet based on probabilities of the symbols. A prefix code, or prefix-free code, is a code in which no codeword is a prefix to another codeword; and a codeword is the individual members of the set of binary sequences assigned to elements of an alphabet. Shannon proposed this method in an article called A Mathematical Theory of Communication, this paper is considered to be the begining of the field of information theory. In this article the idea of a quantitative measure was revolutionized when Shannon defined the quantity called self-information. That is why Shannon is considered to be the father of the field of information theory and data compression. Shannon also taught the first data compression course at MIT, this class would play a vital role in making the field become what it is today. Shortly after Shannon wrote the famous article, Fano published the same method as a technical report. This method garuntees that the codeword lengths are within one bit of the self-informaition but fails to ensure that the length of the codewords are optimal.

9.2 Huffman Coding

In 1951 David Huffman discovered an optimal method of implementing Shannon-Fano techniques. Huffman a student of Shannon's at MIT was motivated by his desire to avoid taking a final exam and instead write a term paper finding the most efficient binary code. Huffman developed this technique using a frequency-sorted binary tree and a bottom up approach. Huffman codes produce variable-length codewords that garuntee the shortest length codewords and is optimal under certain constraints. In the 1970's a method called adaptive Huffman coding that dynamically updated codewords as new symbols were encountered was created. As online storage of files became a common practice, software compression programs were developed most of which used adaptive Huffman coding.

9.3 Shannon-Fano-Elias Coding

Shannon-Fano-Elias Coding is a method for generating uniquely decodable codes using the Cumulative Distribution Function to allocate codewords. This coding scheme achieves codeword lengths within 2 bits of the entropy so it is not readily used but is important because it is a precursor to arithmetic coding.

9.4 Arithmetic coding

The best lossless entropy coding scheme is arithmetic coding because this scheme achieves the best compression ratio. No other algorithm that uses the same model can achieve a higher compression ratio. Modern arithmetic coding owes its birth to Pasco and Rissanen in 1976, until their independent discoveries the problem of finite precision could not be resolved. Rather than assigning codewords to individual symbols like other entropy coding schemes, arithmetic coding encodes the entire source to a single binary number. This allows it to be as if not more efficient than Huffman coding.

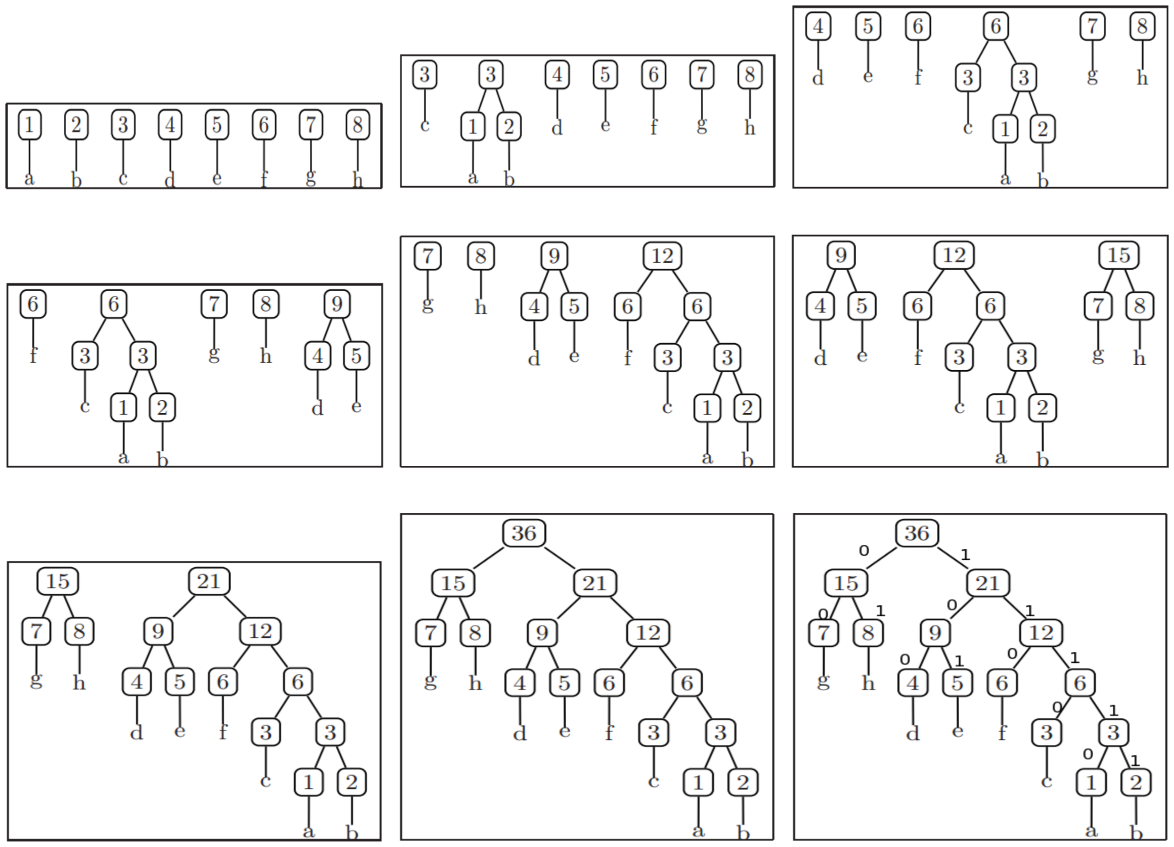
10 **Huffman Coding**

Lets revisits the previously mentioned technique called Huffman coding and explore this method in greater detail. Huffman coding creates and assigns a unique prefix code to each unique symbol from the source. Huffman coding works by creating a unique codeword for each unique symbol using the probability of occurrence of that symbol. This method takes $O(n \log_2 n)$ operations to construct. If the probabilities or weights are already sorted then the codes can be created in linear time.

The technique works by creating a binary tree using a statistical model. The leaf nodes are symbols that have weights that correspond to the probabilities of the symbols. To construct this tree, you have to create a list of the probabilites. Remove the two symbols with the lowest frequency from the remaining symbols in the list. Add the lowest frequency nodes to a new alpha node that has the frequency of the sum of the two lowest frequency nodes together and call this node alpha1 add the new alpha node and its children to the tree. Then add the alpha node to the list. Repeat these steps until all the symbols are in the tree the final alpha node is the root of the tree. When this process is finished there will be N leaf nodes and $N-1$ internal nodes, the leaf nodes will be the symbols and the internal nodes will be the alpha nodes. Now that the tree is complete starting at the root assign a zero to the left branches and ones to the right branches. To find a codeword for a given symbol follow the path from the root of the tree to the given leaf node gathering up the numbers on the branches. This binary string is the codeword for a given symbol.

11 **Huffman Coding Example**

To make this process more concrete we will look at an example. If we have the following alphabet $A = a, b, c, d, e, f, g, h$ with the corresponding frequencies $P = 1, 2, 3, 4, 5, 6, 7, 8$ we could create the following Huffman tree.



Resulting in the codewords $g - 00, h - 01, d - 100, e - 101, f - 110, c - 1110, a - 11110, b - 11111$.

To determine if a Huffman code is optimal we can consider the average codeword length. The average codeword length for a source S with the alphabet $A = a_1, a_2, \ldots, a_k$ and the probability model $P(a_1), P(a_2), \ldots, P(a_k)$ is given by

$$L' = \sum_{i=1}^{k} P(a_i)L(i)$$

where $L(i) = |C(i)|$.

If the Huffman code satisfies the Kraft-McMillan inequality then the following are true.

1. The average codeword length of an optimal code for a source is greater than or equal to the entropy of the source.
2. The average codeword length of an optimal code for a source is strictly less than the entropy of the source plus one.

Therefore we can create an upper and lower bound for a Huffman code. It can be shown that an even tighter upper bound can be found. Huffman code has a code rate within $pmax + .086$ of the entropy, where $pmax$ is the probability of the most frequently occuring symbol. As the alphabet size grows the size of $pmax$ gets quite small. If the alphabet is small and the probability of occurrence of different letters is skewed the value of pmax can be large. Then Huffman code becomes rather inefficient compared to the entropy. To avoid this problem symbols can be blocked together to generate extended Huffman codes.

Lets look at another example of a case when Huffman is not optimal. If we have a source who's output is an iid using the letters from the alphabet $A = a1, a2, a3$ that have the probabilities $P(a_1) = .95$, and $P(a_2) = .02$, $P(a_3) = .03$. The entropy of the source is $.335 bits/symbol$. A Huffman code could be generated that has an entropy of $.715 bits/symbol$ which is 213% of the entropy and would require more than twice the bits promised by the entropy, averaging $1.05 bits/symbol$.

An extended Huffman approach could be used by grouping the symbols together in blocks of two. This code would be an improvement and require $1.22 bits/symbol$ and have an entropy of $.611 bits/symbol$ but this is still 72% more than the entropy. If the Huffman blocks are continually extended blocks of 8 symbols would be needed before acceptable values of entropy could be attained. The problem is that blocks of this size would have an alphabet so big that it would be impractical for many reasons. First of all, a code of this size requires more memory for storage than maybe available. Even if a reasonably efficient encoder code be created, a decoder would be highly inefficient and very time consuming. Lastly if the statistical mode was not almost perfect there would be a huge negative impact on the efficiency. While using extended Huffman codes help reach optimal entropy the consequences is a codeword book that grows exponetionally. In order to create a codeword for a sequence of length m all sequences of length m must be generated and this is not usually practical or efficient.

Huffman Codes are only optimal as symbol codes therefore if the symbols are not independent of one another they are not optimal. Another requirement of these codes is that the probability of each input symbol is a negative power of two, if this utopic condition is not the case then the codes won't be optimal. Despite the pitfalls of Huffman, there are situation were Huffman codes are optimal. The coding scheme garuntees a coding rate R within one bit of the entropy H, in some cases. Recall the coding rate is the average number of bits needed to represent a symbol from the source alphabet and the entropy is the lowest rate at which the source can be coded.

Huffman coding has many applications. It is interesting to note that the Huffman coding algorithm, originally developed for the efficient transmission of data, also has a wide variety of applications outside the sphere of data compression. These include construction of optimal search trees, list merging, and generating optimal evaluation trees in the compilation of expressions. Additional applications involve search for jumps in a monotone function of a single variable, sources of pollution along a river, and leaks in a pipeline. The fact that this elegant combinatorial algorithm has influenced so many diverse areas underscores its importance.

12 **Arithmetic Coding**

Now we will shift our attention from Huffman Coding to another entropy coding technique mentioned earlier called Arithmetic Coding. Arithmetic Coding uses a one-dimensional table of probabilities rather than creating a tree. Arithmetic coding is another way to generate variable-length codes. This method allows a sequence of length m to be created without generating all the sequences of length m, infact arithmetic coding generates one and only one sequence of length m. By creating a unique identifier or tag for a sequence of length m. This tag is a binary fraction which becomes the binary code for the sequence. Unlike Huffman which encodes symbols with individual codewords this method encodes symbols using fragments of bits.

Arithmetic Coding is especially useful when dealing with sources with small alphabets such as binary alphabets and alphabets with highly skewed probabilities. Fax machines with black and white codes have significantly more white symbols than black ones making Arithmetic Coding ideal. Arithmetic coding is near optimal even on large messages. This technique is fast and efficient.

Arithmetic coding accommodates any probabilistic model and can be adapted on the fly and allows for more complicated model than an iid. This makes the technique very compatible with a wide variety of data because arithmetic coding treats the model like a black box. The ability to separate the modeling from the coding is a great feature of Arithmetic coding. Conceptually the coding process can be divided into two parts. During the first part of the process a unique identifier or tag is generated for a given sequence of symbols. The second phase consist of assigning a unique binary code to each of the tags. This phase generates a unique arithmetic code for all sequences of length m without having to generate all possible sequences.

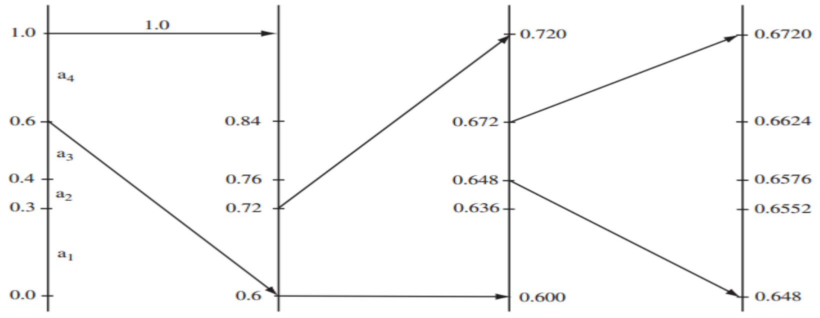
Tagging can be done in a number of ways. One preferred method is to use the set of tags within the interval from zero to one because there is an infinite amount of numbers contained within this interval. Then the Cumulative Distribution Function or CDF is used to subdivide the unit interval into smaller intervals corresponding to the symbols of the source alphabet. The CDF is a function that can map random variables and random sequences of random variables into the unit interval.

Arithmetic coding relies on the fact that there are an innite amount of number between 0 and 1, and exploits this fact. So no matter what the tag becomes no two tags will ”colide” with one another unless they are the same message.

To encode the message start with an interval from 0 to 1. Subdivide the initial interval into smaller intervals that corespond to the probability mass function. After the interval is divided according to probabilities of occurrence of each of the letters of the alphabet select the subinterval that coresponds to the first symbol of the message. The newly selected interval becomes the new interval then the process of subdividing the interval into subintervals coresponding to the probability mass function is repeated until all of the letters of the message have been encountered. To decode the message keep narrowing down the interval in the same manner until finally you have decoded the message.

13 **Arithmetic Coding Example**

Here is an illistration for the source $S = a_4, a_1, a_3, \ldots$ with the alphabet $A = a_1, a_2, a_3, a_4$.



14 Compare Huffman Coding and Arithmetic Coding

Huffman's original algorithm is optimal for a symbol-by-symbol coding (i.e. a stream of unrelated symbols) with a known input probability distribution, it is not optimal when the symbol-by-symbol restriction is dropped, or when the probability mass functions are unknown, not identically distributed, or not independent (e.g., "cat" is more common than "cta"). Each codeword length can have up to 1 bit of inefficientcy per symbol, thus there can be up to n bits of inefficientcy (where n is the length of the source message).

Arithmetic coding can often compress files further. By allowing the entire message to be encoded within a few bits of the entropy; so a greater compression rate maybe achieved than Huffman coding can provide. Also Arithmetic Coding is more flexible allowing the model to be separated from the code. The biggest down fall of Arithmetic Coding is that this process can be slow.

Both algorithms have varrious application and are still being used in many modern applications. Now let us turn our attention to the use of these algorithms to mine data.

15 Application of Data Compression Algorithms to the Process of Data Mining

In the remainder of this paper our attention will be directed to the application of data compression programs in the field of data mining. Data mining is a area of computer science that is rapidly becoming more and more important as the amount of data produced on a daily basis grows at an exponential rate. One technique for dealing with large data sets is clustering. Clustering refers to the process of dividing data into groups of similar objects. Each group, often called a cluster, consists of data that is similar to the data within the cluster and this data is dissimilar to data in other clusters. Often these clusters of data will be useful when trying to recognize patterns within the data that were previously hidden.

Data clustering when applied to data sets may encounter several complications ranging from very large databases that are difficult to deal with, to objects that have many attributes and attributes of many types. To deal with these complications many data mining algorithms require a lot of parameters. This can lead to some major problems if the settings are off, the algorithm may fail to recognize patterns or even recognize patterns that do not exist. Therefore data mining algorithms should use the least amount of parameters as possible, ideally none. Algorithms that are parameter-free are free from prejudices, expectations, and presumptions.

Data compression algorithms fit these requirements and are in fact parameter-free algorithms. This makes them superior to other data mining algorithms that require parameters. But can data compression algorithms accurately cluster data? Well we already know that they are really good at exploiting patterns within data so now we just need a way to cluster the data. If we can determine how similar the data is to one another we can divide the data into groups based on the similarities in the data. It turns out, that an entropy encoder can be used to measure the amount of similarity between streams of data. "By generating an entropy coder/compressor for each class of data; unknown data is then classified by feeding the uncompressed data to each compressor and seeing which compressor yields the highest compression. The coder with the best compression is probably the coder trained on the data that was most similar to the unknown data."

16 Normalized Compression Distance

The metric we will use for determining the similarities within the data and that will allow us to cluster the data is called the Normalized Compression Distance, NCD. The NCD is strongly based in the Kolmogorov theory of information complexity. Kolmogorov's theory tells us that the complexity of information can be measured by the number of symbols required to represent the information. Kolmogorov defined the Kolmogorov complexity $K(x)$ of the information x to be the length of the shortest binary computer program that describes the information. He also defined the Kolmogorov conditional complexity $K(x|y)$ to be the smallest program y that outputs the result x when processed through a universal turning machine.

When two objects have a realitivly small Kolmogorov conditional complexity it is an indication that the objects are similar. This similarity is called the Normalized Information Distance (NID) or the Universal Similarity Metric (USM).

$$NID(x, y) = maxK(x|y), K(y|x)/maxK(x), K(y)$$

However, the Kolmogorov complexity of an object and the Kolmogorov conditional complexity and the NID are not possible to compute. But we do know that the length of the shortest binary computer description of an object is the entropy of that object.

A compressor takes some input y and outputs x, and attempts to encode y in the least number of bits so that the length of x approximates the entropy of y. So it has been suggested the it is possible to approximate the Kolmogorov conditional complexity with the use of a compressor. Given that a compressor acts like a Turning Machine that processes an input y and outputs x, it can be used to approximate K(x—y) or K(x). Using a compressor C to calculate C(x) the NCD can be defined as

$$NCD(x, y) = (|C(xy)| - min|C(x)|, |C(y)|)/max|C(x)|, |C(y)|$$

where xy is the concatenation of the objects x and y.

The NCD should result in a value that is in the range from zero to one. Where a value that is close to zero indicates similarity between the objects x and y and a larger value indicates less similarity between the two objects. In practice, the NCD will be a positive value that is between zero and one plus epsilon, where epsilon is an upper bound created by the imperfections in the compression techniques. For most standard compression algorithms epsilon will be less than 0.1.

17 **Project**

The NCD can be used as a metric of similarity to help classify objects by clustering the more similar objects together. Using the NCD to classify data is a great idea but can the NCD actually classify objects? Several experiments have been done using this approach and have found it to be competitive or even superior to the state-of-the-art approaches in classification and clustering of DNA, text, and video datasets. I want to classify textual data and hopefully be able to achieve automatic authorship attribution.

Automatic authorship attribution is the automation of the process of determining the writer of a text. The technique can be applied when trying to detect plagiarism, verify authorship, profiling and characterizing authors, and detecting stylistic inconsistencies. Traditionally authorship attribution was not an automated process and the beginning attempts to automate the process were not fruitful. As data mining tools improved so did the ability to automate the authorship attribution process. Most of the methods for automating the process are parameter-free.

I want to apply lossless data compression methods to automate authorship attribution. Hopefully by using entropy encoding methods there will be enough similarity within the compressed text to be able to attribute an author. I wanted to use two different entropy encoding techniques to carry out the task and see how well they work compared to one another. I chose to use Huffman Algorithm and the Arithmetic Coding Algorithm to compress the text.

First I gathered downloaded 100 books from Project Gutenberg. Which is a resource for free online books. The books were written by 20 different authors that I randomly chose. Each author had to have written at least 5 text that were available through Project Gutenberg. I tried to select text that were roughly the same size because I was not sure if the text being of different sizes would have any effect on the NCD, and yet this task proved to be difficult and the size of the text varied significantly.

After selecting the text that I wanted to compress, I sanitized the data. By removing portions of the downloaded file that contained the licensing agreements from Project Gutenberg. I also removed the title, author, date, and some of the contents of the files to try to ensure that the results were not contaminated by this data. Then I named the files with the title of the text and an integer value that represented author of the given text. Then I separated the text into two groups the testing set and the paradigm set. For each of the 20 authors I randomly selected one of the five text from the author and put this text into the testing set the remaining four books were put into the paradigm set. When I was finished there were 20 books in the testing set and 80 books in the paradigm set.

Then I implemented my own versions of the Huffman and Arithmetic Coding algorithms using the Python programming language. I wrote the code in such a way that the program would compute the probabilities of the symbols based on the text at hand. After the probabilities of the symbols within the text had been computed then I encoded the text so in the respect I implemented static algorithms rather than choosing to use a dynamic approach.

I was not sure if the differing file length would produce good results. So, I limited the size of the text being compressed to 150,000 characters to ensure that the differing lengths of text did not effect my results. I also removed all symbols that were not ascii letters, spaces, new line characters, or punctuation from the text.

For each algorithm I implemented an encoding and decoding method and a method that analyzed the encoding procedure. In the actual experiment I only used the encoding method and the analysis method to save time because the decoding procedure can be very time consuming to run. The decoding method was merely used to ensure the encoding procedure worked correctly during the testing of the encoder. The analysis method wrote the results to a text file, and named the text file in a way that indicated the author or authors of the text and a symbol that indicated which algorithm had been used to compress the text.

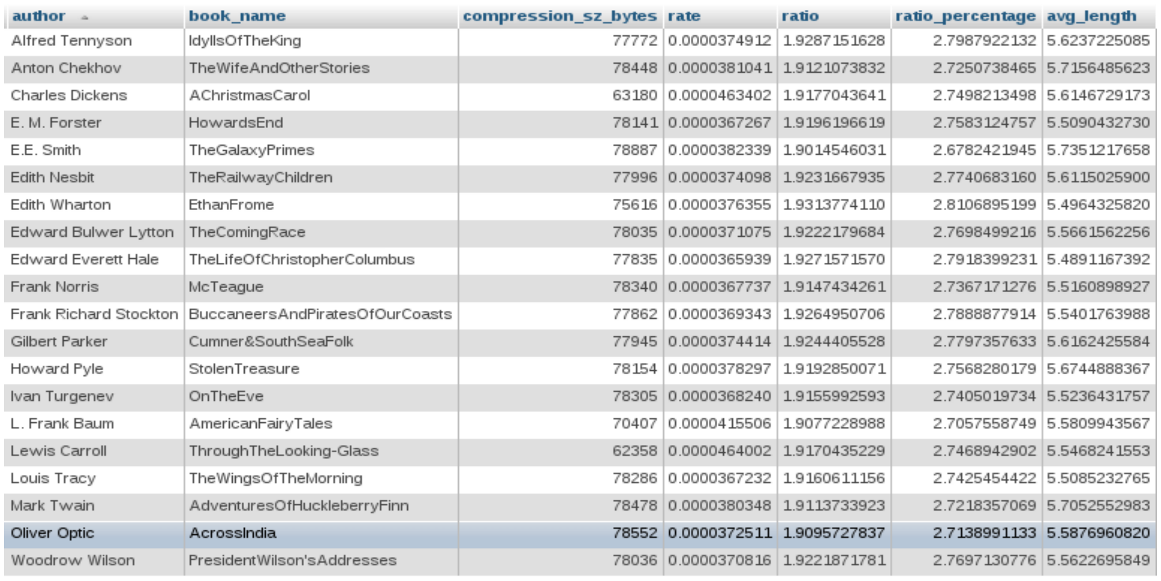
I compressed all of the books in both the testing and paradigm sets. Then I concatenated all of the books in the testing set with each of the books in the paradigm set and compressed the text that was the result of the concatenation. After compressing all of these text I wrote another program compute the NCD of the all of the books in the testing set with each of the books in the paradigm set. The program also stored the results in a database. Then the program tried to use the results to attribute each text in the testing set to an author from the paradigm set.

Then I wanted to know how different the results of the Huffman algorithm would have been had I compressed the full text rather than only a portion of the text. I repeated the experiment using the Huffman algorithm and the full text.

18 **Results**

Here is a list of the author's names that I wanted to attribute the text from the testing set to; along with the book's author; the size of the compressed text in bytes using the arithmetic coding algorthim (which was the classifier); the compression rate; compression ratio, the compression ratio as a percentage; and the average codeword length of the arithmetic coding algorthim (which was the classifier).

Looking at these text they seem very similar in all aspects considered.

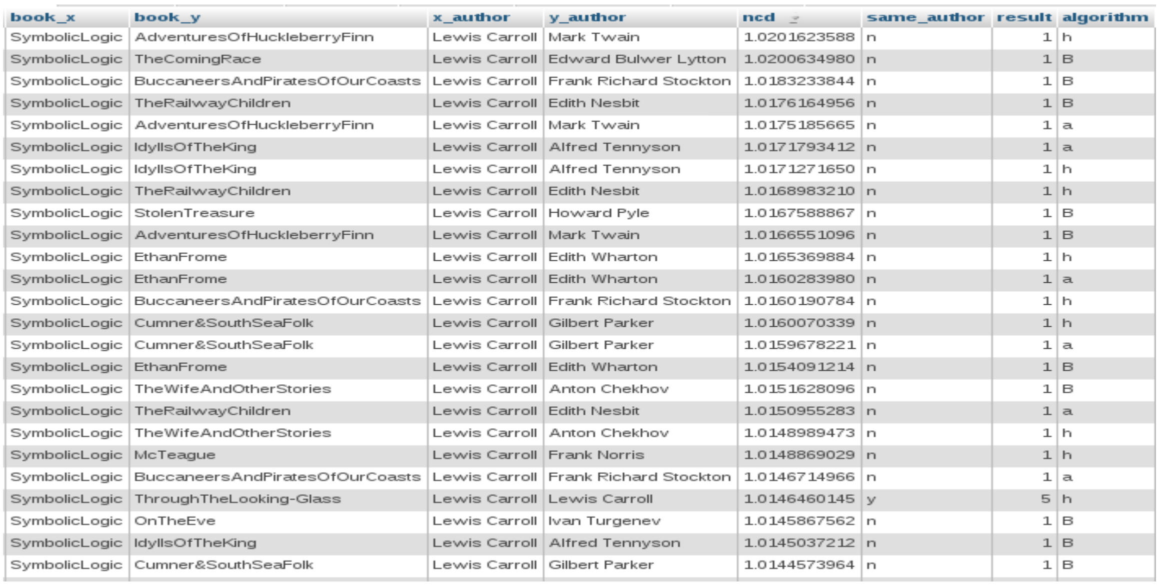


author	book_name	compression_sz_bytes	rate	ratio	ratio_percentage	avg_length
Alfred Tennyson	IdyllsOfTheKing	77772	0.0000374912	1.9287151628	2.7987922132	5.6237225085
Anton Chekhov	TheWifeAndOtherStories	78448	0.0000381041	1.9121073832	2.7250738465	5.7156485623
Charles Dickens	AChristmasCarol	63180	0.0000463402	1.9177043641	2.7498213498	5.6146729173
E. M. Forster	HowardsEnd	78141	0.0000367267	1.9196196619	2.7583124757	5.5090432730
E.E. Smith	TheGalaxyPrimes	78887	0.0000382339	1.9014546031	2.6782421945	5.7351217658
Edith Nesbit	TheRailwayChildren	77996	0.0000374098	1.9231667935	2.7740683160	5.6115025900
Edith Wharton	EthanFrome	75616	0.0000376355	1.9313774110	2.8106895199	5.4964325820
Edward Bulwer Lytton	TheComingRace	78035	0.0000371075	1.9222179684	2.7698499216	5.5661562256
Edward Everett Hale	TheLifeOfChristopherColumbus	77835	0.0000365939	1.9271571570	2.7918399231	5.4891167392
Frank Norris	McTeague	78340	0.0000367737	1.9147434261	2.7367171276	5.5160898927
Frank Richard Stockton	BuccaneersAndPiratesOfOurCoasts	77862	0.0000369343	1.9264950706	2.7888877914	5.5401763988
Gilbert Parker	Cumner&SouthSeaFolk	77945	0.0000374414	1.9244405528	2.7797357633	5.6162425584
Howard Pyle	StolenTreasure	78154	0.0000378297	1.9192850071	2.7568280179	5.6744888367
Ivan Turgenev	OnTheEve	78305	0.0000368240	1.9155992593	2.7405019734	5.5236431757
L. Frank Baum	AmericanFairyTales	70407	0.0000415506	1.9077228988	2.7057558749	5.5809943567
Lewis Carroll	ThroughTheLooking-Glass	62358	0.0000464002	1.9170435229	2.7468942902	5.5468241553
Louis Tracy	TheWingsOfTheMorning	78286	0.0000367232	1.9160611156	2.7425454422	5.5085232765
Mark Twain	AdventuresOfHuckleberryFinn	78478	0.0000380348	1.9113733923	2.7218357069	5.7052552983
Oliver Optic	AcrossIndia	78552	0.0000372511	1.9095727837	2.7138991133	5.5876960820
Woodrow Wilson	PresidentWilson'sAddresses	78036	0.0000370816	1.9221871781	2.7697130776	5.5622695849

I know that I would have a very hard time distingushing these books from one another, lets see how well the NCD does.

18 **Results**

I used the NCD in several way to try and classify the text. First, I looked at how dissimilar the text were overall, so I looked at the text with the highest NCD; the following is what I found.

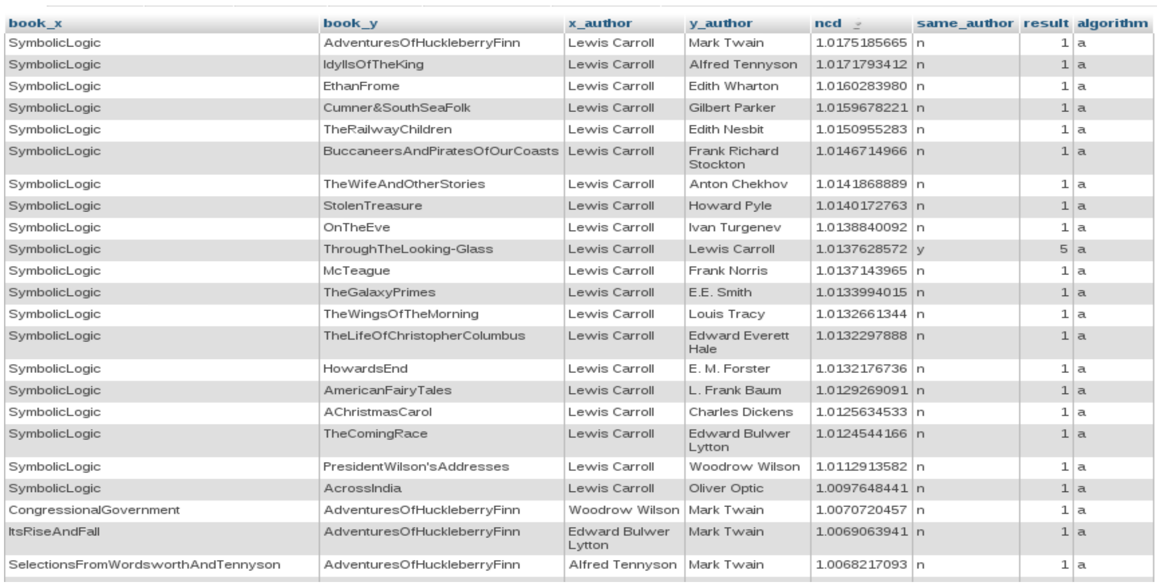


book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
SymbolicLogic	AdventuresOfHuckleberryFinn	Lewis Carroll	Mark Twain	1.0201623588	n	1	h
SymbolicLogic	TheComingRace	Lewis Carroll	Edward Bulwer Lytton	1.0200634980	n	1	B
SymbolicLogic	BuccaneersAndPiratesOfOurCoasts	Lewis Carroll	Frank Richard Stockton	1.0183233844	n	1	B
SymbolicLogic	TheRailwayChildren	Lewis Carroll	Edith Nesbit	1.0176164956	n	1	B
SymbolicLogic	AdventuresOfHuckleberryFinn	Lewis Carroll	Mark Twain	1.0175185665	n	1	a
SymbolicLogic	IdyllsOfTheKing	Lewis Carroll	Alfred Tennyson	1.0171793412	n	1	a
SymbolicLogic	IdyllsOfTheKing	Lewis Carroll	Alfred Tennyson	1.0171271650	n	1	h
SymbolicLogic	TheRailwayChildren	Lewis Carroll	Edith Nesbit	1.0168983210	n	1	h
SymbolicLogic	StolenTreasure	Lewis Carroll	Howard Pyle	1.0167588867	n	1	B
SymbolicLogic	AdventuresOfHuckleberryFinn	Lewis Carroll	Mark Twain	1.0166551096	n	1	B
SymbolicLogic	EthanFrome	Lewis Carroll	Edith Wharton	1.0165369884	n	1	h
SymbolicLogic	EthanFrome	Lewis Carroll	Edith Wharton	1.0160283980	n	1	a
SymbolicLogic	BuccaneersAndPiratesOfOurCoasts	Lewis Carroll	Frank Richard Stockton	1.0160190784	n	1	h
SymbolicLogic	Cumner&SouthSeaFolk	Lewis Carroll	Gilbert Parker	1.0160070339	n	1	h
SymbolicLogic	Cumner&SouthSeaFolk	Lewis Carroll	Gilbert Parker	1.0159678221	n	1	a
SymbolicLogic	EthanFrome	Lewis Carroll	Edith Wharton	1.0154091214	n	1	B
SymbolicLogic	TheWifeAndOtherStories	Lewis Carroll	Anton Chekhov	1.0151628096	n	1	B
SymbolicLogic	TheRailwayChildren	Lewis Carroll	Edith Nesbit	1.0150955283	n	1	a
SymbolicLogic	TheWifeAndOtherStories	Lewis Carroll	Anton Chekhov	1.0148989473	n	1	h
SymbolicLogic	McTeague	Lewis Carroll	Frank Norris	1.0148869029	n	1	h
SymbolicLogic	BuccaneersAndPiratesOfOurCoasts	Lewis Carroll	Frank Richard Stockton	1.0146714966	n	1	a
SymbolicLogic	ThroughTheLooking-Glass	Lewis Carroll	Lewis Carroll	1.0146460145	y	5	h
SymbolicLogic	OnTheEve	Lewis Carroll	Ivan Turgenev	1.0145867562	n	1	B
SymbolicLogic	IdyllsOfTheKing	Lewis Carroll	Alfred Tennyson	1.0145037212	n	1	B
SymbolicLogic	Cumner&SouthSeaFolk	Lewis Carroll	Gilbert Parker	1.0144573964	n	1	B

Surprisingly, the highest 25 NCD's where all over 1.0145. Only one of the predictions was wrong out of 25, which is a 96% of predicting that the text were dissimilar and most likely not written by the same author.

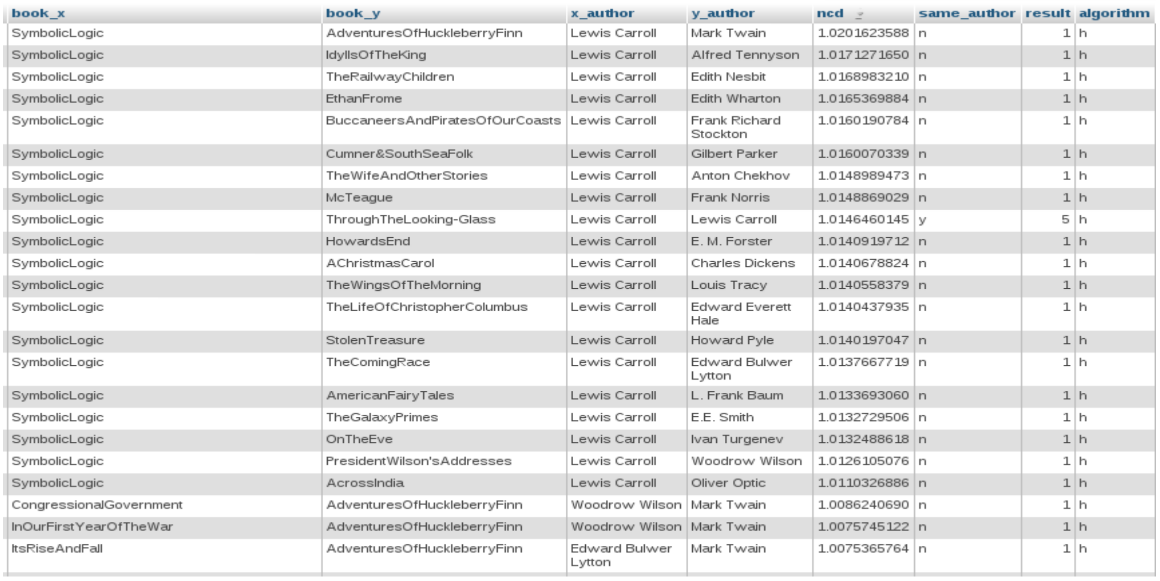
18 **Results**

Then I looked at how dissimilar the partial text compressed with the Arithmetic coding algorithm were, so I looked at the text with the highest NCD that were compressed with the Arithmetic coding algorithm.



book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
SymbolicLogic	AdventuresOfHuckleberryFinn	Lewis Carroll	Mark Twain	1.0175185665	n	1	a
SymbolicLogic	IdyllsOfTheKing	Lewis Carroll	Alfred Tennyson	1.0171793412	n	1	a
SymbolicLogic	EthanFrome	Lewis Carroll	Edith Wharton	1.0160283980	n	1	a
SymbolicLogic	Cumner&SouthSeaFolk	Lewis Carroll	Gilbert Parker	1.0159678221	n	1	a
SymbolicLogic	TheRailwayChildren	Lewis Carroll	Edith Nesbit	1.0150955283	n	1	a
SymbolicLogic	BuccaneersAndPiratesOfOurCoasts	Lewis Carroll	Frank Richard Stockton	1.0146714966	n	1	a
SymbolicLogic	TheWifeAndOtherStories	Lewis Carroll	Anton Chekhov	1.0141868889	n	1	a
SymbolicLogic	StolenTreasure	Lewis Carroll	Howard Pyle	1.0140172763	n	1	a
SymbolicLogic	OnTheEve	Lewis Carroll	Ivan Turgenev	1.0138840092	n	1	a
SymbolicLogic	ThroughTheLooking-Glass	Lewis Carroll	Lewis Carroll	1.0137628572	y	5	a
SymbolicLogic	McTeague	Lewis Carroll	Frank Norris	1.0137143965	n	1	a
SymbolicLogic	TheGalaxyPrimes	Lewis Carroll	E.E. Smith	1.0133994015	n	1	a
SymbolicLogic	TheWingsOfTheMorning	Lewis Carroll	Louis Tracy	1.0132661344	n	1	a
SymbolicLogic	TheLifeOfChristopherColumbus	Lewis Carroll	Edward Everett Hale	1.0132297888	n	1	a
SymbolicLogic	HowardsEnd	Lewis Carroll	E. M. Forster	1.0132176736	n	1	a
SymbolicLogic	AmericanFairyTales	Lewis Carroll	L. Frank Baum	1.0129269091	n	1	a
SymbolicLogic	AChristmasCarol	Lewis Carroll	Charles Dickens	1.0125634533	n	1	a
SymbolicLogic	TheComingRace	Lewis Carroll	Edward Bulwer Lytton	1.0124544166	n	1	a
SymbolicLogic	PresidentWilson'sAddresses	Lewis Carroll	Woodrow Wilson	1.0112913582	n	1	a
SymbolicLogic	AcrossIndia	Lewis Carroll	Oliver Optic	1.0097648441	n	1	a
CongressionalGovernment	AdventuresOfHuckleberryFinn	Woodrow Wilson	Mark Twain	1.0070720457	n	1	a
ItsRiseAndFall	AdventuresOfHuckleberryFinn	Edward Bulwer Lytton	Mark Twain	1.0069063941	n	1	a
SelectionsFromWordsworthAndTennyson	AdventuresOfHuckleberryFinn	Alfred Tennyson	Mark Twain	1.0068217093	n	1	a

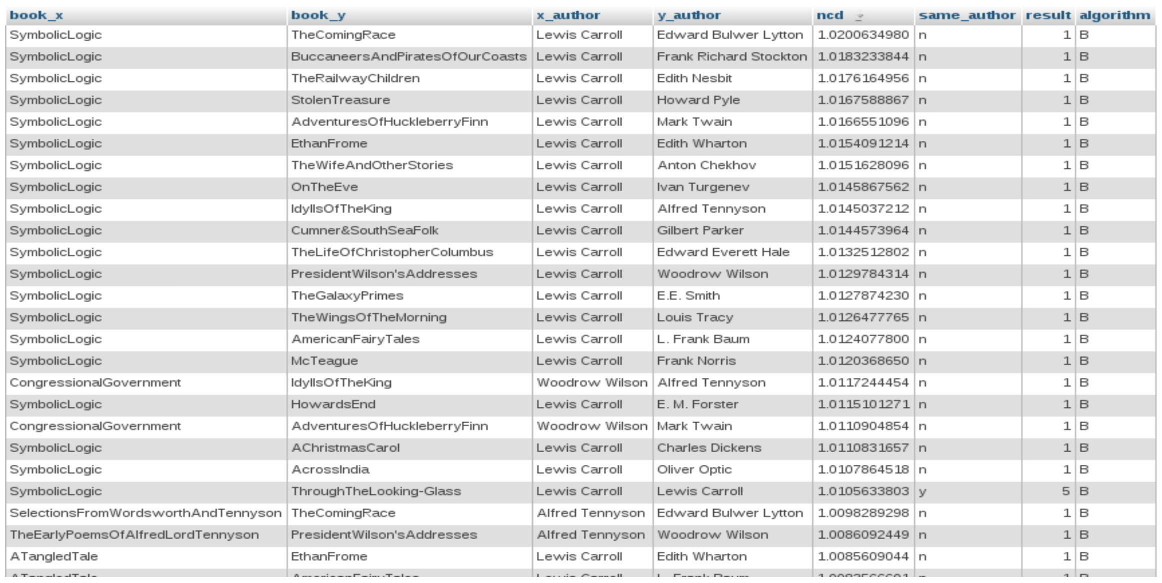
Then I looked at how dissimilar the partial text compressed with the Huffman coding algorithm were, so I looked at the text with the highest NCD that were compressed with the Huffman coding algorithm.



book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
SymbolicLogic	AdventuresOfHuckleberryFinn	Lewis Carroll	Mark Twain	1.0201623588	n	1	h
SymbolicLogic	IdyllsOfTheKing	Lewis Carroll	Alfred Tennyson	1.0171271650	n	1	h
SymbolicLogic	TheRailwayChildren	Lewis Carroll	Edith Nesbit	1.0168983210	n	1	h
SymbolicLogic	EthanFrome	Lewis Carroll	Edith Wharton	1.0165369884	n	1	h
SymbolicLogic	BuccaneersAndPiratesOfOurCoasts	Lewis Carroll	Frank Richard Stockton	1.0160190784	n	1	h
SymbolicLogic	Cumner&SouthSeaFolk	Lewis Carroll	Gilbert Parker	1.0160070339	n	1	h
SymbolicLogic	TheWifeAndOtherStories	Lewis Carroll	Anton Chekhov	1.0148989473	n	1	h
SymbolicLogic	McTeague	Lewis Carroll	Frank Norris	1.0148869029	n	1	h
SymbolicLogic	ThroughTheLooking-Glass	Lewis Carroll	Lewis Carroll	1.0146460145	y	5	h
SymbolicLogic	HowardsEnd	Lewis Carroll	E. M. Forster	1.0140919712	n	1	h
SymbolicLogic	AChristmasCarol	Lewis Carroll	Charles Dickens	1.0140678824	n	1	h
SymbolicLogic	TheWingsOfTheMorning	Lewis Carroll	Louis Tracy	1.0140558379	n	1	h
SymbolicLogic	TheLifeOfChristopherColumbus	Lewis Carroll	Edward Everett Hale	1.0140437935	n	1	h
SymbolicLogic	StolenTreasure	Lewis Carroll	Howard Pyle	1.0140197047	n	1	h
SymbolicLogic	TheComingRace	Lewis Carroll	Edward Bulwer Lytton	1.0137667719	n	1	h
SymbolicLogic	AmericanFairyTales	Lewis Carroll	L. Frank Baum	1.0133693060	n	1	h
SymbolicLogic	TheGalaxyPrimes	Lewis Carroll	E.E. Smith	1.0132729506	n	1	h
SymbolicLogic	OnTheEve	Lewis Carroll	Ivan Turgenev	1.0132488618	n	1	h
SymbolicLogic	PresidentWilson'sAddresses	Lewis Carroll	Woodrow Wilson	1.0126105076	n	1	h
SymbolicLogic	AcrossIndia	Lewis Carroll	Oliver Optic	1.0110326886	n	1	h
CongressionalGovernment	AdventuresOfHuckleberryFinn	Woodrow Wilson	Mark Twain	1.0086240690	n	1	h
InOurFirstYearOfTheWar	AdventuresOfHuckleberryFinn	Woodrow Wilson	Mark Twain	1.0075745122	n	1	h
ItsRiseAndFall	AdventuresOfHuckleberryFinn	Edward Bulwer Lytton	Mark Twain	1.0075365764	n	1	h

18 **Results**

Then I looked at how dissimilar the full text compressed with the Huffman coding algorithm were, so I looked at the text with the highest NCD that were compressed with the Huffman coding algorithm.

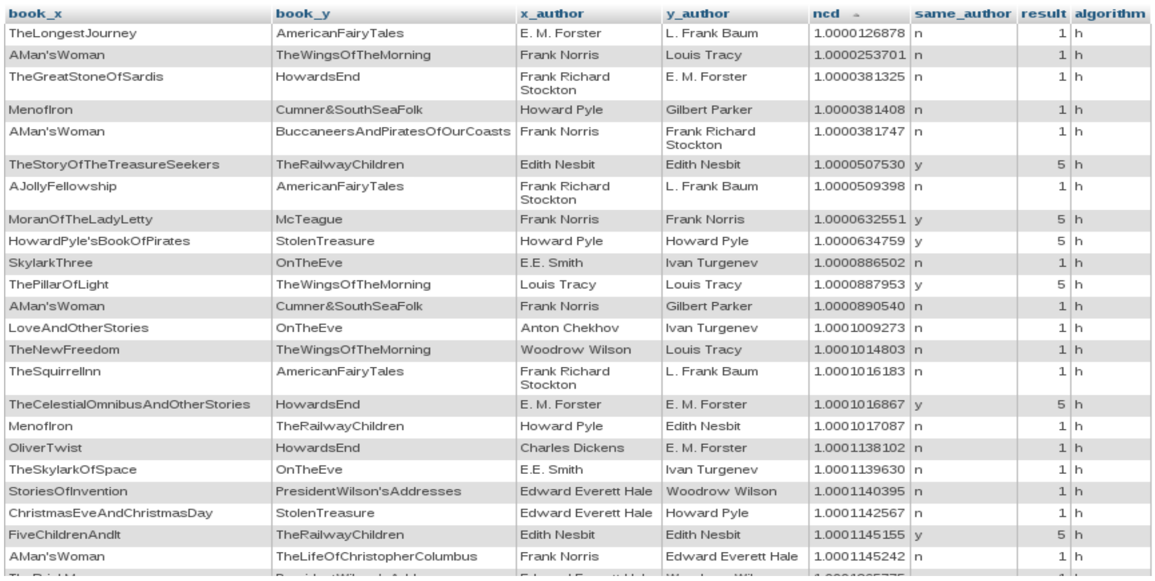


book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
SymbolicLogic	TheComingRace	Lewis Carroll	Edward Bulwer Lytton	1.0200634980	n	1	B
SymbolicLogic	BuccaneersAndPiratesOfOurCoasts	Lewis Carroll	Frank Richard Stockton	1.0183233844	n	1	B
SymbolicLogic	TheRailwayChildren	Lewis Carroll	Edith Nesbit	1.0176164956	n	1	B
SymbolicLogic	StolenTreasure	Lewis Carroll	Howard Pyle	1.0167588867	n	1	B
SymbolicLogic	AdventuresOfHuckleberryFinn	Lewis Carroll	Mark Twain	1.0166551096	n	1	B
SymbolicLogic	EthanFrome	Lewis Carroll	Edith Wharton	1.0154091214	n	1	B
SymbolicLogic	TheWifeAndOtherStories	Lewis Carroll	Anton Chekhov	1.0151628096	n	1	B
SymbolicLogic	OnTheEve	Lewis Carroll	Ivan Turgenev	1.0145867562	n	1	B
SymbolicLogic	IdyllsOfTheKing	Lewis Carroll	Alfred Tennyson	1.0145037212	n	1	B
SymbolicLogic	Cumner&SouthSeaFolk	Lewis Carroll	Gilbert Parker	1.0144573964	n	1	B
SymbolicLogic	TheLifeOfChristopherColumbus	Lewis Carroll	Edward Everett Hale	1.0132512802	n	1	B
SymbolicLogic	PresidentWilson'sAddresses	Lewis Carroll	Woodrow Wilson	1.0129784314	n	1	B
SymbolicLogic	TheGalaxyPrimes	Lewis Carroll	E.E. Smith	1.0127874230	n	1	B
SymbolicLogic	TheWingsOfTheMorning	Lewis Carroll	Louis Tracy	1.0126477765	n	1	B
SymbolicLogic	AmericanFairyTales	Lewis Carroll	L. Frank Baum	1.0124077800	n	1	B
SymbolicLogic	McTeague	Lewis Carroll	Frank Norris	1.0120368650	n	1	B
CongressionalGovernment	IdyllsOfTheKing	Woodrow Wilson	Alfred Tennyson	1.0117244454	n	1	B
SymbolicLogic	HowardsEnd	Lewis Carroll	E. M. Forster	1.0115101271	n	1	B
CongressionalGovernment	AdventuresOfHuckleberryFinn	Woodrow Wilson	Mark Twain	1.0110904854	n	1	B
SymbolicLogic	AChristmasCarol	Lewis Carroll	Charles Dickens	1.0110831657	n	1	B
SymbolicLogic	AcrossIndia	Lewis Carroll	Oliver Optic	1.0107864518	n	1	B
SymbolicLogic	ThroughTheLooking-Glass	Lewis Carroll	Lewis Carroll	1.0105633803	y	5	B
SelectionsFromWordsworthAndTennyson	TheComingRace	Alfred Tennyson	Edward Bulwer Lytton	1.0098289298	n	1	B
TheEarlyPoemsOfAlfredLordTennyson	PresidentWilson'sAddresses	Alfred Tennyson	Woodrow Wilson	1.0086092449	n	1	B
ATangledTale	EthanFrome	Lewis Carroll	Edith Wharton	1.0085609044	n	1	B

Comparing the three algorithms they all did a good job reaching a 96% success rate, the Huffman that compressed the full text only did slightly better at ranking the wrong prediction.

18 **Results**

Next used the NCD to try see how similar the text were overall, so I looked at the text with the lowest NCD; the following is what I found.

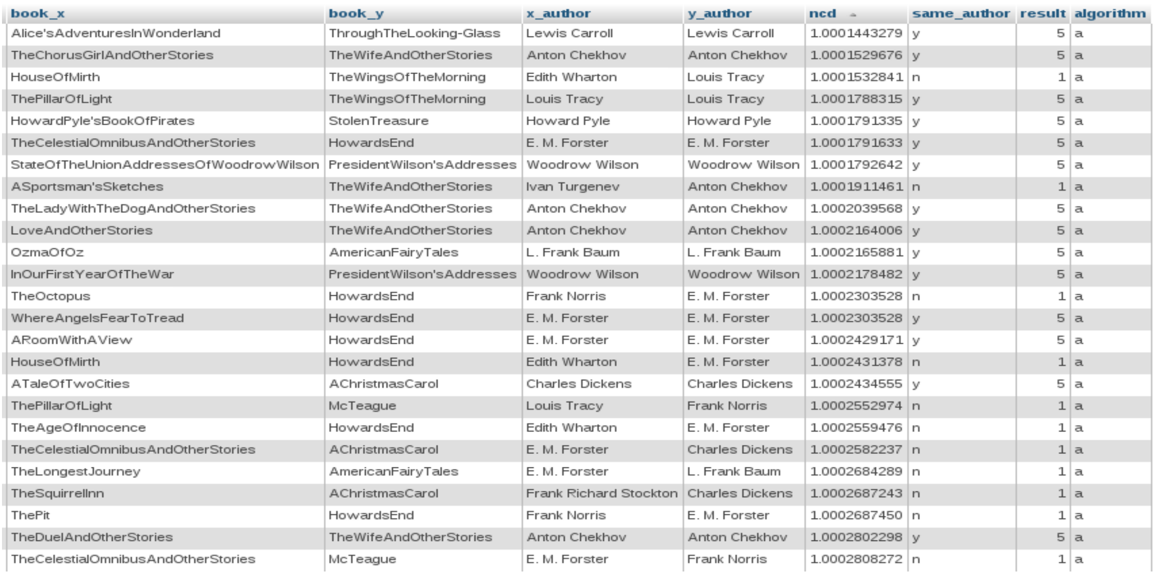


book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
TheLongestJourney	AmericanFairyTales	E. M. Forster	L. Frank Baum	1.0000126878	n	1	h
AMan'sWoman	TheWingsOfTheMorning	Frank Norris	Louis Tracy	1.0000253701	n	1	h
TheGreatStoneOfSardis	HowardsEnd	Frank Richard Stockton	E. M. Forster	1.0000381325	n	1	h
MenofIron	Cumner&SouthSeaFolk	Howard Pyle	Gilbert Parker	1.0000381408	n	1	h
AMan'sWoman	BuccaneersAndPiratesOfOurCoasts	Frank Norris	Frank Richard Stockton	1.0000381747	n	1	h
TheStoryOfTheTreasureSeekers	TheRailwayChildren	Edith Nesbit	Edith Nesbit	1.0000507530	y	5	h
AJollyFellowship	AmericanFairyTales	Frank Richard Stockton	L. Frank Baum	1.0000509398	n	1	h
MoranOfTheLadyLetty	McTeague	Frank Norris	Frank Norris	1.0000632551	y	5	h
HowardPyle'sBookOfPirates	StolenTreasure	Howard Pyle	Howard Pyle	1.0000634759	y	5	h
SkylarkThree	OnTheEve	E.E. Smith	Ivan Turgenev	1.0000886502	n	1	h
ThePillarOfLight	TheWingsOfTheMorning	Louis Tracy	Louis Tracy	1.0000887953	y	5	h
AMan'sWoman	Cumner&SouthSeaFolk	Frank Norris	Gilbert Parker	1.0000890540	n	1	h
LoveAndOtherStories	OnTheEve	Anton Chekhov	Ivan Turgenev	1.0001009273	n	1	h
TheNewFreedom	TheWingsOfTheMorning	Woodrow Wilson	Louis Tracy	1.0001014803	n	1	h
TheSquirrelInn	AmericanFairyTales	Frank Richard Stockton	L. Frank Baum	1.0001016183	n	1	h
TheCelestialOmnibusAndOtherStories	HowardsEnd	E. M. Forster	E. M. Forster	1.0001016867	y	5	h
MenofIron	TheRailwayChildren	Howard Pyle	Edith Nesbit	1.0001017087	n	1	h
OliverTwist	HowardsEnd	Charles Dickens	E. M. Forster	1.0001138102	n	1	h
TheSkylarkOfSpace	OnTheEve	E.E. Smith	Ivan Turgenev	1.0001139630	n	1	h
StoriesOfInvention	PresidentWilson'sAddresses	Edward Everett Hale	Woodrow Wilson	1.0001140395	n	1	h
ChristmasEveAndChristmasDay	StolenTreasure	Edward Everett Hale	Howard Pyle	1.0001142567	n	1	h
FiveChildrenAndIt	TheRailwayChildren	Edith Nesbit	Edith Nesbit	1.0001145155	y	5	h
AMan'sWoman	TheLifeOfChristopherColumbus	Frank Norris	Edward Everett Hale	1.0001145242	n	1	h

I was shocked all the NCD's were over 1.00001267 I was expecting that the NCD would be close to zero when the text were similar and all of them were over 1. So got worried that possibly the NCD was not a good indicator and I could see that the lowest NCD alone was not a very good indicator.

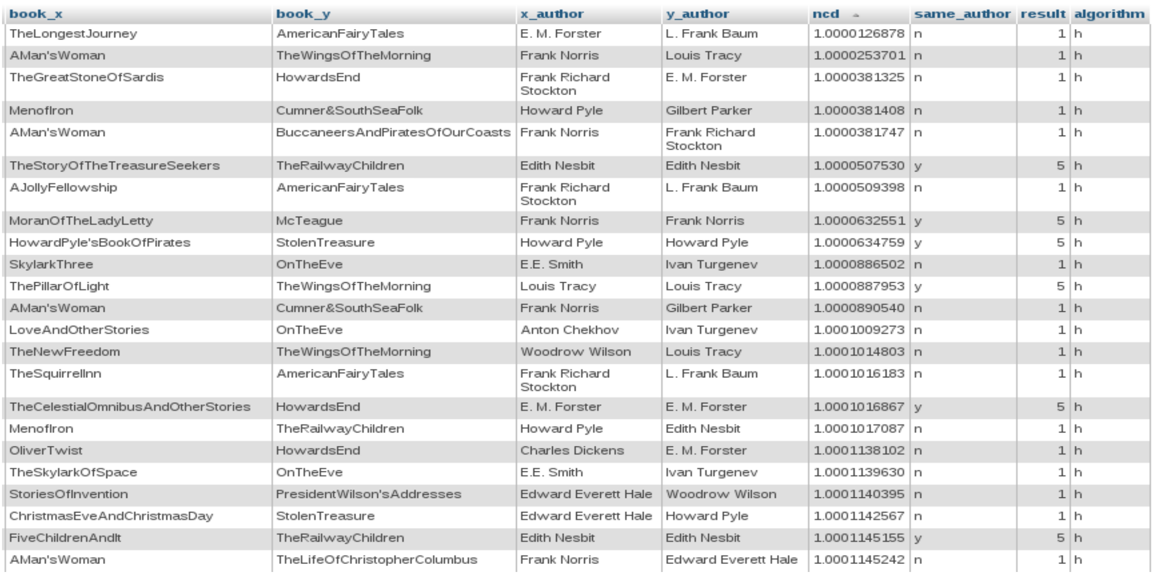
18 Results

I looked at the partial text compressed Arithmetic coding algorithm with the lowest NCD the following is what I found.



book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
Alice'sAdventuresInWonderland	ThroughTheLooking-Glass	Lewis Carroll	Lewis Carroll	1.0001443279	y	5	a
TheChorusGirlAndOtherStories	TheWifeAndOtherStories	Anton Chekhov	Anton Chekhov	1.0001529676	y	5	a
HouseOfMirth	TheWingsOfTheMorning	Edith Wharton	Louis Tracy	1.0001532841	n	1	a
ThePillarOfLight	TheWingsOfTheMorning	Louis Tracy	Louis Tracy	1.0001788315	y	5	a
HowardPyle'sBookOfPirates	StolenTreasure	Howard Pyle	Howard Pyle	1.0001791335	y	5	a
TheCelestialOmnibusAndOtherStories	HowardsEnd	E. M. Forster	E. M. Forster	1.0001791633	y	5	a
StateOfTheUnionAddressesOfWoodrowWilson	PresidentWilson'sAddresses	Woodrow Wilson	Woodrow Wilson	1.0001792642	y	5	a
ASportsman'sSketches	TheWifeAndOtherStories	Ivan Turgenev	Anton Chekhov	1.0001911461	n	1	a
TheLadyWithTheDogAndOtherStories	TheWifeAndOtherStories	Anton Chekhov	Anton Chekhov	1.0002039568	y	5	a
LoveAndOtherStories	TheWifeAndOtherStories	Anton Chekhov	Anton Chekhov	1.0002164006	y	5	a
OzmaOfOz	AmericanFairyTales	L. Frank Baum	L. Frank Baum	1.0002165881	y	5	a
InOurFirstYearOfTheWar	PresidentWilson'sAddresses	Woodrow Wilson	Woodrow Wilson	1.0002178482	y	5	a
TheOctopus	HowardsEnd	Frank Norris	E. M. Forster	1.0002303528	n	1	a
WhereAngelsFearToTread	HowardsEnd	E. M. Forster	E. M. Forster	1.0002303528	y	5	a
ARoomWithAView	HowardsEnd	E. M. Forster	E. M. Forster	1.0002429171	y	5	a
HouseOfMirth	HowardsEnd	Edith Wharton	E. M. Forster	1.0002431378	n	1	a
ATaleOfTwoCities	AChristmasCarol	Charles Dickens	Charles Dickens	1.0002434555	y	5	a
ThePillarOfLight	McTeague	Louis Tracy	Frank Norris	1.0002552974	n	1	a
TheAgeOfInnocence	HowardsEnd	Edith Wharton	E. M. Forster	1.0002559476	n	1	a
TheCelestialOmnibusAndOtherStories	AChristmasCarol	E. M. Forster	Charles Dickens	1.0002582237	n	1	a
TheLongestJourney	AmericanFairyTales	E. M. Forster	L. Frank Baum	1.0002684289	n	1	a
TheSquirrelInn	AChristmasCarol	Frank Richard Stockton	Charles Dickens	1.0002687243	n	1	a
ThePit	HowardsEnd	Frank Norris	E. M. Forster	1.0002687450	n	1	a
TheDuelAndOtherStories	TheWifeAndOtherStories	Anton Chekhov	Anton Chekhov	1.0002802298	y	5	a
TheCelestialOmnibusAndOtherStories	McTeague	E. M. Forster	Frank Norris	1.0002808272	n	1	a

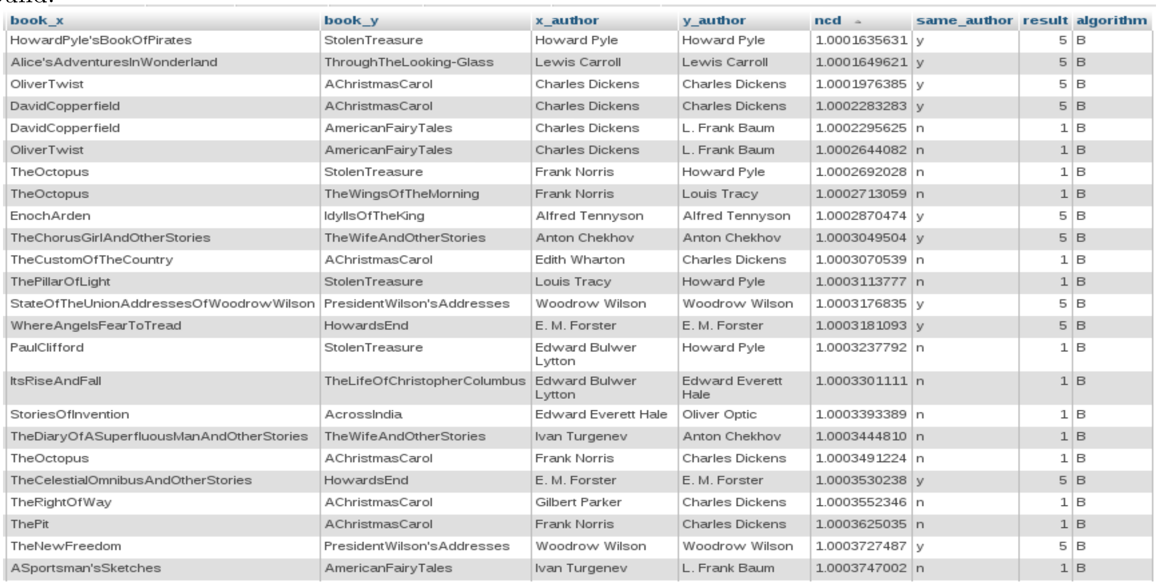
I looked at the partial text compressed Huffman coding algorithm with the lowest NCD the following is what I found.



book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
TheLongestJourney	AmericanFairyTales	E. M. Forster	L. Frank Baum	1.0000126878	n	1	h
AMan'sWoman	TheWingsOfTheMorning	Frank Norris	Louis Tracy	1.0000253701	n	1	h
TheGreatStoneOfSardis	HowardsEnd	Frank Richard Stockton	E. M. Forster	1.0000381325	n	1	h
MenofIron	Cumner&SouthSeaFolk	Howard Pyle	Gilbert Parker	1.0000381408	n	1	h
AMan'sWoman	BuccaneersAndPiratesOfOurCoasts	Frank Norris	Frank Richard Stockton	1.0000381747	n	1	h
TheStoryOfTheTreasureSeekers	TheRailwayChildren	Edith Nesbit	Edith Nesbit	1.0000507530	y	5	h
AJollyFellowship	AmericanFairyTales	Frank Richard Stockton	L. Frank Baum	1.0000509398	n	1	h
MoranOfTheLadyLetty	McTeague	Frank Norris	Frank Norris	1.0000632551	y	5	h
HowardPyle'sBookOfPirates	StolenTreasure	Howard Pyle	Howard Pyle	1.0000634759	y	5	h
SkylarkThree	OnTheEve	E.E. Smith	Ivan Turgenev	1.0000886502	n	1	h
ThePillarOfLight	TheWingsOfTheMorning	Louis Tracy	Louis Tracy	1.0000887953	y	5	h
AMan'sWoman	Cumner&SouthSeaFolk	Frank Norris	Gilbert Parker	1.0000890540	n	1	h
LoveAndOtherStories	OnTheEve	Anton Chekhov	Ivan Turgenev	1.0001009273	n	1	h
TheNewFreedom	TheWingsOfTheMorning	Woodrow Wilson	Louis Tracy	1.0001014803	n	1	h
TheSquirrelInn	AmericanFairyTales	Frank Richard Stockton	L. Frank Baum	1.0001016183	n	1	h
TheCelestialOmnibusAndOtherStories	HowardsEnd	E. M. Forster	E. M. Forster	1.0001016867	y	5	h
MenofIron	TheRailwayChildren	Howard Pyle	Edith Nesbit	1.0001017087	n	1	h
OliverTwist	HowardsEnd	Charles Dickens	E. M. Forster	1.0001138102	n	1	h
TheSkylarkOfSpace	OnTheEve	E.E. Smith	Ivan Turgenev	1.0001139630	n	1	h
StoriesOfInvention	PresidentWilson'sAddresses	Edward Everett Hale	Woodrow Wilson	1.0001140395	n	1	h
ChristmasEveAndChristmasDay	StolenTreasure	Edward Everett Hale	Howard Pyle	1.0001142567	n	1	h
FiveChildrenAndIt	TheRailwayChildren	Edith Nesbit	Edith Nesbit	1.0001145155	y	5	h
AMan'sWoman	TheLifeOfChristopherColumbus	Frank Norris	Edward Everett Hale	1.0001145242	n	1	h

18 **Results**

I looked at the full text compressed Huffman coding algorithm with the lowest NCD the following is what I found.



book_x	book_y	x_author	y_author	ncd	same_author	result	algorithm
HowardPyle'sBookOfPirates	StolenTreasure	Howard Pyle	Howard Pyle	1.0001635631	y	5	B
Alice'sAdventuresInWonderland	ThroughTheLooking-Glass	Lewis Carroll	Lewis Carroll	1.0001649621	y	5	B
OliverTwist	AChristmasCarol	Charles Dickens	Charles Dickens	1.0001976385	y	5	B
DavidCopperfield	AChristmasCarol	Charles Dickens	Charles Dickens	1.0002283283	y	5	B
DavidCopperfield	AmericanFairyTales	Charles Dickens	L. Frank Baum	1.0002295625	n	1	B
OliverTwist	AmericanFairyTales	Charles Dickens	L. Frank Baum	1.0002644082	n	1	B
TheOctopus	StolenTreasure	Frank Norris	Howard Pyle	1.0002692028	n	1	B
TheOctopus	TheWingsOfTheMorning	Frank Norris	Louis Tracy	1.0002713059	n	1	B
EnochArden	IdyllsOfTheKing	Alfred Tennyson	Alfred Tennyson	1.0002870474	y	5	B
TheChorusGirlAndOtherStories	TheWifeAndOtherStories	Anton Chekhov	Anton Chekhov	1.0003049504	y	5	B
TheCustomOfTheCountry	AChristmasCarol	Edith Wharton	Charles Dickens	1.0003070539	n	1	B
ThePillarOfLight	StolenTreasure	Louis Tracy	Howard Pyle	1.0003113777	n	1	B
StateOfTheUnionAddressesOfWoodrowWilson	PresidentWilson'sAddresses	Woodrow Wilson	Woodrow Wilson	1.0003176835	y	5	B
WhereAngelsFearToTread	HowardsEnd	E. M. Forster	E. M. Forster	1.0003181093	y	5	B
PaulClifford	StolenTreasure	Edward Bulwer Lytton	Howard Pyle	1.0003237792	n	1	B
ItsRiseAndFall	TheLifeOfChristopherColumbus	Edward Bulwer Lytton	Edward Everett Hale	1.0003301111	n	1	B
StoriesOfInvention	AcrossIndia	Edward Everett Hale	Oliver Optic	1.0003393389	n	1	B
TheDiaryOfASuperfluousManAndOtherStories	TheWifeAndOtherStories	Ivan Turgenev	Anton Chekhov	1.0003444810	n	1	B
TheOctopus	AChristmasCarol	Frank Norris	Charles Dickens	1.0003491224	n	1	B
TheCelestialOmnibusAndOtherStories	HowardsEnd	E. M. Forster	E. M. Forster	1.0003530238	y	5	B
TheRightOfWay	AChristmasCarol	Gilbert Parker	Charles Dickens	1.0003552346	n	1	B
ThePit	AChristmasCarol	Frank Norris	Charles Dickens	1.0003625035	n	1	B
TheNewFreedom	PresidentWilson'sAddresses	Woodrow Wilson	Woodrow Wilson	1.0003727487	y	5	B
ASportsman'sSketches	AmericanFairyTales	Ivan Turgenev	L. Frank Baum	1.0003747002	n	1	B

I can clearly see that the Arithmetic Coding algorithm does a better job at detecting similarities when looking at the overall lowest NCDs for all authors.

18 **Results**

This left me wondering how similar these compressed text were when looking at the avgerage predictions for a given author. So I looked at this data in several ways.
For a given author I considered the following:

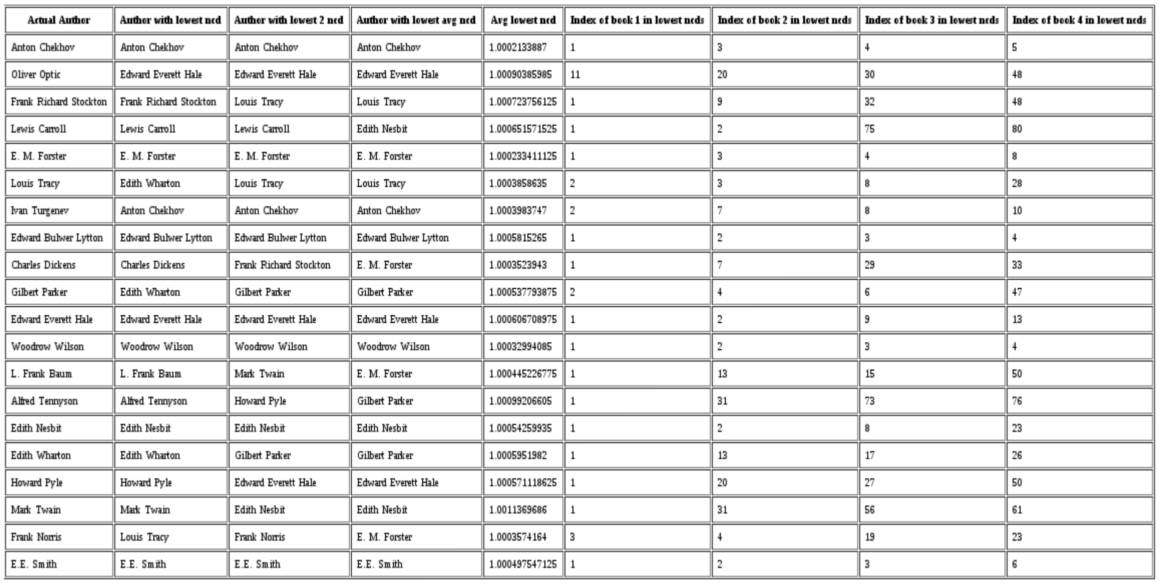
1. I considered the position of the text from the paradigm set with the same author in the list of lowest NCDs, there were four text for each author so four index positions to consider.

2. I considered the author with the overall lowest NCD, I called this Author with lowest ncd.

3. I considered the author that had the two lowest NCD's, I called this Author with lowest 2 ncd.

4. I considered the author that had the four lowest NCD's, I called this Avg lowest NCD.

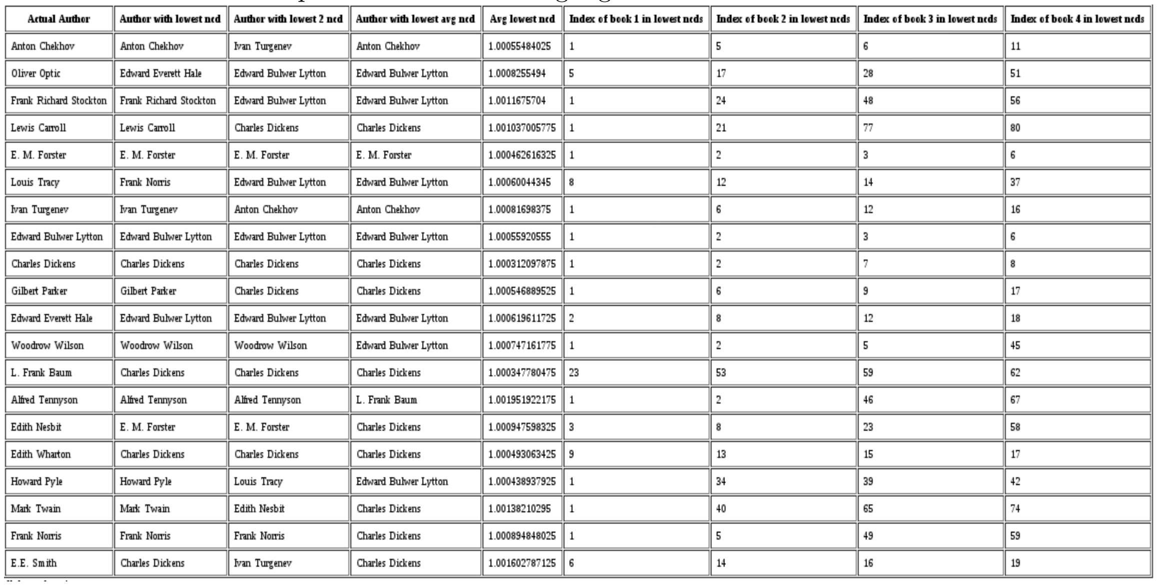
This is the result for the partial Arthimetic coding algorithm:



Actual Author	Author with lowest ncd	Author with lowest 2 ncd	Author with lowest avg ncd	Avg lowest ncd	Index of book 1 in lowest ncds	Index of book 2 in lowest ncds	Index of book 3 in lowest ncds	Index of book 4 in lowest ncds
Anton Chekhov	Anton Chekhov	Anton Chekhov	Anton Chekhov	1.0002133887	1	3	4	5
Oliver Optic	Edward Everett Hale	Edward Everett Hale	Edward Everett Hale	1.00090385985	11	20	30	48
Frank Richard Stockton	Frank Richard Stockton	Louis Tracy	Louis Tracy	1.000723756125	1	9	32	48
Lewis Carroll	Lewis Carroll	Lewis Carroll	Edith Nesbit	1.000651571525	1	2	75	80
E. M. Forster	E. M. Forster	E. M. Forster	E. M. Forster	1.000233411125	1	3	4	8
Louis Tracy	Edith Wharton	Louis Tracy	Louis Tracy	1.0003858635	2	3	8	28
Ivan Turgenev	Anton Chekhov	Anton Chekhov	Anton Chekhov	1.0003983747	2	7	8	10
Edward Bulwer Lytton	Edward Bulwer Lytton	Edward Bulwer Lytton	Edward Bulwer Lytton	1.0005815265	1	2	3	4
Charles Dickens	Charles Dickens	Frank Richard Stockton	E. M. Forster	1.0003523943	1	7	29	33
Gilbert Parker	Edith Wharton	Gilbert Parker	Gilbert Parker	1.000537793875	2	4	6	47
Edward Everett Hale	Edward Everett Hale	Edward Everett Hale	Edward Everett Hale	1.000606708975	1	2	9	13
Woodrow Wilson	Woodrow Wilson	Woodrow Wilson	Woodrow Wilson	1.00032994085	1	2	3	4
L. Frank Baum	L. Frank Baum	Mark Twain	E. M. Forster	1.000445226775	1	13	15	50
Alfred Tennyson	Alfred Tennyson	Howard Pyle	Gilbert Parker	1.00099206605	1	31	73	76
Edith Nesbit	Edith Nesbit	Edith Nesbit	Edith Nesbit	1.00054259935	1	2	8	23
Edith Wharton	Edith Wharton	Gilbert Parker	Gilbert Parker	1.0005951982	1	13	17	26
Howard Pyle	Howard Pyle	Edward Everett Hale	Edward Everett Hale	1.000571118625	1	20	27	50
Mark Twain	Mark Twain	Edith Nesbit	Edith Nesbit	1.0011369686	1	31	56	61
Frank Norris	Louis Tracy	Frank Norris	E. M. Forster	1.0003574164	3	4	19	23
E.E. Smith	E.E. Smith	E.E. Smith	E.E. Smith	1.000497547125	1	2	3	6

18 Results

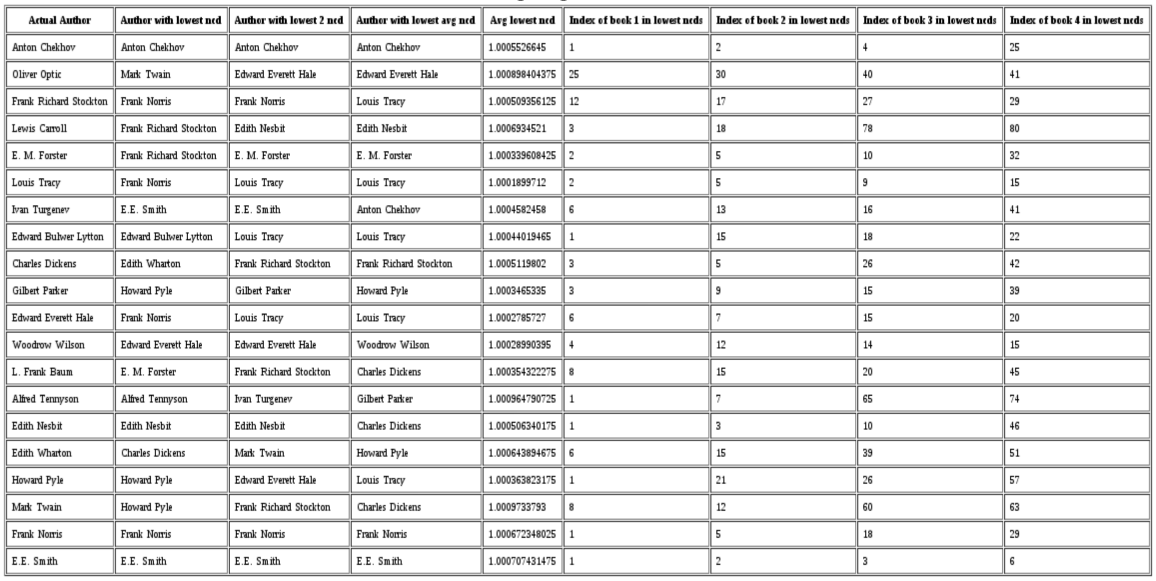
This is the result for the partial Huffman coding algorithm:



Actual Author	Author with lowest ncd	Author with lowest 2 ncd	Author with lowest avg ncd	Avg lowest ncd	Index of book 1 in lowest ncds	Index of book 2 in lowest ncds	Index of book 3 in lowest ncds	Index of book 4 in lowest ncds
Anton Chekhov	Anton Chekhov	Ivan Turgenev	Anton Chekhov	1.00055484025	1	5	6	11
Oliver Optic	Edward Everett Hale	Edward Bulwer Lytton	Edward Bulwer Lytton	1.0008255494	5	17	28	51
Frank Richard Stockton	Frank Richard Stockton	Edward Bulwer Lytton	Edward Bulwer Lytton	1.0011675704	1	24	48	56
Lewis Carroll	Lewis Carroll	Charles Dickens	Charles Dickens	1.001037005775	1	21	77	80
E. M. Forster	E. M. Forster	E. M. Forster	E. M. Forster	1.000462616325	1	2	3	6
Louis Tracy	Frank Norris	Edward Bulwer Lytton	Edward Bulwer Lytton	1.00060044345	8	12	14	37
Ivan Turgenev	Ivan Turgenev	Anton Chekhov	Anton Chekhov	1.00081698375	1	6	12	16
Edward Bulwer Lytton	Edward Bulwer Lytton	Edward Bulwer Lytton	Edward Bulwer Lytton	1.00055920555	1	2	3	6
Charles Dickens	Charles Dickens	Charles Dickens	Charles Dickens	1.000312097875	1	2	7	8
Gilbert Parker	Gilbert Parker	Charles Dickens	Charles Dickens	1.000546889525	1	6	9	17
Edward Everett Hale	Edward Bulwer Lytton	Edward Bulwer Lytton	Edward Bulwer Lytton	1.000619611725	2	8	12	18
Woodrow Wilson	Woodrow Wilson	Woodrow Wilson	Edward Bulwer Lytton	1.000747161775	1	2	5	45
L. Frank Baum	Charles Dickens	Charles Dickens	Charles Dickens	1.000347780475	23	53	59	62
Alfred Tennyson	Alfred Tennyson	Alfred Tennyson	L. Frank Baum	1.001951922175	1	2	46	67
Edith Nesbit	E. M. Forster	E. M. Forster	Charles Dickens	1.000947598325	3	8	23	58
Edith Wharton	Charles Dickens	Charles Dickens	Charles Dickens	1.000493063425	9	13	15	17
Howard Pyle	Howard Pyle	Louis Tracy	Edward Bulwer Lytton	1.000438937925	1	34	39	42
Mark Twain	Mark Twain	Edith Nesbit	Charles Dickens	1.00138210295	1	40	65	74
Frank Norris	Frank Norris	Frank Norris	Charles Dickens	1.000894848025	1	5	49	59
E.E. Smith	Charles Dickens	Ivan Turgenev	Charles Dickens	1.001602787125	6	14	16	19

18 **Results**

This is the result for the full Huffman coding algorithm:



Actual Author	Author with lowest ncd	Author with lowest 2 ncd	Author with lowest avg ncd	Avg lowest ncd	Index of book 1 in lowest ncds	Index of book 2 in lowest ncds	Index of book 3 in lowest ncds	Index of book 4 in lowest ncds
Anton Chekhov	Anton Chekhov	Anton Chekhov	Anton Chekhov	1.0005526645	1	2	4	25
Oliver Optic	Mark Twain	Edward Everett Hale	Edward Everett Hale	1.000898404375	25	30	40	41
Frank Richard Stockton	Frank Norris	Frank Norris	Louis Tracy	1.000509356125	12	17	27	29
Lewis Carroll	Frank Richard Stockton	Edith Nesbit	Edith Nesbit	1.0006934521	3	18	78	80
E. M. Forster	Frank Richard Stockton	E. M. Forster	E. M. Forster	1.000339608425	2	5	10	32
Louis Tracy	Frank Norris	Louis Tracy	Louis Tracy	1.0001899712	2	5	9	15
Ivan Turgenev	E.E. Smith	E.E. Smith	Anton Chekhov	1.0004582458	6	13	16	41
Edward Bulwer Lytton	Edward Bulwer Lytton	Louis Tracy	Louis Tracy	1.00044019465	1	15	18	22
Charles Dickens	Edith Wharton	Frank Richard Stockton	Frank Richard Stockton	1.0005119802	3	5	26	42
Gilbert Parker	Howard Pyle	Gilbert Parker	Howard Pyle	1.0003465335	3	9	15	39
Edward Everett Hale	Frank Norris	Louis Tracy	Louis Tracy	1.0002785727	6	7	15	20
Woodrow Wilson	Edward Everett Hale	Edward Everett Hale	Woodrow Wilson	1.00028990395	4	12	14	15
L. Frank Baum	E. M. Forster	Frank Richard Stockton	Charles Dickens	1.000354322275	8	15	20	45
Alfred Tennyson	Alfred Tennyson	Ivan Turgenev	Gilbert Parker	1.000964790725	1	7	65	74
Edith Nesbit	Edith Nesbit	Edith Nesbit	Charles Dickens	1.000506340175	1	3	10	46
Edith Wharton	Charles Dickens	Mark Twain	Howard Pyle	1.000643894675	6	15	39	51
Howard Pyle	Howard Pyle	Edward Everett Hale	Louis Tracy	1.000363823175	1	21	26	57
Mark Twain	Howard Pyle	Frank Richard Stockton	Charles Dickens	1.0009733793	8	12	60	63
Frank Norris	Frank Norris	Frank Norris	Frank Norris	1.000672348025	1	5	18	29
E.E. Smith	E.E. Smith	E.E. Smith	E.E. Smith	1.000707431475	1	2	3	6

19 **Findings**

Before I conducted the experiment I expected the NCD of two similar text to be close to zero. I also expected the Huffman algorithm to be a better indicator of similarity than the Arithmetic Coding algorithm because it encodes each symbol to a codeword rather than the entire sequence to one binary number. I also thought that the experiment using the full text would be yield a more accurate classification than the shortened version of the text. I found that a combination of all the results from the 4 indicators gave a good indication of the author of a given text.

Overall it seems that the Arithmetic Coder did a better job of classifying the data and resulted in better predictions of the author of the given text despite the fact that it was only using a portion of the full txet. My findings lead me to believe that no one aspect of the NCD was a clear indicator but if you combine all the ways of looking at the NCD you can gain a clear understanding of the true identity of an author.

20 **Retrospect**

Looking back at the experiment there are several things that I would have done differently. First of all, I would have conducted the experiment using the Arithmetic Coding algorithm on the full text not just a portion of the text. Also, after the experiment I was left wondering how my implementation of the algorithms compared to other compressors. I should have compared my compressors to some of the standard compression routines such as compress or gzip.

One thing I would have done differently is that I would have implemented the code using C or C++ rather than Python. One reason is speed, anytime code is very computationally expensive chose a language that is known for its speed. Another reason I would have chose C or C++ is because they are more predictable when it comes to bit operations, Python tries to be too helpful.

In hopes of saving time I chose to divide up the experiment across several computers. I used my laptop as well as several desktop computers. Because I ran the code on different machines I was not able to compare the time it took to run the algorithms. One lesson I learned was that desktops are way faster than laptops. I was shocked that the same code took more than ten times longer to run on my laptop than on an older desktop.

21 **Conclusion**

In conclusion, I was really impressed at how well the NCD worked as a data mining classifier. Using data compression methods to classify data needs to be explored further but the potential is there. Entropy coding methods are deffinately some data compression techniques that are successful at classifing data. I beleive that the reason is that the coding methods exploit the redundancy in the data, therefore they are able to effortlessly recognize the patterns in the data.

Perhaps the ability to automatically attribute the author will never be a full proof science like DNA or finger printing but it can do a good job at being very close like shoe prints and tire impressions so we should continue to exploring the application of the NCD.

22 Bibliography

http://www.csd.uoc.gr/~hy438/lectures/Sayood-DataCompression.pdf
Sayood, Khalid, Introduction to Data Compression second edition
http://www.cs.cmu.edu/~aarti/Class/10704/lec7-kraft.pdf
http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
From: Stephen Wolfram, A New Kind of Science Notes for Chapter 10: Processes of Perception
Compression Page 1069
http://www.ics.uci.edu/~dan/pubs/DC-Sec1.html#Sec_1
http://paginas.fe.up.pt/~vinhoza/itpa/bodden-07-arithmetic-TR.pdf
http://en.wikipedia.org/wiki/Shannon%E2%80%93Fano%E2%80%93Elias_coding
http://en.wikipedia.org/wiki/Self-information
http://en.wikipedia.org/wiki/Huffman_coding#Main_properties
http://en.wikipedia.org/wiki/Variable-length_code
http://en.wikipedia.org/wiki/Shannon's_source_coding_theorem
http://en.wikipedia.org/wiki/Shannon%E2%80%93Fano_coding
http://www.cs.cmu.edu/~aarti/Class/10704/lec7-kraft.pdf
http://www.seas.harvard.edu/courses/emr12/2.pdf
http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf
http://www.binaryessence.com/dct/en000080.htm