Universidad Nacional Jorge Basadre Grohmann - Tacna

Facultad de Ingeniería

Escuela Académico Profesional de Ingeniería en Informática y Sistemas

USO DE UN MODELO DE CODIFICACIÓN DE ALGORITMOS PARA UN COMPRESOR ARITMÉTICO

Tesis

Presentada por:

Bach. Reynaldo Alfonte Zapana

Para obtener el título profesional de:

Ingeniero en Informática y Sistemas

Tacna - Perú

2014

UNIVERSIDAD NACIONAL JORGE BASADRE GROHMANN - TACNA
FACULTAD DE INGENIERÍA

TESIS Nro ..... TÍTULO PROFESIONAL DE

Ingeniero en Informática y Sistemas

La Secretaría Académico de la Facultad de Ingeniería, por resolución de Facultad Nro 01938-2014-FAIN/UNJBG, designó Jurado Calificador Dictaminador para la Sustentación de la Tesis titulada: "Uso de un modelo de algoritmos de codificación para un compresor aritmético".

El mismo que está conformado por:

Presidente: Ing. Gianfranco Alexey Málaga Tejada

Secretario: MSc. Edgar Aurelio Taya Acosta

Vocal: MSc. Erbert Francisco Osco Mamani

Para calificar la sustentación de la tesis en acto público el día 19 de Marzo de 2014 presentado por el Bachiller Reynaldo Alfonte Zapana, de la Escuela Académico Profesional de Ingeniería en Informática y Sistemas.

El Jurado Calificador en forma secreta e individual emitió su calificativo sobre el trabajo expuesto y procedió a obtener el promedio que arrojó el calificativo de Aprobado por Unanimidad con tres votos a favor con la nota de Diecisiete y promedio de Sobresaliente.

Para ratificar lo detallado firman:

Ing. Gianfranco Alexey Málaga Tejada
PRESIDENTE

MSc. Edgar Aurelio Taya Acosta
SECRETARIO

MSc. Erbert Francisco Osco Mamani
VOCAL

A Dios, por todo lo que me ha dado, a mis padres, a todos los profesores por sus enseñanzas y a mis amigos.

Agradecimientos

En primer lugar, deseo agradecer a Dios por haberme guiado a lo largo de estos cinco años de estudio.

Agradezco a mis padres por el apoyo brindado para forjarme como un profesional.

Agradezco a la universidad, mi *alma matter*, por haberme cobijado y brindado la formación que ahora me permitirá ayudar a construir una mejor sociedad.

Agradezco de forma muy especial a mi asesor, Prof. Ing. Edwin A. Hinojosa Ramos, por haberme guiado en esta tesis.

Deseo agradecer de forma especial también a: Prof. Dr. Ernesto Cuadros Vargas, Prof. Lic. Luis A. Amaya Cedrón, Ing. Joel Cotrado Vargas por sus sugerencias.

Deseo agradecer al personal administrativo de la universidad: Asunta del Pilar Alva Santillan. Muchas gracias por la atención brindada.

Contenido

Tablas

Figuras

Resumen

Esta investigación tiene como objetivo el estudio de la compresión de texto plano mediante la codificación aritmética. Se propone una lista de archivos para evaluar los algoritmos de compresión presentados. Se determinó que el símbolo más frecuente de esta lista es “e”.

La idea original de la compresion aritmética no es factible de implementar directamente al ordenador debido a error de redondeo y error de truncamiento. Se explica el proceso de la abstracción haciendo uso de operaciones aritméticas enteras mediante un ejemplo.

Los resultados obtenidos indican que los codificadores influyen en el tiempo de compresión, mas no en la tasa de compresión ni tampoco en el coeficiente de variación. El codificador de Moffat y cols. (1998) es rápido en comparación con Witten y cols. (1987). Los modelos estadisticos influyen en la tasa, tiempo de compresión y coeficiente de variación. El modelo estadistico adaptativo ofrece mejor nivel de compresión, pero es lento. El modelo semi-estático tiene menor dispersión en sus tasas de compresión. El modelo estático es más rápido.

Finalmente, se concluye la combinación entre el modelo y el codificador influye directamente en los resultados de compresión.

Abstract

This research aims to study the plain text compression using arithmetic coding. A file list is proposed to evaluate the compression algorithms presented. It was determined that the most common symbol of this list is “e”.

The original idea of arithmetic compression is not feasible to implement directly to the computer due to rounding error and truncation error. The process of abstraction using integer arithmetic operations is explained by an example.

The results indicate that coders influence the time of compression, but not in the compression rate nor in the coefficient of variation. Moffat y cols. (1998) encoder is fast compared to Witten y cols. (1987). The statistical models influence the rate, time compression and coefficient of variation. Adaptive statistical model offers better compression level but slow. The semi-static model has less dispersion in their rates of compression. The static model is faster.

Finally, the combination between the model and the encoder directly influences the results of compression is concluded.

Introducción

Un compresor de datos en formato digital permite reducir el tamaño de los archivos para su almacenamiento en memoria externa o para la transmisión de la información buscando reducir el tiempo del mismo.

La compresión de datos no es un tema nuevo, uno de los primeros inventos donde implícitamente se observa el uso de compresión de los datos fue en el siglo XIX con el código Morse.

El tema de compresión de datos tiene sus bases en investigaciones realizadas a mediados del siglo pasado destacando principalmente el trabajo de Shannon-Weaver. La compresión aritmética es una de las técnicas de codificación basado en la estadística (conteo de caracteres). Los investigadores de esta técnica indican que se caracteriza por brindar mejores niveles de compresión que otros métodos estadísticos. Este método tiene su origen en los trabajos de Elías, Rissanen y Pasco.

Varias compañías han patentado estudios recientes quedando en secreto las investigaciones más actuales sobre el método.

Las aplicaciones de compresión más utilizados en estos días son WinRAR, WinZip y 7-Zip. Estos, por su popularidad y difusión, estan en los primeros de la lista de aplicaciones existentes. Tambien existen otros no populares que brindan igual nivel de compresión, por ejemplo: KuaiZip, HaoZip, Zip Archiver,

PeaZip, WinUHA, StuffIt, PKZIP, etc. Cualquier persona con poca experiencia en computación puede hacer uso de ellas.

Existen casos en que la compresión debe ser especial, solamente cierto tipo datos con características específicas. Estos son los compresores de propósito específico, por ejemplo: JPG, mp3 y mp4. Estos son utilizados por ingenieros de sonido, desarrolladores web, etc. personas con cierto nivel conocimiento en computación para manipular los parámetros necesarios para mantener la calidad y la información.

Cada uno de las aplicaciones y formatos mencionados hacen uso de algoritmos de compresión. Por ejemplo: el formato *Roshal ARchive* (RAR) en su versión 3 está basado en el algoritmo de compresión LZSS (Storer y Szymanski, 1982) y PPMd (Shkarin, 2001), el formato ZIP puede ser generado a partir del algoritmo de compresión DEFLATE(a su vez basado en LZ77 (Ziv y Lempel, 1977) y Huffman (Huffman, 1952)), Bzip2 (está basado en algoritmo de Burrows-Wheeler block-sorting (Burrows y Wheeler, 1994) y Huffman (Huffman, 1952)), Lzma o PPMd (Shkarin, 2001). Los formatos JPG, mp3 y mp4 usan sus propios algoritmos que tiene la misma denominación.

Los algoritmos de compresión tienen sus propias características, se valen de estrategias que identifican un patron de redundancia en los datos para así reducirlos. El algoritmo de Huffman (Huffman, 1952) genera un árbol binario de acuerdo a la probabilidad de aparición de los símbolos, re-codifica los símbolos asignando código de longitud menor para el símbolo más frecuente y el código de longitud mayor para el menos frecuente. El algoritmo de codi-

ficación aritmético utiliza el mismo principio, pero en vez asignar el código a cada carácter lo asigna a todo el archivo de entrada y se considera como una generalización del algoritmo de Huffman (Huffman, 1952).

Este trabajo está organizado de la siguiente forma:

En el Capítulo I se introduce el tema de compresión de archivos, se plantea el problema de investigación describiendo la necesidad del uso de compresores de datos, se explica las dificultades de implementación del algoritmo aritmético, se revisa algunos trabajos científicos sobre este tema de investigación y se propone una lista de archivos para evaluar los algoritmos de compresión.

En el Capítulo II se trata sobre el estado del arte de compresión de datos y se define conceptualmente términos utilizados en toda la investigación.

En el Capítulo III se abstrae un modelo computacional a partir del algoritmo de codificación aritmético original. Este modelo sirve para entender claramente el funcionamiento de los codificadores aritméticos. Se detallan también los experimentos realizados.

En el Capítulo IV se aprecian los resultados para cada modelo de compresión artimético, se determinan el más rápido, el que ofrece mejor tasa de compresión y el más eficiente de ellos.

En el Capítulo V se muestran discusiones sobre los resultados obtenidos.

Capítulo I: Planteamiento de investigación

1.1 Descripción del problema

1.1.1 Antecedentes del problema

Cada día se generan grandes cantidades de datos en formato digital teniendo como origen diversas fuentes. Hay necesidad de almacenarlos en algún dispositivo de almacenamiento externo y en otros casos trasladarlos de un ordenador hacia otro.

Actualmente, hay compresores que son de libre uso (software libre) y compresores comerciales (software propietario). Estos compresores internamente utilizan algoritmos sofisticados que derivan de algoritmos de compresión clásicos tales como Huffman, LZ77, Aritmético, etc.

Las investigaciones más novedosas del algoritmo de compresión arimética están en secreto debido a las patentes existentes y las lideran *International Business Machines Corporation* (IBM) y *Hewlett-Packard Company* (HP).

Se presentan las investigaciones hechas a lo largo de los últimos cincuenta años proponiendo una alternativa de implementación eficiente del método de codificación aritmética, a continuación:

En Langdon (1984) presenta nociones principales de la codificación aritmética para compresión de símbolos binarios (BAC) y alfabeto de n-símbolos.

Asimismo, Witten y cols. (1987) explican un procedimiento de implementación para el método de compresión aritmética, muestran su implementación en C y analizan el rendimiento sobre algunas arquitecturas de computadoras (AX-11/780, Macintosh 512K y SUN-3/75).

Así también, Howard y Vitter (1994) proveen información sobre el método de compresión aritmético, mostrando compresión óptima y que puede ser relacionado con casi cualquier modelo probabilístico. Proponen el uso de tablas de búsqueda para aumentar la velocidad, con un esquema de estimación de probabilidad determinístico para evitar las operaciones aritméticas.

Moffat y cols. (1998) incorporan mejoras a la implementación de Witten y cols. (1987), replanteando el proceso de codificación. Proponen la introducción al sistema de compresión tres procesos: codificación, modelado, estimación probabilística.

1.1.2 Problemática de la investigación

Un compresor aritmético consiste en operaciones complejas de realizar por el ordenador(multiplicaciones y divisiones), que lo convierten en un algoritmo significativamente lento en comparación con el algoritmo Huffman (Huffman, 1952). Los investigadores desean desarrollar una implementación eficiente del algoritmo.

Los archivos están en continuo almacenamiento y siempre están siendo transmitidos. Se necesita alguna forma de reducir el tamaño de los mismos para agilizar los procesos. Si los archivos están en su forma compacta se convierte

en un archivo fácil de transmitir, trasladar y almacenar.

1.2 Formulación del problema

1.2.1 Problema general

¿Cómo influye el uso de un modelo de compresión aritmético en el desempeño de la compresión basado en caracteres de texto plano?

1.2.2 Problemas específicos

¿Cuál es el nivel de correlación del modelo estadístico (estático, semi-estático y adaptativo) con el desempeño de la compresión basado en caracteres de texto plano?

¿En qué medida influye el codificador aritmético en el desempeño de la compresión basado en caracteres de texto plano?

1.3 Justificación

La capacidad de almacenamiento y tiempos de transmisión no deberían ser un problema debido a la aparición de dispositivos de almacenamiento de mayor capacidad, procesadores de alta velocidad y medios de transmisión cada vez mejores a bajos costos. Pero resulta que se usa al límite los recursos siempre se esta buscando reducir archivos y transmitir rápidamente.

El sistema operativo permite interactuar con dispositivos hardware del ordenador. Este sistema administra de manera eficiente los recursos hardware. El tiempo de lectura de un archivo de texto en el ordenador es proporcional

a su tamaño. Mientras el archivo contenga grandes volumenes de datos se necesita más tiempo para cargar en memoria *Random Access Memory* (RAM). La compresion puede acelerar la lectura del archivo de texto.

Las entidades que administran información en forma masiva adoptan de manera conjunta:

1. Adquirir lo último de las tecnologías de dispositivos de alamacenamiento para guardar la información.

2. Utilizar la técnica de compresión adecuada y almacenar compactamente los datos.

Hoy en día, se está en continua producción de datos e información. Se necesita alguna manera de almacenar compactamente los datos generados con el objeto de accederlos en un futuro no muy lejano sin problemas.

1.4 Alcances y limitaciones

1.4.1 Alcances

Los algoritmos de compresión pueden presentar diferentes eficiencias dependiendo del tipo de datos que va a procesar (texto plano, audio, video e imagen). Esta experiencia se enfoca en medir el desempeño de la compresión **basada en caracteres** de archivos de texto plano. Los experimentos realizados estarán dados bajo la misma condición.

1.4.2 Limitaciones

En el desarrollo de la investigación se presentó principalmente la falta de información y escacez bibliográfica en castellano sobre el tema en estudio.

1.5 Objetivos

1.5.1 Objetivo general

Determinar la influencia del uso de un modelo de compresión aritmético en el desempeño de la compresión basado en caracteres de texto plano.

1.5.2 Objetivos específicos

Objetivo específico 1: Identificar el nivel de correlación del modelo estadístico con el desempeño de la compresión basada en caracteres de texto plano.

Objetivo específico 2: Determinar la influencia del algoritmo codificador aritmético en el desempeño de la compresión basada en caracteres de texto plano.

1.6 Hipótesis

1.6.1 Hipótesis General

El uso de un modelo de compresión aritmético influye significativamente en el desempeño de la compresión basado en caracteres de texto plano.

Hernández Sampieri y cols. (2006, pág. 122), indican: “Las investigaciones cuantitativas que formulan hipótesis son aquellas cuyo planteamiento define que su alcance será correlacional o explicativo, o las que tienen un alcance descriptivo, pero que intentan pronosticar una cifra o un hecho”.

1.6.2 Hipótesis Específicas

Hipótesis Específica 1: Existe correlación directa entre el modelo estadístico y el desempeño de la compresión basado en caracteres de texto plano.

Hipótesis Específica 2: El algoritmo codificador aritmético influye significativamente en el desempeño de la compresión basado en carácteres de texto plano.

1.7 Variables

1.7.1 Identificación de variables

Acontinuación se identifican las variables de estudio.

Variable independiente:
Modelo de compresión aritmético.

- Su naturaleza: Cualitativo
- La amplitud de las unidades de observación: Individual
- Su nivel de abstracción: Intermedia
- La escala de medición: Nominal

Variable dependiente:

Desempeño del compresor arimético aplicado a texto plano.

- Según su naturaleza: Cualitativo
- Según su amplitud: Individual
- Según su nivel de abstracción: Intermedia
- Según escala de medición: Ordinal

1.7.2 Definición de variables

Acontinuación se definen las variables de estudio.

Modelo de compresión arimetico: Esta constituido por dos módulos: El primero es el modelo estadístico que alamacena y provee las frecuencias de los símbolos, y el segundo es el codificador aritmético que transforma la información asignando nuevos códigos a los símbolos.

Desempeño del compresor aritmético aplicado a texto plano: Lo conforman el tiempo de compresión, tiempo de descompresión y el nivel de compresión, son la medidas de eficiencia que permiten cuantificar la compresión de texto plano.

1.7.3 Operacionalización de variables

Se descompone nuestras variables en indicadores, los cuales se detallan a continuación, en la Tabla 1.

1.8 Tipo de la investigación

Descriptivo correlacional. Hernández Sampieri y cols. (2006, Cap. 5) indican que la investigación descriptiva busca especificar propiedades, características y rasgos importantes de cualquier fenómeno que se analice. Tambien, definen la investigación correlacional así: "... tiene como propósito conocer la relación que exista entre dos o más conceptos, categorías o variables en un contexto en particular".

1.9 Diseño de la investigación

1.9.1 Diseño de estudio

Diseño no experimental, transeccionales descriptivos. Hernández Sampieri y cols. (2006, Cap. 7) definen diseño no experimental como estudios donde no se hace variar en forma intencional las variables independientes para ver su efecto sobre otras variables, sino mas bien estudios donde se observan fenómenos tal como se dan en su contexto natural, para después analizarlos. Definen también que los estudios transeccionales descriptivos consisten en ubicar en una o diversas variables a un grupo de personas u otros seres vivos, objetos, situaciones, contextos, fenómenos, comunidades; y así proporcionar su descripción.

1.9.2 Población y muestra

Población

La población en estudio comprende a una colección de archivos de texto plano que se propone en la presente tesis, denominado "Literatura peruana" ver Tabla 2. Esta colección está compuesta por obras literarias de la literatura peruana que permitirá evaluar los modelos de compresión estudiados. Cada carácter es almacenado mediante 1 byte (CP-1252) o 4 bytes (UNICODE) dependiento de la codificación del archivo.

En este trabajo de investigación la muestra sera igual a la población.

1.9.3 Técnicas e instrumentos para recolección de datos

Técnicas: Observación

Instrumentos: Guía de observación

Tabla 1. Operacionalización de variables

Tipo	Variable	Definición Conceptual	Definición Operacional	Indicadores	Subindicadores
V.I.	Modelo de compresión aritmético	Hace mención a los modelos de compresión aritmético.	Secuencia de pasos ordenados (algoritmo) en que consiste la codificación arimética.	Modelo estadístico	- Modelo estático (E) - Modelo semiestático (S) - Modelo adaptativo (A)
				Codificador aritmético	- Codificador basado en Witten y cols. (1987), (W). - Codificador basado en Moffat y cols. (1998), (M).
V.D.	Desempeño del compresor aritmético aplicado a texto plano	Son las medidas de desempeño y eficiencia que se consideran para medir compresores.	Tiempo y tamaño de compresión.	- Tiempo de compresión - Razón de compresión. - Coeficiente de variación - Cuandrante mágico de Gartner	

Fuente: Elaboración propia

Tabla 2. La colección “Literatura peruana”

Título original de la obra	Autor	Fecha de publicación	Género
7 ensayos de interpretación de la realidad peruana	José Carlos Mariátegui	1928	Ensayo
Agua	José María Arguedas	1935	Cuento
Alma América	José Santos Chocano	1906	Poesía
España, Aparta de mí este cáliz	César Vallejo	1937	Poesía
El caballero Carmelo	Abraham Valdelomar	1913	Cuento
Comentarios Reales	Inca Garcilaso de la Vega	1616	Narrativa
Los Heraldos Negros	César Vallejo	1918	Poesía
Los hermanos Ayar	Abraham Valdelomar	1913	Cuento
Poemas humanos	César Vallejo	1938	Poesía
Poemas en prosa	César Vallejo	1968	Poesía
Tradiciones peruanas	Ricardo Palma	1872	Narrativa
Trilce	Cesar Vallejo	1922	Poesía

Fuente: Elaboración propia

Capítulo II: Marco teórico

2.1 Datos en formato digital

2.1.1 Origen de los datos

Los datos en formato digital pueden ser creados por procesadores de texto, editores de imágenes, cámaras digitales, videograbadoras, cámara de vídeo, grabadoras digitales de voz , equipos de tomografía, satélites, etc.

2.2 Compresión de datos

La compresión de datos se define como el proceso de reducir la cantidad de datos necesarios para representar eficazmente una información, con el objetivo de almacenar en un dispositivo de almacenamiento o transmitir a través de la red.

2.3 Teoría de la información

El enfoque de la teoría de la información analiza la estructura matemática y estadística de los mensajes, con independencia de su significado u otros aspectos semánticos. Los aspectos en los que se interesa la teoría de la información son la capacidad de transmisión de los canales, la compresión de datos o la detección y corrección de errores.

La teoría de la información es una disciplina que se encarga de cómo almacenar y transmitir la información entre dispositivos, lo que se origina en

un punto llegue tal cual a otro punto destino. La teoría de la información está ligado al sistema de comunicación, ver Figura 1.

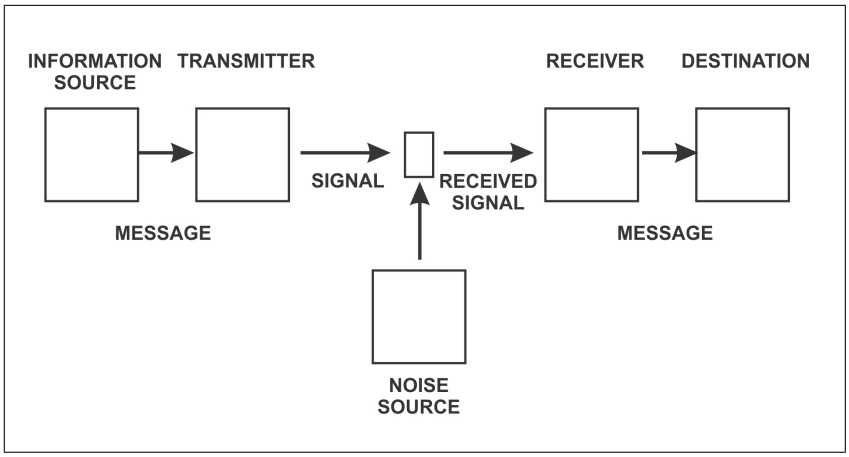


Figura 1. Sistema de comunicación general
Fuente: Shannon (1948)

El sistema de comunicación consiste básicamente de 5 partes (Shannon, 1948).

- La fuente de la información, el cual produce un mensaje o secuencia de mensajes.
- El transmisor opera sobre el mensaje en alguna forma para producir una señal apropiada para transmisión sobre el canal.
- El canal es solamente el medio usado para transmitir la señal que va desde el transmisor hacia el receptor.
- El receptor ejecuta la operación inversa hecha por el transmisor, reconstruyendo el mensaje de la señal.

- El destinatario es la persona o cosa a quien o a cual se espera que llegue el mensaje.

El sistema de comunicación se clasifica en 3 categorías: Sistema discreto (mensaje y señal secuencia de símbolos), sistema continuo (mensaje y señal como funciones continuas) y sistema mixto (aparecen variables discretos y continuos).

Un sistema de comunicación es análogo a un sistema compresión. Por eso, en términos de compresión datos se toma como fuente de información al archivo a comprimir y como mensaje a los caracteres que conforman dicho archivo. El canal viene a ser el medio en donde se almacena la información o por el cual se transmite.

2.3.1 Datos e Información

Es usual que se interpreten datos e información indistintamente, lo cual se debe diferenciar para no permitir que continue esta aberración y relacionarse fácilmente con los términos de este tema.

Los datos son símbolos, letras, números, hechos aislados o valores recolectados del mundo real, estos no tienen contexto ni relación entre sí, que pueden ser leídos y procesados por una computadora para producir información. Por ejemplo: 19, matemática, John.

Cuando los datos son sometidos a un proceso el cual relaciona entre sí, se obtiene la información. Por ejemplo: John ganó el concurso de matemática

con 19 puntos.

Desde otro punto de vista, la información es un conocimiento explícito extraído por sistemas como resultado de interacción con el entorno.

Las personas interactuan con la información, se percibe la información en los medios de comunicación, en el entorno que nos rodea y se reacciona de acuerdo a ella todo el tiempo (Pu, 2006, Pág. 19).

Un conjunto de datos organizados son una forma de representar la información, una misma información puede ser representada por otros conjuntos de datos y algunas de estas representaciones pueden contener datos redundantes.

La información es el resultado del procesamiento de una representación particular de datos, ver Figura 2. La información cambia el estado de conoci-

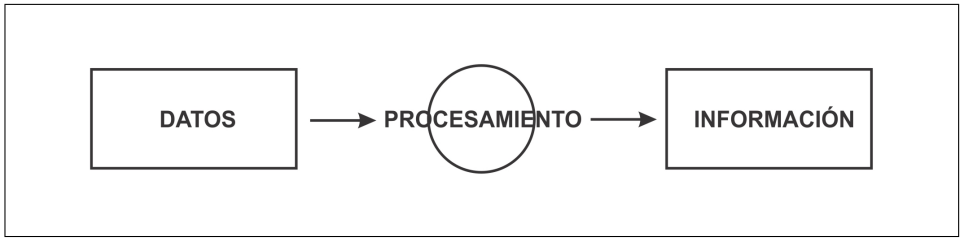


Figura 2. Relación entre datos e información
Fuente: Elaboración propia

miento de un sujeto o sistema que lo percibe, ver Figura 3. La inversión de la pirámide del conocimiento(*Datos Información Conocimiento Sabiduría* (DICS)) produce una serie de términos opuestos incluyendo la desinformación, el error, la ignorancia y la estupidez (Bernstein, 2009). Definiciones de datos, informacion y conocimiento se pueden encontrar con detalle en Zins (2007).

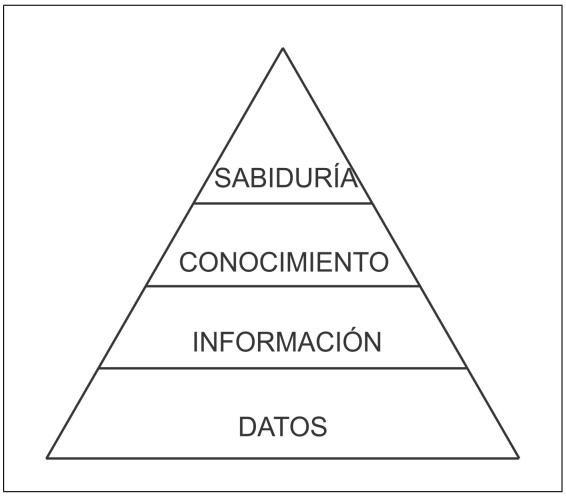


Figura 3. Pirámide del conocimiento
Fuente: Rowley (2007)

La información permite aumentar el conocimiento y reducir la incertidumbre entre un conjunto de alternativas posibles. Una forma de caracterizar nuestro estado de conocimiento del mundo, es a través de las probabilidades. Si se sabe que en el futuro pueden suceder n cosas o eventos diferentes $e_1, \ldots, e_n$, cada una con probabilidad $p_1, \ldots, p_n$ ese conjunto de probabilidades constituyen el conocimiento del mundo, la información debería reducir nuestra incertidumbre, variando las probabilidades a $\bar{a}_1, \ldots, \bar{a}_n$. Si el segundo estado tiene menos incertidumbre es porque algunas cosas se han hecho más probables frente a otras alternativas que se han hecho menos probables.

La ocurrencia de eventos de alta probabilidad de aparición aporta menos información que la ocurrencia de eventos menos probables. Por ejemplo: ¿Cuántos bits son necesarios para transmitir el estado del clima de Tacna a Arequipa? Si se tiene las siguientes probabilidades, ver Tabla 3.

Una alternativa es asignar índice (0,1,2,3) a los estados del tiempo, y

Tabla 3. Pronóstico del tiempo

Pronóstico del clima	Probabilidad
Soleado	0,5
Nublado	0,25
Lluvioso	0,125
Ventoso	0,125

Fuente: Elaboración propia

luego enviar solo el índice. Esta opción necesita 2 bits. Pero las probabilidades del estado del clima no son uniformes, es más probable que se envie el estado de clima soleado. Por eso es necesario establecer descripciones pequeñas para los más probables y descripciones largas para los menos probables, de la siguiente forma: Soleado 0, Nublado 10, Lluvioso 110 y Ventoso 111.

En la compresión de datos el canal se considera sin ruido (Gonzáles-Ruiz, 2000), entonces se define la información de cualquier evento x, como:

$$I(x) = \log_2 \frac{1}{p(x)} = -\log_2 p(x) \tag{1}$$

La Ecuación 1 describe el número de bits necesarios para representar un evento.

Como se puede observar, la definición dada de la información es coherente si se tiene en cuenta que:

- Si $p(x) = 1$ (el suceso x siempre ocurre), entonces $I(x) = 0$ (la ocurrencia del suceso no proporciona ninguna información)
- Si $p(x) \to 0$ (el suceso x casi nunca ocurre), entonces $I(x) \to \infty$ (la

ocurrencia del suceso proporciona mucha información).

La información de dos eventos independientes es aditiva, ver Ecuación 2.

$$I(AB) = \log_2 \frac{1}{p(A)\,p(B)} = \log_2 \frac{1}{p(A)} + \log_2 \frac{1}{p(B)} = I(A) + I(B) \qquad (2)$$

2.3.2 Entropía

Supóngase que X es una variable aleatoria discreta que toma eventos del alfabeto χ y que este alfabeto $\chi = \{x_1, x_2, \ldots, x_n\}$ contiene un conjunto de eventos independientes o símbolos . Además, téngase en cuenta que $p(x) = P[X = x]$ para $x \in \chi$.

La entropía $H(X)$ según Cover y Thomas (2006), de una variable aleatoria discreta X es definido por, ver Ecuación 3.

$$H(x) = \sum_{x \in \chi} p(x)\, I(x) = -\sum_{x \in \chi} p(x) \log_2 p(x) \qquad (3)$$

La entropía varía en el siguiente rango, ver Ecuación 4

$$0 \leq H(X) \leq \log_2 n \qquad (4)$$

donde $\chi = \{x_1, x_2, \ldots, x_n\}$ y X variable aleatoria. La Ecuación 4 se demuestra de la siguiente manera:

- $H(X) = 0$, si y solo si $p(x) = 1, \exists x \in \chi$

- $H(X) = \log_2 n$, si y solo si $p(x) = \frac{1}{n}, \forall x \in \chi$

La importancia de la entropía en compresión de datos, es que permite estimar el número promedio de bits mínimo necesarios para representar un símbolo compactamente.

Desde la Ecuación 4 , obtenemos la Ecuación 5.

$$0 \leq \frac{H(X)}{\log_2 n} \leq 1 \tag{5}$$

Shannon (1948) demostró que no es posible comprimir una fuente estadísticamente menos del nivel indicado por su entropía, ver Ecuación 6.

$$\frac{H(X)}{\log_2 n} \leq tasa\_compresión \tag{6}$$

2.3.3 Entropía condicional

La entropía $H(X,Y)$ de dos variables aleatorias discretas X y Y esta dado por, ver Ecuación 7.

$$H(X,Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log_2 \frac{1}{p(x,y)} \tag{7}$$

Por el Teorema de Bayes se sabe que $p(x,y) = p(y)p(x|y)$ donde $p(x|y)$ es la probabilidad de que se dé un estado x dado que ya se conoce y.

La entropía condicional $H(X|Y)$ se define como, ver Ecuación 8.

$$H(X|Y) = \sum_{(x,y)\in(X,Y)} p(x,y) \log_2 \frac{p(y)}{p(x,y)} \tag{8}$$

Propiedades de la Entropía condicional

$$H(X,Y) = H(X) + H(Y|X) \, (Regla\ de\ Chain) \tag{9}$$

De la Ecuación 9 se obtiene la Ecuación 10.

$$H(X|Y) \leq H(X) \tag{10}$$

2.4 Clasificación de algoritmos de compresión

2.4.1 Compresión con y sin pérdida de datos

Las técnicas de compresión pueden ser divididos en dos grupos; técnicas de compresión que no producen pérdida de datos y los que sí producen pérdida (Nelson y Gailly, 1995).

El primer grupo supone la no pérdida de la información debido a que los datos originales pueden ser recuperados de los datos comprimidos Sayood (2006), e interpretarse la información que contiene. Su uso es principalmente en la compresión de texto plano, pero también se ha visto este tipo de técnicas en la compresión de imágenes (Lossless JPEG compression) y compresión de audio (FLAC compression).

El segundo grupo implica pérdida de datos, los datos originales no pueden ser recuperados o reconstruidos exactamente Sayood (2006) de los datos comprimidos, aunque sí una aproximación cuya semejanza con la original (la calidad de la información) dependerá del tipo de compresión. Se utilizan principalmente en la compresión de imágen (JPG), compresión de audio (mp3) y video (mp4).

2.4.2 Compresión de códigos fijos o variables

Los algoritmos de compresión también se pueden clasificar por la longitud de los códigos antes y después de la compresión, ver Tabla 4.

Fijo-a-Fijo: Cada símbolo es representado por una longitud fija de bits antes y después de la compresión, por ejemplo: si tenemos A, B, C y D que consisten de 8 bits entonces podria ser codificado como A: 00, B: 01, C: 10, D: 11.

Fijo-a-Variable: Cada símbolo antes de la compresión es representado por un número fijo de bits y es codificado por una secuencia de bits variable. Ejemplo: A: 0, B: 10, C: 101, D: 0101

Variable-a-Fijo: Una secuencia de símbolos que consiste de varios bits es codificado en secuencias bits de longitud fija. Ejemplo: ABCD: 00, ABCDE: 01, BC: 11

Variable-a-Variable: Una secuencia de símbolos representados en número diferente de bits es codificado en una secuencia de bits de longitud variable. Ejemplo: ABCD: 0; ABCDE: 01; BC: 1; BBB: 0001;

Tabla 4. Ejemplos de clasificar por tamaños de código.

Método	Tamaño de grupo	
	Entrada	Salida
CS&Q	Fijo	Fijo
Huffmann	Fijo	Variable
Aritmético	Variable	Variable
Run-Length, LZW	Variable	Fijo

Fuente: Smith (1999)

2.4.3 Clasificación en cuanto a su operación

Los métodos de compresión también se clasifican por la forma en que operan, y el objetivo que busca alcanzar. Esta forma de clasificación incluye:

Métodos estadísticos o métodos de aproximación a su entropía: Son métodos que utilizan la probabilidad de ocurrencia de los símbolos en la secuencia de datos y alterar la representación de cada símbolo o grupo de símbolos. Así que tienen por objeto reducir el número de bits utilizados para representar cada símbolo o grupo de símbolos, y así aproximar el tamaño medio de los símbolos a su entropía. Son los métodos que se basan directamente en teoría de la información como la codificación Huffman o codificación aritmética.

Los métodos basados en diccionário o los métodos de eliminación de datos duplicados: Son métodos que utilizan diccionarios u otras estructuras similares para eliminar repeticiones de símbolos (frases) redundantes o repetitivas, los métodos LZ77 y LZ78 (Ziv y Lempel, 1978) son parte de ese grupo. Los programas más utilizados de compresión sin pérdidas asocian una técnica basada en diccionario con una técnica estadística.

Transformaciones: Son métodos que por sí sola no comprimen los datos, pero son capaces de transformar los datos que no se comprimen o son ligeramente comprimidos, en datos que pueden ser más fácilmente comprimidos. Se utilizan generalmente para la compresión con pérdida para eliminar la correlación entre los datos adyacentes y así identificar qué datos se pueden eliminar sin perjuicio del resultado final. Un ejemplo es la transformada discreta del coseno , utilizado por JPEG y MPEG (Gall, 1991). Una excepción es el método de Burrows-Wheeler utilizado en la compresión sin pérdida de datos.

2.4.4 Compresión estática y dinámica

Los métodos de compresión son clasificados también como estáticos y dinámicos Lelewer y Hirschberg (1987).

Los estáticos codifican antes de realizar la transmisión, de manera que un mensaje es representado por el mismo código cada vez que aparece en el archivo, ver en Figura 4.

Los dinámicos cambian el código de los mensajes con el tiempo y reciben también el nombre de adaptativos, ver en Figura 5.

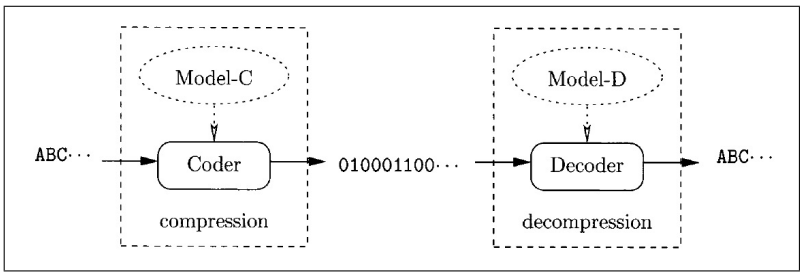


Figura 4. Sistema de compresión estático
Fuente: Pu (2006)

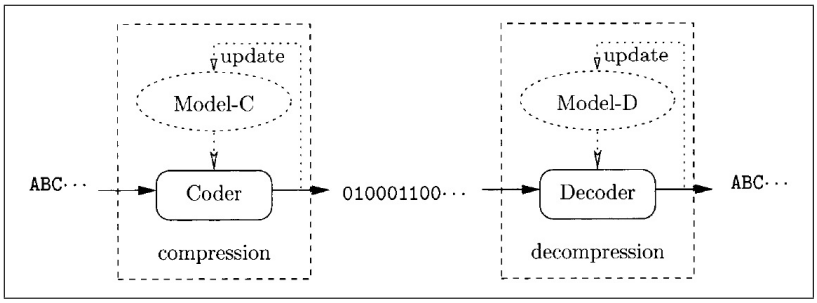


Figura 5. Sistema de compresión dinámico
Fuente: Pu (2006)

2.5 Implementación de algoritmos de compresión

Para implementar un algoritmo de compresión de datos para un problema específico se deben seguir los siguientes pasos Pu (2006).

1. Descripción del problema
2. Modelamiento matemático
3. Diseño del algoritmo
4. Verificación del algoritmo
5. Estimación de la complejidad computacional
6. Implementación
7. Testeo del programa
8. Documentación

2.6 Uso de los algoritmos de compresión

El uso que se da principalmente es almacenamiento y transmisión de información.

Los algoritmos de compresión han permitido que haya sido posible el crecimiento de Internet, masificación de la multimedia a través de Internet, aparición de TV digital, video conferencia, comunicación satelital y otros avances tecnológicos actuales.

2.7 Eficiencia y desempeño de los algoritmos de compresión

2.7.1 Tasa de compresión

Es una de las medidas que son comúnmente usadas para expresar la eficiencia de un método de compresión, ver Ecuación 11.

$$tasa\_compresión = \frac{tamaño\_archivo\_comprimido}{tamaño\_archivo\_original} \tag{11}$$

2.7.2 Factor de compresión

Indica la porción existente entre el tamaño del fichero sin comprimir y el tamaño del fichero comprimido Gonzáles-Ruiz (2000) se expresa como, ver Ecuación 12.

$$factor\_compresión = \frac{tamaño\_archivo\_original}{tamaño\_archivo\_comprimido} \tag{12}$$

Si $factor\_compresión \leq 1$ entonces se produce expansión del archivo. El tamaño del archivo comprimido es mayor que el original, por tanto, ya no se considera como compresión.

2.7.3 Promedio de bits por símbolo

Es el número de bits en promedio para representar un símbolo, se obtiene de la Ecuación 13.

$$promedio\_bits = \frac{tamaño\_archivo\_comprimido}{tamaño\_archivo\_original} * \log_2 N \tag{13}$$

2.7.4 % de espacio ahorrado

Es el porcentaje de espacio libre despues de la compresión, se obtiene mediante la Ecuación 14.

$$\%\_espacio\_ahorrado = 1 - tasa\_compresión \tag{14}$$

2.8 El compresor aritmético

El codificador aritmético necesita de un modelo estadístico.

2.8.1 Modelo estadístico

Modelo estadístico estático

En este modelo las frecuencias de los símbolos no cambian, estan previamente definidas. Estas deben ser buenas estimaciones generadas a partir de archivos similares al que se va comprimir.

Modelo estadístico semi-estático

Este modelo necesita recorrer el archivo para determinar la frecuencia de aparición real de los símbolos. Las frecuencias deben ser enviadas junto con los datos comprimidos para usar nuevamente en el proceso de descompresión. Se le considera un proceso de dos pasos.

Modelo estadístico adaptativo

Este modelo inicia con alta incertidumbre, no se conoce las frecuencias de los símbolos. Puede iniciar con frecuencias estimadas o simplemente con 1s. No necesita que estas frecuencias vayan junto con los datos comprimidos para su descompresión. Se le considera un proceso de un solo paso.

2.8.2 Codificador aritmético

El mensaje es representado por un número real entre $[Inf, Sup) = [0, 1)$. Mientras el mensaje va en aumento, el intervalo necesitado para representarlo

se convierte pequeño y los límites $[Inf, Sup)$ números de punto flotante con bastantes decimales (Salomon, 2008).

Se empieza calculando las frecuencias de aparición de los símbolos desde el archivo de entrada. Pero en caso de poseer de buenas estimaciones de las frecuencias este paso puede ser omitido.

Los pasos que involucra el proceso de codificación, se resumen a continuación

Paso 1. Se define los límites dentro del intervalo $[0, 1)$.

$$Inf = 0, Sup = 1$$

Paso 2. Se repite el proceso hasta que el archivo entero sea leído.

Se halla el rango a partir del intervalo.

$$Rango = Sup - Inf$$

Se actualiza el intervalo, de acuerdo al símbolo de entrada s.

$$Sup = Inf + Rango * LimSup\,(s)$$

$$Inf = Inf + Rango * LimInf\,(s)$$

Paso 3. Cuando el archivo de entrada haya sido leído, la salida del proceso

debería ser cualquier número real que esté dentro del intervalo actual *código*.

El proceso de decodificación es un proceso inverso, los pasos son los siguientes:

Paso 1. Se necesita el código de salida del proceso de compresión *código*.

Paso 2. Se repite hasta que hayamos leído el archivo. Se encuentra el símbolo, cuyos rangos contengan el valor *código*.

Se calcula el rango del símbolo s decodificado.

$$Rango = LimSup\,(s) - LimInf\,(s)$$

Se obtiene el nuevo *código* .

$$código = \frac{código - LimInf\,(s)}{Rango}$$

Paso 3. Debería haber sido decodificado el archivo.

2.9 Definición conceptual de términos

2.9.1 BIT

Es un dígito del sistema de numeración binario, puede ser 0 o 1. Es la mínima unidad de información empleada en informática, en cualquier dispositivo digital o en la teoría de la información.

2.9.2 Byte

Es la secuencia de 8 bits que se utiliza para representar un símbolo en las computadoras modernas. Las equivalencias del Byte se listan a continuación, en Tabla 5.

Tabla 5. Equivalentes del BYTE

Según SI		Según ISO/IEC 80000-13	
Kilobyte	10^3 byte	Kibibyte	2^{10} byte
Megabyte	10^6 byte	Mebibyte	2^{20} byte
Gigabyte	10^9 byte	Gibibyte	2^{30} byte
Terabyte	10^{12} byte	Tebibyte	2^{40} byte
Petabyte	10^{15} byte	Pebibyte	2^{50} byte
Exabyte	10^{18} byte	Exbibyte	2^{60} byte
Zettabyte	10^{21} byte	Zebibyte	2^{70} byte
Yottabyte	10^{24} byte	Yobibyte	2^{80} byte

Fuente: `http://physics.nist.gov/cuu/Units/binary.html`

Los equivalentes del *Sistema Internacional* (SI) se utilizan en capacidad de almacenamiento, transmisión de datos y velocidad del procesador, mientras tanto los equivalentes de ISO/IEC 80000-13 se usan para medir la capacidad de la RAM.

2.9.3 ASCII

Es el código de caracteres estándar que son usados en las computadoras modernas ver Anexo 1.1. Cada carácter es representado por 7 bits, como resultado 128 símbolos pueden ser codificados. *American Standard Code for Information Interchange* (ASCII) puede manejar letras mayúsculas y minúsculas, números, signos de puntuación y algunos caracteres de control. Empleaba un

bit adicional (bit de paridad) que se usa para detectar errores en la transmisión o para almacenar 128 caracteres adicionales. ASCII extendido, ver Anexo 1.2, fue un intento de estandarizar el conjunto de caracteres a 256, pero no fue aceptado internacionalmente.

2.9.4 CP-1252

Windows-1252 o CP-1252 es una codificación de caracteres del alfabeto latino, alfabeto Inglés y otros idiomas occidentales. Es una versión en el grupo de páginas de códigos de Windows. Esta codificación es un superconjunto de la norma ISO 8859-1, pero difiere de la ISO-8859-1 debido al uso de caracteres que se pueden mostrar en lugar de los caracteres de control en el rango de 0080 a 009F, ver Anexo B.

2.9.5 ISO/IEC 8859-1

ISO/IEC 8859-1:1998 es el conjunto de caracteres de 8 bits. Es parte de la serie ISO/IEC 8859 de la codificación de caracteres estándar basados en ASCII. Por lo general, está destinada a las lenguas de "Europa Occidental". Es la base para la mayoría de los juegos de caracteres de 8 bits más populares, incluyendo Windows 1252 y es el primer bloque de caracteres Unicode.

2.9.6 Unicode

Unicode es un estándar de codificación de caracteres. Define la forma en que los caracteres individuales se representan en archivos de texto, páginas web, y otros tipos de documentos Christensson (2012).

Fue creado por el Consorcio Unicode para la codificación universal completa de los carácteres. Unicode es un conjunto de 4 bytes, compatible con ASCII y ISO/IEC 8859-1, ver Figura 6.

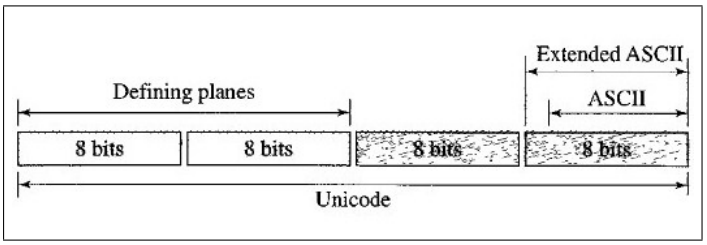


Figura 6. Rango de Unicode
Fuente: Forouzan (2007)

Cada carácter o símbolo en esta codificación es definido por un número de 32-bits. Esta codificación puede definir hasta 4 294 967 296 caracteres o símbolos. La notación usa dígitos hexadecimales en el siguiente formato U-XXXXXXXX.

2.9.7 Archivo

Tanenbaum (2009, pág. 256) " ... los archivos son unidades lógicas de información creada por los procesos. ... un disco contiene miles o incluso millones de archivos independientes ... si se considera a cada archivo como un espacio e direcciones no se estará tan alejado de la realidad ... ".

Un archivo es una colección de datos estructurados almacenados en algún medio de alamacenamiento como una unidad bajo un formato.

2.9.8 Formato de archivo

Todo archivo está constituido por bytes que deben ser interpretados en conjunto para abrir, ejecutar o visualizar. El tipo de interpretación de cada archivo está dado por el formato que utiliza (un archivo gráfico de formato GIF se interpretará como tal y no como si fuese de formato ZIP). El formato permite identificar la aplicación adecuada para visualizar el contenido del archivo.

Inicialmente, cada formato de compresión estaba diseñado para seguir un determinado algoritmo de compresión. Ahora, la mayoría de los formatos de compresión pueden utilizar diferentes algoritmos. Por ejemplo, varios programas permiten comprimir en formato ZIP pudiendo elegir algoritmo LZMA, deflate o bzip2. Esto explica por qué algunos compresores comprimen en cierto formato y los archivos no son reconocidos por otros compresores.

2.9.9 Compresores y archivadores

Es necesario distinguir qué es un compresor y archivador ya que se utilizan indistintamente estos términos. Los archivadores agrupan varios archivos en un solo (TAR y GTAR), mientras que los compresores reducen el tamaño de un archivo mediante algoritmos que permitan conservar la información (GZIP, BZIP2 y LZIP). Los compresores de propósito general, tales como: WinZip, WinRAR, PKZIP, WinAce, etc., se caracterizan porque comprimen cualquier tipo de archivo y devuelven el archivo tal cual después de la descompresión, no producen pérdida de datos. En tanto, los compresores específicos pueden tener pérdida de datos disminuyendo su calidad: JPEG Wallace (1992), MPEG,

FLAC (compresor de audio), etc.

2.9.10 Incertidumbre

Se entiende por incertidumbre una situación en la cual no se conoce completamente la probabilidad de que ocurra un determinado evento. Entonces, si se conoce la probabilidad se disminuye la incertidumbre.

2.10 Abreviaturas

IBM *International Business Machines Corporation*

HP *Hewlett-Packard Company*

RAR *Roshal ARchive*

RAM *Random Access Memory*

UNICODE *Unique, Universal and Uniform character enCoding*

ASCII *American Standard Code for Information Interchange*

CV *Coeficiente de Variación*

GMQ *Gartner Magic Quadrants*

DICS *Datos Información Conocimiento Sabiduría*

SI *Sistema Internacional*

Capítulo III: Desarrollo del algoritmo computacional

3.1 Característica de la colección "Literatura peruana"

Se determina el tamaño de los archivos a comprimir, ¿Cuántos bytes contiene el archivo? Ver Tabla 6.

El tamaño máximo y mínimo son: 5 913 274 bytes y 21 838 bytes, pertenecen respectivamente a los archivos "tradiciones.txt" y "carmelo.txt".

Tabla 6. Tamaño de la colección "Literatura peruana"

Título de la obra	Nombre del archivo	Tamaño (Bytes)
7 ensayos de interpretación de la realidad peruana	7ensayos.txt	1 284 348
Agua	agua.txt	45 423
Alma América	alma_america.txt	286 253
España, Aparta de mí este cáliz	caliz.txt	27 886
El caballero Carmelo	carmelo.txt	21 838
Comentarios Reales	comentarios_reales.txt	250 416
Los Heraldos Negros	heraldos.txt	49 106
Los hermanos Ayar	hermanos_ayar.txt	30 154
Poemas humanos	humanos.txt	85 958
Poemas en prosa	prosa.txt	27 960
Tradiciones peruanas	tradiciones.txt	5 913 274
Trilce	trilce.txt	59 740
	TOTAL	8 082 356

Fuente: Elaboración propia

Todos estos archivos están en formato TXT y contienen párrafos de texto

literario. Cada carácter es almacenado mediante un byte (CP-1252) en casi la mayoria de los archivos, excepto dos de ellos: 7ensayos.txt y tradiciones.txt que contienen caracteres *Unique, Universal and Uniform character enCoding* (UNICODE) que utilizan 4 bytes para su almacenamiento.

3.2 Modelamiento matemático

Se necesita definir un modelo estadístico que va proveer información al codificador. Se debe definir la estructura de datos adecuada para acceder a las frecuencias y las probabilidades.

Ejemplo: Se toma la palabra "ciencia". Se obtiene las frecuencias contando la aparición de los símbolos, ver Tabla 7.

Tabla 7. Frecuencias y probabilidades de las letras de la palabra "ciencia"

Carácter	Frecuencia	Frecuencia acumulada	Probabilidad	Probabilidad acumulada
c	2	2	0,286	0,286
i	2	4	0,286	0,571
e	1	5	0,143	0,714
n	1	6	0,143	0,857
a	1	7	0,143	1,000

Fuente: Elaboración propia

La probabilidad de cada símbolo se obtiene de su frecuencia dividido por el total de símbolos que conforman la frase, ver Tabla 8.

Se distribuye cada carácter en un rango $[0; 1)$ de acuerdo a su probabilidad.

El proceso de división hace perder dígitos decimales en los límites inferior

Tabla 8. Límites de los símbolos de “ciencia”

Carácter	Límite inferior	Límite superior
c	0,000	0,286
i	0,286	0,571
e	0,571	0,714
n	0,714	0,857
a	0,857	1,000

Fuente: Elaboración propia

Tabla 9. Almacenamiento de frecuencia acumulada

Índice	Carácter	Frecuencia acumulada
0	c	0
1	i	2
2	e	4
3	n	5
4	a	6
5		7

Fuente: Elaboración propia

y superior debido al error redondeo y truncamiento. Para evitar esta pérdida será mejor trabajar con sus frecuencias y el total de símbolos.

La representación de las frecuencias en la Tabla 9, permite obtener las frecuencias acumuladas correspondientes a los límites inferior y superior de un determinado símbolo.

Véase cómo se obtiene los límites inferior y superior para la letra “c” que tiene el índice “0”, estarían dados por: $LimInf\,(c) = \frac{frec\_acum[0]}{frec\_acum[5]} = \frac{0}{7}$ y $LimSup\,(c) = \frac{frec\_acum[1]}{frec\_acum[5]} = \frac{2}{7}$.

En general, para determinar los límites de cualquier símbolo s_i, estarían dados por: ver Ecuación 15 y Ecuación 16, donde N es el total símbolos

diferentes y $frec\_acum\,[N]$ es total de símbolos de la fuente a comprimir.

$$LimInf\,(s_i) = \frac{frec\_acum\,[i]}{frec\_acum\,[N]} \tag{15}$$

$$LimSup\,(s_i) = \frac{frec\_acum\,[i+1]}{frec\_acum\,[N]} \tag{16}$$

La implementación de la idea original del algoritmo en estudio no se puede llevar directamente al ordenador.

Continuando con el ejemplo para observar qué ocurre en la codificación de la palabra "ciencia", ver Figura 7 y Figura 8.

El código que representa la compresión estaría dado por un valor en el rango $\left[\frac{112712}{813543}; \frac{16104}{117649}\right)$ que es equivalente a $[0,136862313; 0,136881741)$. Se elige $código = \frac{inf+sup}{2} = 0.136872027$ en este caso.

El proceso de descompresión es un procedimiento inverso, ver Figura 9 y Figura 10.

Computacionalmente el proceso descrito anteriormente no es factible de implementar en el ordenador.

Es necesario entonces definir un proceso adecuado. Una forma es mediante el uso de aritmética entera. Se verá como se puede abstraer hacia un modelo computacional evitando operaciones en punto flotante, ver Tabla 10.

En la Tabla 10 el valor de sup es 4 y el valor inf es 0. Si se agrega un bit (0) por la derecha a los cinco puntos, entonces estaría constituido de 4 dígitos binarios el valor de $sup = 8$ y el valor $inf = 0$, ver Tabla 11.

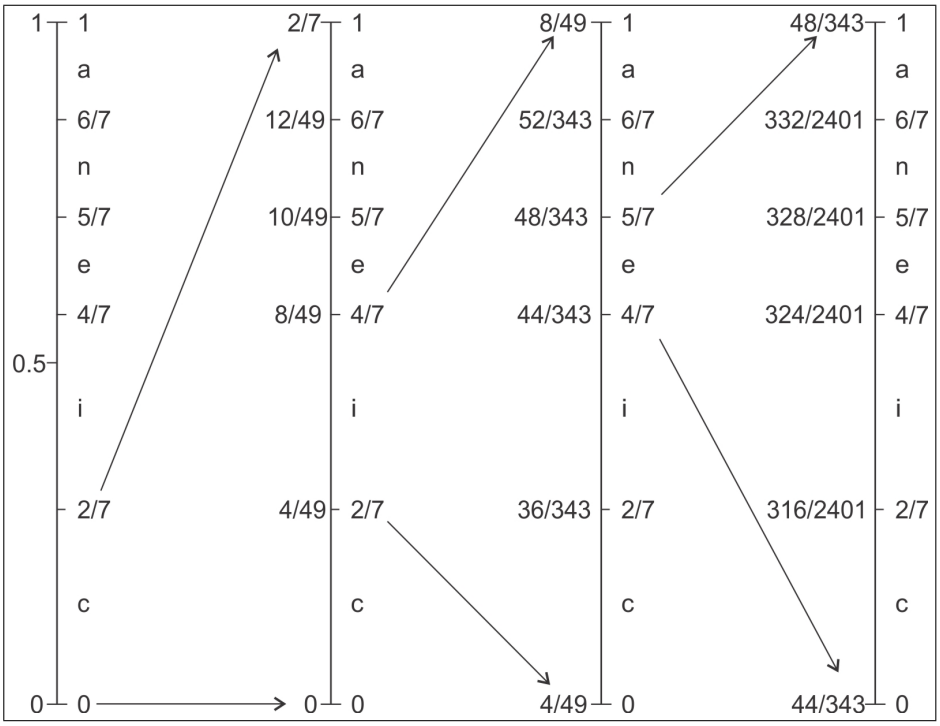


Figura 7. Proceso de codificación de la palabra "ciencia"
Fuente: Elaboración propia

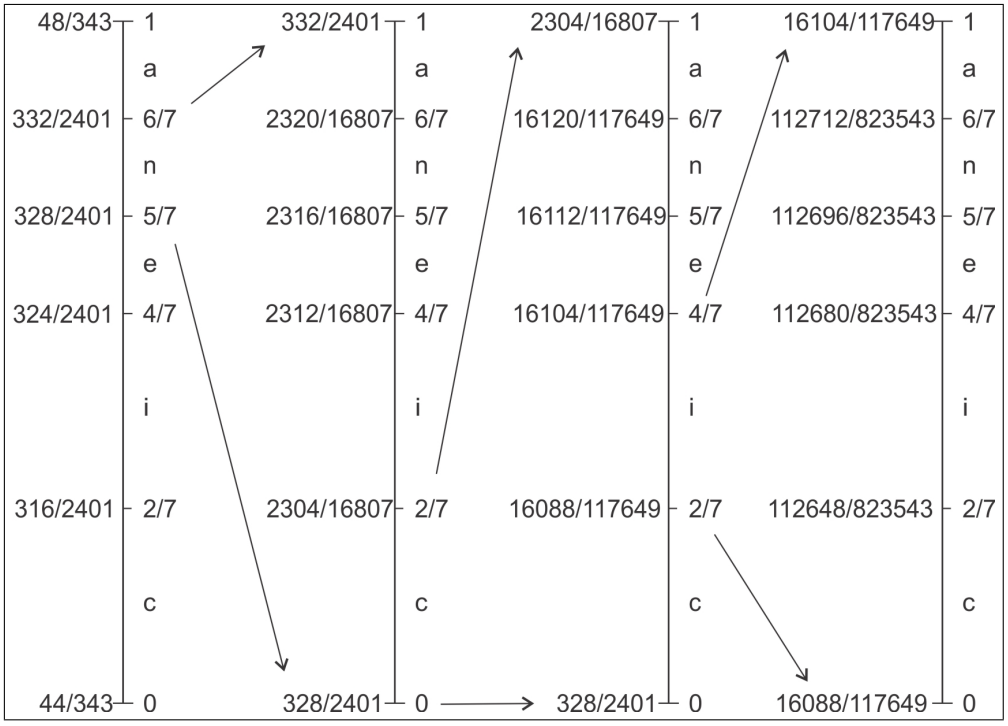


Figura 8. Continuación proceso de codificación de la palabra "ciencia"
Fuente: Elaboración propia

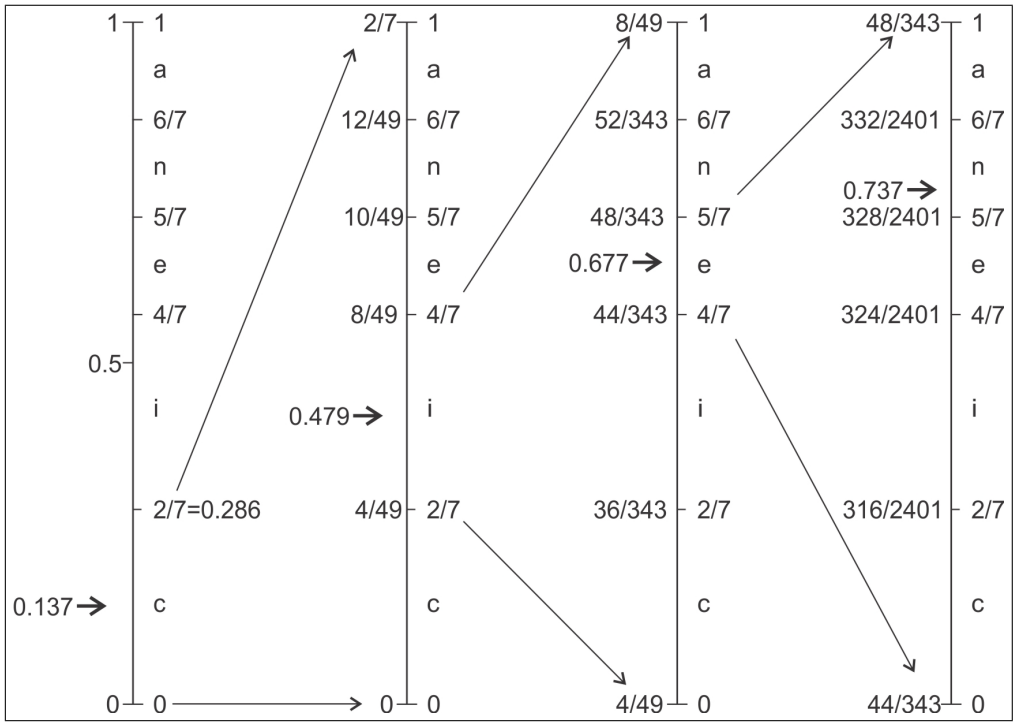


Figura 9. Proceso de decodificación del *código* $= 0,136872027$
Fuente: Elaboración propia

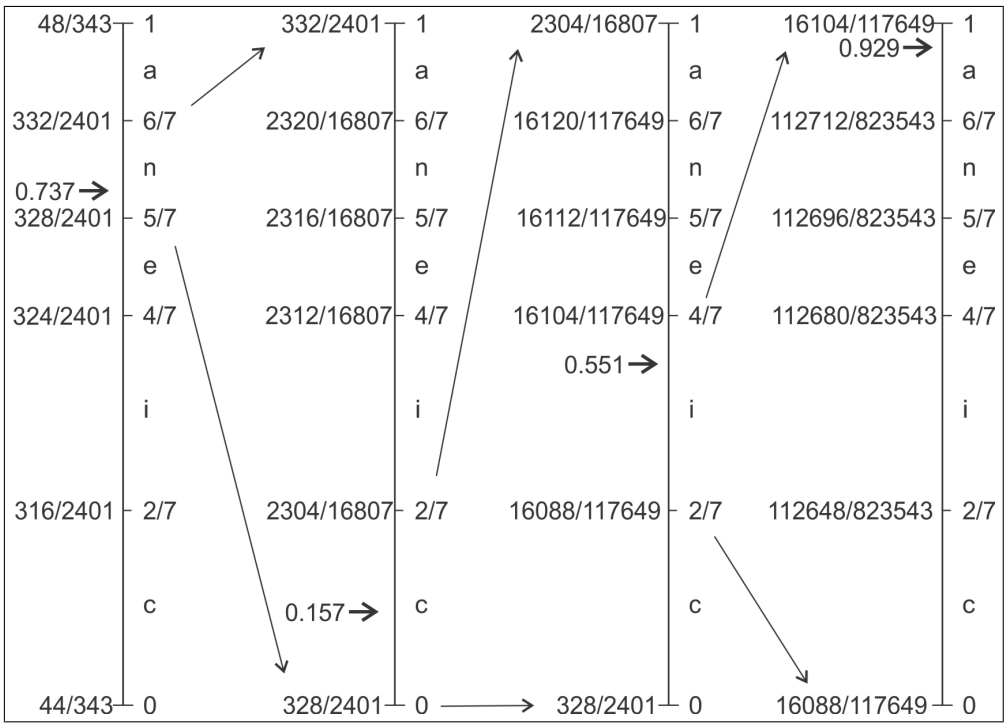


Figura 10. Continuación proceso de decodificación del *código* $= 0,136872027$
Fuente: Elaboración propia

Tabla 10. Abstracción al uso de aritmética entera.

Decimal	Decimal binario	Binario	Entero
1	1,00	100	4
0,75	0,11	011	3
0,5	0,10	010	2
0,25	0,01	001	1
0	0,00	000	0

Fuente: Elaboración propia

Tabla 11. Abstrayendo enteros con más dígitos binarios

Decimal binario	Binario	Entero
1,000	1000_2	8
0,110	0110_2	6
0,100	0100_2	4
0,010	0010_2	2
0,000	0000_2	0

Fuente: Elaboración propia

Si seguimos agregando más dígitos binarios por la derecha, entonces los valores de *sup* se van a inicializar con estos valores $sup = 4, 8, 16, 32, 64, \ldots = 2^k, k = 2, 3, 4, 5, 6 \ldots$ y los valores de inf se inician siempre con 0.

¿Qué sucede si $k = 8$ dígitos binarios? Véase la Tabla 12, se tendrá $sup = 100000000_2 = 256$ con 9 dígitos binarios. El intervalo debe ser $[inf, sup)$. Para uniformizar disminuimos en 1 el valor de *sup* quedando así: $sup = 2^8 - 1 = 11111111_2$. El valor inf no necesita ajuste, entonces $inf = 00000000_2 = 0$. Si *sup* y inf aparecen en el rango $[0; 127]$ entonces se codificará con 0 mientras que si aparecen en el rango $[128; 255]$ entonces se codificará con 1, ver Tabla 13.

Entonces resultan las siguientes ecuaciones, ver Ecuación 17 y Ecuación

Tabla 12. Abstrayendo enteros con 8 dígitos binarios

Decimal	Decimal binario	Binario	Entero
1	1,00000000	100000000_2	256
0,75	0,11000000	11000000_2	192
0,5	0,10000000	10000000_2	128
0,25	0,01000000	10000000_2	64
0	0,00000000	00000000_2	0

Fuente: Elaboración propia

Tabla 13. Aproximando a 8 dígitos binarios

Decimal	Entero	Binario	Entero	Binario
1	255	$\underline{1}1111111_2$		
0,75	192	$\underline{1}1000000_2$		
0,5	128	$\underline{1}0000000_2$	127	$\underline{0}1111111_2$
0,25			64	$\underline{0}1000000_2$
0			0	$\underline{0}0000000_2$

Fuente: Elaboración propia

18. El valor de *sup* siempre será mayor que el valor de inf, ver Ecuación 19.

$$sup = 2^k - 1,\ k\ =\ número\ de\ bits \tag{17}$$

$$inf = 0 \tag{18}$$

$$sup \geq inf \tag{19}$$

El codificador arimético consiste de las siguientes operaciones, ver Ecuación 20, Ecuación 21 y Ecuación 22

$$sup = sup - inf + 1 \tag{20}$$

$$sup = inf + r * frec\_acum[símbolo + 1]/frec\_acum[N] - 1 \tag{21}$$

$$inf = inf + r * frec\_acum[símbolo]/frec\_acum[N] \tag{22}$$

Sí el límite superior del símbolo a codificar es menor que 0,5 entonces devolverá como resultado de codificación 0, ver Ecuación 23.

$$Si\ sup < 128\ entonces\ bit\_siguiente(0) \tag{23}$$

En este caso solo es necesario reescalar el rango $[0; 0, 5)$ al rango $[0; 1)$, ver Ecuación 24 y Ecuación 25.

$$sup = 2 * sup + 1 \tag{24}$$

$$inf = 2 * inf \tag{25}$$

Si el límite inferior es mayor o igual que 0,5 entonces el resultado de codificación será 1, ver Ecuación 26. Después se tendría que disminuir los límites en 128 y multiplicar por 2, con esto se estaría reescalando el rango $[0, 5; 1)$ hacia el nuevo rango $[0; 1)$, ver Ecuación 29 y Ecuación 30.

$$Si\ inf \geq 128\ entonces\ bit\_siguiente(1) \tag{26}$$

$$sup = sup - 128 \tag{27}$$

$$inf = inf - 128 \tag{28}$$

$$sup = 2 * sup + 1 \tag{29}$$

$$inf = 2 * inf \tag{30}$$

En el caso que los límites estén en el rango $[0,25; 0,75)$, se aumenta el contador $bps = bps + 1$ ver Ecuación 31. Este contador llevará un conteo de las veces en que es ambiguo dicidir por 1 o 0. El reescalamiento consiste en disminuir 0,25 a los límites para luego duplicarlos, es decir el rango $[0,25; 0,75)$ se reescala al rango $[0; 1)$, ver Ecuación 34 y Ecuación 35.

$$Si\ sup < 192\ y\ inf \geq 64\ entonces\ bps = bps + 1 \tag{31}$$

$$sup = sup - 64 \tag{32}$$

$$inf = inf - 64 \tag{33}$$

$$sup = 2 * sup + 1 \tag{34}$$

$$inf = 2 * inf \tag{35}$$

Se muestra la codificación arimética con el uso de números enteros.

Inicializando

$$sup = 255, inf = 0, bps = 0$$

Codificando símbolo : **c**

$$r = 255 - 0 + 1 = 256$$

$$sup = 0 + 256 * \frac{2}{7} - 1 = 72$$

$$inf = 0 + 256 * \frac{0}{7} = 0$$

$$\boxed{0}$$

$$sup = 144 + 1 = 145$$

$$inf = 0$$

Codificando símbolo : **i**

$$r = 145 - 0 + 1 = 146$$

$$sup = 0 + 146 * \frac{4}{7} - 1 = 82$$

$$inf = 0 + 146 * \frac{2}{7} = 41$$

$$\boxed{0}$$

$$sup = 164 + 1 = 165$$

$$inf = 82$$

$$bps = 1$$

$$sup = 165 - 64 = 101$$

$$inf = 82 - 64 = 18$$

$$sup = 202 + 1 = 203$$

$$inf = 36$$

Codificando símbolo : **e**

$$r = 203 - 36 + 1 = 168$$

$$sup = 36 + 168 * \frac{5}{7} - 1 = 155$$

$$inf = 36 + 168 * \frac{4}{7} = 132$$

$$\boxed{1}\boxed{0}, bps = 0$$

$$sup = 155 - 128 = 27$$

$$inf = 132 - 128 = 4$$

$$sup = 2 * 27 + 1 = 55$$

$$inf = 8$$

$$\boxed{0}$$

$$sup = 110 + 1 = 111$$

$$inf = 16$$

$$\boxed{0}$$

$$sup = 222 + 1 = 223$$

$$inf = 32$$

Codificando símbolo : **n**

$$r = 223 - 32 + 1 = 192$$

$$sup = 32 + 192 * \frac{6}{7} - 1 = 195$$

$$inf = 32 + 192 * \frac{5}{7} = 169$$

$$\boxed{1}$$

$$sup = 195 - 128 = 67$$

$$inf = 169 - 128 = 41$$

$$sup = 2 * 67 + 1 = 135$$

$$inf = 82$$

$$bps = 1$$

$$sup = 135 - 64 = 71$$

$$inf = 82 - 64 = 18$$

$$sup = 142 + 1 = 143$$

$$inf = 36$$

Codificando símbolo : **c**

$$r = 143 - 36 + 1 = 108$$

$$sup = 36 + 108 * \frac{2}{7} - 1 = 65$$

$$inf = 36 + 108 * \frac{0}{7} = 36$$

$$\boxed{0}\boxed{1}, bps = 0$$

$$sup = 130 + 1 = 131$$

$$inf = 72$$

$$bps = 1$$

$$sup = 131 - 64 = 67$$

$$inf = 72 - 64 = 8$$

$$sup = 2 * 67 + 1 = 135$$

$$inf = 16$$

Codificando símbolo : **i**

$$r = 135 - 16 + 1 = 120$$

$$sup = 16 + 120 * \frac{4}{7} - 1 = 83$$

$$inf = 16 + 120 * \frac{2}{7} = 50$$

$$\boxed{0}\boxed{1}, bps = 0$$

$$sup = 166 + 1 = 167$$

$$inf = 100$$

$$bps = 1$$

$$sup = 167 - 64 = 103$$

$$inf = 100 - 64 = 36$$

$$sup = 2 * 103 + 1 = 207$$

$$inf = 72$$

Codificando símbolo : **a**

$$r = 207 - 72 + 1 = 136$$

$$sup = 72 + 136 * \frac{7}{7} - 1 = 207$$

$$inf = 72 + 136 * \frac{6}{7} = 188$$

$$\boxed{1}\boxed{0}, bps = 0$$

$$sup = 207 - 128 = 79$$

$$inf = 188 - 128 = 60$$

$$sup = 158 + 1 = 159$$

$$inf = 120$$

$$bps = 1$$

$$sup = 159 - 64 = 95$$

$$inf = 120 - 64 = 56$$

$$sup = 2 * 95 + 1 = 191$$

$$inf = 112$$

$$bps = 2$$

$$sup = 191 - 64 = 127$$

$$inf = 112 - 64 = 48$$

$$sup = 2 * 127 + 1 = 255$$

$$inf = 96$$

$$\frac{sup + inf + 1}{2} = 176 = \boxed{1}0110000_2$$

El resultado de la codificación aritmética para este caso da como resultado: $código = 00100010\ 101101\boxed{0}\boxed{0}\ 0110000|0$.

El proceso de decodificación con aritmética entera es inverso a la codificación.

La decodificación comienza con los siguientes valores iniciales, ver Ecuación 36 y Ecuación 37. El valor *val* comenzará con el primer byte de la archivo comprimido, ver Ecuación 38.

$$sup = 2^k - 1,\ k\ =\ número\ de\ bits \tag{36}$$

$$inf = 0 \tag{37}$$

$$val = (1er\ byte\ del\ archivo\ comprimido) \tag{38}$$

Las operaciones en que consiste la decodificación son, ver Ecuación 39, Ecuación 40, Ecuación 41, Ecuación 42 y Ecuación 43

$$r = sup - inf + 1 \tag{39}$$

$$acum = ((val - inf + 1) * frec\_acum[N] - 1)/r \tag{40}$$

$$símbolo = obtener\_símbolo(acum) \tag{41}$$

$$sup = inf + r * frec\_acum[símbolo + 1]/frec\_acum[N] - 1 \tag{42}$$

$$inf = inf + r * frec\_acum[símbolo]/frec\_acum[N] \tag{43}$$

Si el límite superior del símbolo a codificar es menor que 0,5 entonces solo es necesario reescalar el rango [0; 0,5) hacia al rango [0; 1), ver Ecuación 44 y Ecuación 45.

$$sup = 2 * sup + 1 \tag{44}$$

$$inf = 2 * inf \tag{45}$$

Si el límite inferior es mayor o igual que 0,5 entonces se tendría que disminuir los límites en 128 y multiplicar por 2, con esto se estaría reescalando el rango $[0,5; 1)$ hacia el nuevo rango $[0; 1)$, ver Ecuación 48 y Ecuación 49.

$$sup = sup - 128 \tag{46}$$

$$inf = inf - 128 \tag{47}$$

$$sup = 2 * sup + 1 \tag{48}$$

$$inf = 2 * inf \tag{49}$$

En el caso que los límites estén en el rango $[0,25; 0,75)$, entonces el reescalamiento consiste en disminuir 0,25 a los límites para luego duplicarlos, es decir el rango $[0,25; 0,75)$ se reescala al rango $[0; 1)$, ver Ecuación 52 y

Ecuación 53.

$$sup = sup - 64 \tag{50}$$

$$inf = inf - 64 \tag{51}$$

$$sup = 2 * sup + 1 \tag{52}$$

$$inf = 2 * inf \tag{53}$$

Se muestra la decodificación arimética con el uso de números enteros.

Inicializando

$$val = 34(00100010), sup = 255, inf = 0$$

Decodificando el primer símbolo

$$r = 255 - 0 + 1 = 256$$

$$acum = \frac{(34 - 0 + 1) * 7 - 1}{256} = \frac{35 * 7 - 1}{256}$$

$$acum = \frac{244}{256} \approx 0,95$$

Símbolo : **c**

$$sup = 0 + 256 * \frac{2}{7} - 1 = 72$$

$$inf = 0 + 256 * \frac{0}{7} = 0$$

$$sup = 144 + 1 = 145$$

$$inf = 0$$

$$val = 68 + \underline{1} = 69$$

Decodificando siguiente símbolo

$$r = 145 - 0 + 1 = 146$$

$$acum = \frac{(69 - 0 + 1) * 7 - 1}{146} = \frac{70 * 7 - 1}{146}$$

$$acum = \frac{489}{146} \approx 3,34$$

Símbolo : **i**

$$sup = 0 + 146 * \frac{4}{7} - 1 = 82$$

$$inf = 0 + 146 * \frac{2}{7} = 41$$

$$sup = 164 + 1 = 165$$

$$inf = 82$$

$$val = 138 + \underline{0} = 138$$

$$sup = 165 - 64 = 101$$

$$inf = 82 - 64 = 18$$

$$val = 138 - 64 = 74$$

$$sup = 202 + 1 = 203$$

$$inf = 36$$

$$val = 148 + \underline{1} = 149$$

Decodificando siguiente símbolo

$$r = 203 - 36 + 1 = 168$$

$$acum = \frac{(149 - 36 + 1) * 7 - 1}{168} = \frac{114 * 7 - 1}{168}$$

$$acum = \frac{797}{168} \approx 4,74$$

Símbolo : **e**

$$sup = 36 + 168 * \frac{5}{7} - 1 = 155$$

$$inf = 36 + 168 * \frac{4}{7} = 132$$

$$sup = 155 - 128 = 27$$

$$inf = 132 - 128 = 4$$

$$val = 149 - 128 = 21$$

$$sup = 2 * 27 + 1 = 55$$

$$inf = 8$$

$$val = 42 + \underline{1} = 43$$

$$sup = 110 + 1 = 111$$

$$inf = 16$$

$$val = 86 + \underline{0} = 86$$

$$sup = 222 + 1 = 223$$

$$inf = 32$$

$$val = 172 + \underline{1} = 173$$

Decodificando siguiente símbolo

$$r = 223 - 32 + 1 = 192$$

$$acum = \frac{(173 - 32 + 1) * 7 - 1}{192} = \frac{142 * 7 - 1}{192}$$

$$acum = \frac{993}{192} \approx 5,17$$

Símbolo : **n**

$$sup = 32 + 192 * \frac{6}{7} - 1 = 195$$

$$inf = 32 + 192 * \frac{5}{7} = 169$$

$$sup = 195 - 128 = 67$$

$$inf = 169 - 128 = 41$$

$$val = 173 - 128 = 45$$

$$sup = 2 * 67 + 1 = 135$$

$$inf = 82$$

$$val = 90 + \underline{0} = 90$$

$$sup = 135 - 64 = 71$$

$$inf = 82 - 64 = 18$$

$$val = 90 - 64 = 26$$

$$sup = 142 + 1 = 143$$

$$inf = 36$$

$$val = 52 + \underline{0} = 52$$

Decodificando siguiente símbolo

$$r = 143 - 36 + 1 = 108$$

$$acum = \frac{(52 - 36 + 1) * 7 - 1}{108} = \frac{17 * 7 - 1}{108}$$

$$acum = \frac{118}{108} \approx 1,09$$

Símbolo : **c**

$$sup = 36 + 108 * \frac{2}{7} - 1 = 65$$

$$inf = 36 + 108 * \frac{0}{7} = 36$$

$$sup = 130 + 1 = 131$$

$$inf = 72$$

$$inf = 104 + \underline{0} = 104$$

$$sup = 131 - 64 = 67$$

$$inf = 72 - 64 = 8$$

$$val = 104 - 64 = 40$$

$$sup = 2 * 67 + 1 = 135$$

$$inf = 16$$

$$val = 80 + \underline{1} = 81$$

Decodificando siguiente símbolo

$$r = 135 - 16 + 1 = 120$$

$$acum = \frac{(81 - 16 + 1) * 7 - 1}{120} = \frac{66 * 7 - 1}{120}$$

$$acum = \frac{461}{120} \approx 3,84$$

Símbolo : **i**

$$sup = 16 + 120 * \frac{4}{7} - 1 = 83$$

$$inf = 16 + 120 * \frac{2}{7} = 50$$

$$sup = 166 + 1 = 167$$

$$inf = 100$$

$$val = 162 + \underline{1} = 163$$

$$sup = 167 - 64 = 103$$

$$inf = 100 - 64 = 36$$

$$val = 163 - 64 = 99$$

$$sup = 2 * 103 + 1 = 207$$

$$inf = 72$$

$$val = 198 + \underline{0} = 198$$

Decodificando siguiente símbolo

$$r = 207 - 72 + 1 = 136$$

$$acum = \frac{(198 - 72 + 1) * 7 - 1}{136} = \frac{127 * 7 - 1}{136}$$

$$acum = \frac{888}{136} \approx 6,52$$

Símbolo : **a**

$$sup = 72 + 136 * \frac{7}{7} - 1 = 207$$

$$inf = 72 + 136 * \frac{6}{7} = 188$$

$$sup = 207 - 128 = 79$$

$$inf = 188 - 128 = 60$$

$$val = 198 - 128 = 70$$

$$sup = 158 + 1 = 159$$

$$inf = 120$$

$$val = 140 + \underline{0} = 140$$

$$sup = 159 - 64 = 95$$

$$inf = 120 - 64 = 56$$

$$val = 140 - 64 = 76$$

$$sup = 2 * 95 + 1 = 191$$

$$inf = 112$$

$$val = 152 + \underline{0} = 152$$

$$sup = 191 - 64 = 127$$

$$inf = 112 - 64 = 48$$

$$val = 152 - 64 = 88$$

$$sup = 2 * 127 + 1 = 255$$

$$inf = 96$$

$$val = 176 + \underline{0} = 176$$

3.3 Diseño del algoritmo

3.3.1 Algoritmo codificador y decodificador basado en Witten y cols. (1987)

Se pueden observar el algoritmo computacional del codificador Witten y cols. (1987): Datos de entrada y salida en Figura 11 y el diagrama de flujo en la Figura 12.

Se pueden apreciar también el algoritmo computacional del decodificador Witten y cols. (1987): los datos de E/S se pueden ver en Figura 13 y el diagrama

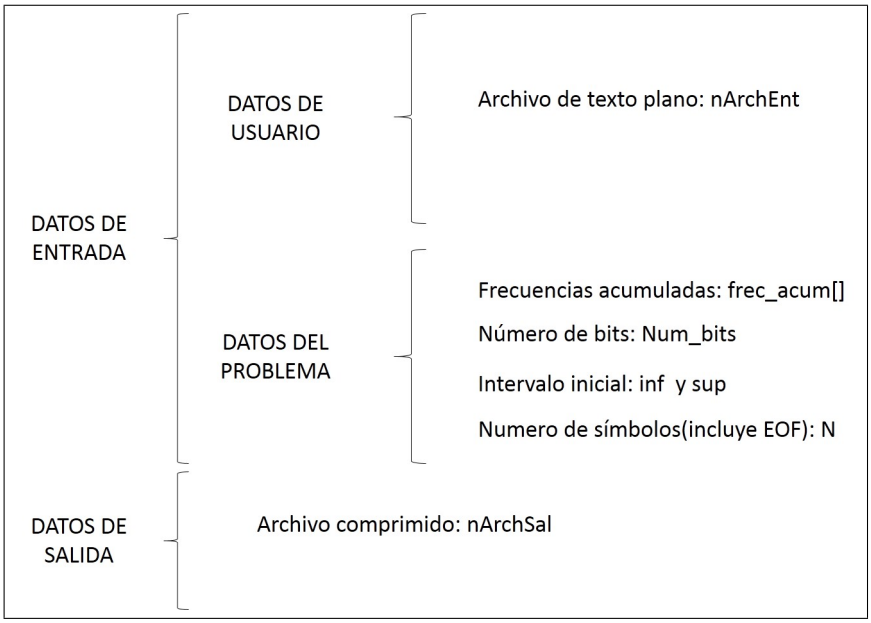


Figura 11. Datos de entrada y salida del codificador basado en Witten y cols. (1987)
Fuente: Elaboración propia

de flujo en la Figura 14.

3.3.2 Algoritmo codificador y decodificador basado en Moffat y cols. (1998)

Los algoritmos computacionales del codificador Moffat y cols. (1998): Se pueden ver datos de E/S en Figura 15 y el diagrama de flujo en Figura 16.

El algoritmo computacional del decodificador Moffat y cols. (1998): Se pueden ver datos de E/S en Figura 17, el diagrama de flujo en Figura 18 y Figura 19.

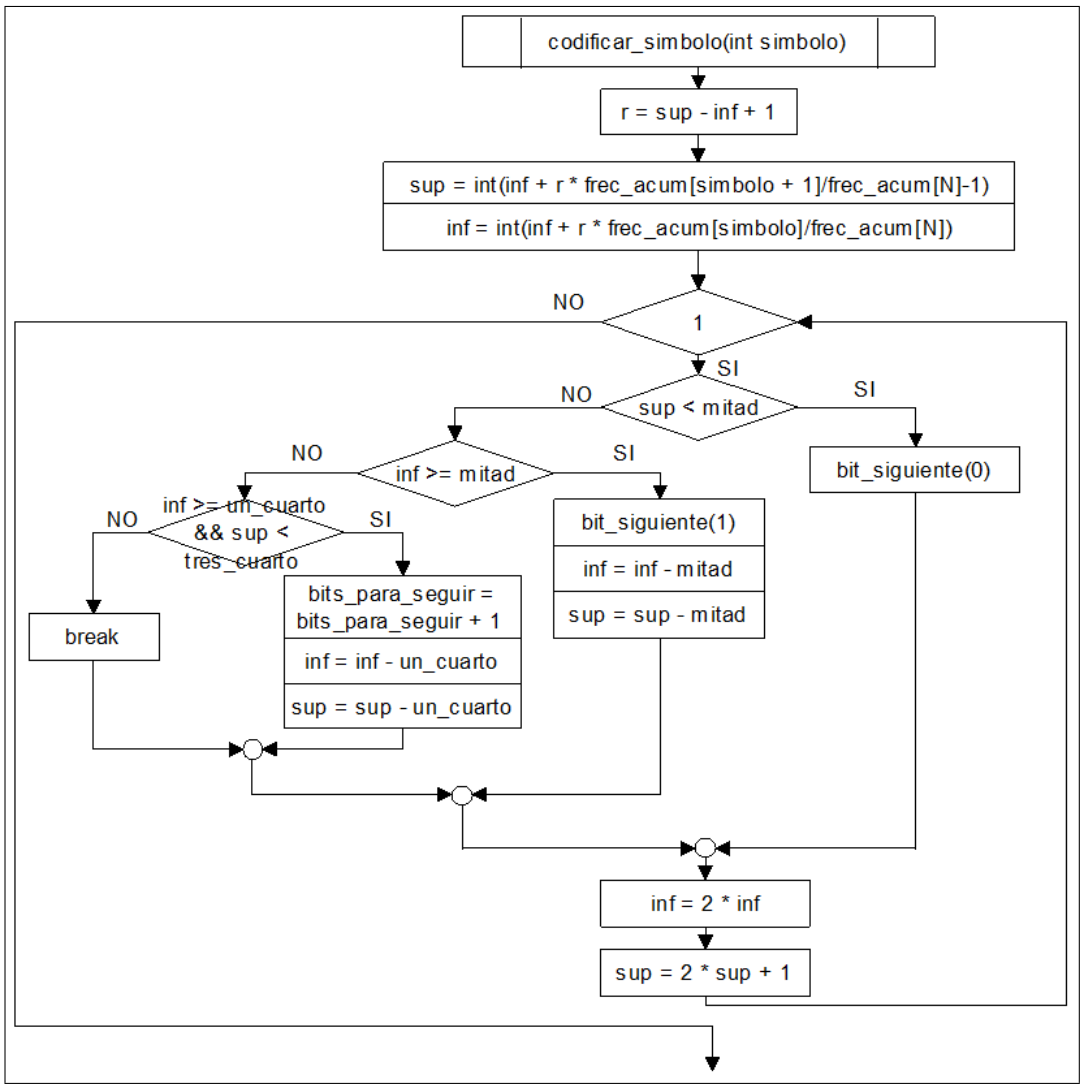


Figura 12. Diagrama de flujo del codificador basado en Witten y cols. (1987)
Fuente: Elaboración propia

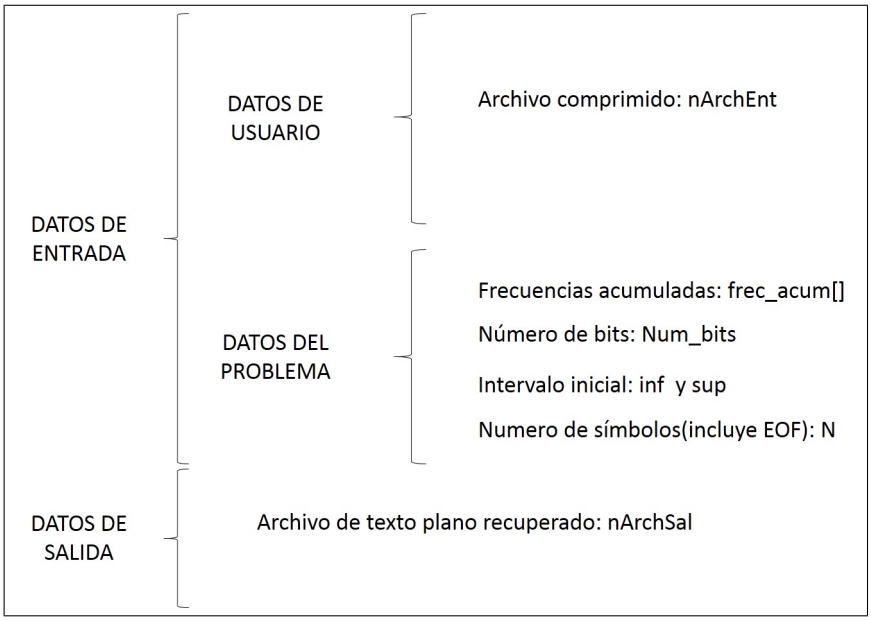


Figura 13. Datos de entrada y salida del decodificador basado en Witten y cols. (1987)
Fuente: Elaboración propia

3.4 Implementación del algoritmo

Se han implementado los codificadores y sus respectivos decodificadores, ver en Anexo C

Se implementaron los modelos estadísticos en el lenguaje de programación C++, ver Anexo D.

Los programas para medición del desempeño se implementaron en lenguaje C++, se puede encontrar en Anexo E.

El modelo estático necesita de un arreglo de frecuencias de aparición de símbolos con anticipación. Este arreglo se determinó mediante el conteo de los caracteres de los archivos que conforman la colección “Literatura peruana”,

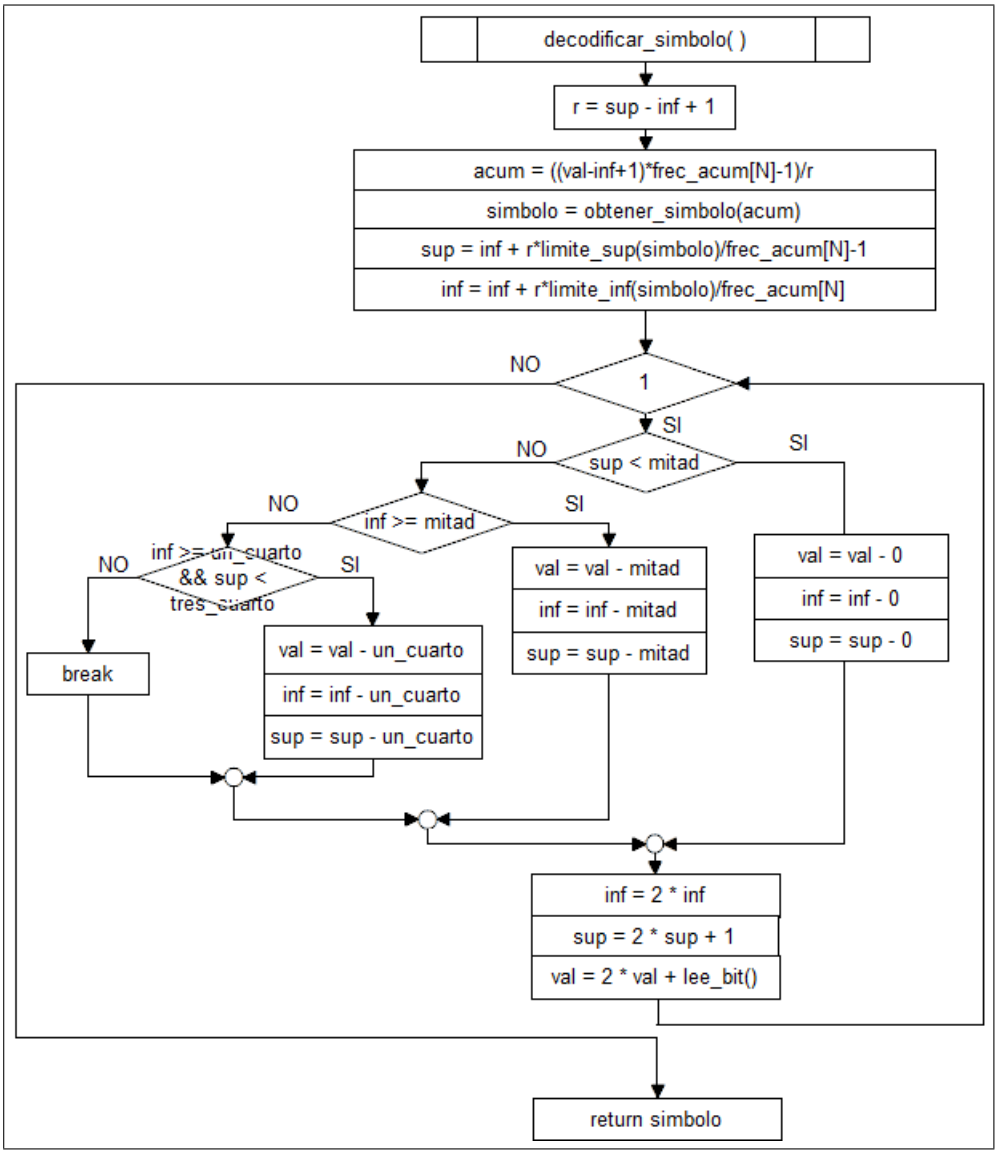


Figura 14. Diagrama de flujo del decodificador basado en Witten y cols. (1987)
Fuente: Elaboración propia

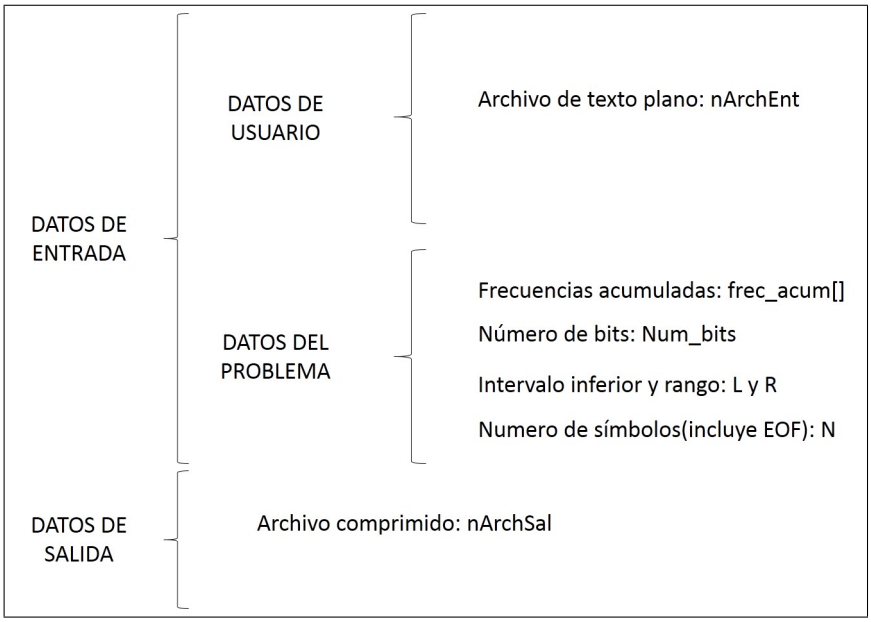


Figura 15. Datos de entrada y salida del codificador basado en Moffat y cols. (1998)
Fuente: Elaboración propia

ver Anexo F, obteniendose así los modelos estadísticos ver Anexo K.

Las respectivas gráficas de frecuencias de símbolos para cada uno de los archivos antes de la compresión se pueden ver en G, después de la compresión se pueden ver Anexo H y la comparación de antes y después de la compresión se puede ver en I.

Las gráficas muestran claramente que existen símbolos con más frecuencia que otros. Se determinó el símbolo más frecuente, ver Anexo J, siendo este la letra “e” con 12,01 % seguida de la letra “a” con 11,87 %.

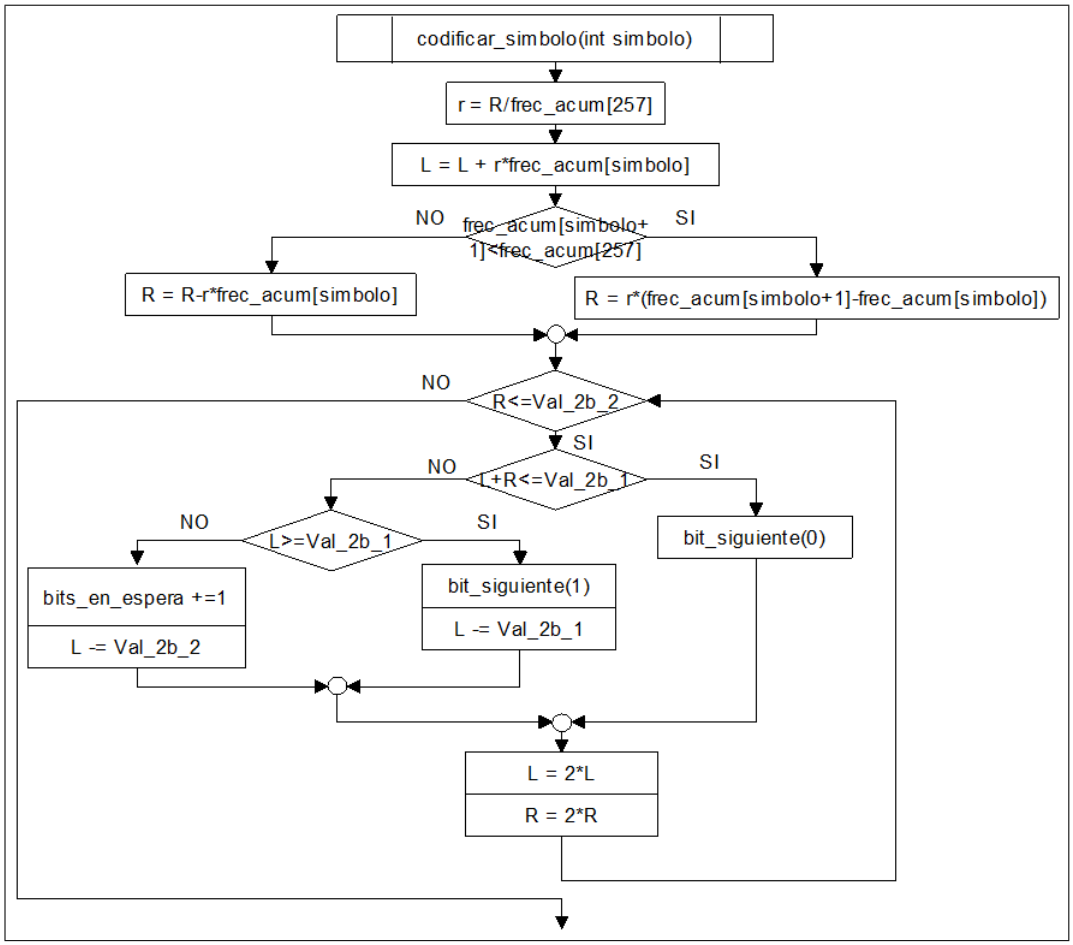


Figura 16. Diagrama de flujo del codificador basado en Moffat y cols. (1998)
Fuente: Elaboración propia

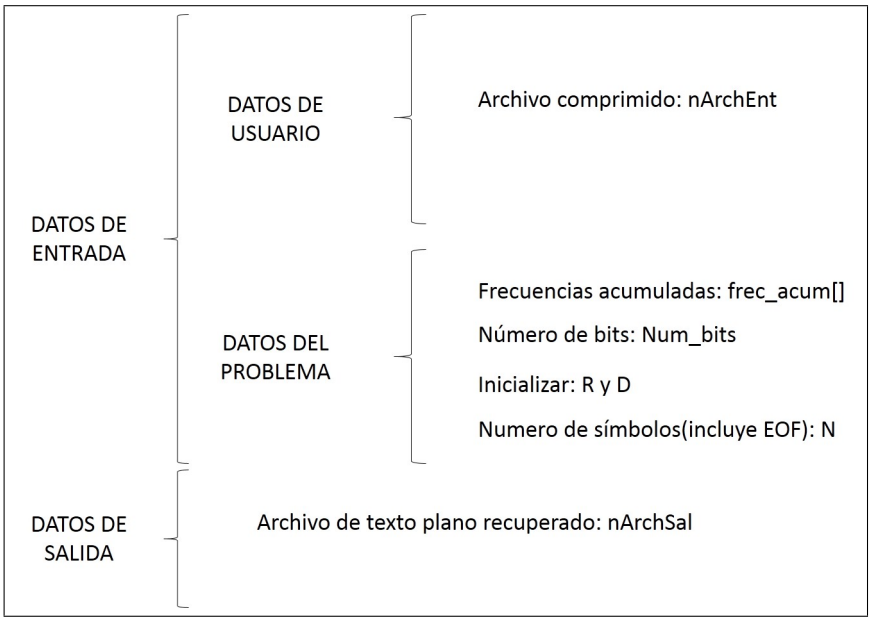


Figura 17. Datos de entrada y salida del decodificador basado en Moffat y cols. (1998)
Fuente: Elaboración propia

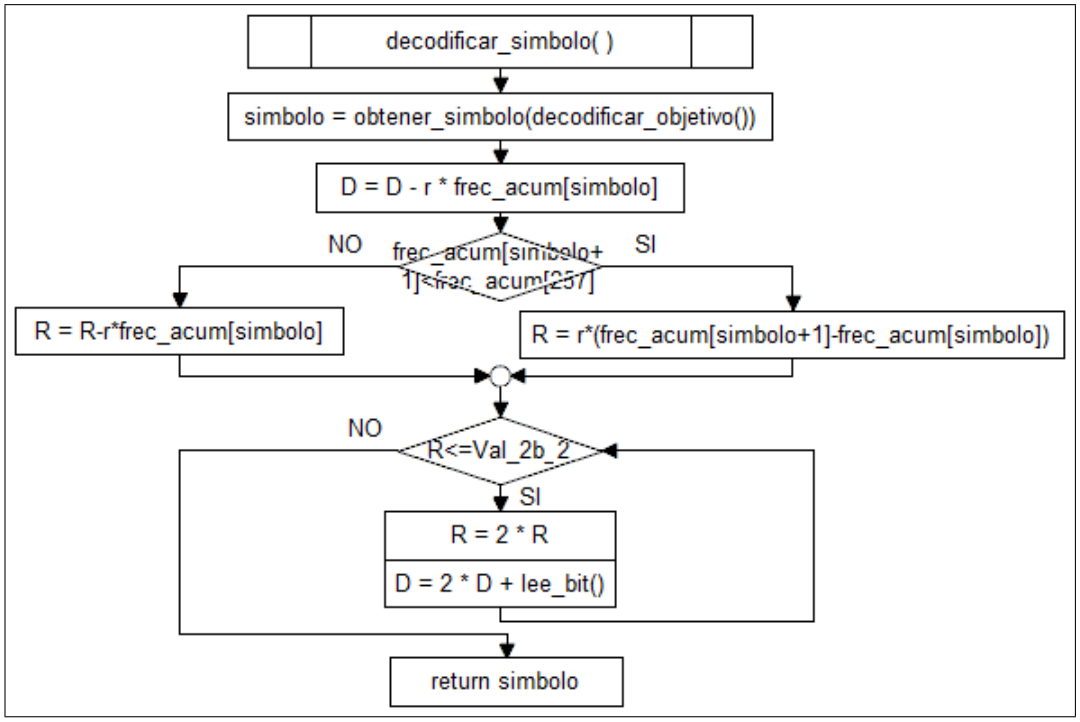


Figura 18. Diagrama de flujo del decodificador basado en Moffat y cols. (1998)
Fuente: Elaboración propia

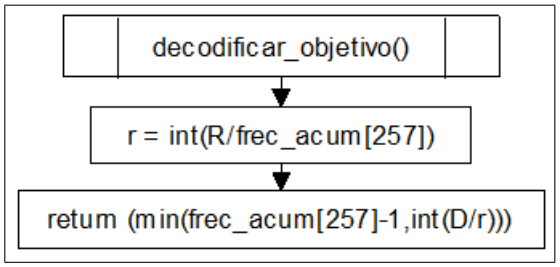


Figura 19. Función utilizada por el decodificador Moffat y cols. (1998)
Fuente: Elaboración propia

Capítulo IV: Resultados

Resultados bajo el modelo estático. Se observa los tamaños después de la compresión, ver Tabla 14.

Tabla 14. Tamaños después de la compresión bajo el modelo estático

Nombre de archivo	Witten y cols. (1987), (bytes)	Moffat y cols. (1998), (bytes)
7ensayos.txt	517 320	517 321
agua.txt	31 270	31 270
alma_america.txt	201 038	201 038
caliz.txt	19 426	19 426
carmelo.txt	14 460	14 460
comentarios_reales.txt	163 592	163 592
heraldos.txt	34 158	34 158
hermanos_ayar.txt	20 029	20 029
humanos.txt	59 377	59 377
prosa.txt	18 545	18 545
tradiciones.txt	2 388 684	2 388 685
trilce.txt	40 157	40 157

Fuente: Elaboración propia

Se determina la tasa de compresión, ver Tabla 15.

El tiempo de compresión se puede ver en la Tabla 16 y el tiempo de descompresión se aprecia en Tabla 17.

Bajo el modelo semi-estático. Los resultados tras la compresión al conbinar con los codificadores aritméticos se pueden ver en la Tabla 18.

Se aprecia la tasa de compresión, ver Tabla 19.

Tabla 15. Tasa de compresión bajo el modelo estático

Nombre de archivo	Witten y cols. (1987), (%)	Moffat y cols. (1998), (%)
7ensayos.txt	40,28	40,28
agua.txt	68,84	68,84
alma_america.txt	70,23	70,23
caliz.txt	69,66	69,66
carmelo.txt	66,21	66,21
comentarios_reales.txt	65,33	65,33
heraldos.txt	69,56	69,56
hermanos_ayar.txt	66,42	66,42
humanos.txt	69,08	69,08
prosa.txt	66,33	66,33
tradiciones.txt	40,40	40,40
trilce.txt	67,22	67,22

Fuente: Elaboración propia

Tabla 16. El tiempo de compresión bajo el modelo estático.

Nombre de archivo	Witten y cols. (1987), (ms)	Moffat y cols. (1998), (ms)
7ensayos.txt	354,89	321,98
agua.txt	29,8	28,86
alma_america.txt	109,83	101,1
caliz.txt	24,02	23,39
carmelo.txt	21,22	21,52
comentarios_reales.txt	94,71	87,98
heraldos.txt	31,5	31,66
hermanos_ayar.txt	24,48	24,02
humanos.txt	43,51	42,27
prosa.txt	23,07	23,86
tradiciones.txt	1 573,11	1 402,74
trilce.txt	34,79	33,24

Fuente: Elaboración propia

Tabla 17. El tiempo de descompresión bajo el modelo estático.

Nombre de archivo	Witten y cols. (1987), (ms)	Moffat y cols. (1998), (ms)
7ensayos.txt	549,28	467,22
agua.txt	41,96	39,62
alma_america.txt	168,48	153,81
caliz.txt	31,03	29,94
carmelo.txt	27,14	26,82
comentarios_reales.txt	153,5	142,59
heraldos.txt	43,21	41,02
hermanos_ayar.txt	32,44	31,51
humanos.txt	62,25	59,27
prosa.txt	31,51	30,11
tradiciones.txt	2 448,73	2 065,74
trilce.txt	49,29	46,16

Fuente: Elaboración propia

Tabla 18. Tamaños después de la compresión bajo el modelo semi-estático

Nombre de archivo	Witten y cols. (1987), (bytes)	Moffat y cols. (1998), (bytes)
7ensayos.txt	588 282	588 282
agua.txt	26 960	26 960
alma_america.txt	172 023	172 023
caliz.txt	16 878	16 878
carmelo.txt	12 796	12 796
comentarios_reales.txt	141 936	141 936
heraldos.txt	29 489	29 489
hermanos_ayar.txt	17 567	17 568
humanos.txt	51 076	51 076
prosa.txt	16 383	16 383
tradiciones.txt	2 722 219	2 722 219
trilce.txt	34 775	34 775

Fuente: Elaboración propia

Tabla 19. Tasa de compresión bajo el modelo semi-estático

Nombre de archivo	Witten y cols. (1987), (%)	Moffat y cols. (1998), (%)
7ensayos.txt	45,80	45,80
agua.txt	59,35	59,35
alma_america.txt	60,09	60,09
caliz.txt	60,52	60,52
carmelo.txt	58,60	58,60
comentarios_reales.txt	56,68	56,68
heraldos.txt	60,05	60,05
hermanos_ayar.txt	58,26	58,26
humanos.txt	59,42	59,42
prosa.txt	58,59	58,59
tradiciones.txt	46,04	46,04
trilce.txt	58,21	58,21

Fuente: Elaboración propia

El tiempo de compresión se pueden ver en Tabla 20 y el tiempo de descompresión se puede ver en la Tabla 21

Bajo el modelo adaptativo. El resultado de los tamaños de compresión resultantes se puede observar en la Tabla 22.

Véase la tasa de compresión, en Tabla 23.

El tiempo de compresión y descompresión se pueden ver en Tabla 24 y Tabla 25 respectivamente.

Se selecciona el archivo de mayor y de menor tamaño para analizar la compresión, debido a que los 100 datos recopilados del tiempo compresión mediante el programa (rcrono, ver Anexo 5.1) presentan menor dispersión o ruido en archivos de gran tamaño que en archivos de menor cantidad de bytes.

Tabla 20. El tiempo de compresión bajo el modelo semi-estático

Nombre de archivo	Witten y cols. (1987), (ms)	Moffat y cols. (1998), (ms)
7ensayos.txt	487,34	453,47
agua.txt	34,14	33,84
alma_america.txt	130,4	122,62
caliz.txt	26,98	26,83
carmelo.txt	24,48	24,18
comentarios_reales.txt	112,95	106,4
heraldos.txt	35,87	34,95
hermanos_ayar.txt	28,39	28,08
humanos.txt	50,39	48,21
prosa.txt	26,66	26,83
tradiciones.txt	2 184,47	2 025,97
trilce.txt	40,27	38,84

Fuente: Elaboración propia

Tabla 21. El tiempo de descompresión bajo el modelo semi-estático

Nombre de archivo	Witten y cols. (1987), (ms)	Moffat y cols. (1998), (ms)
7ensayos.txt	549,27	477,68
agua.txt	42,11	38,83
alma_america.txt	166,44	152,1
caliz.txt	31,04	29,33
carmelo.txt	27,91	26,05
comentarios_reales.txt	150,55	138,21
heraldos.txt	43,52	39,93
hermanos_ayar.txt	32,6	31,35
humanos.txt	62,08	57,72
prosa.txt	30,58	28,87
tradiciones.txt	2 445,93	2 115,81
trilce.txt	48,52	44,92

Fuente: Elaboración propia

Tabla 22. Tamaños después de la compresión bajo el modelo adaptativo

Nombre de archivo	Witten y cols. (1987), (bytes)	Moffat y cols. (1998), (bytes)
7ensayos.txt	513 812	513 812
agua.txt	26 050	26 050
alma_america.txt	166 642	166 642
caliz.txt	16 365	16 365
carmelo.txt	12 280	12 280
comentarios_reales.txt	136 329	136 329
heraldos.txt	28 551	28 551
hermanos_ayar.txt	16 891	16 891
humanos.txt	49 543	49 543
prosa.txt	15 679	15 679
tradiciones.txt	2 380 448	2 380 449
trilce.txt	33 346	33 346

Fuente: Elaboración propia

Tabla 23. Tasa de compresión bajo el modelo adaptativo

Nombre de archivo	Witten y cols. (1987), (%)	Moffat y cols. (1998), (%)
7ensayos.txt	40,01	40,01
agua.txt	57,35	57,35
alma_america.txt	58,21	58,21
caliz.txt	58,69	58,69
carmelo.txt	56,23	56,23
comentarios_reales.txt	54,44	54,44
heraldos.txt	58,14	58,14
hermanos_ayar.txt	56,02	56,02
humanos.txt	57,64	57,64
prosa.txt	56,08	56,08
tradiciones.txt	40,26	40,26
trilce.txt	55,82	55,82

Fuente: Elaboración propia

Tabla 24. El tiempo de compresión bajo el modelo adaptativo

Nombre de archivo	Witten y cols. (1987), (ms)	Moffat y cols. (1998), (ms)
7ensayos.txt	1 357,85	1 319,61
agua.txt	66,29	65,05
alma_america.txt	326,98	321,51
caliz.txt	47,11	45,7
carmelo.txt	40,08	39,93
comentarios_reales.txt	286,41	279,71
heraldos.txt	69,57	68,47
hermanos_ayar.txt	49,77	48,2
humanos.txt	109,04	108,12
prosa.txt	47,74	45,7
tradiciones.txt	6 181,35	6 001,04
trilce.txt	81,12	79,24

Fuente: Elaboración propia

Tabla 25. El tiempo de descompresión bajo el modelo adaptativo

Nombre de archivo	Witten y cols. (1987), (ms)	Moffat y cols. (1998), (ms)
7ensayos.txt	1 539,56	1 427,39
agua.txt	75,19	71,29
alma_america.txt	386,74	357,24
caliz.txt	52,27	49,92
carmelo.txt	44,93	43,36
comentarios_reales.txt	342,26	316,34
heraldos.txt	79,87	74,41
hermanos_ayar.txt	55,22	52,88
humanos.txt	127,13	120,74
prosa.txt	51,79	50,22
tradiciones.txt	7 015,94	6 490,86
trilce.txt	93,45	87,35

Fuente: Elaboración propia

Objetivo 1: Identificar el nivel de correlación del modelo estadístico con el desempeño del compresor aritmético.

Tabla 26. Nivel de relación del modelo estadístico con el desempeño del compresor

	"tradiciones.txt"		"carmelo.txt"		CV
	Tiempo (ms)	Tasa	Tiempo (ms)	Tasa	
WE	4 021,84	40,40 %	48,36	66,21 %	0,17128
WS	4 630,40	46,04 %	52,39	58,60 %	0,091318
WA	13 197,29	40,26 %	85,01	56,23 %	0,12250
ME	3 468,48	40,40 %	48,34	66,21 %	0,17128
MS	4 141,78	46,04 %	50,23	58,60 %	0,091318
MA	12 491,90	40,26 %	83,29	56,23 %	0,12250

Fuente: Elaboración propia

El modelo adaptativo ofrece mejor nivel de compresión, el modelo estático brinda rapidez en la compresión y el modelo semi estático ofrece menor CV en sus tasas de compresión (estable).

Objetivo 2: Determinar la influencia del algoritmo codificador aritmético en el desepeño del compresor aritmético.

Tabla 27. Influencia del codificador arimético con el desempeño del compresor

	"tradiciones.txt"		"carmelo.txt"		CV
	Tiempo (ms)	Tasa	Tiempo (ms)	Tasa	
EW	4 021,84	40,40 %	48,36	66,21 %	0,17128
EM	3 468,48	40,40 %	48,34	66,21 %	0,17128
SW	4 630,40	46,04 %	52,39	58,60 %	0,091318
SM	4 141,78	46,04 %	50,23	58,60 %	0,091318
AW	13 197,29	40,26 %	85,01	56,23 %	0,12250
AM	12 491,90	40,26 %	83,29	56,23 %	0,12250

Fuente: Elaboración propia

El codificador de Moffat y cols. (1998) es más rapido, no hay diferencias en sus tasas de compresión ni tampoco en sus *Coeficiente de Variación* (CV).

Hasta ahora no se ha definido el modelo de compresión más eficiente. Este será analizado utilizando el Cuadrante Mágico de Gartner *Gartner Magic Quadrants* (GMQ). Se aplicará en dos archivos, el de tamaño pequeño y el de tamaño grande.

El porcentaje de espacio ahorrado del archivo "carmelo.txt" se encuentra en la Tabla 28.

Tabla 28. % de espacio ahorrado del archivo "carmelo.txt"

Modelo	Witten y cols. (1987)	Moffat y cols. (1998)
estático	33,79 %	33,79 %
semi-estático	41,40 %	41,40 %
adaptativo	43,77 %	43,77 %

Fuente: Elaboración propia

El Cuadrante Mágico de Gartner para el archivo "camelo.txt", ver Figura 20, se grafica haciendo uso de los tiempos de compresión y el porcentaje de espacio ahorrado de la Tabla 29.

Tabla 29. Datos para graficar Cuadrante Mágico de Gartner para "camelo.txt"

Modelo de compresión	Tiempo de compresión (ms)	% de espacio ahorrado
EW	48,36	33,79 %
EM	48,34	33,79 %
SW	52,39	41,40 %
SM	50,23	41,40 %
AW	85,01	43,77 %
AM	83,29	43,77 %

Fuente: Elaboración propia

El porcentaje de espacio ahorrado para el archivo "tradiciones.txt" se encuentra en la Tabla 30.

Se hace uso del porcentaje de espacio ahorrado y los tiempos de compre-

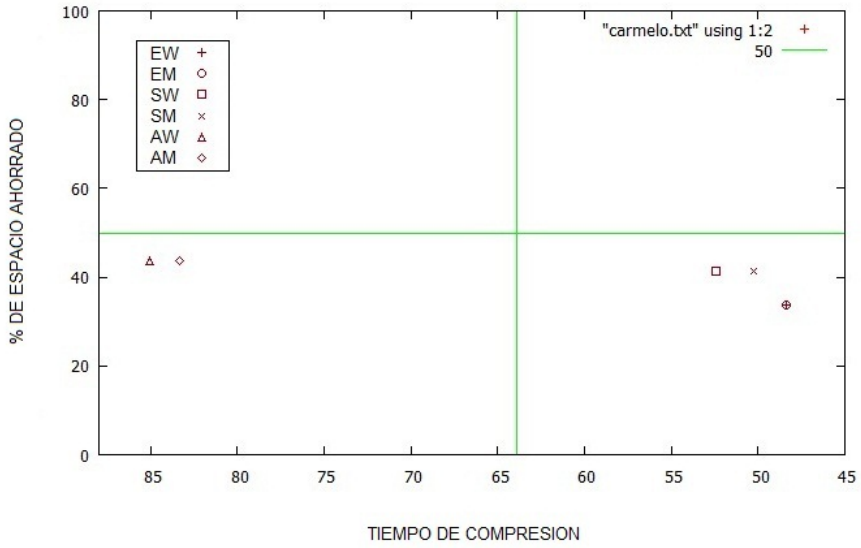


Figura 20. Cuadrante Mágico de Gartner para el archivo "carmelo.txt"
Fuente: Elaboración propia

Tabla 30. % de espacio ahorrado del archivo "tradiciones.txt"

Modelo	Witten y cols. (1987)	Moffat y cols. (1998)
estático	59,6 %	59,6 %
semi-estático	53,96 %	53,96 %
adaptativo	59,74 %	59,74 %

Fuente: Elaboración propia

Tabla 31. Datos para graficar Cuadrante Mágico de Gartner para “tradiciones.txt”

Modelo de compresión	Tiempo de compresión (ms)	% de espacio ahorrado
EW	4021,84	59,6 %
EM	3468,48	59,6 %
SW	4630,40	53,96 %
SM	4141,78	53,96 %
AW	13197,29	59,74 %
AM	12491,9	59,74 %

Fuente: Elaboración propia

sión de la Tabla 31 para graficar el Cuadrante Mágico de Gartner del archivo “tradiciones.txt”, ver Figura 21.

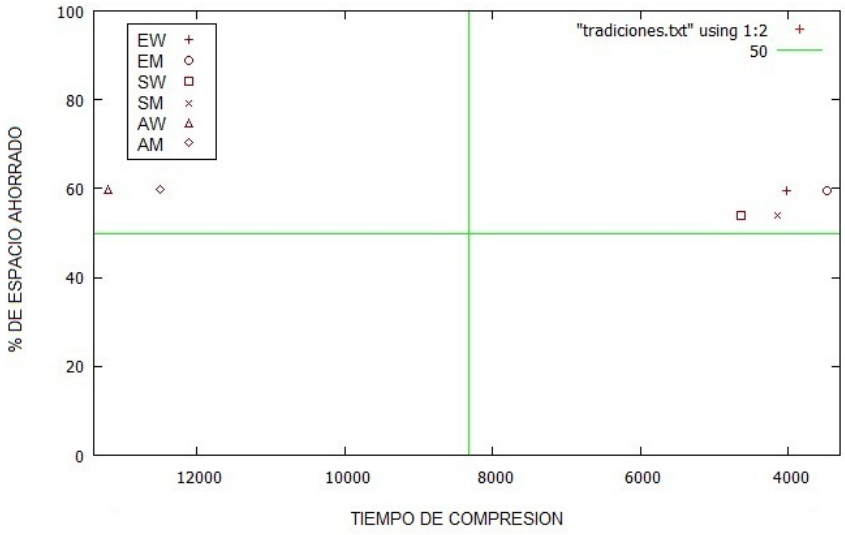


Figura 21. Cuadrante Mágico de Gartner para el archivo “tradiciones.txt”
Fuente: Elaboración propia

Capítulo V: Discusiones

En los resultados de compresión se observa una combinación de los tres modelos (estático, semi-estático y dinámico) con los dos codificadores, originando seis modelos de compresión aritmética.

El modelo estadístico adaptativo ofrece mejor tasa de compresión convirtiendose en una alternativa interesante para futuros proyectos relacionados a la compresión, el modelo estático brinda rapidez en la compresión pero no es una alternativa prometedora para la compresión de datos su uso puede ser orientado a la criptologia. El modelo semi-estático brinda menor CV que lo convierte en el más confiable.

El codificador de Moffat y cols. (1998) es más rápido que el codificador de Witten y cols. (1987), esto se debe a que optimiza sus operaciones aritmeticas, reduciendo algunas operaciones redundantes. No hay diferencias en las tasas de compresion ni en su CV.

En general, el modelo de compresión aritmético más rápido es, el codificador basado en Moffat y cols. (1998) bajo el modelo estático y el más lento es el codificador basado en Witten y cols. (1987) bajo el modelo adaptativo.

El modelo de compresión arimético que brinda mejores tasas de compresión estan bajo el modelo adaptativo.

El modelo de compresión con menor CV en sus tasas de compresión son

los dos codificadores bajo el modelo semi-estático.

Considerando el resultado de los cuadrantes mágicos de Gartner, ver Figura 20 y Figura 21, los tiempos de compresión, las tasas de compresión y los Coeficientes de Variación de las tasas de compresión ver Tabla 26 y Tabla 27, se puede inducir que el más eficiente es el codificador basado en Moffat y cols. (1998) bajo el modelo semi-estático.

Conclusiones

1.- Existe correlación directa entre el modelo estadístico y el desempeño de la compresión basado en caracteres de texto plano.

2.- El codificador aritmético influye directamente en el tiempo de la compresión basado en caracteres de texto plano.

3.- El uso de un modelo de compresión artimético influye significativamente en el desempeño de la compresión basada en caracteres de texto plano.

Recomendaciones

1.- Se recomienda a los investigadores en computación de la Universidad Nacional Jorge Basadre Grohmann el uso apropiado de las estructuras de datos en la parte del modelo estadístico adaptativo para actualizar, almacenar y acceder a las frecuencias de los símbolos rápidamente y asi reducir el tiempo de compresión. Por ejemplo el uso de Splay Trees (Jones, 1988), Binary Indexed Tree (Fenwick, 1994) o Cumulative Frequency Matrix (Doshi y Gandhi, 2012).

2.- Se recomienda a los investigadores en computación de la Universidad Nacional Jorge Basadre Grohmann considerar la compresión de cada carácter bajo diferentes contexto de probabilidades, para mejorar la tasa de compresión.

3.- Se recomienda a los investigadores en computación de la Universidad Nacional Jorge Basadre Grohmann la utilización de operaciones a nivel de bits (desplazamientos de bits en vez de multiplicaciones), para disminuir el tiempo de compresión.

4.- Se recomienda a los investigadores en computación de la Universidad Nacional Jorge Basadre Grohmann considerar estructuras de datos dinámica (listas enlazadas o arboles binarios), para mejorar el tiempo y tasa de compresión (Nelson y Gailly, 1995).

5.- Se recomienda a los investigadores en computación de la Universidad Na-

cional Jorge Basadre Grohmann el uso de la compresión basada en palabras en vez de caracteres, para mejorar la compresión más aún (Moffat y cols., 1998).

Referencias bibliográficas

Bernstein, J. H. (2009). The data-information-knowledge-wisdom hierarchy and its antithesis. *Digital Library of Information Science and Technology (DLIST)*.

Burrows, M., y Wheeler, D. J. (1994, May). *A block-sorting lossless data compression algorithm.* (Research report n.º 124). 130 Lytton Avenue Palo Alto, California 94301: Digital Equipment Corporation Systems Research Center. (HP Labs Technical Reports)

Christensson, P. (2012, April). *Unicode definition.* April 20, 2012. Descargado de `http://www.techterms.com/definition/unicode`

Cover, T. M., y Thomas, J. A. (2006). *Elements of information theory* (2nd ed.). Hoboken, New Jersey: John Wiley & Sons, Inc.

Doshi, J., y Gandhi, S. (2012, March). Implementing a novel data structure for maintaining cumulative frequency of symbols. *International Journal of Computer Applications*, *41*(15).

Fenwick, P. M. (1994). A new data structure for cumulative frequency tables. *Software: Practice and Experience*, *24*(3), 327-336.

Forouzan, B. A. (2007). *Data communications and networking* (4th ed.; R. Olson, Ed.). Alan R. Apt.

Gall, D. L. (1991, April). Mpeg: a video compression standard for multimedia applications. *Communications of the ACM*, *34*(4), 46-58.

Gonzáles-Ruiz, V. (2000). *Compresión reversible y transmisión de inágenes* (Tesis de Doctorado). Universidad de Almería.

Hernández Sampieri, R., Fernández Collado, C., y Baptista Lucio, P. (2006). *Metodología de la investigación* (4th ed.). McGraw-Hill Interamericana.

Howard, P. G., y Vitter, J. S. (1994). Arithmetic coding for data compression. *Proceedings of the IEEE*, *82*(6), 857-865.

Huffman, D. A. (1952, September). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, *40*(9), 1098 - 1101.

Jones, D. W. (1988, August). Application of splay trees to data compression. *Communications of the ACM*, *31*(8), 996-1007.

Langdon, G. (1984). An introduction to arithmetic coding. *IBM Journal of Research and Development*, *28*(2), 135-149.

Lelewer, D. A., y Hirschberg, D. S. (1987). Data compression. *ACM Computing Surveys*, *19*(3), 261-296. Descargado de `http://www.ics.uci.edu/ dan/pubs/DataCompression.html`

Moffat, A., Neal, R. M., y Witten, I. H. (1998, Julio). Arithmetic coding revisited. *ACM Transactions on Information Systems*, *16*(3), 256-294.

Nelson, M., y Gailly, J.-L. (1995). *The data compression book* (2nd ed.; M. . T. Books, Ed.). IDG Books Worldwide, Inc.

Pu, I. M. (2006). *Fundamental data compression* (1st ed.). Butterworth-Heinemann.

Rowley, J. (2007, April). The wisdom hierarchy: representations of the dikw hierarchy. *Journal of Information Science*, *33*(2), 163-180. Descargado de `http://wisdomresearch.org/forums/storage/26/220/rowley_jis_042007.pdf`

Salomon, D. (2008). *A consice introduction to data compression.* Springer London.

Sayood, K. (2006). *Introduction to data compression* (3rd ed.). Elsevier Inc.

Shannon, C. E. (1948). A mathematical theory of communication. *m The Bell System Technical Journal*, *27*, 379-423,623-656. Descargado de `http://cm.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf`

Shkarin, D. A. (2001, March). Improving the efficiency of the ppm algorithm. *Problems of Information Transmission*, *37*(3), 44-54.

Smith, S. W. (1999). *The scientist and engineer's guide to digital signal processing* (2nd ed.). California Technical Publishing.

Storer, J. A., y Szymanski, T. G. (1982, October). Data compression via textual substitution. *Journal of the Association for Computing Machinery*, *19*(4), 928-951.

Tanenbaum, A. S. (2009). *Sistemas operativos modernos* (3ra ed.; P. E. de México, Ed.). Pearson Education, Inc.

Wallace, G. K. (1992, February). The jpeg still picture compression standard. *IEEE Transactions on Consumer Electronics*, *38*(1), xviii - xxxiv.

Witten, I., Neal, R., y Cleary, J. (1987). Arithmetic coding for data compression. *Communications of the Association for Computing Machinery*, *30*(6), 520-540.

Zins, C. (2007, February). Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, *58*(4), 479-493. Descargado de `http://www.success.co.il/is/zins_definitions_dik.pdf`

Ziv, J., y Lempel, A. (1977, May). A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, *23*(3), 337-343.

Ziv, J., y Lempel, A. (1978, September). Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, *24*(5), 530-536.

ANEXOS

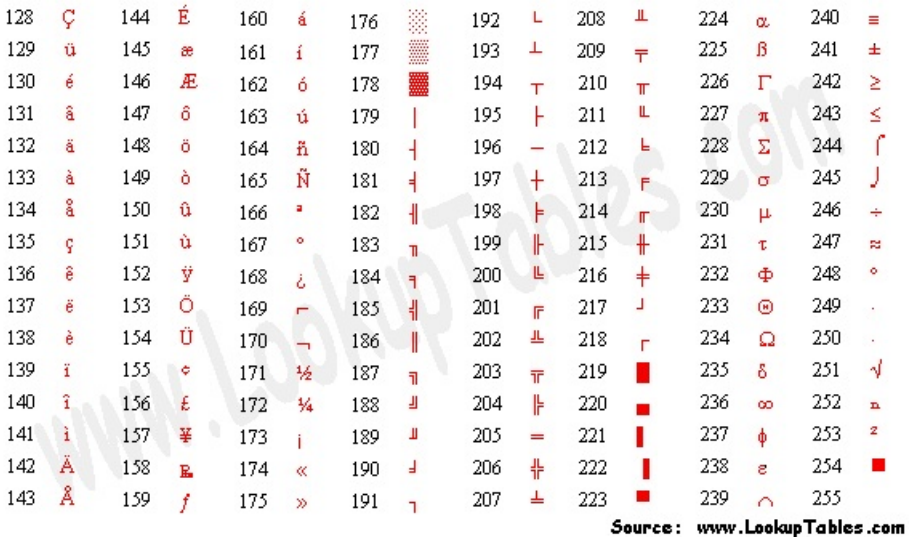
Anexo A: Tabla ASCII

1.1 Tabla ASCII original

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

1.2 Tabla ASCII extendido



Anexo B: CP-1252

2.1 Windows 1252

Windows-1252 (CP1252)																
	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1x	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2x	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3x	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4x	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5x	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6x	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7x	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL
8x	€		‚	ƒ	„	…	†	‡	ˆ	‰	Š	‹	Œ		Ž	
9x		‘	’	“	”	•	–	—	˜	™	š	›	œ		ž	Ÿ
Ax	NBSP	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬		®	¯
Bx	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
Cx	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Dx	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
Ex	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
Fx	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Fuente: *http* : *//www.cp*1252*.com/*

Anexo C: Código fuente de los codificadores aritméticos en lenguaje C++

3.1 Basado en Witten y cols. (1987)

```
/*Archivo: codif.h*/

#ifndef CODIF_H
#define CODIF_H

#include "bitio.h"
#include "modelo.h"

#define Un_cuarto (unsigned long)((Max_val+1)/4)
#define Mitad (unsigned long)(2*Un_cuarto)
#define Tres_cuarto (unsigned long)(3*Un_cuarto)

typedef unsigned int LIM, RAN;

void iniciar_codificador();
void codificar_simbolo(SMB);
void finalizar_codificador();
void bit_siguiente(BIT);

#endif
```

```
/* Archivo: codif.cpp */

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <assert.h>

#include "codif.h"

static CONTADOR bits_para_seguir;
static LIM inf, sup;

void iniciar_codificador(){
```

```
14      inf = 0;
15      sup = Max_val;
16      bits_para_seguir = 0;
17  }
18
19  void codificar_simbolo(SMB simbolo){
20      RAN r;
21
22      r = (RAN)(sup - inf + 1);
23
24      sup = (LIM)(inf + r * ((FLO)frec_acum[simbolo+1]/
            frec_acum[N]) - 1);
25      inf = (LIM)(inf + r * ((FLO)frec_acum[simbolo]/
            frec_acum[N]));
26
27      if(sup < inf){
28          printf("Error de compresion!\n");
29          exit(EXIT_FAILURE);
30      }
31
32      while (1){
33          if (sup < Mitad){
34              bit_siguiente(0);
35          }else if (inf >= Mitad){
36              bit_siguiente(1);
37              inf -= Mitad;
38              sup -= Mitad;
39          }else if ((inf>=Un_cuarto)&&(sup<Tres_cuarto)){
40              bits_para_seguir +=1;
41              inf -= Un_cuarto;
42              sup -= Un_cuarto;
43          }else{
44              break;
45          }
46          inf = 2*inf;
47          sup = 2*sup + 1;
48      }
49  }
50
51  void finalizar_codificador(){
52      bits_para_seguir += (Num_bits-1); //
53      if (inf<Un_cuarto)
54          bit_siguiente(0);
```

```
55      else
56          bit_siguiente(1);
57  }
58
59  void bit_siguiente(BIT bit){
60      escribe_bit(bit&1);
61      while (bits_para_seguir>0){
62          escribe_bit(~bit&1);
63          bits_para_seguir -= 1;
64      }
65  }
```

```
1   /*Archivo: decodif.h */
2
3   #ifndef DECODIF_H
4   #define DECODIF_H
5
6   #include "bitio.h"
7   #include "modelo.h"
8
9   #define Un_cuarto (unsigned long)((Max_val+1)/4)
10  #define Mitad (unsigned long)(2*Un_cuarto)
11  #define Tres_cuarto (unsigned long)(3*Un_cuarto)
12
13  typedef unsigned int LIM, RAN, VAL;
14
15  void iniciar_decodificador();
16  int decodificar_simbolo();
17  void finalizar_decodificador();
18
19  #endif
```

```
1   /* Archivo: decodif.cpp */
2
3   #include <stdio.h>
4   #include <stdlib.h>
5   #include <conio.h>
6   #include <assert.h>
7
8   #include "decodif.h"
9
```

```
10  static LIM inf, sup;
11  static VAL val;
12
13  void iniciar_decodificador(){
14      IND i;
15
16      val = 0;
17      for (i = 1; i<= Num_bits; i++){
18          val = 2*val + lee_bit();
19      }
20      inf = 0;
21      sup = Max_val;
22  }
23
24  SMB decodificar_simbolo(){
25      RAN r;
26      SMB simbolo;
27      FRA acum;
28
29      r = sup - inf + 1;
30      acum = (FRA)((FLO)(val-inf+1)/r*frec_acum[N]-(FLO)1/r
            );
31      simbolo = obtener_simbolo(acum);
32
33      sup = (LIM)(inf + r*((FLO)frec_acum[simbolo+1]/
            frec_acum[N])-1);
34      inf = (LIM)(inf + r*((FLO)frec_acum[simbolo]/
            frec_acum[N]));
35
36      if(val > sup){
37          printf("Error de descompresion ... \n");
38          exit(EXIT_FAILURE);
39      }
40
41      while(1){
42          if (sup<Mitad){
43              /* nada de codigo */
44          }else if(inf >= Mitad){
45              val -= Mitad;
46              inf -= Mitad;
47              sup -= Mitad;
48          }else if (inf>=Un_cuarto && sup<Tres_cuarto){
49              val -= Un_cuarto;
```

```
50              inf -= Un_cuarto;
51              sup -= Un_cuarto;
52          }else{
53              break;
54          }
55          inf = 2*inf;
56          sup = 2*sup + 1;
57          val = 2*val + lee_bit();
58      }
59      return simbolo;
60  }
61
62  void finalizar_decodificador(){
63      /* nada de codigo */
64  }
```

3.2 Basado en Moffat y cols. (1998)

```
1   /*Archivo: codif.h */
2
3   #ifndef CODIF_H
4   #define CODIF_H
5
6   #include "bitio.h"
7   #include "modelo.h"
8
9   #define Val_2b_1 ((unsigned long)1<<(Num_bits-1)) /* la
        mitad */
10  #define Val_2b_2 ((unsigned long)1<<(Num_bits-2)) /*
        cuarta parte */
11
12  typedef unsigned int LR;
13  typedef int CONTADOR, MASCARA;
14
15  void iniciar_codificador();
16  void codificar_simbolo(SMB);
17  void finalizar_codificador();
18  void bit_siguiente(BIT);
19
20  #endif // CODIF_H
```

```
/*Archivo: codif.cpp */

#include "codif.h"

static LR L, R;
static CONTADOR bits_en_espera;

void iniciar_codificador(){
    L = 0;
    R = Val_2b_1;
    bits_en_espera = 0;
}

void codificar_simbolo(SMB simbolo){
    FLO r;
    r = (FLO)R/frec_acum[257];

    L = L + (LR)(r*frec_acum[simbolo]);

    if (frec_acum[simbolo+1]<frec_acum[257])
        R = (LR)(r*(frec_acum[simbolo+1]-frec_acum[
           simbolo]));
    else
        R = R-(LR)(r*frec_acum[simbolo]);
    while (R<=Val_2b_2){
        if (L+R<=Val_2b_1){
            bit_siguiente(0);
        }else if(L>=Val_2b_1){
            bit_siguiente(1);
            L -= Val_2b_1;
        }else{
            bits_en_espera +=1;
            L -= Val_2b_2;
        }
        L = 2*L;
        R = 2*R;
    }
}

void finalizar_codificador(){
    MASCARA mask;
    mask = 1<<(Num_bits-1);
```

```
42      while (mask){
43          if ((mask&L) > 0)
44              bit_siguiente(1);
45          else
46              bit_siguiente(0);
47          mask >>= 1;
48      }
49  }
50
51  void bit_siguiente(BIT bit){
52      escribe_bit(bit&1);
53      while (bits_en_espera>0){
54          escribe_bit(~bit&1);
55          bits_en_espera -= 1;
56      }
57  }
```

```
1   /* Archivo: decodif.h
2
3   #ifndef DECODIF_H
4   #define DECODIF_H
5
6   #include "bitio.h"
7   #include "modelo.h"
8
9   #define minimo(a,b) a<b?a:b
10
11  #define Val_2b_1 ((unsigned long)1<<(Num_bits-1)) /* la
        mitad */
12  #define Val_2b_2 ((unsigned long)1<<(Num_bits-2)) /*
        cuarta parte */
13
14  typedef unsigned int RD;
15
16  void iniciar_decodificador();
17  SMB decodificar_simbolo();
18  void finalizar_decodificador();
19  FRA decodificar_objetivo();
20
21  #endif
```

```
/*Archivo: decodif.cpp*/

#include "decodif.h"

static RD R, D;
static FLO r;

void iniciar_decodificador(){
    IND i;
    R = Val_2b_1;
    D = 0;
    for (i = 1; i<= Num_bits; i++){
        D = 2*D + lee_bit();
    }
}

SMB decodificar_simbolo(){
    SMB simbolo;
    simbolo = obtener_simbolo(decodificar_objetivo());
    D = D - (RD)(r * frec_acum[simbolo]);

    if (frec_acum[simbolo+1]<frec_acum[257])
        R = (RD)(r * (frec_acum[simbolo+1]-frec_acum[
           simbolo]));
    else
        R = R - (RD)(r * frec_acum[simbolo]);

    while (R<=Val_2b_2){
        R = 2 * R;
        D = 2 * D + lee_bit();
    }
    return simbolo;
}

void finalizar_decodificador(){
    /* nada de codigo */
}

FRA decodificar_objetivo(){
    r = (FLO)R/frec_acum[257];
    return (minimo(frec_acum[257]-1,D/r));
}
```

Anexo D: Código fuente de los modelos estadísticos en lenguaje C++

4.1 Modelo estático

```
/*Archivo: modelo.h */

#ifndef MODELO_H
#define MODELO_H

#define N 257 /* Numero total de simbolos */
#define EOF_simbolo 256

#define Num_bits 28
#define Max_val (((unsigned long)1<<Num_bits)-1)

typedef unsigned int FRE, FRA;
typedef int IND, SMB;
typedef float FLO;

extern FRA frec_acum[N+1];

void iniciar_modelo();
SMB obtener_simbolo(FRA );

#endif
```

```
/*Archivo: modelo.cpp */

#include "modelo.h"

FRA frec_acum[N+1];
FRE frec[N] = {
        65535, 1, 1, 1, 1, 1, 1, 1, 1, 1, 749, 1, 1, 749,
            1, 1,
        1, 1, 1, 1, 6, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        13616, 64, 13, 1, 1, 1, 1, 8, 19, 18, 2, 1, 1077,
            90, 627, 2,
```

```
10          11, 49, 19, 10, 9, 13, 14, 13, 17, 9, 57, 92, 1,
               1, 1, 23,
11          1, 147, 35, 138, 85, 170, 29, 34, 35, 74, 29, 1,
               134, 82, 50, 33,
12          136, 20, 59, 113, 54, 24, 53, 1, 5, 46, 4, 15, 1,
                15, 8, 4,
13          1, 7849, 927, 2592, 3174, 7941, 390, 700, 576,
               3635, 296, 7, 3748, 1634, 4248, 5542,
14          1499, 659, 4170, 4501, 2501, 2626, 615, 1, 67,
               673, 299, 1, 1, 1, 1, 1,
15          1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
16          1, 3, 1, 1, 1, 1, 1, 10, 1, 1, 1, 1, 1, 1, 1, 1,
17          1, 56, 1, 1, 1, 1, 1, 1, 1, 1, 1, 23, 1, 1, 1, 1,
18          1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 24, 1, 1, 1, 23,
19          1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 2, 1, 1,
20          1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
21          1, 268, 1, 1, 1, 1, 1, 1, 1, 202, 1, 1, 1, 422,
               1, 1,
22          1, 172, 1, 438, 1, 1, 1, 1, 1, 1, 76, 1, 4, 1, 1,
                1,
23          1 // <- Para el simbolo EOF
24  };
25
26  void iniciar_modelo(){
27          IND i;
28          frec_acum[0] = 0;
29          for(i=0;i<N;i++){
30                  frec_acum[i+1] = frec_acum[i] + frec[i];
                       //Generamos la frecuencia acumulada
31          }
32  }
33
34  SMB obtener_simbolo(FRA acum){
35          IND i=0;
36          while(acum>=frec_acum[i+1])     i++;
37          return i;
38  }
```

4.2 Modelo semi-estático

```
1   /*Archivo: modelo.h */
```

```
2
3  #ifndef MODELO_H
4  #define MODELO_H
5
6  #define N 257 /* Numero total de simbolos */
7  #define EOF_simbolo 256
8
9  #define Num_bits 28
10 #define Max_val (((unsigned long)1<<Num_bits)-1)
11
12 typedef unsigned int FRA, FRE;
13 typedef int IND, SMB;
14 typedef char NOM;
15 typedef float FLO;
16
17 extern FRA frec_acum[N+1];
18
19 void iniciar_modelo(NOM *);
20 void iniciar_modelo();
21 void agregar_simbolo(SMB );
22 SMB obtener_simbolo(FRA );
23 FRA limite_sup(SMB );
24 FRA limite_inf(SMB );
25
26 #endif
```

```
1  /*Archivo: modelo.cpp */
2
3  #include "modelo.h"
4  #include "bitio.h"
5
6  FRA frec_acum[N+1];
7  FRE frec[N];
8
9  void iniciar_modelo(NOM *nArchEnt){
10     FILE *f;
11         IND i;
12         BYTE c;
13         FRE mayor, menor;
14
15         for (i=0; i<N; i++)     frec[i] = 0;
16
```

```
17      f = fopen(nArchEnt,"rb");
18      c = fgetc(f);
19      while(!feof(f)){
20          agregar_simbolo(c);
21          c = fgetc(f);
22      }
23      fclose(f);
24
25      mayor = frec[0];
26      for(i=1; i<N-1; i++)
27          if(mayor<frec[i])
28              mayor = frec[i];
29
30      if(mayor>255)
31          for(i=0;i<N-1;i++)  frec[i]=(FRE)(((FLO)frec[i]/
               mayor)*255);
32
33          for(i=0;i<N-1;i++)  // escribiendo las frec al
               archivo comprimidp
34          escribe_byte(frec[i]);
35
36      menor = frec[0];
37      for(i=1; i<N-1; i++)
38          if(menor>frec[i])
39              menor = frec[i];
40
41      if(menor==0)
42          for(i=0;i<N-1;i++)  frec[i]++;
43
44      frec[N-1]=1;
45      frec_acum[0]=0;
46      for(i=0; i<N; i++)
47          frec_acum[i+1]=frec_acum[i] + frec[i];
48  }
49
50  void iniciar_modelo(){
51      IND i;
52      FRE menor;
53
54          for(i=0;i<N-1;i++) frec[i]=(FRE)lee_byte();
55
56      menor = frec[0];
57      for(i=1; i<N-1; i++)
```

```
58          if(menor>frec[i])
59              menor = frec[i];
60      if(menor==0)
61          for(i=0;i<N-1;i++)  frec[i]++;
62
63      frec[N-1]=1; //Frecuencia del Simbolo artificial
            EOF_simbolo
64      frec_acum[0]=0;
65      for(i=0; i<N; i++)
66          frec_acum[i+1]=frec_acum[i] + frec[i];
67  }
68
69  void agregar_simbolo(SMB simbolo){
70      frec[simbolo]++;
71  }
72
73  SMB obtener_simbolo(FRA acum){
74      int i;
75      for(i=0; frec_acum[i+1]<=acum; i++);
76      return i;
77  }
78
79  FRA limite_sup(SMB simbolo){
80      return frec_acum[simbolo+1];
81  }
82
83  FRA limite_inf(SMB simbolo){
84      return frec_acum[simbolo];
85  }
```

4.3 Modelo adaptativo

```
1  /*Archivo: modelo.h*/
2
3  #ifndef MODELO_H
4  #define MODELO_H
5
6  #define N 257
7  #define EOF_simbolo 256
8
9  #define Num_bits 28
```

```
10 #define Max_val (((unsigned long)1<<Num_bits)-1)
11
12 typedef unsigned int FRA, FRE;
13 typedef int IND, SMB;
14 typedef float FLO;
15
16 extern FRA frec_acum[N+1];
17
18 void iniciar_modelo();
19 void agregar_simbolo(SMB );
20 SMB obtener_simbolo(FRA );
21 FRA limite_sup(SMB );
22 FRA limite_inf(SMB );
23 void mostrar_modelo();
24
25 #endif
```

```
1  /*Archivo: modelo.cpp*/
2
3  #include <stdio.h>
4  #include <string.h>
5
6  #include "modelo.h"
7
8  FRA frec_acum[N+1];
9  static FRE frec[N];
10
11 void iniciar_modelo(){
12     IND i;
13     for (i=0; i<N; i++) frec[i]=0;
14     for(i=0; i<N; i++)  frec_acum[i+1]=frec_acum[i]+frec[
          i]+1;
15 }
16
17 void agregar_simbolo(SMB simbolo){
18     IND i;
19     frec[simbolo]++;
20     for(i=0; i<N; i++)  frec_acum[i+1]=frec_acum[i]+frec[
          i]+1;
21 }
22
23 SMB obtener_simbolo(FRA acum){
```

```
24      IND i;
25      for(i=0; frec_acum[i+1]<=acum; i++);
26      return i;
27  }
28
29  FRA limite_sup(SMB simbolo){
30      return frec_acum[simbolo+1];
31  }
32
33  FRA limite_inf(SMB simbolo){
34      return frec_acum[simbolo];
35  }
36
37  void mostrar_modelo(){
38      for(IND i=0; i<N; i++){
39          printf("%d \t%d \t[%d %d>\n",i,frec[i],frec_acum[
               i],frec_acum[i+1]);
40      }
41  }
```

Anexo E: Programas para obtener el desempeño de los compresores

5.1 rcrono

```
1  /*rcrono*/
2
3  #include <iostream>
4  #include <time.h>
5  #include <stdlib.h>
6  #include <string.h>
7
8  #define N 100
9
10 typedef double SUMADOR, TIEMPO;
11 typedef char COMANDO;
12 typedef int IND;
13
14 using namespace std;
15
16 int main(int argc,char **argv)
17 {
18     SUMADOR s;
19     TIEMPO t;
20     IND i;
21     clock_t ini, fin;
22     COMANDO cmdComp[234];
23
24     if (argc != 3) {
25           cout << "Uso : rcrono.exe <nombre_programa> <
                parametros_programa>" << endl;
26           return 1;
27     }
28
29     strcpy(cmdComp,argv[1]);
30     strcat(cmdComp," ");
31     strcat(cmdComp,argv[2]);
32
33     s = 0;
34     i = 0;
```

```
35
36      while(i<N){
37           ini = clock(); // Empieza correr el reloj
38           system(cmdComp); // Ejecutamos comando
39           fin = clock();  // Se detiene reloj
40           t = (TIEMPO) (fin-ini) / CLK_TCK ;
41
42           cout << " t= " << t << endl;
43
44           s = s + t;
45           i++;
46       }
47
48       cout<<"Tiempo de ejecucion : "<<((TIEMPO)s/N)<<" s"
            <<endl;
49       cout<<"Tiempo de ejecucion : "<<((TIEMPO) s)/N *
            1000<<" ms"<< endl;
50       return 0;
51  }
```

5.2 rtam

```
1   /*rtam*/
2
3   #include <iostream>
4   #include <stdio.h>
5
6   typedef unsigned long TAM;
7
8   using namespace std;
9
10  int main (int argc, char **argv)
11  {
12      FILE * pArch;
13      TAM t;
14
15      if(argc!=2){
16          printf("Uso: tam.exe nombre_archivo \n");
17          return 1;
18      }
19      pArch = fopen (argv[1],"rb");
```

```
20      if (pArch==NULL){
21          printf("No se encuentra el archivo. \n");
22          return 1;
23      }else{
24          fseek (pArch, 0, SEEK_END);
25          t=ftell (pArch);
26          fclose (pArch);
27          printf ("TamaÃśo del %s: %ld bytes.\n",argv[1],t)
                ;
28      }
29      return 0;
30  }
```

Anexo F: Conteo de los símbolos de la colección "Literatura peruana"

Tabla 32. Frecuencia de los símbolos en los archivos de la colección “Literatura peruana”.

CP-1252	**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	**a10**	**a11**	**a12**	**TOTAL**
0	641664	0	0	0	0	0	0	0	0	0	2956626	0	3598290
1	0	0	0	0	0	0	0	0	0	0	1	0	1
2	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	83	0	0	0	0	0	0	0	0	7	0	91
10	1641	748	11095	864	338	709	1798	98	2824	307	18622	2111	41155
11	0	0	0	0	0	0	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1641	748	11095	864	338	709	1798	98	2824	307	18621	2111	41154
14	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0
19	1	0	0	0	0	0	0	0	0	0	0	0	1
20	332	0	0	0	0	0	0	0	0	0	2	0	334
21	0	0	0	0	0	0	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	0	0	0	0	0	0

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
23	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0
25	0	0	0	0	0	0	0	0	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0
28	0	0	0	0	0	0	0	0	0	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0
32	101650	7406	48909	3958	3731	44651	7471	5131	12343	5464	495374	11518	747606
33	210	110	387	162	18	1	156	31	297	46	2076	66	3560
34	677	1	13	1	16	25	0	4	0	0	9	0	746
35	0	0	10	0	0	0	0	0	0	0	0	0	10
36	0	0	5	0	0	0	0	0	0	0	0	0	5
37	10	0	6	0	0	0	0	0	0	0	0	0	16
38	0	0	23	0	0	0	0	0	0	0	0	0	23
39	102	0	286	0	0	0	1	0	7	0	44	0	440
40	91	15	323	10	0	73	4	0	23	4	504	7	1054
41	94	15	286	8	0	72	4	0	23	4	501	6	1013
42	62	0	46	0	0	0	0	0	0	0	36	0	144
43	0	0	0	0	0	0	0	0	0	0	4	0	4
44	7342	636	3696	713	424	3851	687	433	2302	474	37666	934	59158
45	347	85	152	2	34	4	10	13	4	2	4308	4	4965
46	5193	504	3960	130	199	1022	719	370	672	311	20648	725	34453

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
47	6	1	71	4	0	4	4	4	4	4	6	4	112
48	105	2	61	2	0	3	0	0	0	0	439	6	618
49	333	10	458	11	0	6	6	0	8	5	1889	15	2741
50	205	4	322	2	3	0	1	0	2	1	500	9	1049
51	115	2	62	8	1	0	1	0	5	4	368	2	568
52	93	2	46	0	1	0	0	0	1	1	351	1	496
53	94	2	44	0	1	4	0	0	1	0	589	0	735
54	72	2	29	0	1	5	0	0	1	1	683	2	796
55	83	2	38	5	0	0	0	0	0	0	624	4	756
56	92	2	49	2	2	0	3	0	4	3	780	2	939
57	199	2	39	7	0	2	3	0	6	4	230	11	503
58	295	40	412	27	21	74	34	27	31	25	2150	25	3161
59	342	165	872	47	28	586	135	24	189	9	2627	32	5056
60	2	0	49	0	0	0	0	0	0	0	2	0	53
61	2	0	1	0	0	0	0	0	0	0	0	0	3
62	2	0	58	1	0	0	0	0	0	0	2	0	63
63	65	20	157	4	5	19	17	15	102	27	851	26	1308
64	0	0	8	0	0	0	0	0	0	0	0	0	8
65	682	70	1165	51	39	452	182	137	177	45	5025	96	8121
66	150	90	130	15	12	57	29	1	21	4	1399	18	1926
67	673	52	561	43	45	410	60	70	113	27	5464	68	7586
68	337	75	380	32	15	267	82	12	60	25	3360	39	4684
69	1889	76	893	74	34	545	157	41	147	71	5317	98	9342
70	141	10	101	2	4	73	10	2	14	8	1263	10	1638

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
71	285	3	126	19	1	90	24	17	11	5	1289	6	1876
72	200	14	139	12	7	125	54	20	52	26	1252	28	1929
73	463	32	598	33	3	302	66	24	51	15	2398	112	4097
74	104	39	138	2	8	36	17	3	6	3	1255	6	1617
75	5	22	45	0	0	0	0	0	0	0	9	0	81
76	1399	78	757	43	24	353	115	31	84	46	4343	85	7358
77	417	29	474	37	12	169	70	66	59	47	3118	58	4556
78	477	23	321	22	10	211	59	17	58	44	1488	46	2776
79	203	5	381	26	5	224	82	8	66	24	790	42	1856
80	1084	153	432	49	20	348	71	31	76	28	5153	50	7495
81	55	7	104	7	5	52	12	15	58	18	780	33	1146
82	434	20	345	35	12	207	59	10	64	14	2065	19	3284
83	590	84	571	30	17	406	110	72	88	43	4157	56	6224
84	289	42	334	25	6	195	61	26	72	22	1879	35	2986
85	300	19	188	11	9	93	35	28	43	26	596	7	1355
86	344	77	276	37	7	86	34	11	42	12	1945	56	2927
87	12	27	24	0	0	0	0	0	2	0	24	0	89
88	46	0	12	7	0	0	0	0	0	0	147	103	315
89	284	15	257	10	4	112	122	21	75	41	1527	92	2560
90	16	0	27	2	0	39	5	0	3	1	160	2	255
91	176	0	60	0	0	0	0	0	0	0	641	0	877
92	0	0	31	0	0	0	0	0	0	0	1	0	32
93	176	1	9	0	0	0	0	0	0	0	641	0	827
94	0	0	441	0	0	0	0	0	0	0	0	0	441

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
95	219	0	0	0	0	1	0	2	0	0	1	10	233
96	0	0	0	0	0	0	0	0	0	0	3	0	3
97	59772	4902	23660	2539	2300	24716	4561	3070	7452	2435	290060	5500	430967
98	5688	570	2849	350	290	2758	510	451	932	268	35647	624	50937
99	23697	1351	7028	658	682	7749	1161	851	2276	635	94878	1374	142340
100	25333	1408	8741	1079	795	9806	1719	1081	3026	1074	118032	2196	174290
101	64155	3533	24784	2463	1906	24351	4227	2750	7857	2743	292263	4985	436017
102	3584	91	1597	116	100	958	244	128	424	122	13823	247	21434
103	5494	310	2289	221	211	1842	460	232	605	187	26199	386	38436
104	3018	425	1713	215	169	2422	433	293	704	271	21515	485	31663
105	37679	1480	9878	1026	725	9545	1693	1092	3176	1151	130192	1967	199604
106	1508	236	1229	99	121	912	220	163	353	140	11109	203	16293
107	144	217	25	1	0	2	1	2	3	1	38	0	434
108	28943	2144	12873	1199	1187	11934	1946	1457	3768	1161	136921	2291	205824
109	13842	860	4685	625	486	5019	986	609	2190	739	58520	1196	89757
110	37389	2401	13128	1252	1037	12971	2382	1512	4235	1373	152864	2701	233245
111	43090	3428	16507	1836	1506	17361	3104	2045	5917	1838	203909	3751	304292
112	12853	785	3959	450	368	4417	627	525	1416	437	55615	876	82328
113	3218	157	1901	168	105	3310	345	170	592	176	25619	445	36206
114	32953	2382	12565	1345	1064	12424	2366	1669	3979	1315	154189	2733	228984
115	37963	2664	16188	1524	1273	15466	2642	1985	4457	1446	158204	3375	247187
116	24737	1319	7314	950	564	6877	1399	745	2810	840	88040	1730	137325
117	19523	1170	9242	914	627	8795	1709	958	2694	835	96085	1639	144191
118	4787	267	2293	263	144	1734	428	244	742	266	22175	468	33811

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
119	19	15	23	4	0	2	3	2	5	3	27	4	107
120	929	7	137	24	9	57	18	9	72	37	2387	33	3719
121	3961	304	2030	222	192	2880	419	386	741	229	25163	433	36960
122	1975	171	1090	95	80	870	215	107	291	77	11275	188	16434
123	0	0	19	0	0	0	0	0	0	0	0	0	19
124	0	0	22	0	0	0	0	0	0	0	0	0	22
125	0	0	2	0	0	0	0	0	0	0	0	0	2
126	0	0	2	0	0	0	0	0	0	0	0	0	2
127	0	0	0	0	0	0	0	0	0	0	0	0	0
128	0	0	1	0	0	0	0	0	0	0	0	0	1
129	0	0	0	0	0	0	0	0	0	0	0	0	0
130	0	0	0	0	0	0	0	0	0	0	0	0	0
131	0	0	0	0	0	0	0	0	0	0	0	0	0
132	0	0	0	0	0	0	0	0	0	0	0	0	0
133	0	0	0	2	0	0	0	0	1	0	0	0	3
134	0	0	0	0	0	0	0	0	0	0	0	0	0
135	0	0	0	0	0	0	0	0	0	0	0	0	0
136	0	0	0	0	0	0	0	0	0	0	0	0	0
137	0	0	0	0	0	0	0	0	0	0	0	0	0
138	0	0	0	0	0	0	0	0	0	0	0	0	0
139	0	0	0	0	0	0	0	0	0	0	0	0	0
140	0	0	0	0	0	0	0	0	0	0	0	0	0
141	0	0	0	0	0	0	0	0	0	0	0	0	0
142	0	0	0	0	0	0	0	0	0	0	0	0	0

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
143	0	0	0	0	0	0	0	0	0	0	0	0	0
144	0	0	0	0	0	0	0	0	0	0	0	0	0
145	176	0	0	0	0	0	0	0	0	0	0	0	176
146	0	97	0	0	0	0	0	0	0	0	2	0	99
147	0	3	0	3	0	23	8	0	4	1	0	2	44
148	0	4	0	3	0	23	8	0	4	1	0	2	45
149	0	0	10	0	0	0	0	0	0	0	0	0	10
150	0	0	0	0	0	0	0	42	0	0	0	0	42
151	0	219	321	8	0	0	7	0	8	38	0	2	603
152	0	0	0	0	0	0	0	0	0	0	0	0	0
153	0	0	0	0	0	0	0	0	0	0	0	0	0
154	0	0	0	0	0	0	0	0	0	0	0	0	0
155	0	0	0	0	0	0	0	0	0	0	0	0	0
156	0	0	0	0	0	0	0	0	0	0	0	0	0
157	0	0	0	0	0	0	0	0	0	0	0	0	0
158	0	0	0	0	0	0	0	0	0	0	0	0	0
159	0	0	0	0	0	0	0	0	0	0	0	0	0
160	0	0	0	0	0	0	0	0	0	0	0	0	0
161	16	110	381	130	17	1	27	31	272	36	2070	22	3113
162	0	0	0	0	0	0	0	0	0	0	0	0	0
163	0	0	1	0	0	0	0	0	0	0	0	0	1
164	0	0	0	0	0	0	0	0	0	0	0	0	0
165	0	0	4	0	0	0	0	0	0	0	0	0	4
166	0	0	6	0	0	0	0	0	0	0	0	0	6

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
167	0	0	2	0	0	0	0	0	0	0	0	0	2
168	0	0	0	0	0	0	0	0	0	0	0	0	0
169	0	0	6	0	0	0	0	0	0	0	0	0	6
170	8	0	0	0	0	0	0	0	0	0	10	0	18
171	234	0	31	6	0	0	3	0	1	2	1001	0	1278
172	0	0	0	0	0	0	0	0	0	0	0	0	0
173	0	0	0	0	0	0	0	0	0	9	0	0	9
174	0	0	28	0	0	0	0	0	0	0	0	0	28
175	0	0	0	0	0	0	0	0	0	0	0	0	0
176	2	0	2	0	0	0	0	0	0	0	0	0	4
177	0	0	0	0	0	0	0	0	0	0	0	0	0
178	0	0	0	0	0	0	0	0	0	0	0	0	0
179	0	0	0	0	0	0	0	0	0	0	0	0	0
180	0	0	0	0	0	0	0	0	0	0	5	0	5
181	0	0	0	0	0	0	0	0	0	0	0	0	0
182	0	0	0	0	0	0	0	0	0	0	0	0	0
183	0	0	0	0	0	0	0	0	0	0	0	0	0
184	0	0	0	0	0	0	0	0	0	0	0	0	0
185	0	0	0	0	0	0	0	0	0	0	0	0	0
186	37	0	0	0	0	0	0	0	0	0	50	0	87
187	234	1	47	6	0	0	3	0	1	2	1042	0	1336
188	0	0	0	0	0	0	0	0	0	0	0	0	0
189	0	0	0	0	0	0	0	0	0	0	0	0	0
190	0	0	0	0	0	0	0	0	0	0	0	0	0

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
191	58	20	156	4	6	19	7	15	103	24	846	22	1280
192	0	0	0	0	0	0	0	0	0	0	0	0	0
193	10	0	19	1	0	9	3	3	0	2	60	1	108
194	0	0	0	0	0	0	0	0	0	0	0	0	0
195	0	0	0	0	0	0	0	0	0	0	0	0	0
196	0	0	0	0	0	0	0	0	0	0	0	0	0
197	0	0	0	0	0	0	0	0	0	0	0	0	0
198	0	0	0	0	0	0	0	0	0	0	0	0	0
199	0	0	0	0	0	0	0	0	0	0	0	0	0
200	0	0	0	0	0	0	0	0	0	0	0	0	0
201	13	1	53	1	0	12	1	4	3	0	126	0	214
202	0	0	0	0	0	0	0	0	0	0	0	0	0
203	0	0	0	0	0	0	0	0	0	0	0	0	0
204	0	0	0	0	0	0	0	0	0	0	0	0	0
205	10	2	30	1	0	16	3	0	3	0	53	0	118
206	0	0	0	0	0	0	0	0	0	0	0	0	0
207	0	0	0	0	0	0	0	0	0	0	0	0	0
208	0	0	0	0	0	0	0	0	0	0	0	0	0
209	3	0	20	3	0	4	1	0	1	0	24	0	56
210	0	0	0	0	0	0	0	0	0	0	0	0	0
211	11	0	17	0	0	17	6	0	5	1	20	1	78
212	0	0	0	0	0	0	0	0	0	0	0	0	0
213	0	0	0	0	0	0	0	0	0	0	0	0	0
214	0	0	0	0	0	0	0	0	0	0	0	0	0

Continua en la siguiente página

Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
215	0	0	0	0	0	0	0	0	0	0	0	0	0
216	0	0	0	0	0	0	0	0	0	0	0	0	0
217	0	0	0	0	0	0	0	0	0	0	0	0	0
218	7	0	9	3	0	6	0	0	2	0	7	0	34
219	0	0	0	0	0	0	0	0	0	0	0	0	0
220	1	0	2	0	0	0	0	0	0	0	0	0	3
221	0	0	0	0	0	0	0	0	0	0	0	0	0
222	0	0	0	0	0	0	0	0	0	0	0	0	0
223	0	0	0	0	0	0	0	0	0	0	0	0	0
224	1	0	0	0	0	0	0	0	0	0	3	0	4
225	2042	145	1262	164	73	783	215	122	464	103	9135	254	14762
226	0	0	0	0	0	0	0	0	0	0	1	0	1
227	0	0	0	0	0	0	0	0	0	0	0	0	0
228	0	0	0	0	0	0	0	0	0	0	0	0	0
229	0	0	0	0	0	0	0	0	0	0	0	0	0
230	0	0	0	0	0	0	0	0	0	0	7	0	7
231	2	0	0	0	0	0	0	0	0	0	2	0	4
232	1	0	0	0	0	0	0	0	0	0	6	0	7
233	1314	73	687	85	45	540	144	91	331	69	7564	172	11115
234	0	0	0	0	0	0	0	0	0	0	3	0	3
235	0	0	0	0	0	0	0	0	0	0	0	0	0
236	0	0	0	0	0	0	0	0	0	0	0	0	0
237	3312	203	1207	135	133	1444	166	198	362	125	15615	282	23182
238	0	0	0	0	0	0	0	0	1	0	0	0	1

Continua en la siguiente página

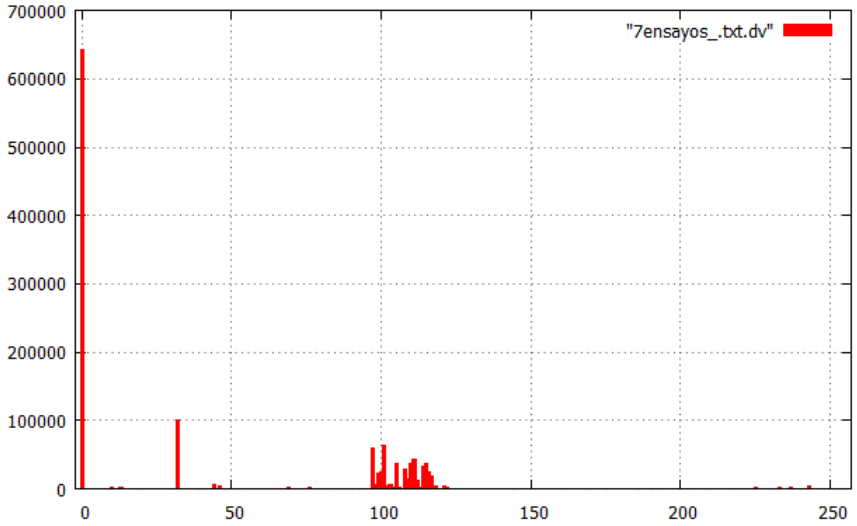
Tabla 32 – Continuación de la anterior página

CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL
239	0	0	0	0	0	0	0	0	0	0	0	0	0
240	0	0	0	0	0	0	0	0	0	0	0	0	0
241	810	58	512	91	42	587	100	58	141	31	6909	114	9453
242	0	0	0	0	0	0	0	0	0	0	0	0	0
243	4303	194	983	122	109	841	144	124	327	95	16668	160	24070
244	0	0	0	0	0	0	0	0	1	0	0	0	1
245	0	0	0	0	0	0	0	0	0	0	0	0	0
246	0	0	0	0	0	0	0	0	0	0	0	0	0
247	0	0	0	0	0	0	0	0	0	0	0	0	0
248	0	0	0	0	0	0	0	0	0	0	0	0	0
249	0	0	0	0	0	0	0	0	0	0	0	0	0
250	824	26	262	21	12	233	74	17	89	24	2522	79	4183
251	0	0	0	0	0	0	0	0	0	0	1	0	1
252	50	0	33	3	0	7	0	0	2	0	141	1	237
253	0	0	0	0	0	0	0	0	0	0	0	0	0
254	1	0	0	0	0	0	0	0	0	0	1	0	2
255	1	0	0	0	0	0	0	0	0	0	1	0	2
													8082356

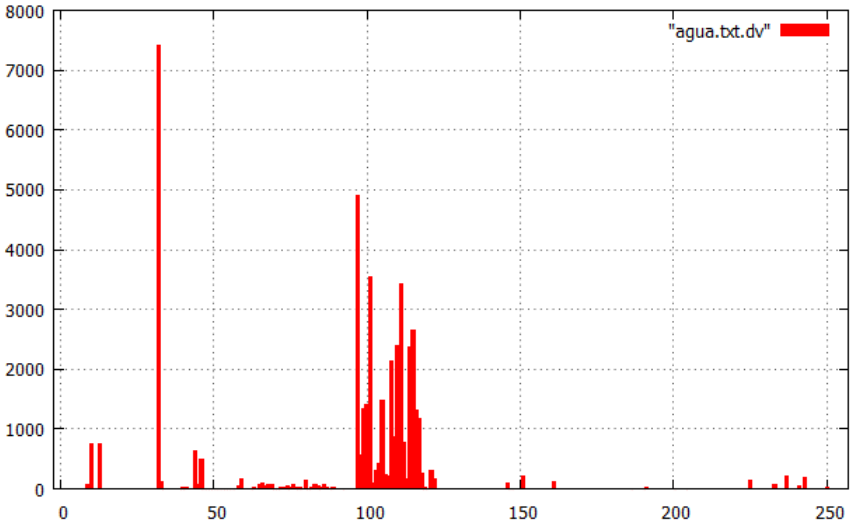
Tabla 33. Lista de indetificadores

Nombre de archivo	Identificador
7ensayos.txt	a1
agua.txt	a2
alma_america.txt	a3
caliz.txt	a4
carmelo.txt	a5
comentarios_reales.txt	a6
heraldos.txt	a7
hermanos_ayar.txt	a8
humanos.txt	a9
prosa.txt	a10
tradiciones.txt	a11
trilce.txt	a12

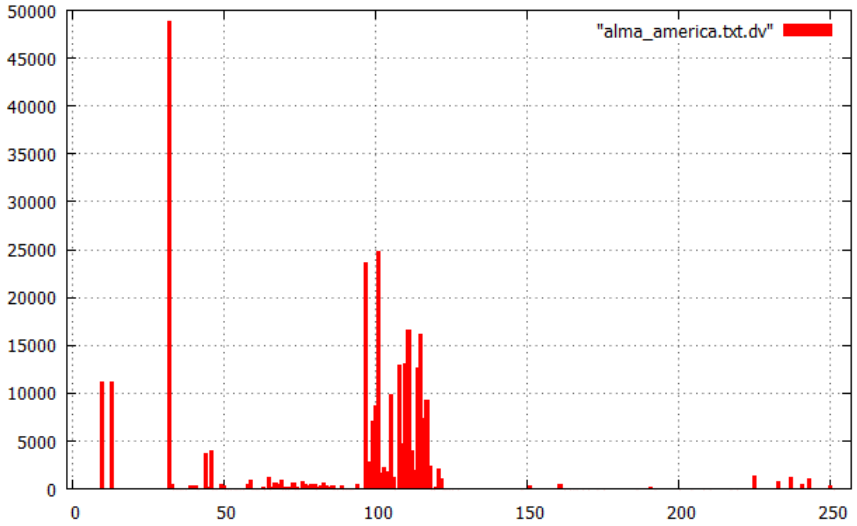
Anexo G: Gráficas de la frecuencia de símbolos de los archivos de la colección "Literatura peruana" antes compresión



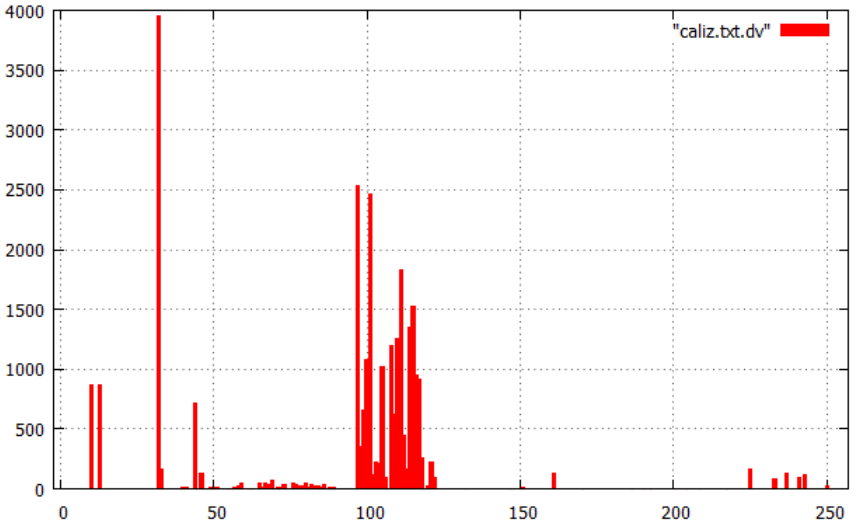
(a) Frecuencias del archivo “7ensayos.txt”



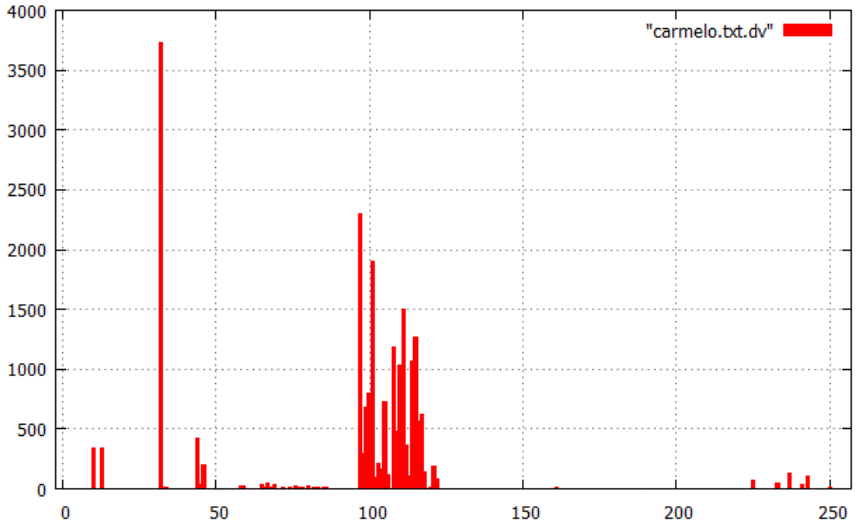
(b) Frecuencias del archivo “agua.txt”



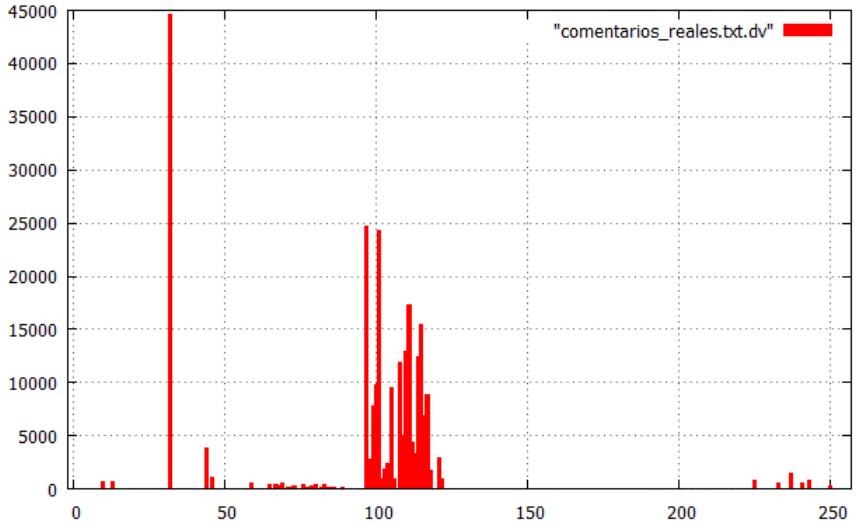
(c) Frecuencias del archivo “alma_america.txt”



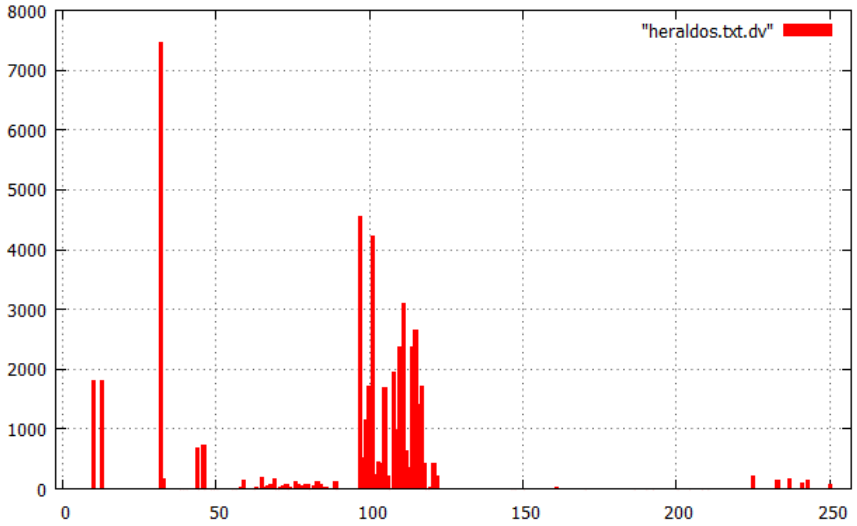
(d) Frecuencias del archivo “caliz.txt”



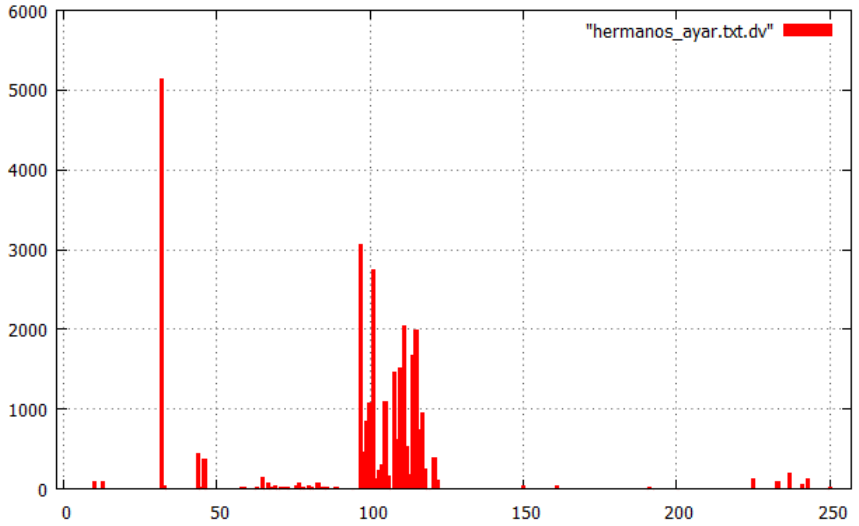
(f) Frecuencias del archivo “carmelo.txt”



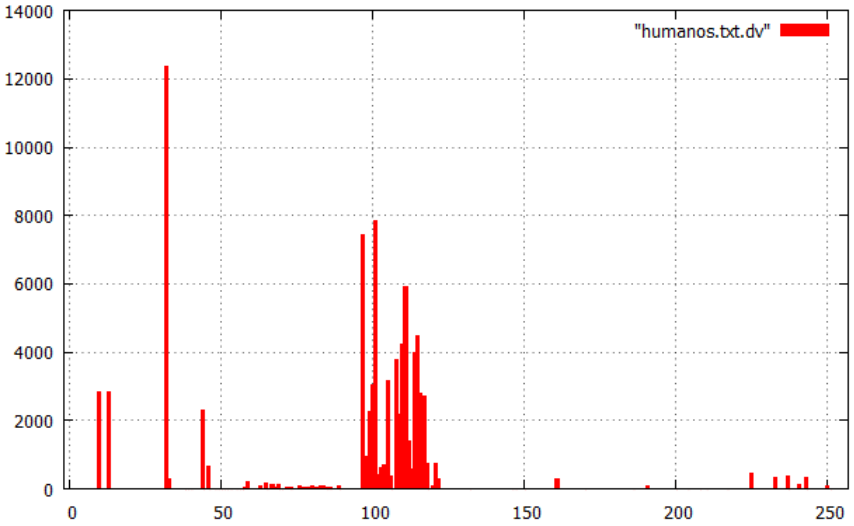
(g) Frecuencias del archivo “comentarios_reales.txt”



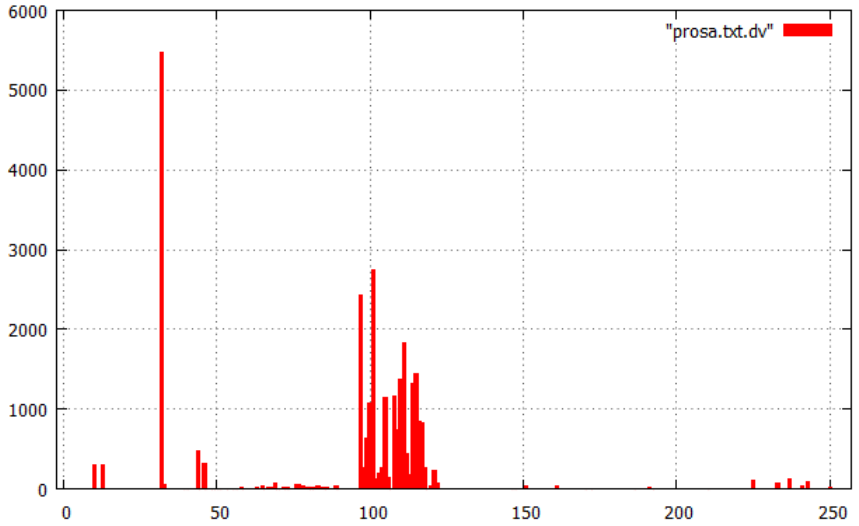
(h) Frecuencias del archivo “heraldos.txt”



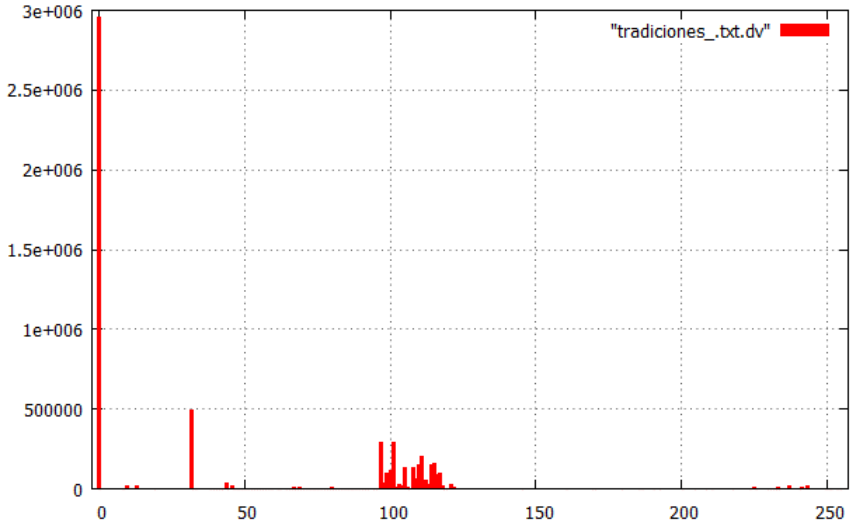
(i) Frecuencias del archivo “hermanos_ayar.txt”



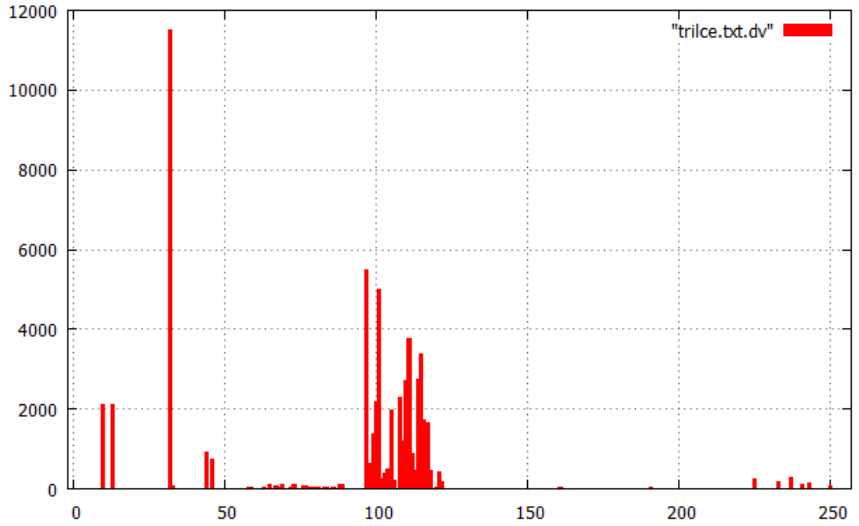
(j) Frecuencias del archivo “humanos.txt”



(k) Frecuencias del archivo “prosa.txt”

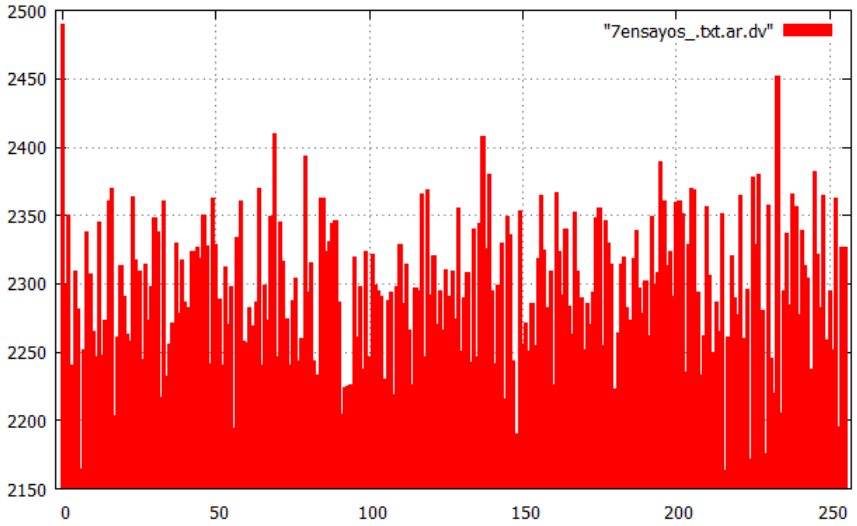


(l) Frecuencias del archivo “tadiciones.txt”

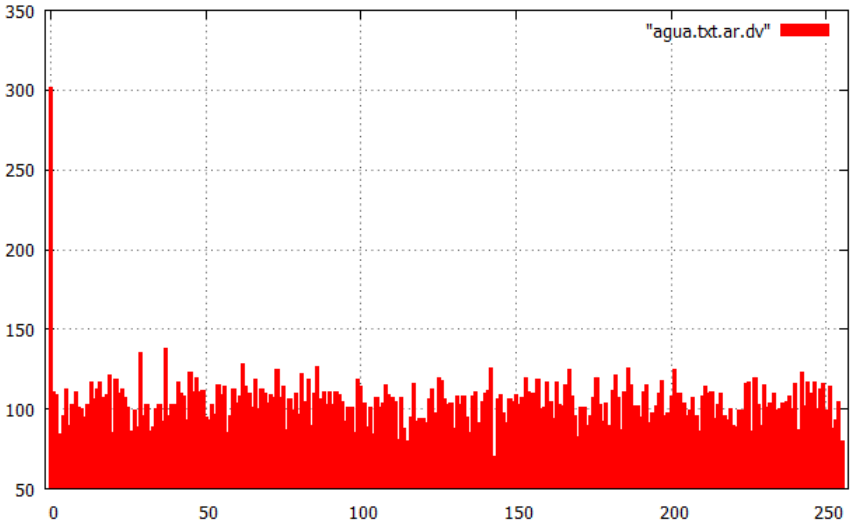


(m) Frecuencias del archivo “trilce.txt”

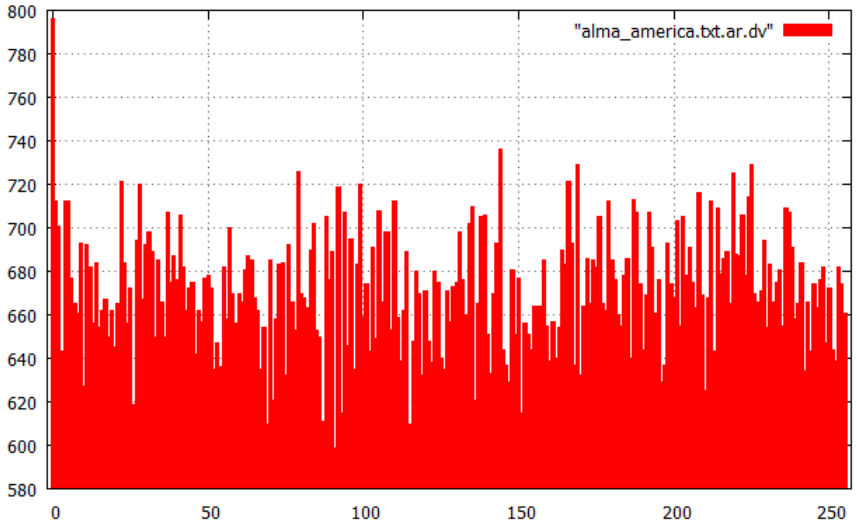
Anexo H: Gráficas de la frecuencia de símbolos de los archivos de la colección "Literatura peruana" después de la compresión



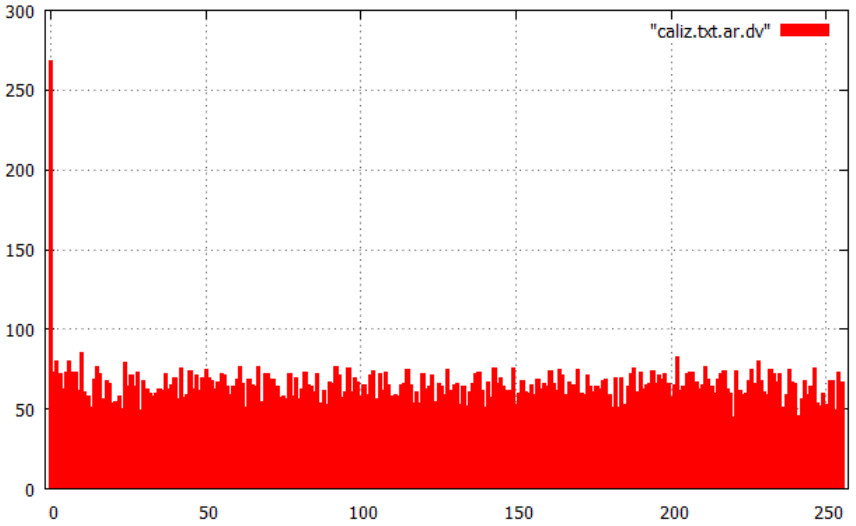
(a) Frecuencias del archivo “7ensayos.txt”



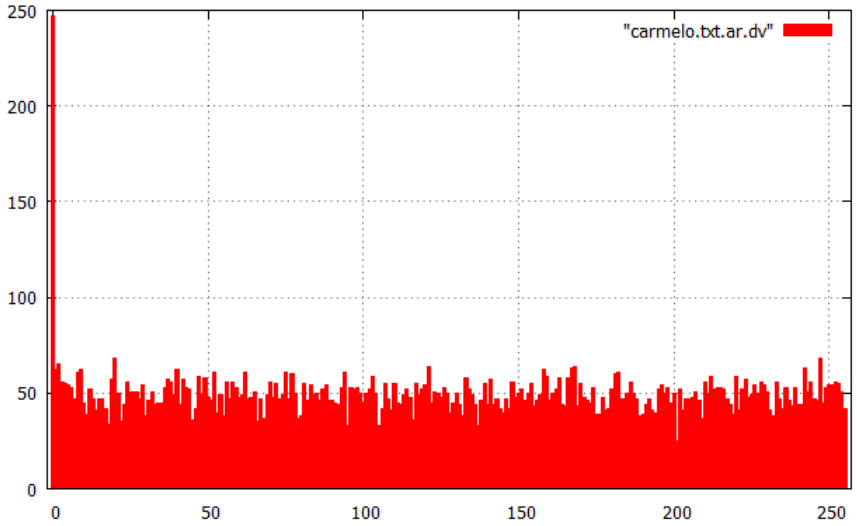
(b) Frecuencias del archivo “agua.txt”



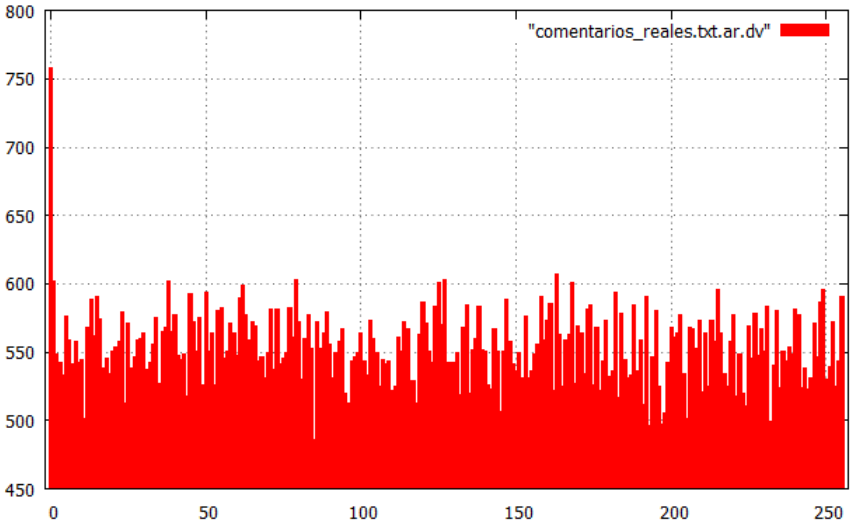
(c) Frecuencias del archivo “alma_america.txt”



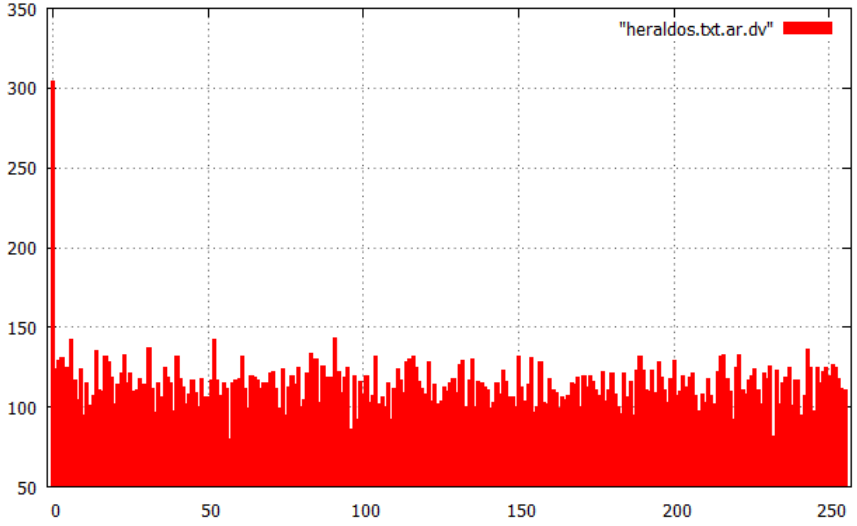
(d) Frecuencias del archivo “caliz.txt”



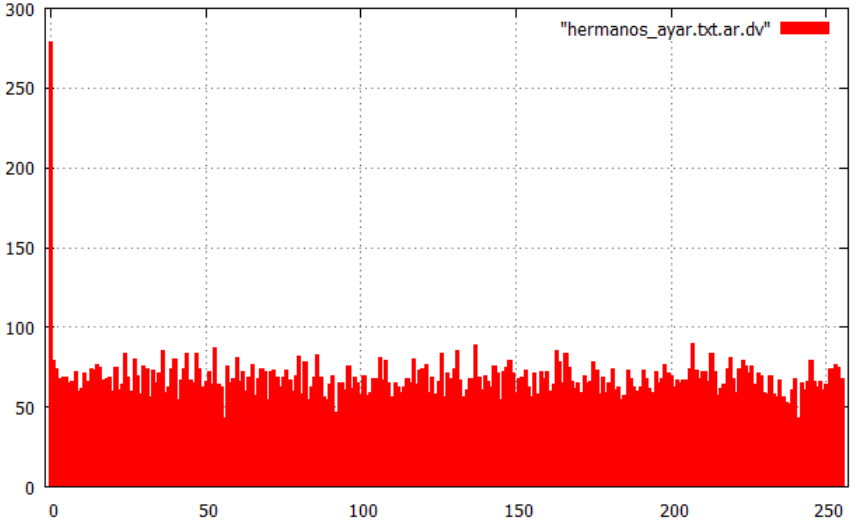
(f) Frecuencias del archivo “carmelo.txt”



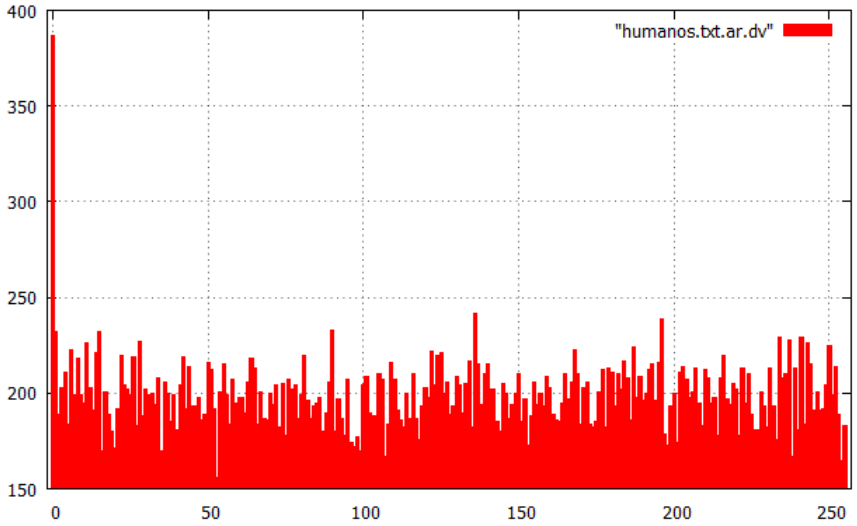
(g) Frecuencias del archivo “comentarios_reales.txt”



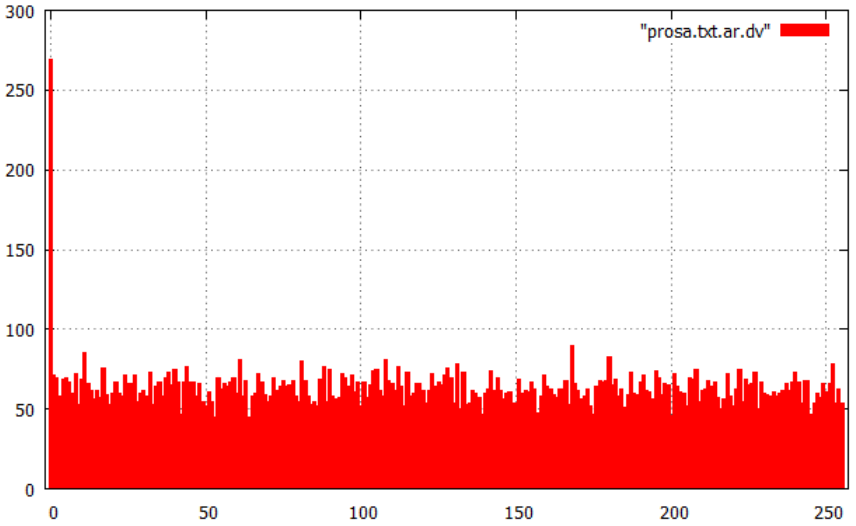
(h) Frecuencias del archivo “heraldos.txt”



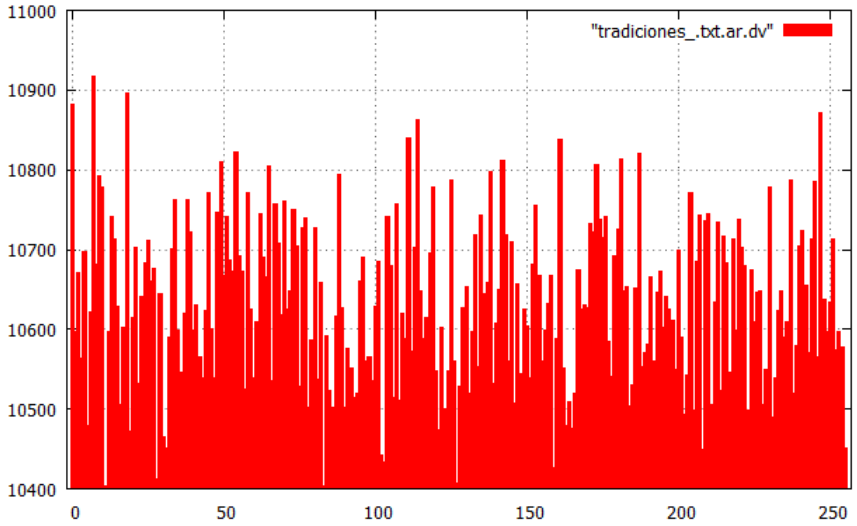
(i) Frecuencias del archivo “hermanos_ayar.txt”



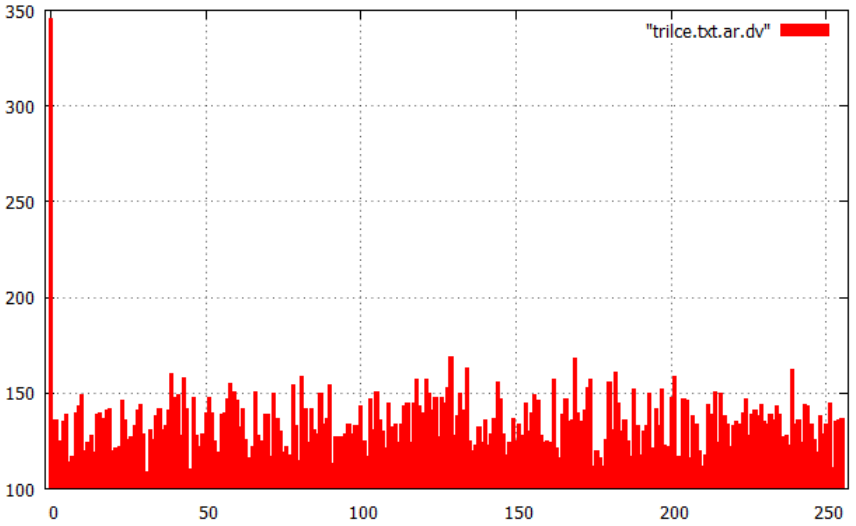
(j) Frecuencias del archivo “humanos.txt”



(k) Frecuencias del archivo “prosa.txt”

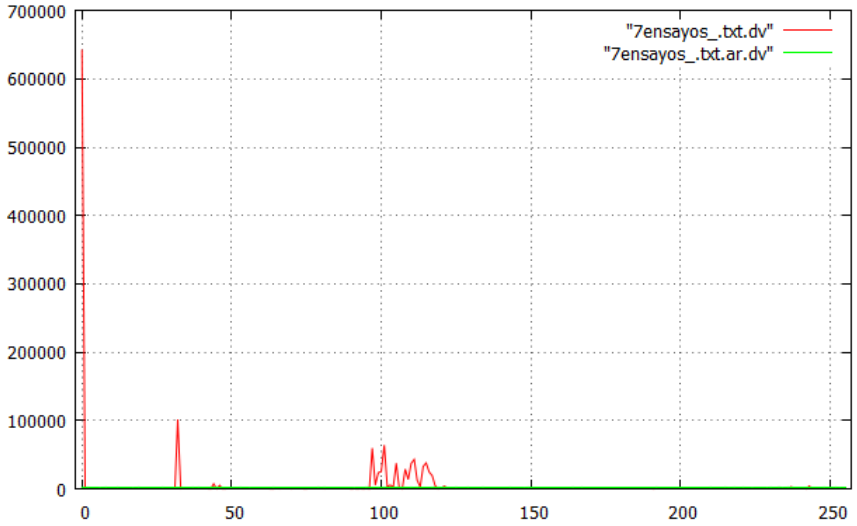


(l) Frecuencias del archivo “tadiciones.txt”

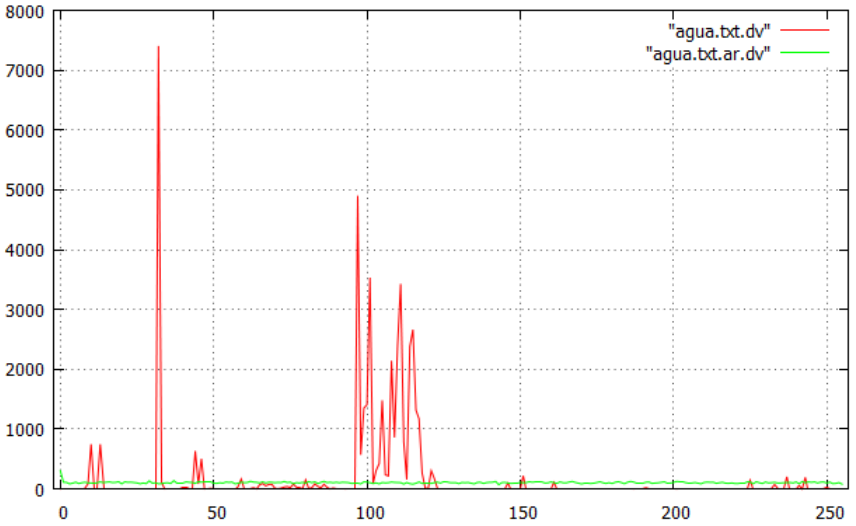


(m) Frecuencias del archivo “trilce.txt”

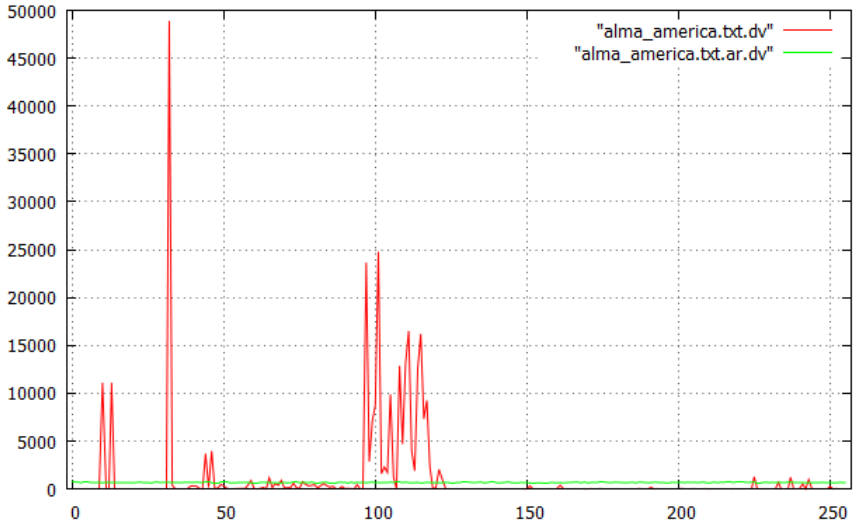
Anexo I: Comparación de las frecuencias antes y después de la compresión



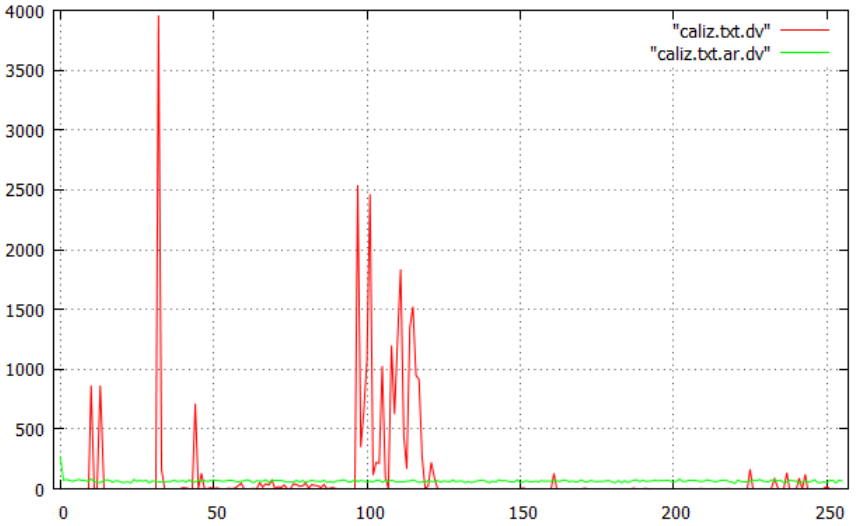
(a) Frecuencias del archivo “7ensayos.txt”



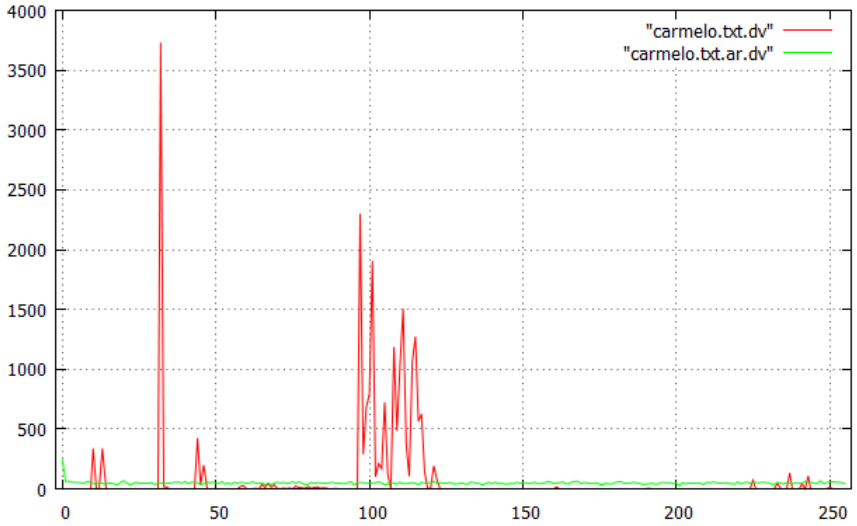
(b) Frecuencias del archivo “agua.txt”



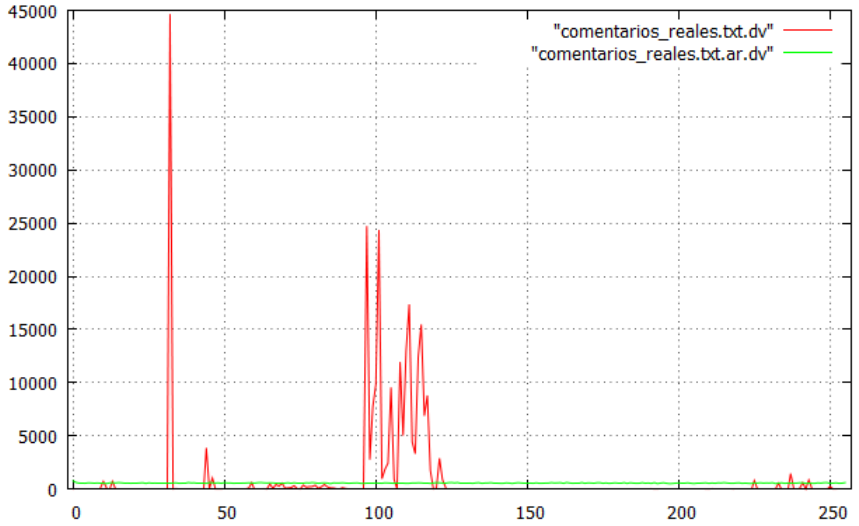
(c) Frecuencias del archivo “alma_america.txt”



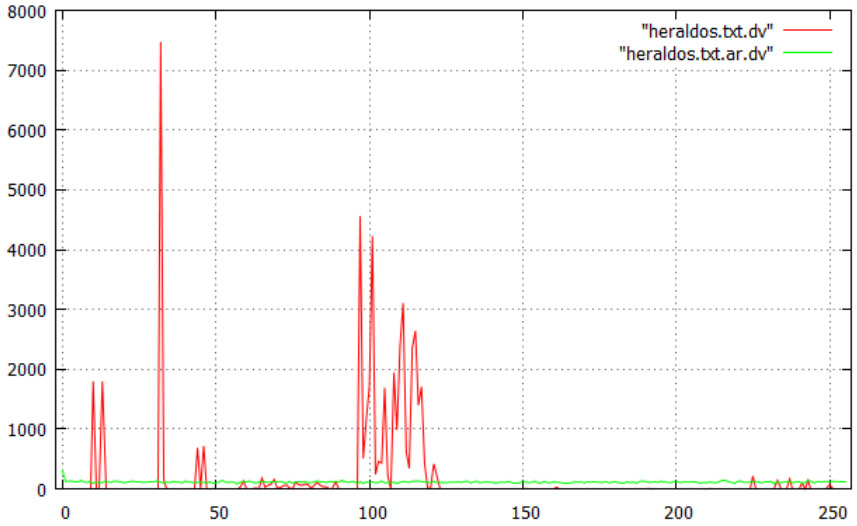
(d) Frecuencias del archivo “caliz.txt”



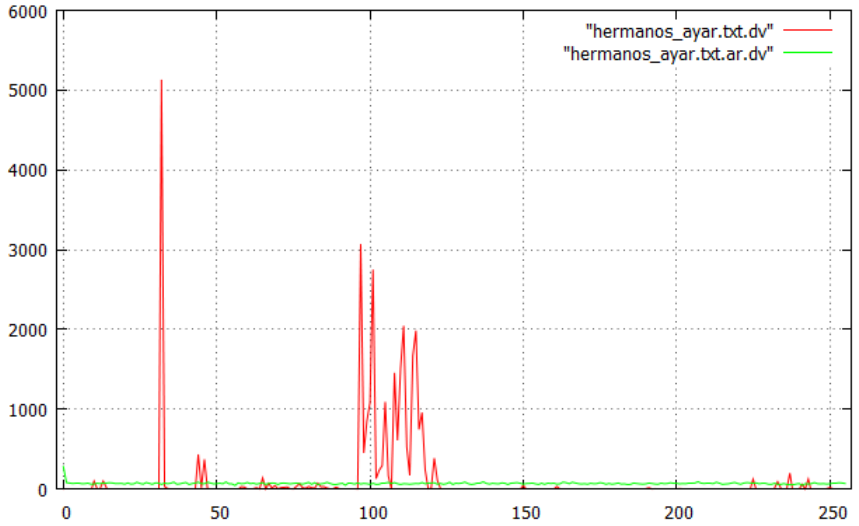
(f) Frecuencias del archivo “carmelo.txt”



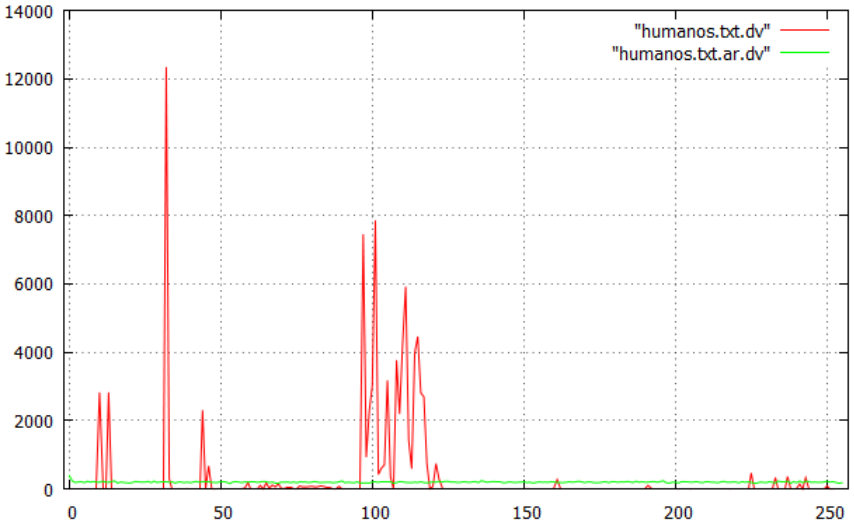
(g) Frecuencias del archivo “comentarios_reales.txt”



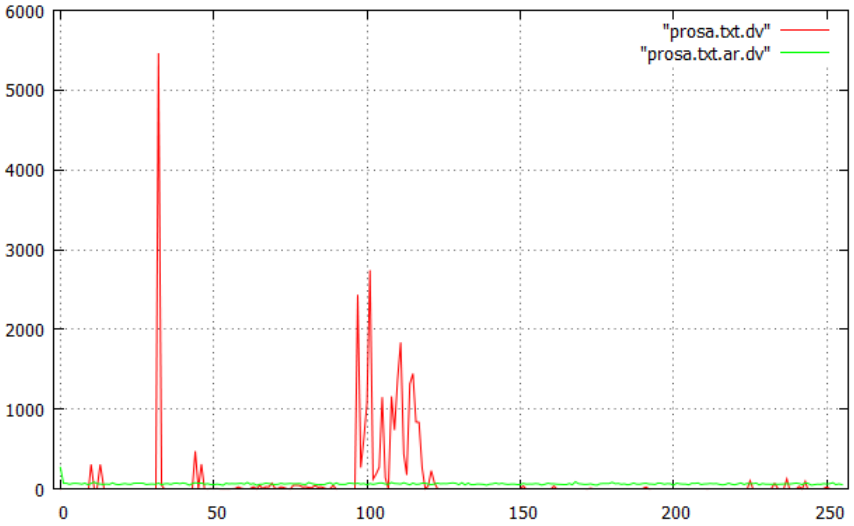
(h) Frecuencias del archivo “heraldos.txt”



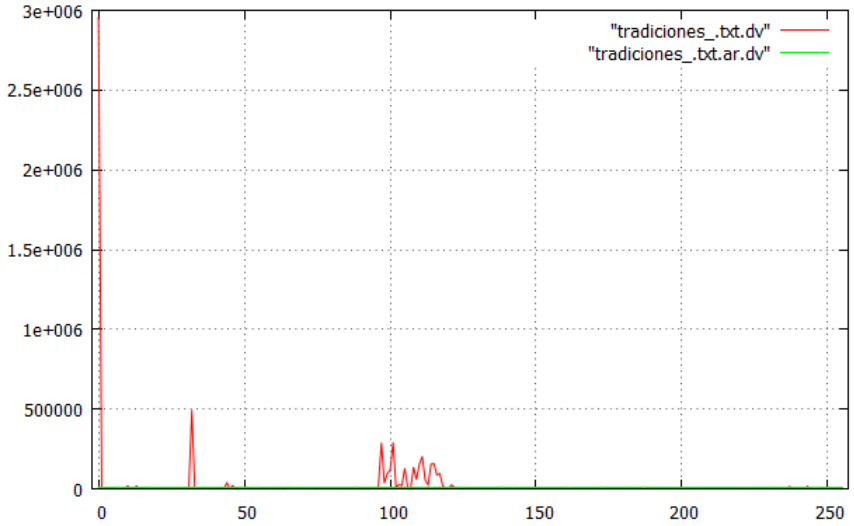
(i) Frecuencias del archivo “hermanos_ayar.txt”



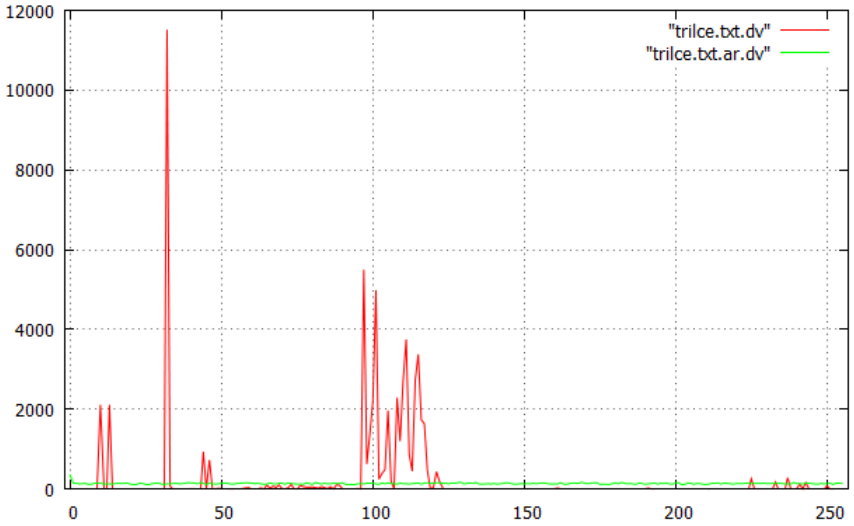
(j) Frecuencias del archivo “humanos.txt”



(k) Frecuencias del archivo “prosa.txt”



(l) Frecuencias del archivo “tadiciones.txt”



(m) Frecuencias del archivo “trilce.txt”

Anexo J: Símbolos más frecuentes de la colección "Literatura peruana"

Tabla 34. Letra más frecuente en los archivos de la colección “Literatura peruana”.

Letra	**CP-1252**	**a1**	**a2**	**a3**	**a4**	**a5**	**a6**	**a7**	**a8**	**a9**	**a10**	**a11**	**a12**	**TOTAL**	**(%)**
e	101	64155	3533	24784	2463	1906	24351	4227	2750	7857	2743	292263	4985	436017	12.01 %
a	97	59772	4902	23660	2539	2300	24716	4561	3070	7452	2435	290060	5500	430967	11.87 %
o	111	43090	3428	16507	1836	1506	17361	3104	2045	5917	1838	203909	3751	304292	8.38 %
s	115	37963	2664	16188	1524	1273	15466	2642	1985	4457	1446	158204	3375	247187	6.81 %
n	110	37389	2401	13128	1252	1037	12971	2382	1512	4235	1373	152864	2701	233245	6.42 %
r	114	32953	2382	12565	1345	1064	12424	2366	1669	3979	1315	154189	2733	228984	6.31 %
l	108	28943	2144	12873	1199	1187	11934	1946	1457	3768	1161	136921	2291	205824	5.67 %
i	105	37679	1480	9878	1026	725	9545	1693	1092	3176	1151	130192	1967	199604	5.50 %
d	100	25333	1408	8741	1079	795	9806	1719	1081	3026	1074	118032	2196	174290	4.80 %
u	117	19523	1170	9242	914	627	8795	1709	958	2694	835	96085	1639	144191	3.97 %
c	99	23697	1351	7028	658	682	7749	1161	851	2276	635	94878	1374	142340	3.92 %
t	116	24737	1319	7314	950	564	6877	1399	745	2810	840	88040	1730	137325	3.78 %
m	109	13842	860	4685	625	486	5019	986	609	2190	739	58520	1196	89757	2.47 %
p	112	12853	785	3959	450	368	4417	627	525	1416	437	55615	876	82328	2.27 %
.	44	7342	636	3696	713	424	3851	687	433	2302	474	37666	934	59158	1.63 %
b	98	5688	570	2849	350	290	2758	510	451	932	268	35647	624	50937	1.40 %
g	103	5494	310	2289	221	211	1842	460	232	605	187	26199	386	38436	1.06 %
y	121	3961	304	2030	222	192	2880	419	386	741	229	25163	433	36960	1.02 %
q	113	3218	157	1901	168	105	3310	345	170	592	176	25619	445	36206	1.00 %
.	46	5193	504	3960	130	199	1022	719	370	672	311	20648	725	34453	0.95 %
v	118	4787	267	2293	263	144	1734	428	244	742	266	22175	468	33811	0.93 %
h	104	3018	425	1713	215	169	2422	433	293	704	271	21515	485	31663	0.87 %
ó	243	4303	194	983	122	109	841	144	124	327	95	16668	160	24070	0.66 %

Continua en la siguiente página

Tabla 34 – Continuación de la anterior página

Letra	CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL	(%)
í	237	3312	203	1207	135	133	1444	166	198	362	125	15615	282	23182	0.64 %
f	102	3584	91	1597	116	100	958	244	128	424	122	13823	247	21434	0.59 %
z	122	1975	171	1090	95	80	870	215	107	291	77	11275	188	16434	0.45 %
j	106	1508	236	1229	99	121	912	220	163	353	140	11109	203	16293	0.45 %
á	225	2042	145	1262	164	73	783	215	122	464	103	9135	254	14762	0.41 %
é	233	1314	73	687	85	45	540	144	91	331	69	7564	172	11115	0.31 %
ñ	241	810	58	512	91	42	587	100	58	141	31	6909	114	9453	0.26 %
E	69	1889	76	893	74	34	545	157	41	147	71	5317	98	9342	0.26 %
A	65	682	70	1165	51	39	452	182	137	177	45	5025	96	8121	0.22 %
C	67	673	52	561	43	45	410	60	70	113	27	5464	68	7586	0.21 %
P	80	1084	153	432	49	20	348	71	31	76	28	5153	50	7495	0.21 %
L	76	1399	78	757	43	24	353	115	31	84	46	4343	85	7358	0.20 %
S	83	590	84	571	30	17	406	110	72	88	43	4157	56	6224	0.17 %
;	59	342	165	872	47	28	586	135	24	189	9	2627	32	5056	0.14 %
D	68	337	75	380	32	15	267	82	12	60	25	3360	39	4684	0.13 %
M	77	417	29	474	37	12	169	70	66	59	47	3118	58	4556	0.13 %
ú	250	824	26	262	21	12	233	74	17	89	24	2522	79	4183	0.12 %
I	73	463	32	598	33	3	302	66	24	51	15	2398	112	4097	0.11 %
x	120	929	7	137	24	9	57	18	9	72	37	2387	33	3719	0.10 %
!	33	210	110	387	162	18	1	156	31	297	46	2076	66	3560	0.10 %
R	82	434	20	345	35	12	207	59	10	64	14	2065	19	3284	0.09 %
:	58	295	40	412	27	21	74	34	27	31	25	2150	25	3161	0.09 %
¡	161	16	110	381	130	17	1	27	31	272	36	2070	22	3113	0.09 %
T	84	289	42	334	25	6	195	61	26	72	22	1879	35	2986	0.08 %

Continua en la siguiente página

Tabla 34 – Continuación de la anterior página

Letra	CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL	(%)
V	86	344	77	276	37	7	86	34	11	42	12	1945	56	2927	0.08 %
N	78	477	23	321	22	10	211	59	17	58	44	1488	46	2776	0.08 %
Y	89	284	15	257	10	4	112	122	21	75	41	1527	92	2560	0.07 %
H	72	200	14	139	12	7	125	54	20	52	26	1252	28	1929	0.05 %
B	66	150	90	130	15	12	57	29	1	21	4	1399	18	1926	0.05 %
G	71	285	3	126	19	1	90	24	17	11	5	1289	6	1876	0.05 %
O	79	203	5	381	26	5	224	82	8	66	24	790	42	1856	0.05 %
F	70	141	10	101	2	4	73	10	2	14	8	1263	10	1638	0.05 %
J	74	104	39	138	2	8	36	17	3	6	3	1255	6	1617	0.04 %
U	85	300	19	188	11	9	93	35	28	43	26	596	7	1355	0.04 %
?	63	65	20	157	4	5	19	17	15	102	27	851	26	1308	0.04 %
¿	191	58	20	156	4	6	19	7	15	103	24	846	22	1280	0.04 %
Q	81	55	7	104	7	5	52	12	15	58	18	780	33	1146	0.03 %
”	34	677	1	13	1	16	25	0	4	0	0	9	0	746	0.02 %
k	107	144	217	25	1	0	2	1	2	3	1	38	0	434	0.01 %
X	88	46	0	12	7	0	0	0	0	0	0	147	103	315	0.01 %
Z	90	16	0	27	2	0	39	5	0	3	1	160	2	255	0.01 %
ü	252	50	0	33	3	0	7	0	0	2	0	141	1	237	0.01 %
É	201	13	1	53	1	0	12	1	4	3	0	126	0	214	0.01 %
Í	205	10	2	30	1	0	16	3	0	3	0	53	0	118	0.00 %
Á	193	10	0	19	1	0	9	3	3	0	2	60	1	108	0.00 %
w	119	19	15	23	4	0	2	3	2	5	3	27	4	107	0.00 %
W	87	12	27	24	0	0	0	0	0	2	0	24	0	89	0.00 %

Continua en la siguiente página

Tabla 34 – Continuación de la anterior página

Letra	CP-1252	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	TOTAL	(%)
K	75	5	22	45	0	0	0	0	0	0	0	9	0	81	0.00 %
Ó	211	11	0	17	0	0	17	6	0	5	1	20	1	78	0.00 %
Ñ	209	3	0	20	3	0	4	1	0	1	0	24	0	56	0.00 %
Ú	218	7	0	9	3	0	6	0	0	2	0	7	0	34	0.00 %

Tabla 35. Lista de identificadores

Nombre de archivo	Identificador
7ensayos.txt	a1
agua.txt	a2
alma_america.txt	a3
caliz.txt	a4
carmelo.txt	a5
comentarios_reales.txt	a6
heraldos.txt	a7
hermanos_ayar.txt	a8
humanos.txt	a9
prosa.txt	a10
tradiciones.txt	a11
trilce.txt	a12

Anexo K: Frecuencias para usarse en el modelo estático

Tabla 36. Primer grupo de frecuencias. Se consideran todos los archivos de la colección.

Símbolo	**Frecuencia de aparición**
0	65535
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	749
11	1
12	1
13	749
14	1
15	1
16	1
17	1
18	1
19	1
20	6
21	1
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
30	1
31	1
32	13616
33	64
34	13
35	1
36	1
37	1
38	1
39	8
40	19
41	18
42	2
43	1
44	1077
45	90
46	627
47	2
48	11
49	49
50	19
51	10
52	9
53	13
54	14
55	13
56	17
57	9
58	57
59	92
60	1
61	1
62	1
63	23
64	1
65	147

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
66	35
67	138
68	85
69	170
70	29
71	34
72	35
73	74
74	29
75	1
76	134
77	82
78	50
79	33
80	136
81	20
82	59
83	113
84	54
85	24
86	53
87	1
88	5
89	46
90	4
91	15
92	1
93	15
94	8
95	4
96	1
97	7849
98	927
99	2592
100	3174
101	7941

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
102	390
103	700
104	576
105	3635
106	296
107	7
108	3748
109	1634
110	4248
111	5542
112	1499
113	659
114	4170
115	4501
116	2501
117	2626
118	615
119	1
120	67
121	673
122	299
123	1
124	1
125	1
126	1
127	1
128	1
129	1
130	1
131	1
132	1
133	1
134	1
135	1
136	1
137	1

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
138	1
139	1
140	1
141	1
142	1
143	1
144	1
145	3
146	1
147	1
148	1
149	1
150	1
151	10
152	1
153	1
154	1
155	1
156	1
157	1
158	1
159	1
160	1
161	56
162	1
163	1
164	1
165	1
166	1
167	1
168	1
169	1
170	1
171	23
172	1
173	1

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
174	1
175	1
176	1
177	1
178	1
179	1
180	1
181	1
182	1
183	1
184	1
185	1
186	1
187	24
188	1
189	1
190	1
191	23
192	1
193	1
194	1
195	1
196	1
197	1
198	1
199	1
200	1
201	3
202	1
203	1
204	1
205	2
206	1
207	1
208	1
209	1

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
210	1
211	1
212	1
213	1
214	1
215	1
216	1
217	1
218	1
219	1
220	1
221	1
222	1
223	1
224	1
225	268
226	1
227	1
228	1
229	1
230	1
231	1
232	1
233	202
234	1
235	1
236	1
237	422
238	1
239	1
240	1
241	172
242	1
243	438
244	1
245	1

Continua en la siguiente página

Tabla 36 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
246	1
247	1
248	1
249	1
250	76
251	1
252	4
253	1
254	1
255	1

Tabla 37. Segundo grupo de frecuencias. Se excluyen archivos con caracteres UNICODE

Símbolo	Frecuencia de aparición
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	83
10	20892
11	1
12	1
13	20892
14	1
15	1
16	1
17	1
18	1
19	1
20	1
21	1

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
22	1
23	1
24	1
25	1
26	1
27	1
28	1
29	1
30	1
31	1
32	150582
33	1274
34	60
35	10
36	5
37	6
38	23
39	294
40	459
41	418
42	46
43	1
44	14150
45	310
46	8612
47	100
48	74
49	519
50	344
51	85
52	52
53	52
54	41
55	49
56	67
57	74

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
58	716
59	2087
60	49
61	1
62	59
63	392
64	8
65	2414
66	377
67	1449
68	987
69	2136
70	234
71	302
72	477
73	1236
74	258
75	67
76	1616
77	1021
78	811
79	863
80	1258
81	311
82	785
83	1477
84	818
85	459
86	638
87	53
88	122
89	749
90	79
91	60
92	31
93	10

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
94	441
95	13
96	1
97	81135
98	9602
99	23765
100	30925
101	79599
102	4027
103	6743
104	7130
105	31733
106	3676
107	252
108	39960
109	17395
110	42992
111	57293
112	13860
113	7369
114	41842
115	51020
116	24548
117	28583
118	6849
119	61
120	403
121	7836
122	3184
123	19
124	22
125	2
126	2
127	1
128	1
129	1

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
130	1
131	1
132	1
133	3
134	1
135	1
136	1
137	1
138	1
139	1
140	1
141	1
142	1
143	1
144	1
145	1
146	97
147	44
148	45
149	10
150	42
151	603
152	1
153	1
154	1
155	1
156	1
157	1
158	1
159	1
160	1
161	1027
162	1
163	1
164	1
165	4

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
166	6
167	2
168	1
169	6
170	1
171	43
172	1
173	9
174	28
175	1
176	2
177	1
178	1
179	1
180	1
181	1
182	1
183	1
184	1
185	1
186	1
187	60
188	1
189	1
190	1
191	376
192	1
193	38
194	1
195	1
196	1
197	1
198	1
199	1
200	1
201	75

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
202	1
203	1
204	1
205	55
206	1
207	1
208	1
209	29
210	1
211	47
212	1
213	1
214	1
215	1
216	1
217	1
218	20
219	1
220	2
221	1
222	1
223	1
224	1
225	3585
226	1
227	1
228	1
229	1
230	1
231	1
232	1
233	2237
234	1
235	1
236	1
237	4255

Continua en la siguiente página

Tabla 37 – Continuación de la anterior página

Símbolo	Frecuencia de aparición
238	1
239	1
240	1
241	1734
242	1
243	3099
244	1
245	1
246	1
247	1
248	1
249	1
250	837
251	1
252	46
253	1
254	1
255	1

Anexo L: Matriz de consistencia

TÍTULO: *"Uso de un modelo de codificación de algoritmos para un compresor aritmético"*

PROBLEMA	OBJETIVO	HIPÓTESIS	VARIABLES	TIPO DE INVESTIGACIÓN	DISEÑO DE INVESTIGACIÓN	TÉCNICAS E INSTRUMENTOS
General: - ¿Cómo influye el uso de un modelo de compresión aritmética en el desempeño de compresión basado en caracteres de texto plano?	General: - Determinar la influencia del uso de un modelo de compresión aritmético en el desempeño de la compresión basado en caracteres de texto plano.	General: - El uso de un modelo de compresión aritmético influye significativamente en el desempeño de la compresión basado en caracteres de texto plano.	* *V. Independiente:* Modelo de compresión aritmético. Indicadores: - Modelo estadístico. - Codificador aritmético. * *V. Dependiente:* Desempeño del compresor aritmético. Indicadores: - Tiempo de compresión - Tasa de compresión - Coeficiente de variación (CV) - Cuadrante mágico de gartner (CMG)	Descriptivo correlacional	No experimental, transeccional descriptivo	*Técnica:* Observación *Herramientas:* Guía de observación
Específico: - ¿Cuál es el nivel de correlación del modelo estadístico con el desempeño de la compresión basado en caracteres de texto plano? - ¿En que medida influye el algoritmo codificador aritmético en el desempeño de la compresión basado en caracteres de texto plano?	Específico: - Identificar el nivel de correlación del modelo estadístico con el desempeño de la compresión basado en caracteres de texto plano. - Determinar la influencia del algoritmo codificador aritmético en el desempeño de la compresión basado en caracteres de texto plano.	Específico: - Existe correlación directa entre el modelo estadístico y el desempeño de la compresión basado en caracteres de texto plano. - El algoritmo codificador aritmético influye significativamente en el desempeño de la compresión basado en caracteres de texto plano.				