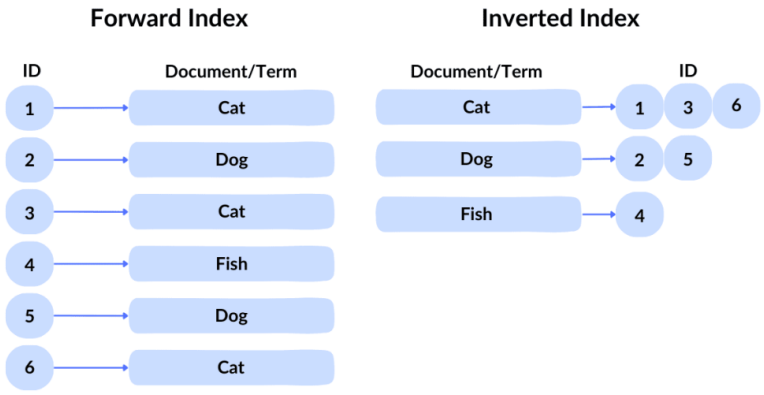
Escuela de Informática
Estructura de datos
Prof. Steven Brenes Chavarría
Primer proyecto de programación

Estructura de datos: índice invertido

Un **índice invertido** (en inglés *inverted index*) es una estructura de datos utilizada para **buscar rápidamente palabras o frases** en un conjunto de documentos. A diferencia de un índice tradicional (como el de un libro, que te lleva de un tema a una página), el índice invertido **asocia cada término con los documentos en los que aparece**.



Fuente de imagen: spotintelligence.com/2023/10/30/inverted-indexing/

El índice invertido surge del mundo de las **bibliotecas** y la **recuperación de información**. Su uso se remonta a los **catálogos bibliográficos** y más adelante, a los **sistemas computacionales de búsqueda de texto** en los años 60-70. Sin embargo, cobró verdadera relevancia con el crecimiento de la **web** y los **motores de búsqueda**, como AltaVista, Google, y otros, que necesitaban una forma rápida y eficiente de buscar palabras clave en miles o millones de documentos.

El índice invertido se utiliza ampliamente en diversas aplicaciones, como en los motores de búsqueda (por ejemplo, Google o Bing), que indexan miles de millones de páginas web para permitir búsquedas instantáneas; en sistemas de recuperación de información, como bibliotecas digitales, bases de datos académicas o plataformas legales; en el procesamiento de lenguaje natural (NLP), para tareas como análisis de textos, detección de temas o extracción de información; en plataformas de comercio electrónico, donde permite a los usuarios encontrar productos por nombre, descripción o atributos; y en buscadores de archivos locales, como Windows Search o Spotlight en macOS.

Procesamiento de documentos

Antes de construir el índice invertido, cada documento pasa por una serie de procesos:

- **Tokenización:** separar el texto en palabras (tokens).
- **Normalización:** convertir todo a minúsculas, eliminar tildes, signos de puntuación, etc.
- **Eliminación de stopwords:** omitir palabras comunes como "el", "la", "de", etc.
- **Stemming o lematización:** reducir las palabras a su raíz (ej: *correr*, *corriendo* → *corr*).
  Este último proceso no será necesario realizarlo para el proyecto.

Proceso de recuperación

A continuación, se expone un ejemplo completo para recuperar un documento a partir de una consulta q.

1. Documentos de entrada
D1: "el gato come pescado"
D2: "el perro come carne"
D3: "el gato duerme"
Para efectos del proyecto, se han subido

2. Preprocesamiento
Se eliminan las *stopwords* (como "el") y se tokenizan los documentos:

Documento	***Tokens***
D1	gato, come, pescado
D2	perro, come, carne
D3	gato, duerme

⚠ **Es proyecto de programación debe aplicar la Ley de Zipf, borrando el percentil indicado por el usuario en la construcción del índice invertido.**

3. Índice Invertido

Término	***Documentos***
gato	D1, D3
come	D1, D2
pescado	D1
perro	D2
carne	D2
duerme	D3

4. Matriz de Frecuencia de Términos (TF)

Término	***D1***	***D2***	***D3***
gato	1	0	1
come	1	1	0
pescado	1	0	0
perro	0	1	0
carne	0	1	0
duerme	0	0	1

5. Cálculo de TF-IDF
IDF se calcula con la fórmula:

$$IDF_t = \log_{10}\left(\frac{N}{DF(t)}\right)$$

TF: número de veces que aparece el término en el documento.
DF (document frequency): número de documentos en los que aparece el término.
N: número total de documentos.

Ejemplos:

$$IDF_{gato} = \log_{10}\left(\frac{3}{2}\right) = 0.176$$

$$IDF_{\text{come}} = \log_{10}\left(\frac{3}{2}\right) = 0.176$$
$$IDF_{\text{pescado}} = \log_{10}\left(\frac{3}{1}\right) = 0.477$$
$$IDF_{\text{perro}} = \log_{10}\left(\frac{3}{1}\right) = 0.477$$
$$IDF_{\text{carne}} = \log_{10}\left(\frac{3}{1}\right) = 0.477$$
$$IDF_{\text{duerme}} = \log_{10}\left(\frac{3}{1}\right) = 0.477$$

Matriz TF-IDF:

En este punto se multiplica cada TF por su IDF:

Término	***D1***	***D2***	***D3***
gato	1 x 0.176 = 0.176	0	0.176
come	0.176	0.176	0
pescado	0.477	0	0
perro	0	0.477	0
carne	0	0.477	0
duerme	0	0	0.477

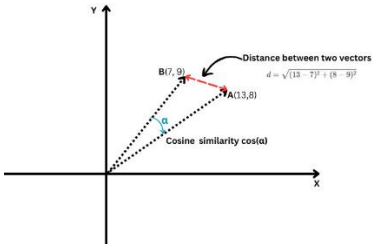
Es decir, cada entrada de esta matriz está definida por:

$$M_{t,d} = \log_{10}\left(\frac{N}{DF(t)}\right) * TF(t,d)$$

6. Similitud del Coseno entre Documentos

Para determinar similitud coseno se utiliza la fórmula:

$$Cos(\theta) = \frac{A * B}{||A|| * ||B||}$$



Consulta: "el gato come"
Después de eliminar la stopword "el", queda: ["gato", "come"]
TF-IDF de la consulta q = [0.176, 0.176, 0, 0, 0, 0]

Similitud coseno entre consulta y documentos:

Consulta vs Documento	***Similitud Coseno***
q vs D1	≈ 0.462
q vs D3	≈ 0.244
q vs D2	≈ 0.175

Por último, se ordena de mayor a menor el valor de similitud para ser mostrado al usuario.

Sobre el proyecto

Alcance y requisitos funcionales

El proyecto debe cumplir con los siguientes elementos:

- Todo el código debe ser realizado por los estudiantes, no se acepta el uso de librerías externas o internas del lenguaje.
- El lenguaje de programación puede ser Java, C# o C++
- En caso de usar C++ el uso de punteros inteligentes es obligatorio.
- El proyecto puede ser en consola, escritorio o web.
- Es necesario implementar la estructura de datos: índice invertido, esto incluye los métodos:
    - Crear desde una ruta (primera vez)
    - Actualizar el índice (agregar nuevos documentos)
    - Aplicar ley de Zipf
    - Guardar en archivo binario
    - Cargar desde archivo binario
    - Otros métodos necesarios de la interface Collection
- Es necesario aplicar la recuperación mediante la distancia coseno de la búsqueda vectorial.
- El proyecto debe estar conformado por al menos:
    - Lista doblemente enlazada circular
    - Vectores: debe estar sobrecargado el operador * para multiplicar dos vectores.
    - Cualquier otra estructura de datos que sea necesaria.
- El uso de la IA queda terminantemente prohibido, si se detecta plagio se calificará con 0 y se abrirá un proceso disciplinario.

Estructura de la entrega

- Carpeta raíz con el proyecto, debe contener un manual en formato PDF en donde se analice el O(n) de cada función, arquitectura de código, UML, análisis de patrones de diseño.
- Es necesario una presentación oral para la defensa del proyecto.
- Un máximo de tres integrantes, sin mínimo.

Política de retrasos

- Penalización por día de atraso o condiciones para entregas extemporáneas. Se restará un 10% por cada día de entrega tardía.
- La fecha de entrega es el día 07 de setiembre a medianoche por aula virtual.

Normas de codificación

- Estándares de indentación, nombres de variables, comentarios y organización del código según normas Camel case.
- Todo código debe cumplir con principios SOLID
- El uso de patrones de diseño es obligatorio: Iterador, Singlenton, Strategy, etc

Biografía

Manning, C. D., Raghavan, P., & Schütze, H. (2025). *Modern information retrieval: The concepts and technology behind search* (2ª ed.). Cambridge University Press.