Codificación para fuentes con memoria

Autor: Gerez Jimenez, Juan Jose Armando

Resumen

Este informe explora la codificación de fuentes con memoria, enfocándose en las fuentes de Markov y el algoritmo de compresión LZ77. Se investiga el concepto de cadenas de Markov / fuentes de Markov, entropía condicional y LZ77, y cómo estos conceptos se aplican en la compresión de datos. Además, se desarrolla una implementación en Python que analiza un texto para generar una tabla de frecuencias condicionales, modela una fuente de Markov para generar texto aleatorio y aplica el algoritmo LZ77 para comprimir y descomprimir cadenas de caracteres. Los resultados muestran la eficacia de LZ77 en la compresión de datos generados a partir de fuentes con memoria, acercándose al límite teórico de compresión dado por la entropía condicional.

Introducción

Este informe se basa en los principios establecidos en el libro "Principles of Digital Communication" de Gallager (2008), centrándose en las secciones que abordan las fuentes de Markov y el algoritmo LZ77.

La codificación eficiente de fuentes con memoria es esencial en sistemas de comunicaciones digitales para reducir la redundancia de datos y optimizar el uso del ancho de banda.

Las fuentes de Markov son modelos matemáticos que describen procesos donde la probabilidad de transición a un nuevo estado depende del estado actual (capturando así la memoria inmediata del sistema), pero no de la secuencia de estados anteriores. Al utilizar cadenas de Markov para modelar fuentes de información, se pueden representar datos donde existe una dependencia entre elementos consecutivos, reflejando la memoria inherente del proceso (proceso tiene dependencia temporal, es decir, que el estado actual o futuro del sistema depende de estados pasados). Esto es especialmente útil en la compresión y transmisión de datos que presentan patrones o correlaciones temporales, ya que permite diseñar algoritmos que aprovechen estas dependencias para mejorar la eficiencia. La entropía condicional es una medida clave para cuantificar la incertidumbre restante en una fuente dada la memoria del sistema.

El algoritmo LZ77 es un método de compresión sin pérdida que aprovecha las repeticiones en los datos para reducir el tamaño de la información transmitida o almacenada. A diferencia de los códigos de longitud variable tradicionales, LZ77 es universal y no requiere conocimiento previo de las estadísticas de la fuente, adaptándose a la información entrante.

Metodología

Basándonos en el libro de Gallagher previamente mencionado, definimos los siguientes conceptos:

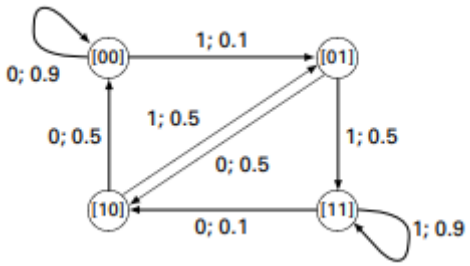
Proceso de Markov y sus propiedades

Una cadena de Markov es un proceso estocástico donde la probabilidad de transición a un estado futuro depende únicamente del estado presente y no de los estados pasados. Esto se conoce como propiedad de Markov. Las cadenas de Markov se caracterizan por tener un conjunto finito de estados y transiciones probabilísticas entre ellos. Las transiciones entre estados se describen mediante una matriz de probabilidades de transición, y las cadenas pueden ser ergódicas, lo que significa que desde cualquier estado es posible llegar a cualquier otro

Fuente de Markov

Una fuente de Markov es una extensión de la cadena de Markov aplicada a la generación de símbolos. Cada transición entre estados, como se ve en la figura 1, está etiquetada con un símbolo de salida, y la probabilidad de emisión de un símbolo depende del estado actual. Esto permite modelar fuentes con memoria, ya que la emisión actual está condicionada al estado previo del sistema.

figura 1 - Fuente de Markov (Gallagher (2008) - 2.8 Markov sources - Principles of digital communication)



Entropía Condicional

La entropía condicional H(X|S) cuantifica la incertidumbre promedio del símbolo de salida X dado el estado actual S de la fuente de Markov. Se calcula como en la ecuación 1 :

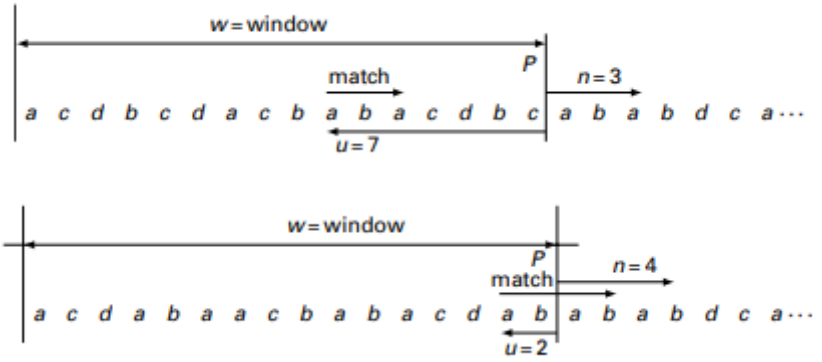
$$\mathrm{H}[X|S] = \sum_{s\in\mathcal{S}} \sum_{x\in\mathcal{X}} q(s) P(x|s) \log \frac{1}{P(x|s)}. \qquad \textit{(Ecuación 1)}$$

donde q_s es la probabilidad de estar en el estado *s* y $P_{X|S}(x|s)$ es la probabilidad condicional de emitir el símbolo *x* dado el estado *s*. En el contexto de las fuentes de Markov, la entropía condicional desempeña un rol similar al de la entropía en fuentes sin memoria. Para una fuente de Markov, la entropía condicional es el promedio ponderado de la entropía de la distribución condicional de los símbolos

Algoritmo LZ77

El algoritmo LZ77 es un método de compresión que utiliza una ventana deslizante para buscar y codificar repeticiones en los datos. Al encontrar coincidencias de cadenas previas en la ventana *w*, LZ77 reemplaza la cadena repetida por un par *(n,u)*, donde *n* es la longitud de la coincidencia y *u* es la distancia desde la posición actual hasta la coincidencia anterior, esto puede verse un poco más claro en la figura 2.

figura 2 - w, n y u para una cadena de caracteres (Gallagher (2008) - 2.9.1 The Pz77 algorithm - Principles of digital communication)



Resultados y discusión

"Haciendo uso del código en Python desarrollado, se pudo ver que el algoritmo LZ77 utilizado sobre la tabla de frecuencias obtenida de nuestro libro de prueba, efectivamente logró realizar la compresión de datos reduciendo así la cantidad de bits utilizados"

Desarrollo del análisis y generación de la tabla de frecuencias condicionales

Se desarrolló una rutina en Python que lee un archivo de texto en formato UTF-8 y genera una tabla de frecuencias condicionales basada en las últimas N palabras encontradas. Se utilizó expresiones regulares para procesar el texto y eliminar puntuaciones, obteniendo así una lista de palabras.

La tabla de frecuencias se implementó como un diccionario anidado, donde la clave externa es una tupla de N palabras (el estado) y la clave interna es la palabra siguiente, almacenando la frecuencia de aparición.

Modelado de la fuente de Markov y generación de texto aleatorio

Utilizando la tabla de frecuencias condicionales, se modeló una fuente de Markov que genera texto aleatorio. Se selecciona aleatoriamente un estado inicial y, basándose en las probabilidades condicionales, se elige la siguiente palabra. Este proceso se repite hasta alcanzar la longitud deseada del texto generado.

Implementación del algoritmo LZ77

Se implementó el algoritmo LZ77 para la compresión y descompresión de cadenas de caracteres. La compresión se realiza identificando la coincidencia más larga dentro de una ventana deslizante de tamaño w y codificando las repeticiones utilizando el código unario-binario para la longitud de la coincidencia y un código de longitud fija para el desplazamiento u.

La descompresión reconstruye el texto original utilizando la información codificada de las coincidencias y los literales.

Evaluación del índice de compresión

Se evaluó el índice de compresión de la implementación de LZ77 comparando el número de bits ocupados por el texto original codificado en UTF-8 y el número de bits utilizados por LZ77. Para ello, se utilizó el texto generado por la fuente de Markov previamente desarrollada.

Los resultados obtenidos del código desarrollado mostraron que el algoritmo LZ77 logró una compresión efectiva, alcanzando aproximadamente un 78.88% del tamaño original. Esto demuestra la capacidad de LZ77 para aprovechar las redundancias presentes en fuentes con memoria y acercarse al límite teórico establecido por la entropía condicional.

Interfaz de Programación de Aplicaciones (API) del software desarrollado

A continuación, se detalla la API del software, describiendo cada función, sus parámetros y su funcionalidad:

1. **generar_tabla_frecuencias_condicionales(archivo, N)**
    - **Descripción**: Genera una tabla de frecuencias condicionales de palabras a partir de un archivo de texto.
    - **Parámetros**:
        - `archivo` (str): Ruta del archivo de texto en formato UTF-8 a analizar.
        - `N` (int): Número de palabras anteriores (contexto) a considerar para calcular las frecuencias condicionales.
    - **Retorno**: Un diccionario anidado (`defaultdict`) que representa la tabla de frecuencias condicionales.
2. **guardar_tabla_frecuencias_en_archivo(tabla_frecuencias, nombre_archivo="Tablas_Frecuencias.txt")**
    - **Descripción**: Guarda la tabla de frecuencias condicionales generada en un archivo de texto.
    - **Parámetros**:
        - `tabla_frecuencias` (dict): La tabla de frecuencias condicionales a guardar.
        - `nombre_archivo` (str, opcional): Nombre del archivo de destino. Por defecto es "Tablas_Frecuencias.txt".
    - **Retorno**: None.
3. **generar_texto_markov(tabla_frecuencias, longitud_texto, N)**

- **Descripción**: Genera un texto aleatorio basado en un modelo de Markov utilizando la tabla de frecuencias condicionales.
- **Parámetros**:
  - `tabla_frecuencias` (dict): La tabla de frecuencias condicionales utilizada para el modelo de Markov.
  - `longitud_texto` (int): Número de palabras que contendrá el texto generado.
  - `N` (int): Número de palabras en el estado (contexto) del modelo de Markov.
- **Retorno**: Una cadena de texto con el texto generado.

4. **guardar_texto_en_archivo(texto, nombre_archivo='Fuente_Markov.txt')**
   - **Descripción**: Guarda el texto generado en un archivo de texto.
   - **Parámetros**:
     - `texto` (str): El texto a guardar.
     - `nombre_archivo` (str, opcional): Nombre del archivo de destino. Por defecto es "Fuente_Markov.txt".
   - **Retorno**: None.
5. **utf8_a_binary(texto)**
   - **Descripción**: Convierte una cadena de texto en una representación binaria utilizando codificación UTF-8.
   - **Parámetros**:
     - `texto` (str): El texto a convertir.
   - **Retorno**: Una cadena de caracteres '0' y '1' representando el texto en binario.
6. **codigo_unario_binario(n)**
   - **Descripción**: Codifica un entero positivo utilizando el código unario-binario.
   - **Parámetros**:
     - `n` (int): El entero a codificar.
   - **Retorno**: Una cadena binaria que representa el entero codificado.
7. **decodif_unario_binario(binario, i)**
   - **Descripción**: Decodifica un número codificado en unario-binario desde una cadena binaria a partir de una posición dada.
   - **Parámetros**:
     - `binario` (str): La cadena binaria que contiene el número codificado.
     - `i` (int): Índice inicial en la cadena binaria desde donde comienza la decodificación.
   - **Retorno**: Una tupla `(n, i)` donde `n` es el número decodificado y `i` es el índice actualizado después de la decodificación.
8. **lz77_comprimir(texto, ventana_tamano)**
   - **Descripción**: Aplica el algoritmo LZ77 para comprimir una cadena de texto.
   - **Parámetros**:
     - `texto` (str): El texto a comprimir.
     - `ventana_tamano` (int): Tamaño de la ventana deslizante utilizada en el algoritmo.
   - **Retorno**: Una cadena binaria que representa el texto comprimido.
9. **lz77_descomprimir(binario, ventana_tamano)**

- **Descripción**: Descomprime una cadena binaria que fue comprimida utilizando el algoritmo LZ77.
- **Parámetros**:
  - `binario` (str): La cadena binaria comprimida.
  - `ventana_tamano` (int): Tamaño de la ventana deslizante utilizada durante la compresión.
- **Retorno**: Una cadena de texto que representa el texto descomprimido.

Estas funciones permiten modularizar el software y facilitan su reutilización y mantenimiento. Además, proporcionan una interfaz clara para interactuar con el software, permitiendo a otros desarrolladores o usuarios integrar estas funcionalidades en otros proyectos.

Conclusiones

La exploración de las fuentes de Markov y la entropía condicional proporciona una base sólida para comprender cómo las dependencias en los datos pueden ser explotadas para la compresión. El algoritmo LZ77, al no requerir conocimiento previo de las estadísticas de la fuente y adaptarse dinámicamente, es efectivo para comprimir datos generados por fuentes con memoria.

La implementación práctica de estos conceptos demostró que es posible lograr una comprensión adecuada. El desarrollo del modelo de fuente de Markov y la generación de texto aleatorio permitieron visualizar cómo las probabilidades condicionales influyen en la estructura de los datos y, por ende, en su compresibilidad.

Referencias

Gallager, R. G. (2008). *Principles of Digital Communication*. Cambridge University Press.