Diseño e Implementación de un Sistema RI

Índice

Introducción

Introducción

- La implementación de un sistema de recuperación de información contiene multitud de retos que tienen que solucionarse
- A nivel teórico, las definiciones de los modelos son claras
- A nivel práctico, el desarrollo no es tan sencillo
  - ¿Cómo procesar los documentos? ¿Cómo eliminamos los signos de puntuación? ¿Cómo dividimos el texto en términos? ¿Qué algoritmo de stemming utilizar?
  - ¿Qué estructuras de datos utilizar para almacenar el índice invertido?
  - ¿Cómo calculamos el TF-IDF? ¿Se puede calcular de una sola vez?
  - ¿Cómo realizamos la búsqueda?
  - ¿Cómo ordenamos los resultados?

Introducción

- En la actualidad, existen herramientas de software libre para implementar un sistema de recuperación de información
  - Apache Lucene
    - http://lucene.apache.org/core/
  - Solr
    - http://lucene.apache.org/solr/
  - LIRE
    - http://www.lire-project.net/
  - Terrier
    - http://terrier.org/
  - Lemur
    - http://www.lemurproject.org/

Introducción

- Herramientas para el procesamiento del lenguaje natural
  - Stanford's Core NLP Suite
    - http://nlp.stanford.edu/software/
  - Apache Lucene
  - Apache OpenNLP
    - http://opennlp.apache.org/
  - GATE
    - https://gate.ac.uk/
- Herramientas para extracción de texto
  - Apache Tika
    - https://tika.apache.org/

Esquema del Sistema

Esquema del Sistema

- Características del sistema
  - Colección de documentos en texto plano
  - Esquema de ponderación TF-IDF

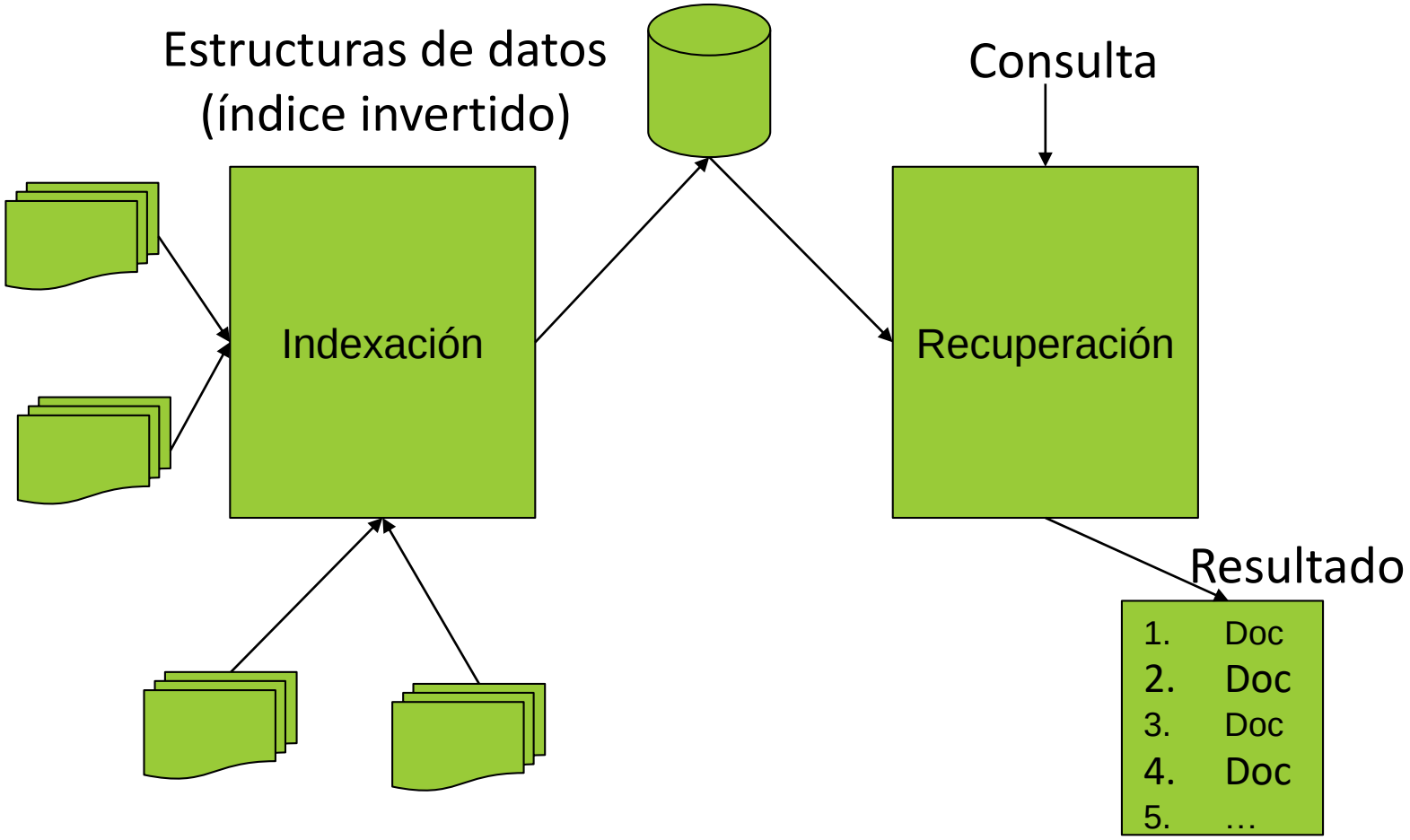
$$w_{i,j} = \begin{cases} (1 + \log f_{i,j}) \times \log \frac{N}{n_i} & Si\ f_{i,j} > 0 \\ 0 & en\ otro\ caso \end{cases}$$

  - Modelo espacio vectorial

$$\cos\theta = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

  - Sistema de recuperación de información basado en texto

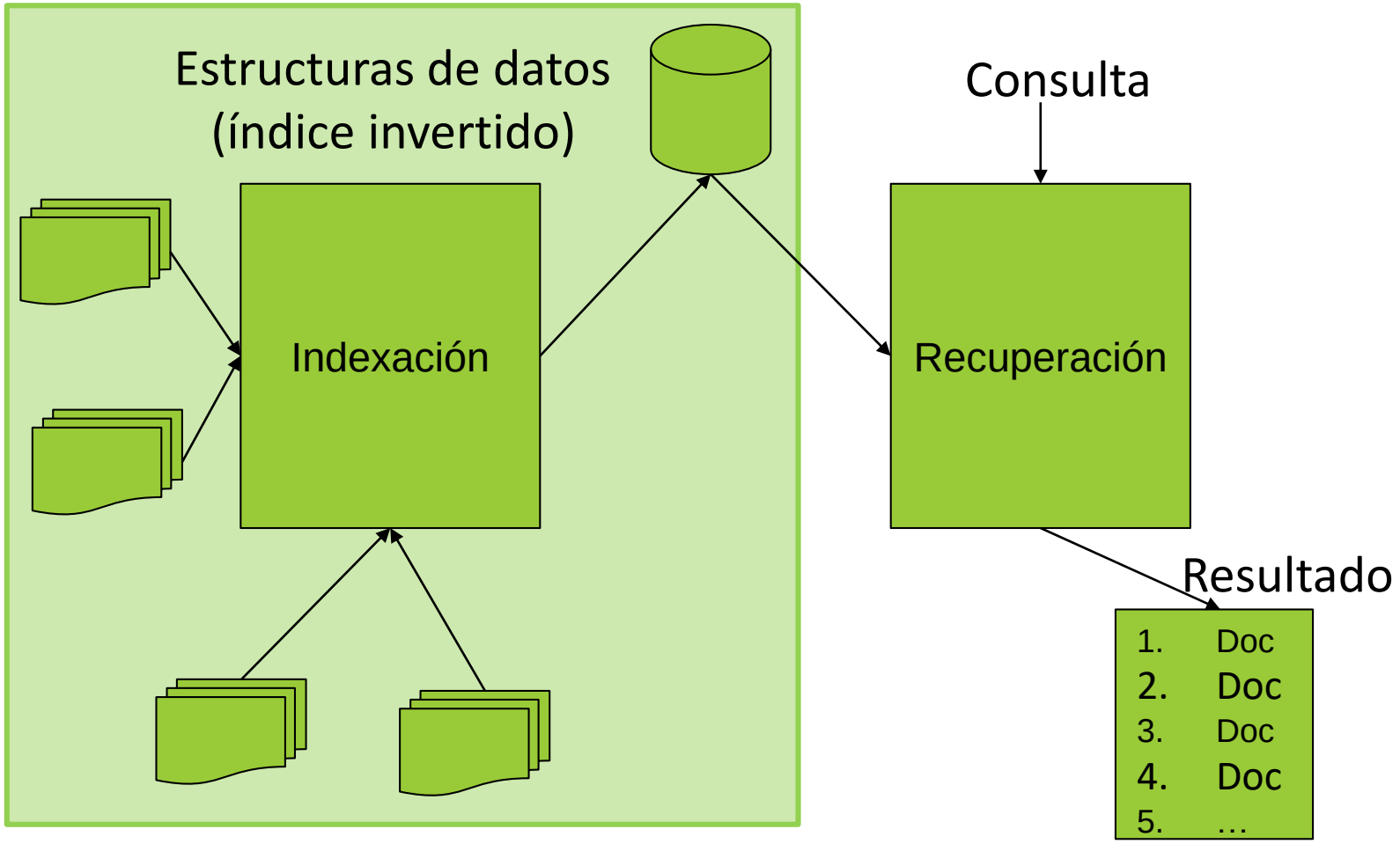
Esquema del Sistema



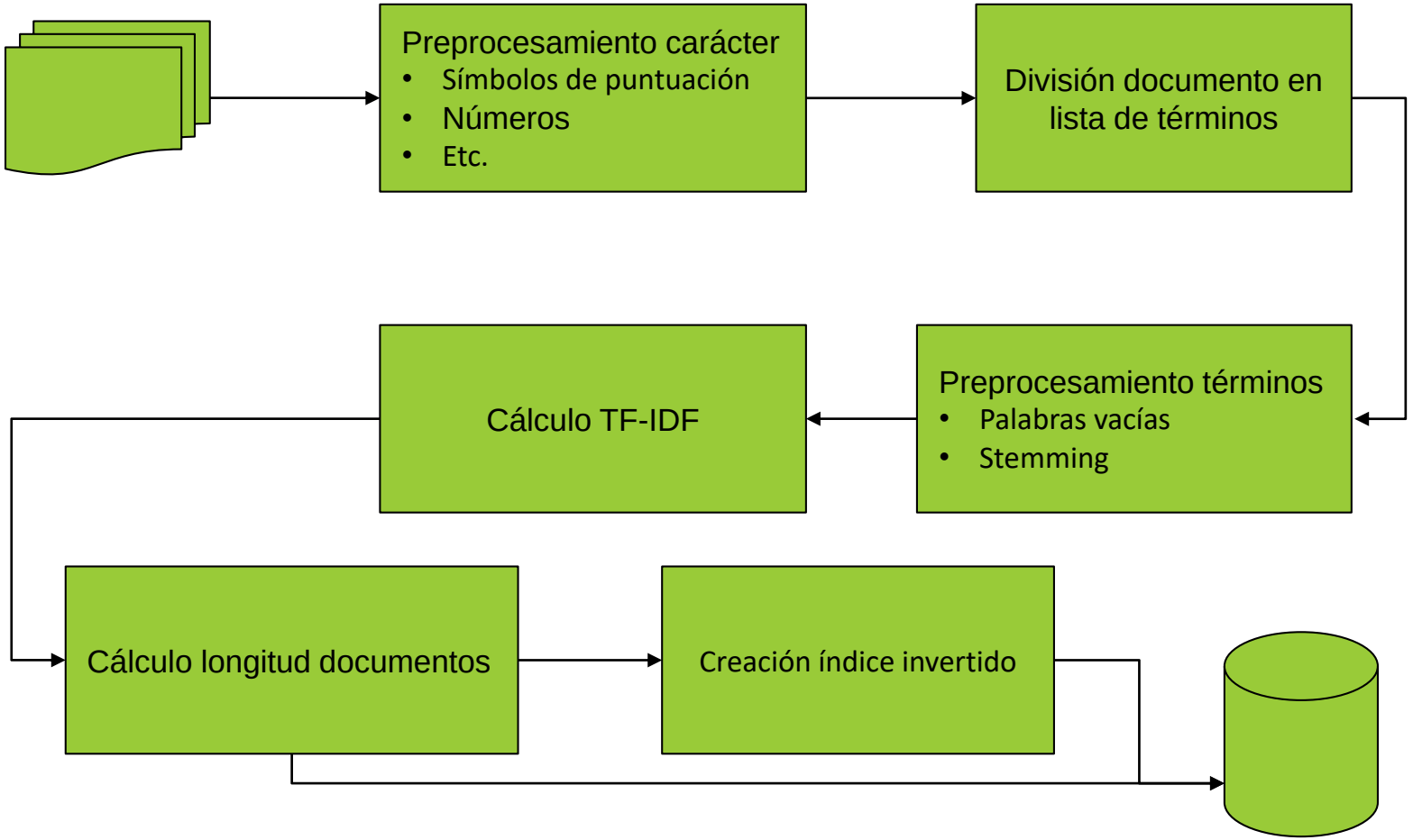
Esquema del Sistema

INDEXACIÓN

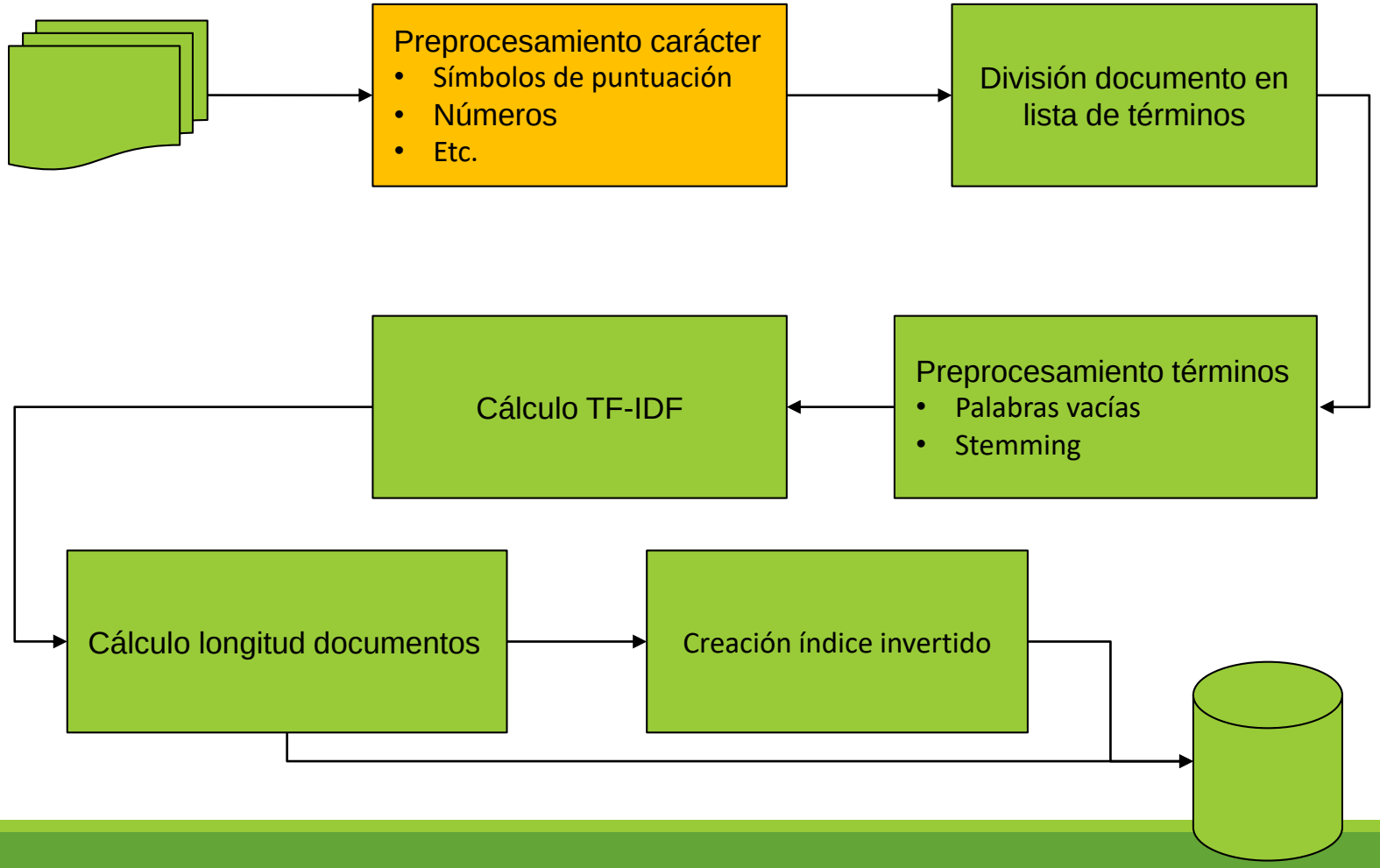
Esquema del Sistema Indexación



Esquema del Sistema Indexación



Esquema del Sistema Indexación



Esquema del Sistema Indexación

- El preprocesamiento se debe realizar para cada documento

Pseudocódigo. Esquema básico indexación

```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  …
Fin
```

Java. Lectura de todo el texto de un documento

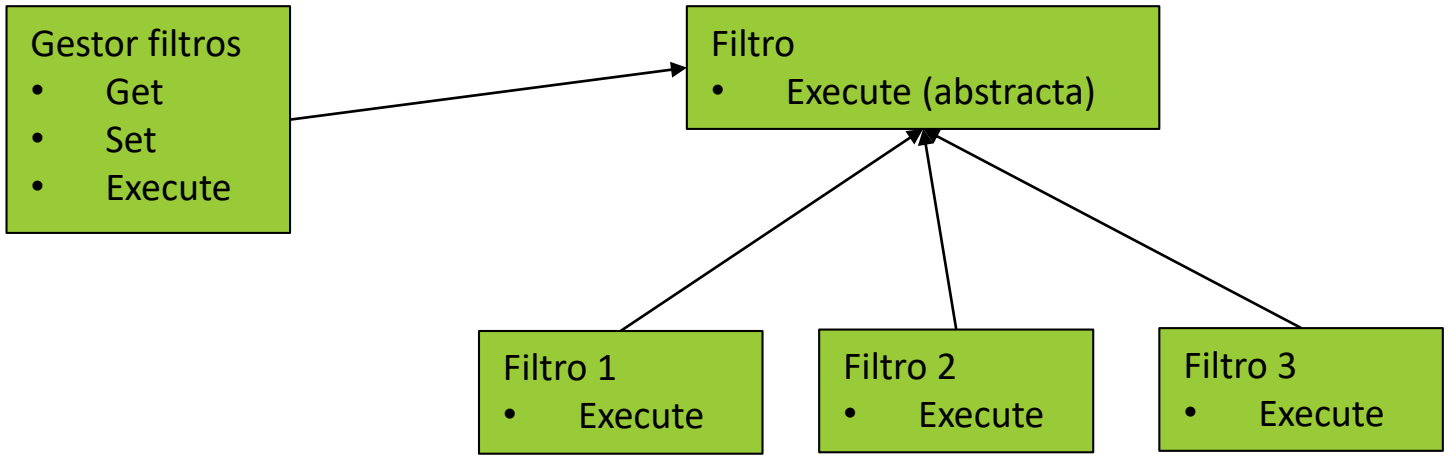
```
texto = new String(Files.readAllBytes(Paths.get(rutaFichero)))
```

Esquema del Sistema Indexación

- Se debe de aplicar la misma secuencia de preprocesamiento a todos los documentos
  - La consulta también tendrá que ser preprocesada siguiendo los mismos pasos
- El preprocesamiento a nivel de carácter se puede realizar utilizando las funciones y métodos asociados al tipo de dato *String* de los distintos lenguajes de programación
  - *replaceAll(patrón, reemplazo)*
- La organización del código para su posterior reutilización es importante
  - Patrón de diseño: cadena de filtros

Esquema del Sistema Indexación

- Una cadena de filtros es una secuencia de filtros que se aplicarán en cascada
  - Cada filtro es una clase
  - El gestor de la secuencia de filtros se encargará de ejecutar cada uno de los filtros de forma secuencia



Esquema del Sistema Indexación

- El gestor de filtros contiene la secuencia de filtros a ejecutar

Pseudocódigo. Ejecutar cadena de filtros

```
Función ejecutar(String s): String

  sProcesada ← s

  Para cada filtro ∈ filtros
    sProcesada ← filtro.ejecutar(sProcesada)
  Fin_para

  devolver sProcesada

Fin
```

Esquema del Sistema Indexación

- El gestor de filtros se crea vacío
- Se tienen que añadir un conjunto de filtros
- El orden en que se añadan, será el orden en el que se ejecuten
  - El orden es muy importante

Pseudocódigo. Añadir filtros al gestor de filtros

```
gestor.añadir(nuevo Filtro(patrón1, reemplazo1))
gestor.añadir(nuevo Filtro(patrón2, reemplazo2))
gestor.añadir(nuevo Filtro(patrón3, reemplazo3))
gestor.añadir(nuevo Filtro(patrón4, reemplazo4))
…
```

- Consejo: agrupar una secuencia de filtros bajo una misma función o clase
  - Se podrá repetir la misma secuencia fácilmente

Esquema del Sistema Indexación

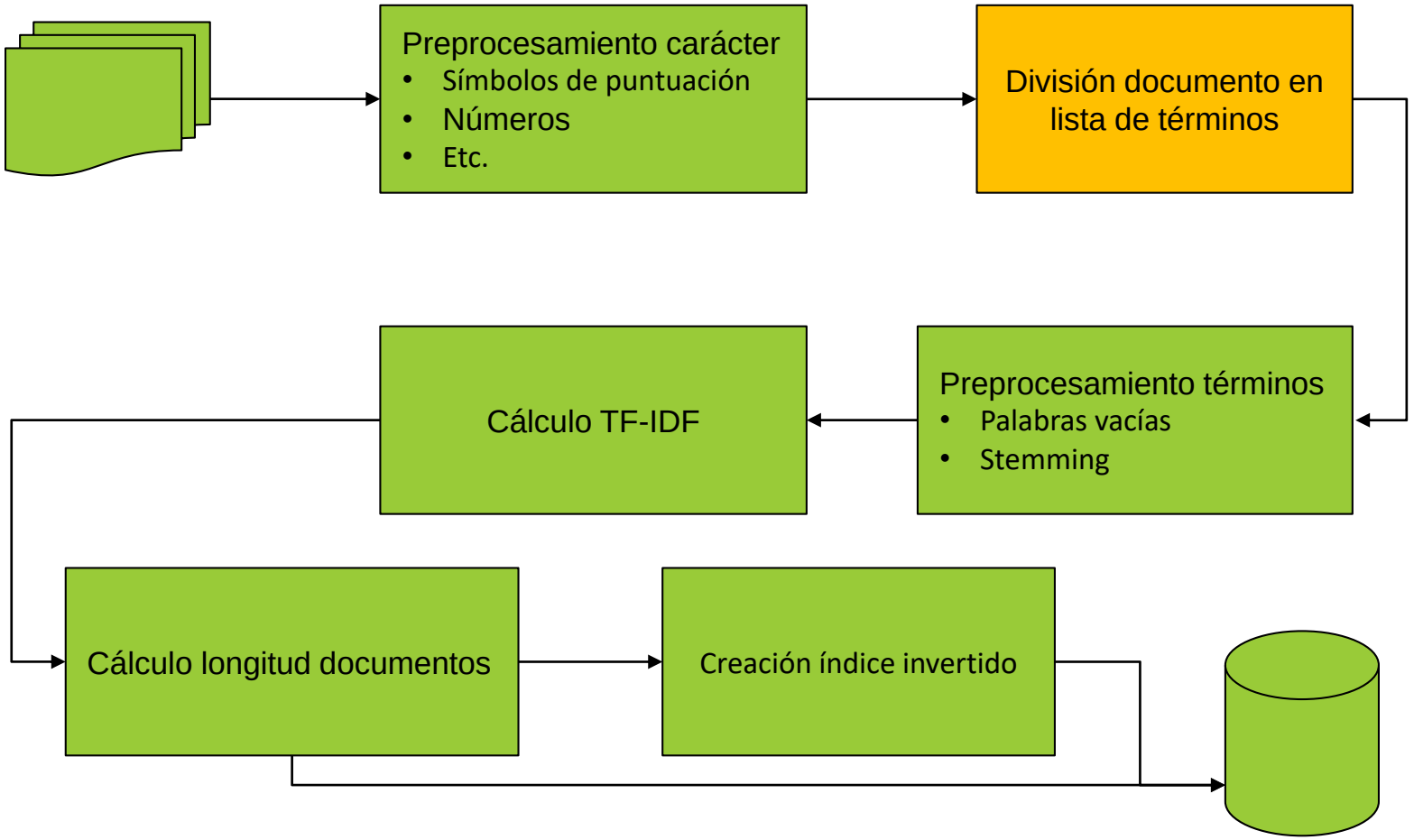
- Ejemplo de secuencia de preprocesamiento de caracteres:
  1. Convertir todo el texto a minúsculas o mayúsculas
     - Los lenguajes de programación suelen ser sensibles a mayúsculas y minúsculas
  2. Eliminar los signos de puntuación
     - ¿Todos? Depende de la colección
     - En Twitter, @ y # son importantes
     - El guion es un caso especial: si une dos palabras no lo deberíamos quitar
  3. Eliminar los número que no formen parte de una palabra
     - CO2 es un término válido
     - 2016 no es un término significativo
  4. Eliminar los “-” que no formen parte de una palabra
  5. Eliminar los espacios duplicados

Esquema del Sistema Indexación

Java. Ejemplo de secuencia de filtros

```
characterFilter.addFilter(new ToLowerCaseFilter());
characterFilter.addFilter(new SpecialCharacterFilter("[^-\\w]", " "));
characterFilter.addFilter(new SpecialCharacterFilter("\\b[0-9]+\\b", " "));
characterFilter.addFilter(new SpecialCharacterFilter("-+ | -+", " "));
characterFilter.addFilter(new SpecialCharacterFilter(" +", " "));
```

Esquema del Sistema Indexación



Esquema del Sistema Indexación

Pseudocódigo. Esquema básico indexación

```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  vTerms ← extraerPalabras(textoProcesado)
  …
Fin
```

Esquema del Sistema Indexación

- La división del texto procesado en una secuencia de términos dependerá del carácter que separe cada palabra
  - En el ejemplo anterior, se ha utilizado el carácter espacio para reemplazar las diferentes secuencias de caracteres

Pseudocódigo. División de texto en términos

```
vectorTérminos ← dividirCadena(sPreprocesada, " ")
```

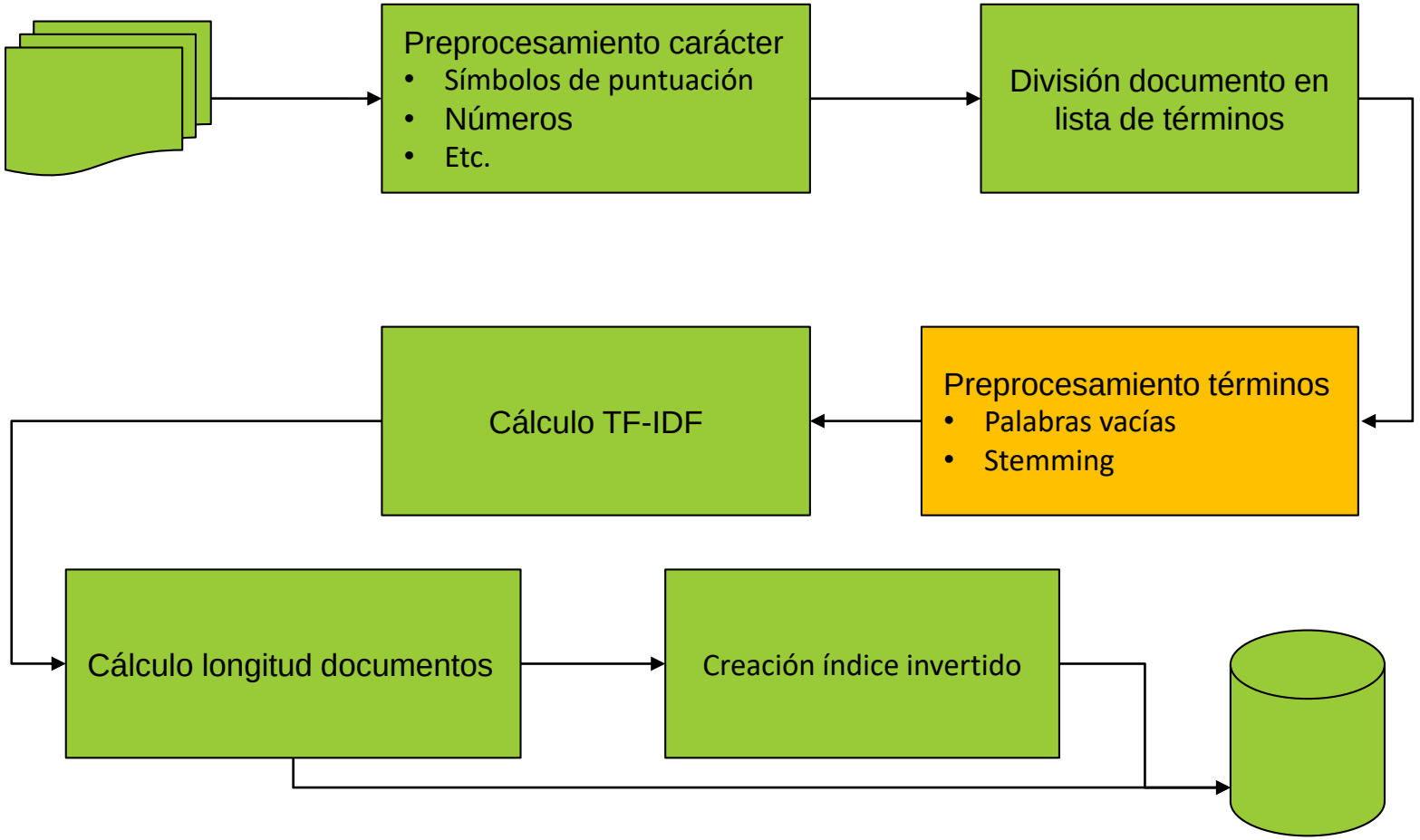
Java. División de texto en términos

```
String[] vTerms = sPreprocesada.split(" ");
```

Java. División de texto en términos y construcción de un Array

```
vTerm = new ArrayList<>(Arrays.asList(s.split(" ")));
```

Esquema del Sistema Indexación



Esquema del Sistema Indexación

- Se debe de realizar la misma secuencia de preprocesamiento a nivel de término a todos los documentos
- Se puede utilizar el patrón filtro
  - No se puede utilizar el mismo gestor de filtros para el preprocesamiento a nivel de carácter y a nivel de términos
    - Preprocesamiento caracteres: la entrada es una secuencia de caracteres
    - Preprocesamiento términos: la entrada es una secuencia (vector) de términos
- Consejo: crear una clase o función con la secuencia de filtros

Esquema del Sistema Indexación

- Los filtros pueden eliminar términos del vector y modificar sus caracteres
  - Es necesario una estructura que permita eliminar elementos
    - El vector/array clásico no lo permite
    - Utilizar un ArrayList o cualquier estructura similar
    - El recorrido será secuencial, por lo que la estructura no tiene por qué ordenar los elementos

Esquema del Sistema Indexación

Pseudocódigo. Esquema básico indexación

```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  vTerms ← extraerPalabras(textoProcesado)
  vTermsProcesados ← preprocesarTérminos(vTerms)
  …
Fin
```

Esquema del Sistema Indexación

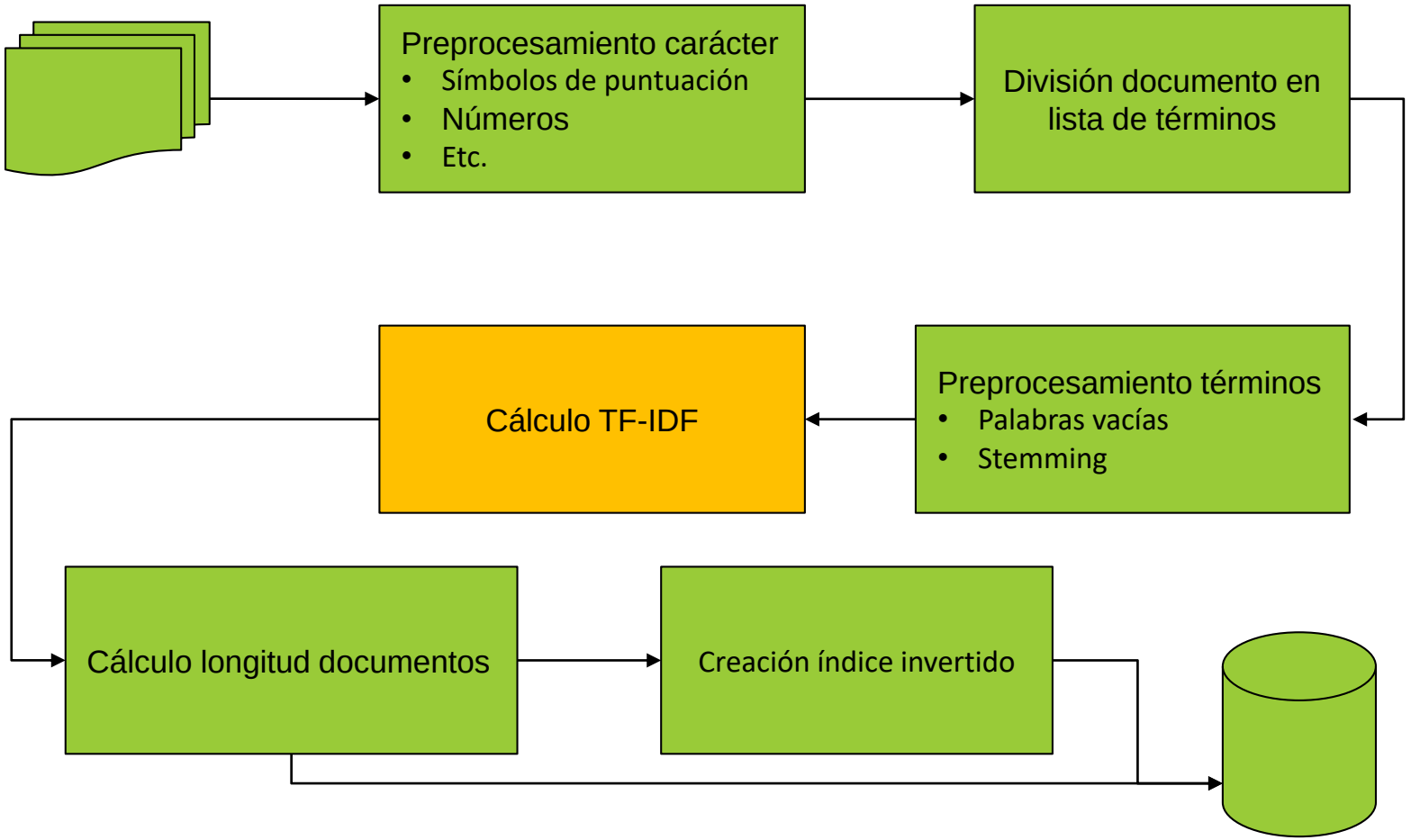
- Ejemplo de secuencia de preprocesamiento de términos
  - Palabras vacías
    - Existen listas de palabras vacías:
      - http://www.ranks.nl/stopwords
    - Dependiendo de la colección, se deberá de añadir o quitar ciertas palabras
  - Stemming
    - Algoritmo de Porter
    - http://snowball.tartarus.org
  - Umbral de longitud
    - Eliminar aquellos términos que no superen una longitud determinada
    - Es importante, tras el stemming a veces se obtienen términos de una sola letra o términos sin contenido

Esquema del Sistema Indexación

Java. Ejemplo de uso del algoritmo Porter Stemmer

```
public ArrayList<String> execute(ArrayList<String> terms) {

  int i;

  for (i = 0; i < terms.size(); i++) {

    if (! terms.get(i).contains("-")) {

      this.stemmer.setCurrent(terms.get(i));
      this.stemmer.stem();
      terms.set(i, stemmer.getCurrent());
    }
  }

  return terms;
}
```

Esquema del Sistema Indexación



Esquema del Sistema Indexación

Pseudocódigo. Esquema básico indexación

```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  vTerms ← extraerPalabras(textoProcesado)
  vTermsProcesados ← preprocesarTérminos(vTerms)
  calcularTF(vTermsProcesados)
  …
Fin
```

Esquema del Sistema Indexación

- El mejor esquema de ponderación de los términos en un documento es el TF-IDF

$$w_{i,j} = \begin{cases} (1 + \log f_{i,j}) \times \log \frac{N}{n_i} & Si\ f_{i,j} > 0 \\ 0 & en\ otro\ caso \end{cases}$$

- El TF se calcula a nivel de documento
  - Importancia del término dentro del documento
- El IDF se calcula a nivel de colección
  - Importancia del término en la colección
- ¿Se puede calcular el TF-IDF en una sola iteración?
  - NO!!!
- El TF-IDF se tiene que calcular en una secuencia de iteraciones

Esquema del Sistema Indexación

- Para el calculo del TF necesitamos saber la frecuencia del término dentro del documento
  - No lo podemos saber hasta que no hayamos leído todo el documento
- Para el cálculo del IDF necesitamos saber el tamaño del vocabulario de la colección (número de términos de indexación)
  - No lo podemos saber hasta que toda la colección no haya sido procesada
- Secuencia de iteraciones para el cálculo del TF (para cada documento)
  1. Contar la frecuencia de aparición de los términos en el documento actual
  2. Calcular el TF para los términos del documento actual

Esquema del Sistema Indexación

Pseudocódigo. Esquema básico indexación

```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  vTerms ← extraerPalabras(textoProcesado)
  vTermsProcesados ← preprocesarTérminos(vTerms)
  termsFrecuencia ← calcularTF_paso1(vTermsProcesados)
  …
Fin
```

Esquema del Sistema Indexación

- El algoritmo para el cálculo de la frecuencia de los términos en un documento puede realizarse de diversas formas
  - Utilizar una mapa (HashMap)
    - Clave: término
    - Valor: frecuencia
- Se debe utilizar instancia diferente del mapa para cada documento de la colección
  - Para no crear y destruir tantos objetos, se puede crear una sola instancia y vaciar la estructura para cada nuevo documento

Esquema del Sistema Indexación

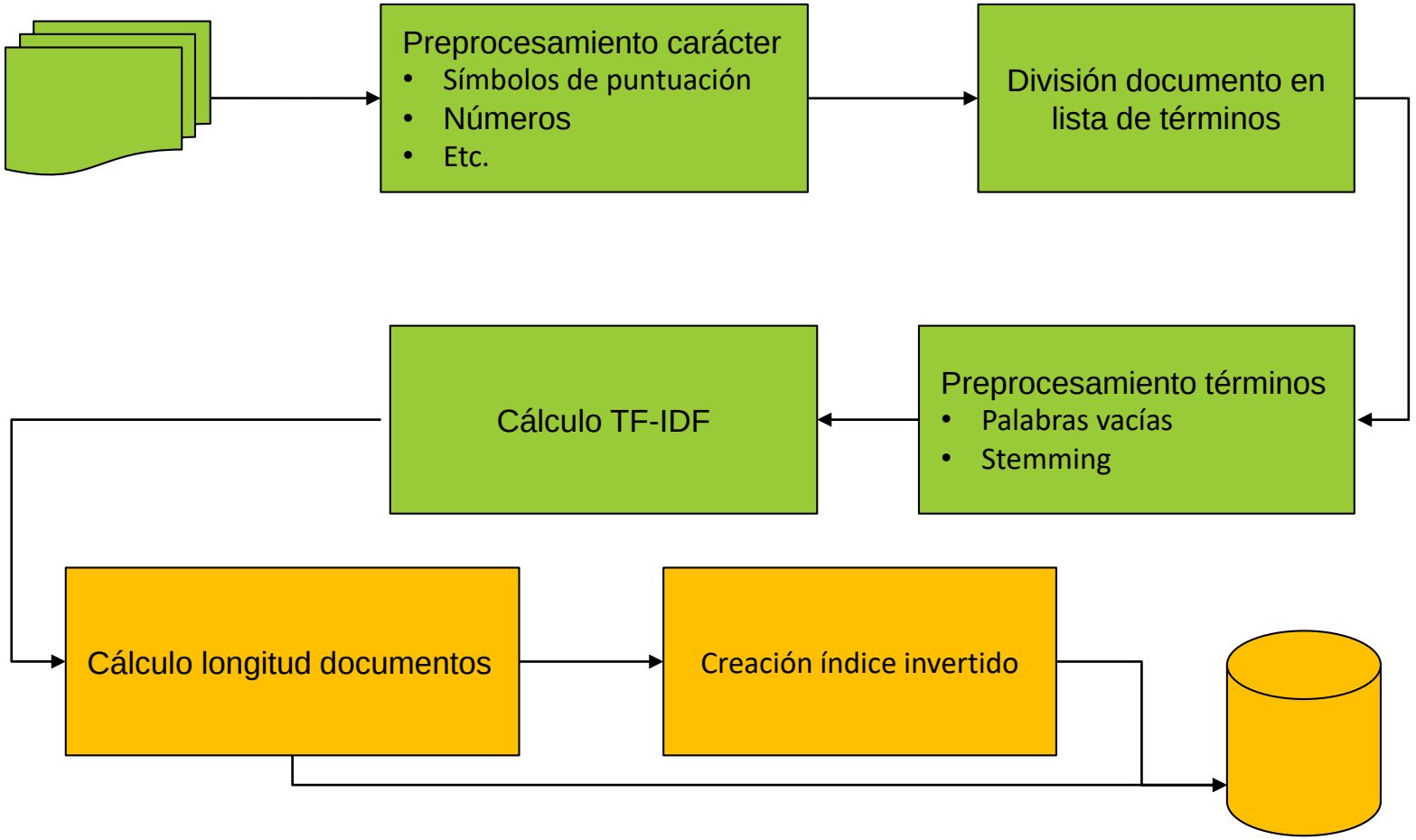
Pseudocódigo. Algoritmo básico para el conteo de elementos

```
Para cada término ∈ vTermsProcesados

  Si (mapaFrecuenciasTerminos.contiene(término))

    f ← mapaFrecuenciasTerminos.obtener(término)
    f ← f + 1
    mapaFrecuenciasTerminos.poner(término, f)

  Sino

    mapaFrecuenciasTerminos.poner(término, 1.0)

  Fin_Si
Fin
```

Esquema del Sistema Indexación

- Una vez procesados todos los documentos, se puede calcular el IDF de cada término
  - Problema: para calcular el IDF hace falta conocer el número de documentos en los que aparece una palabra
  - Solución: calcular el IDF, y TF-IDF, con la ayuda del índice invertido
- Consejo: el índice invertido puede empezar a construirse en el segundo paso del cálculo del TF
  - Se recorren todos los términos preprocesados del documento

Esquema del Sistema Indexación



Esquema del Sistema Indexación

Pseudocódigo. Esquema básico indexación

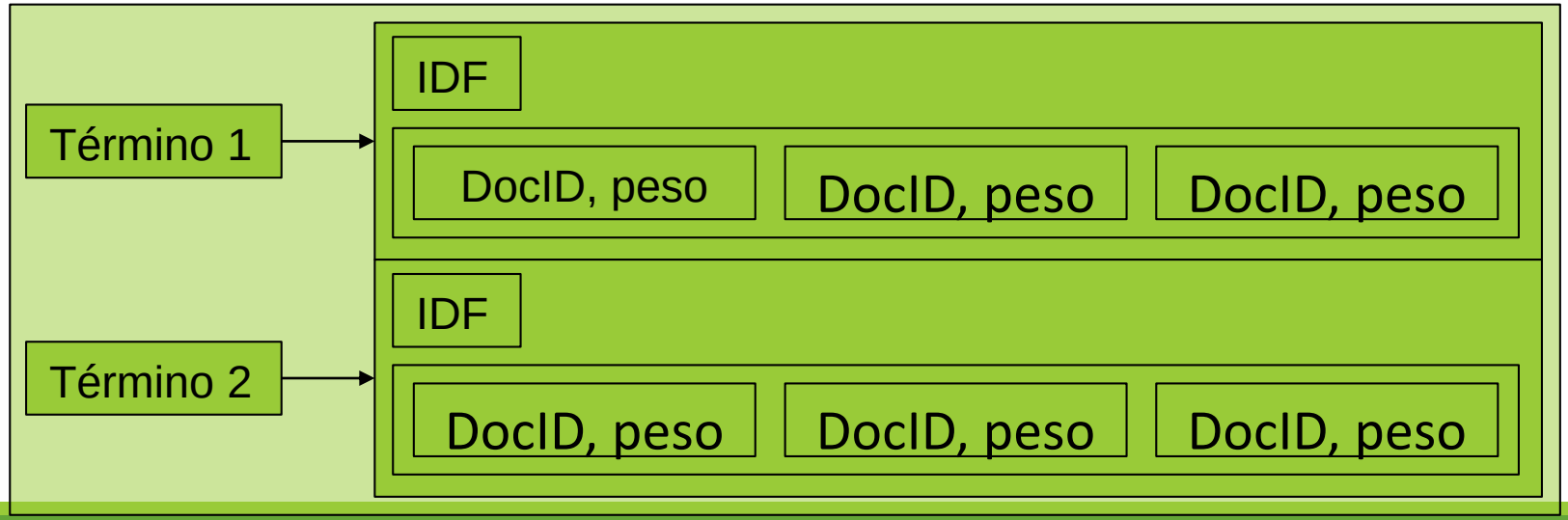
```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  vTerms ← extraerPalabras(textoProcesado)
  vTermsProcesados ← preprocesarTérminos(vTerms)
  termsFrecuencia ← calcularTF_paso1(vTermsProcesados)
  indiceInvertido ← calcularTF_paso2(termsFrecuencia)
  …
Fin
```

Esquema del Sistema Indexación

- Estructura básica de un índice invertido
  - Lista de términos
  - Cada término tiene asociado una lista de documentos

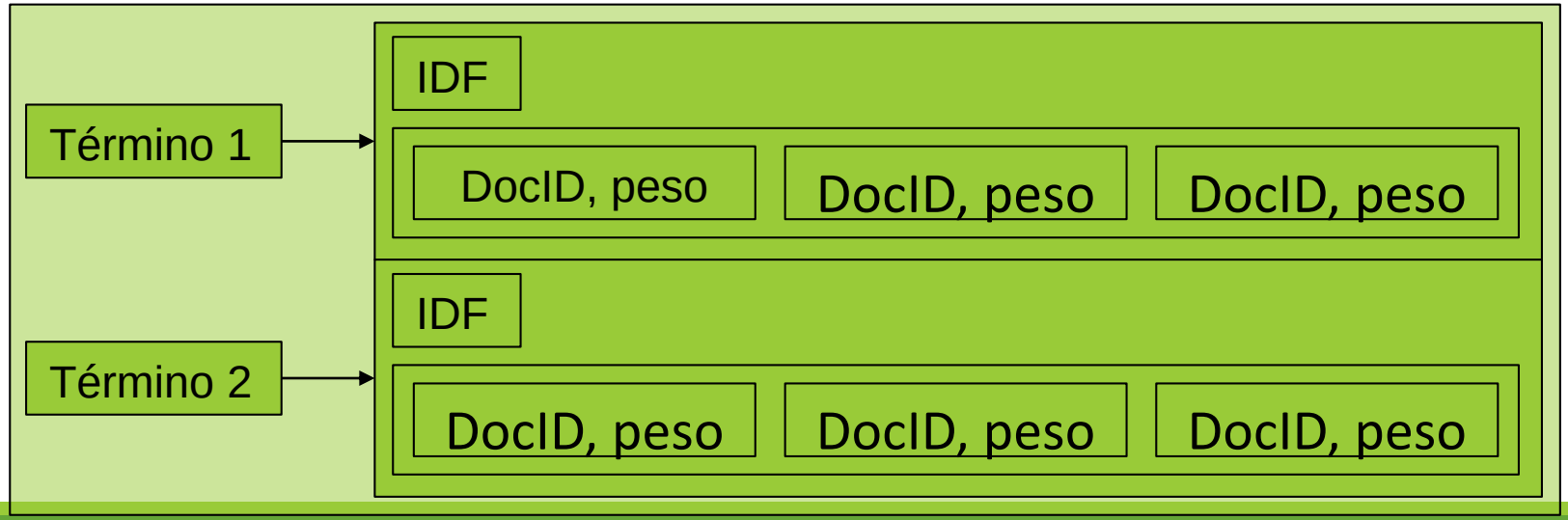
Esquema del Sistema Indexación

- La estructura se puede completar para mejorar la eficiencia en el módulo de recuperación
  - Para cada término, almacenar parejas de DocID-peso
  - Para cada término, almacenar el IDF
    - El IDF será necesario en el ranking ya que las consultas no tendrán TF
      - Las palabras no se repiten dentro de una consulta



Esquema del Sistema Indexación

- Estructura de datos para almacenar el índice invertido
  - Mapa (HashMap)
    - Clave: término
    - Valor: tupla
      - IDF
      - Mapa (HashMap) con las parejas docID-peso
        - Clave: docID
        - Valor: peso



Esquema del Sistema Indexación

Pseudocódigo. Calculo TF y creación índice invertido

```
Para cada término ∈ termsFrecuencia
  tf ← 1 + log(termsFrecuencia.obtener(término) / log(2)

  Si (indiceInvertido.contiene(término))

    parejaDocPeso ← indiceInvertido.obtener(término)

  Sino

    Crear estructura pareja docID-peso

  Fin_Si

  parejaDocPeso.poner(docID, tf)
  indiceInvertido.poner(término, parejaDocPeso)

Fin
```

Esquema del Sistema Indexación

Pseudocódigo. Esquema básico indexación

```
Para cada documento
  texto ← leer texto de documento
  textoProcesado ← preprocesarCaracteres(texto)
  vTerms ← extraerPalabras(textoProcesado)
  vTermsProcesados ← preprocesarTérminos(vTerms)
  termsFrecuencia ← calcularTF_paso1(vTermsProcesados)
  indiceInvertido ← calcularTF_paso2(termsFrecuencia)
Fin

calcularIDF(indiceInvertido)
longDocumentos ← calcularLongDocumentos(indiceInvertido)

escribirFichero(indiceInvertido)
escribirFichero(longDocumentos)
```

Esquema del Sistema Indexación

- Tras el segundo paso del cálculo del TF, el índice contiene:
  - IDF = 0
  - Listado con las parejas docID-TF
- Para el cálculo del IDF se debe recorrer el índice invertido y para cada término calcular su IDF
  - Numerador: número total de documentos
    - Tamaño del mapa principal
  - Denominador: número de parejas docID-peso para un término dado
    - Tamaño del mapa secundario asociado al término

Esquema del Sistema Indexación

- Para el cálculo de la longitud del peso del documento se debe recorrer el índice invertido y todas las parejas docID-peso
  - El peso a tener en cuenta tiene que ser el IDF-TF
    - Multiplicar el TF de la pareja por el IDF del término
  - Utilizar el algoritmo básico de conteo
    - Clave: ID del documento
    - Valor: peso del documento

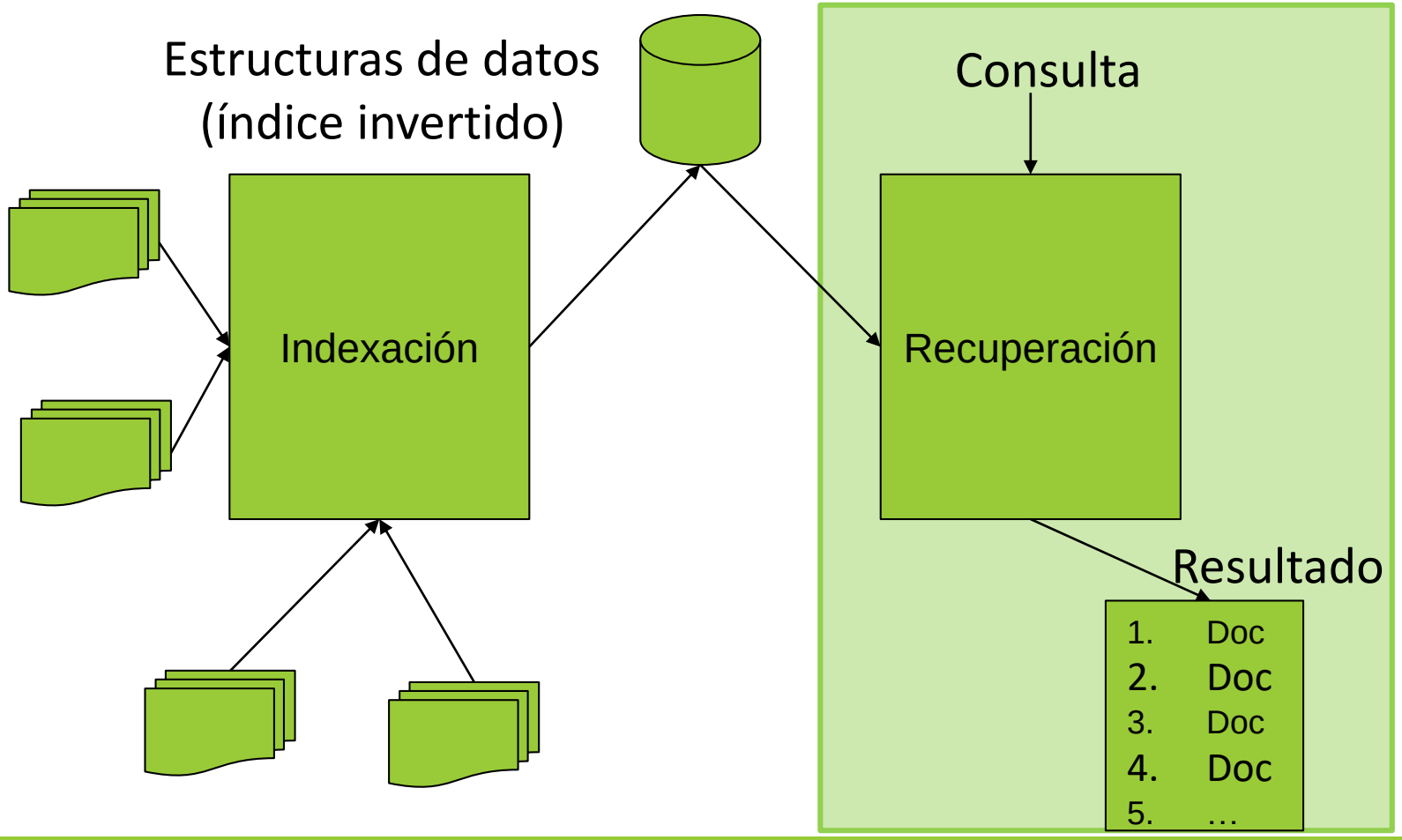
$$\left|\overrightarrow{d_j}\right| = \sqrt{\sum_{i}^{t} w_{i,j}^2}$$

- Consejo: el cálculo del IDF y de la longitud del documento puede realizarse en el mismo recorrido del índice invertido

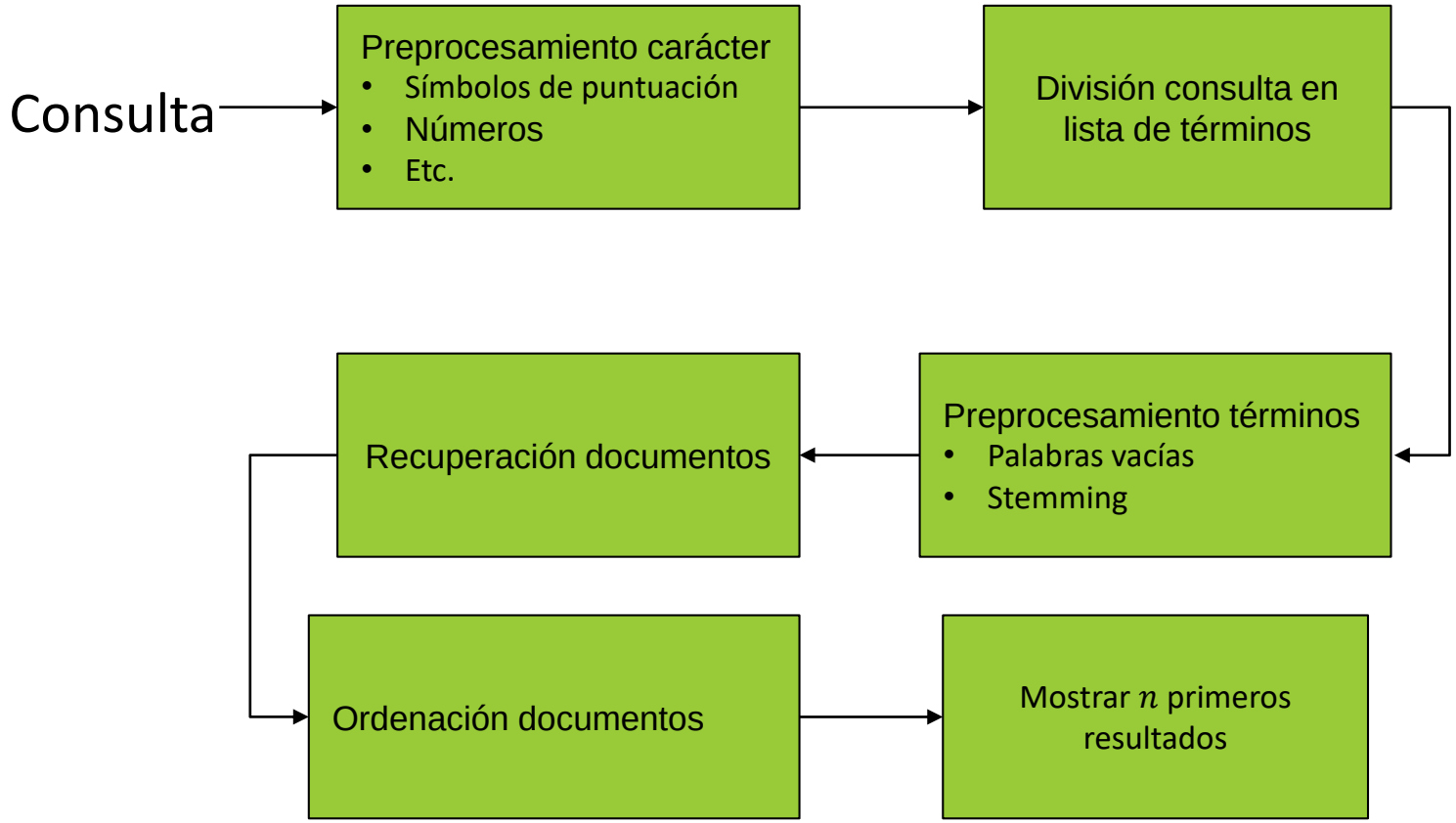
Esquema del Sistema

RECUPERACIÓN

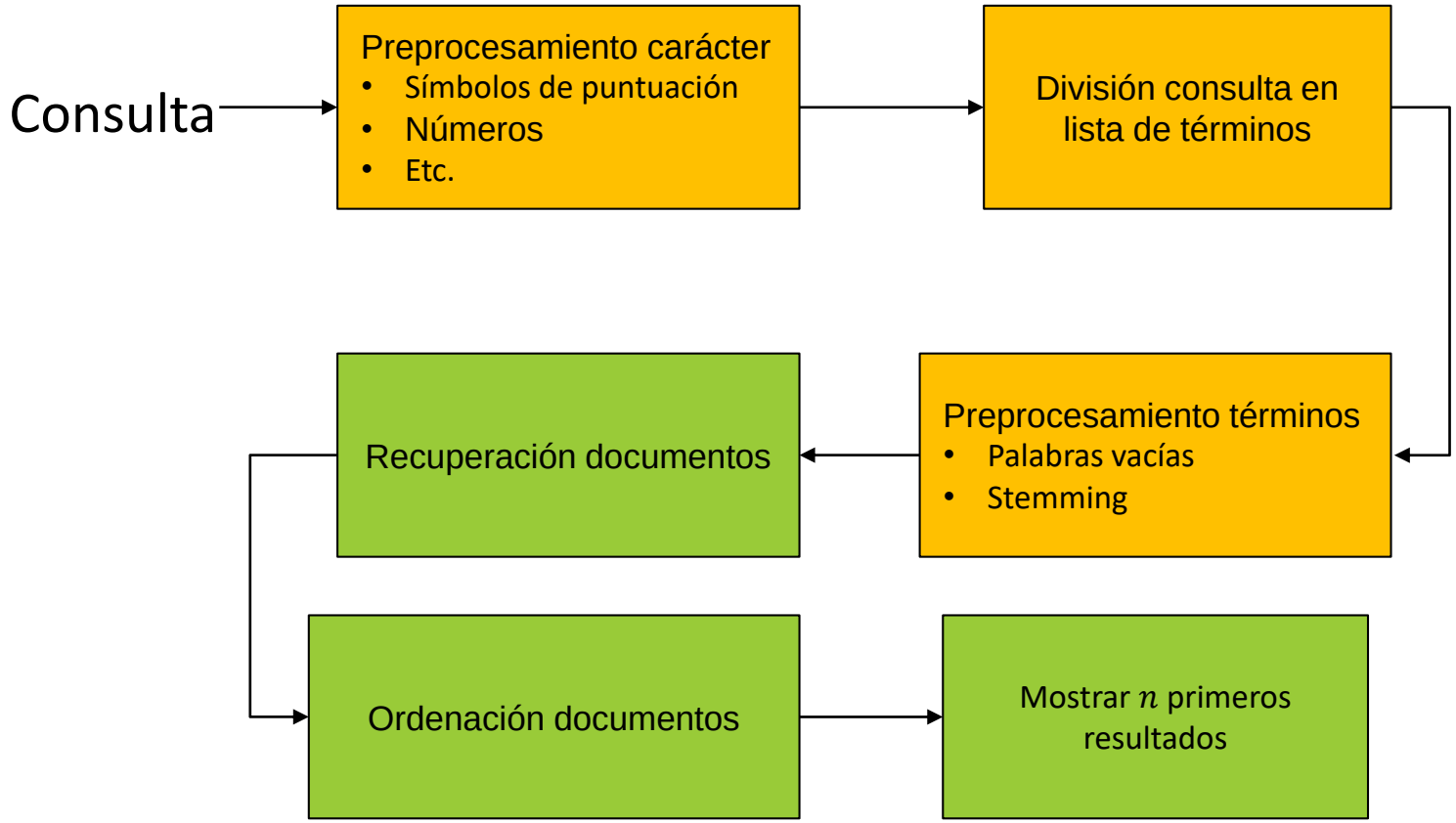
Esquema del Sistema Recuperación



Esquema del Sistema Recuperación



Esquema del Sistema Recuperación

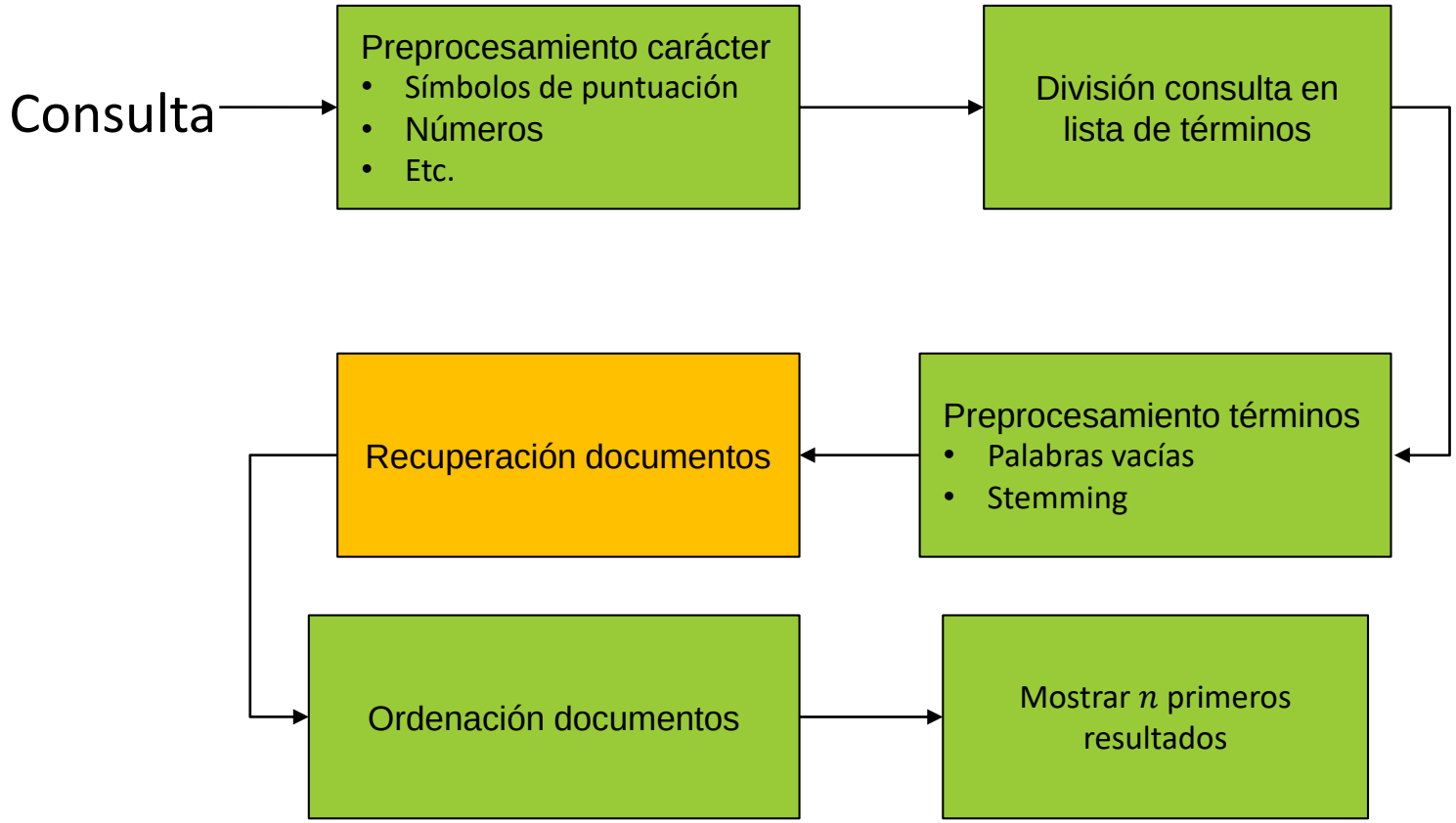


Esquema del Sistema Recuperación

Pseudocódigo. Esquema básico indexación

```
longitudDocumento ← leerFichero()
indiceInvertido ← leerFichero()

consulta ← leer()
textoProcesado ← preprocesarCaracteres(consulta)
vTerms ← extraerPalabras(textoProcesado)
vTermsProcesados ← preprocesarTérminos(vTerms)

…
```

Esquema del Sistema Recuperación



Esquema del Sistema Recuperación

Pseudocódigo. Esquema básico indexación

```
longitudDocumento ← leerFichero()
indiceInvertido ← leerFichero()

consulta ← leer()
textoProcesado ← preprocesarCaracteres(consulta)
vTerms ← extraerPalabras(textoProcesado)
vTermsProcesados ← preprocesarTérminos(vTerms)
docRecuperados ← recuperarDocumentos(vTermsProcesados)

…
```

Esquema del Sistema Recuperación

#	Término	IDF	d_1	d_2	d_3	d_4
1	to	1	3	2		
2	do	0,415	0,830		1,073	1,073
3	is	2	4			
4	be	0				
5	or	2		2		
6	not	2		2		
7	I	1		2	2	
8	am	1		2	1	
9	what	2		2		
10	think	2			2	
11	therefore	2			2	
12	da	2				5,170
13	let	2				4
14	it	2				4

Longitud	d_1	d_2	d_3	d_4
Vector normal	5,068	4,899	3,762	7,738

$$d_1 = \frac{1 \times 3 + 0{,}415 \times 0{,}830}{5{,}068} = 0{,}660$$

$$d_2 = \frac{1 \times 2 + 0{,}415 \times 0}{4{,}899} = 0{,}048$$

$$d_3 = \frac{1 \times 0 + 0{,}415 \times 1{,}073}{3{,}762} = 0{,}118$$

$$d_4 = \frac{1 \times 0 + 0{,}415 \times 1{,}073}{7{,}738} = 0{,}058$$

Esquema del Sistema Recuperación

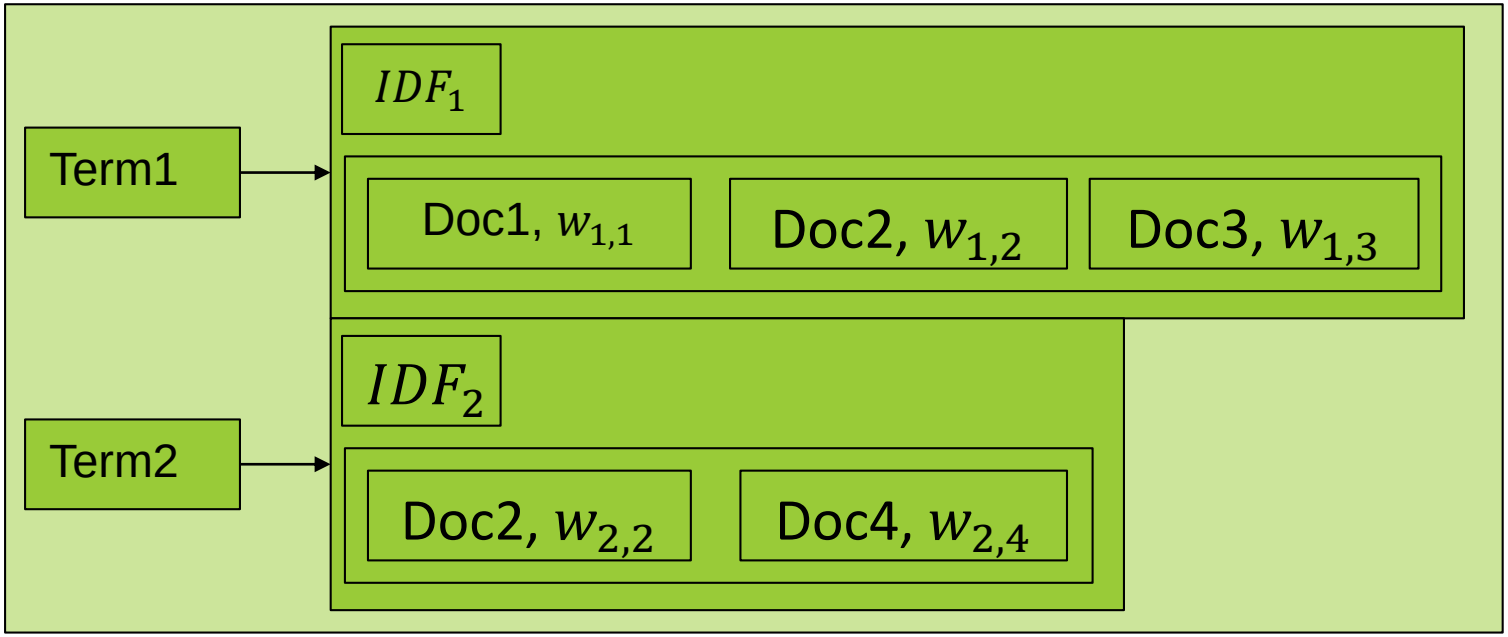
- Computo del peso de cada documento para los términos introducidos
  - Para cada documento, obtener el peso del término en el documento
  - Calcular el peso total
- Problemas:
  - La estructura de índice invertido no permite navegar fácilmente por los documentos
    - Permite navegar por los términos
  - Los documentos que contengan los términos serán solo un pequeño subconjunto de la colección total
- La búsqueda por documentos no es un buen enfoque

Esquema del Sistema Recuperación

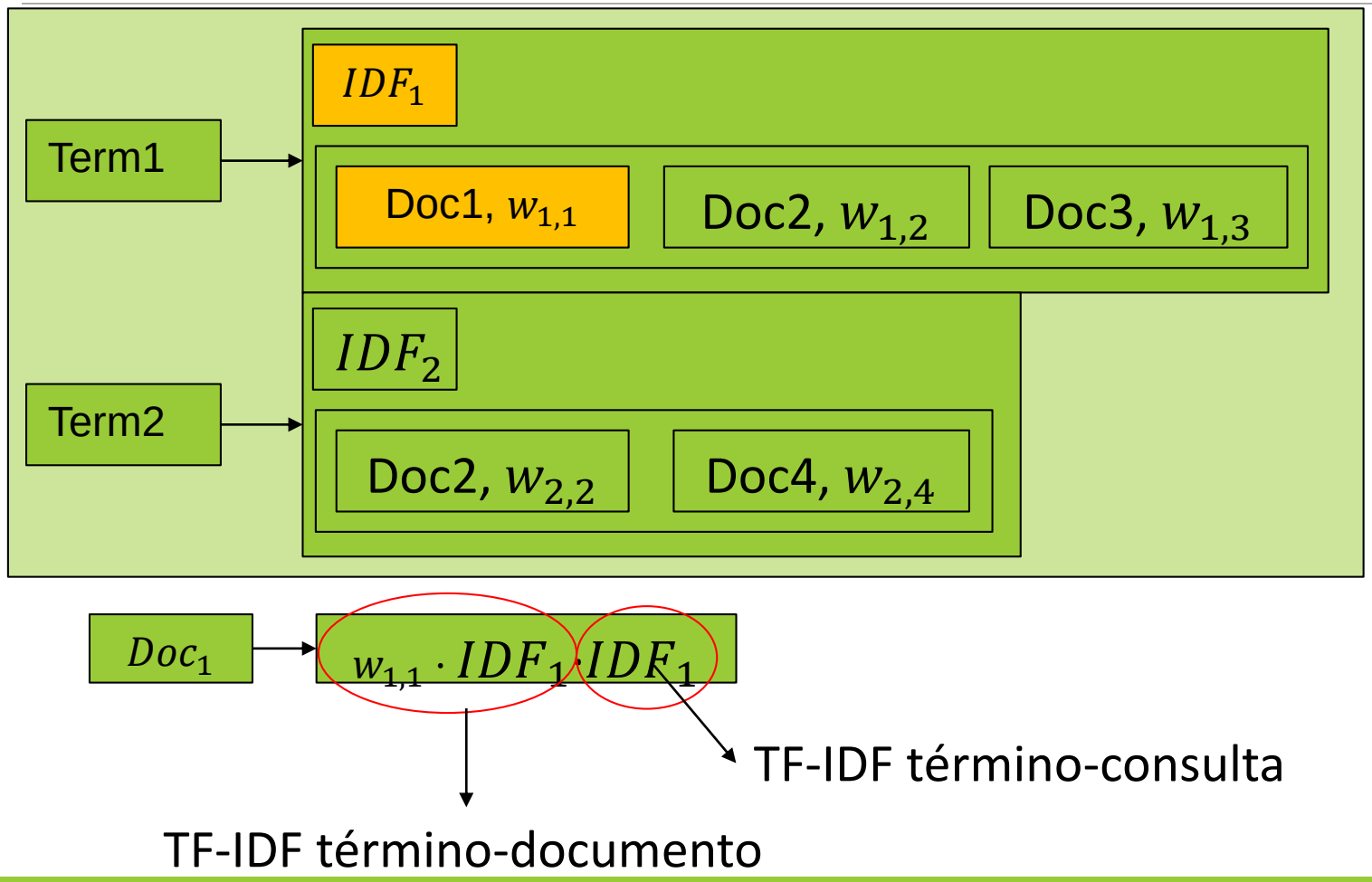
- Para mejorar la eficiencia, intercambiaremos el enfoque
  - Buscaremos por términos (de la consulta)
- La fórmula para el calculo del ranking de un documento no variará
  - Calcular el numerador de cada documento de forma iterativa
    - Algoritmo básico de conteo
  - Cuando el numerador esté completo, dividir por la longitud del documento

Esquema del Sistema Recuperación

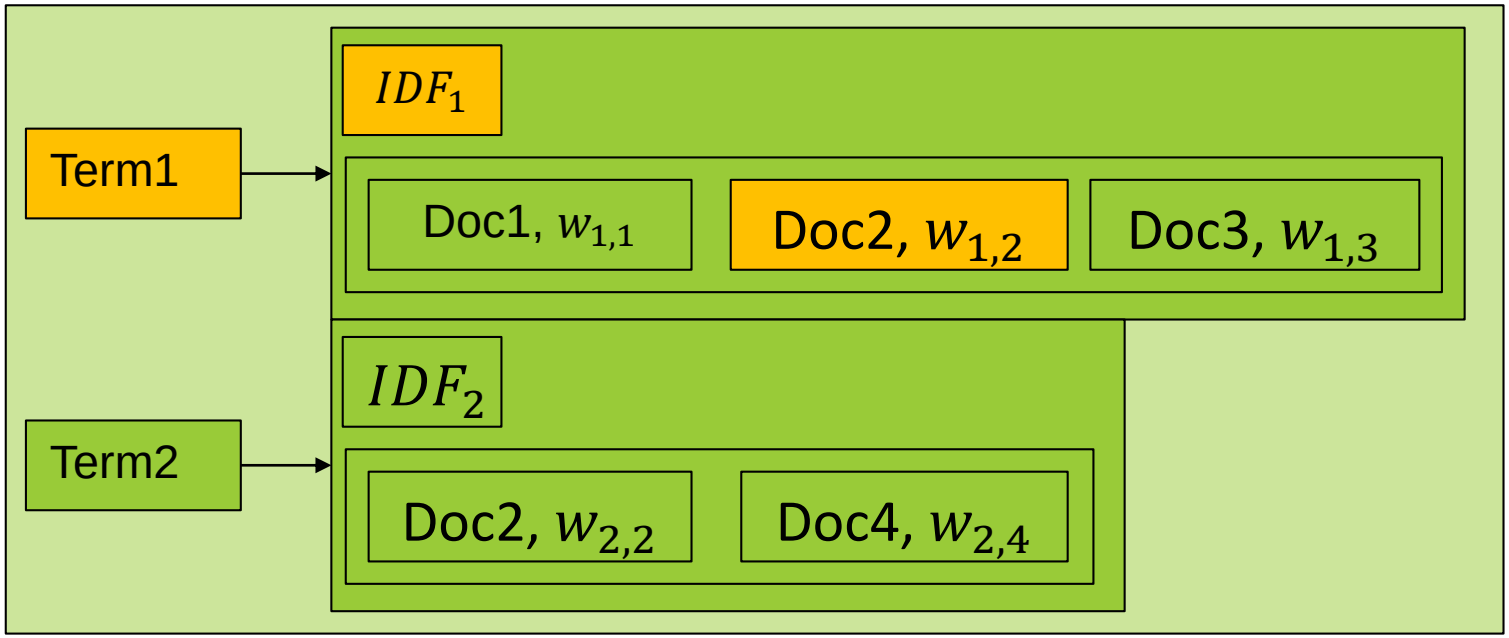
- Consulta: Term1 Term2



Esquema del Sistema Recuperación



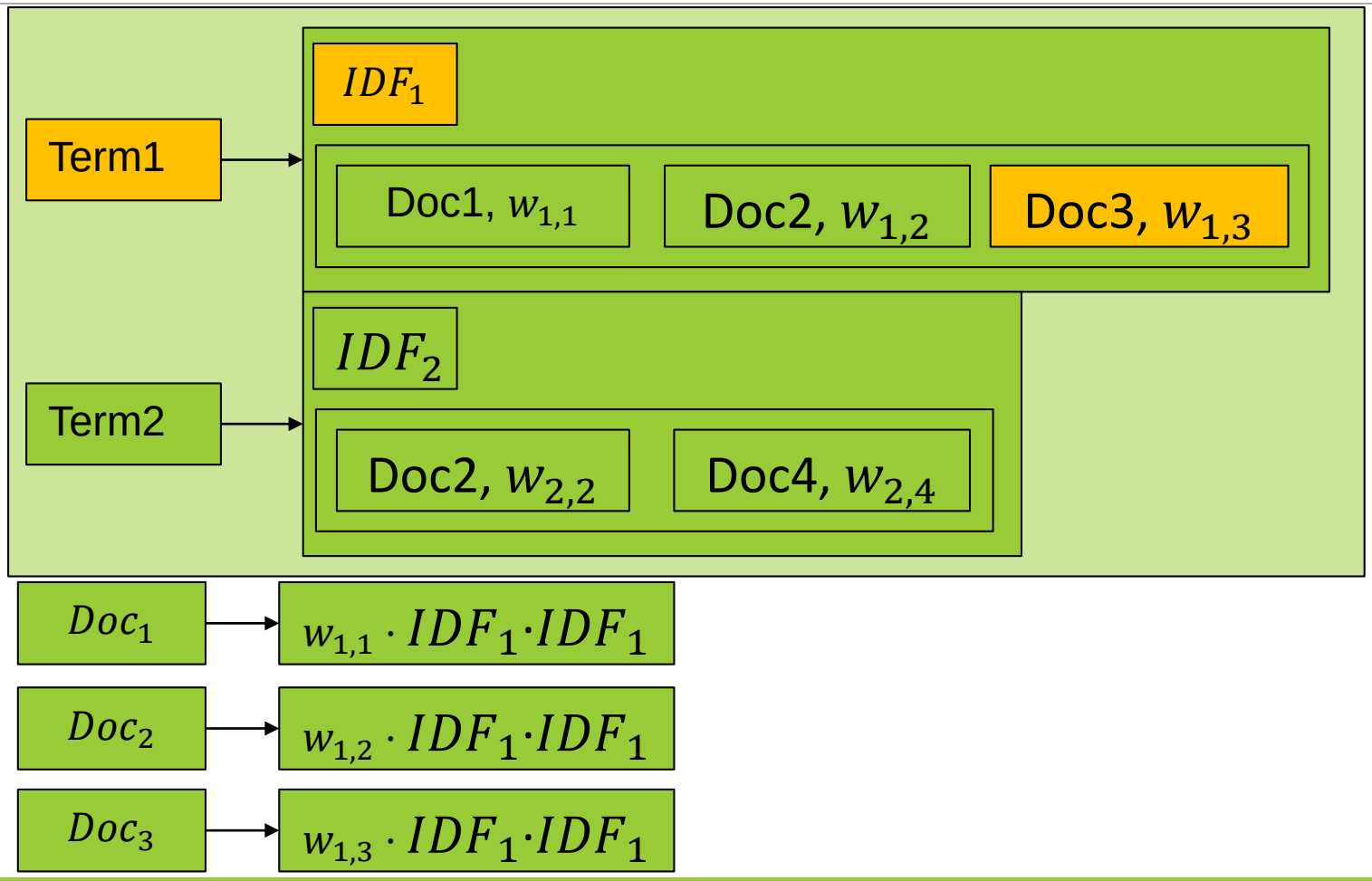
Esquema del Sistema Recuperación



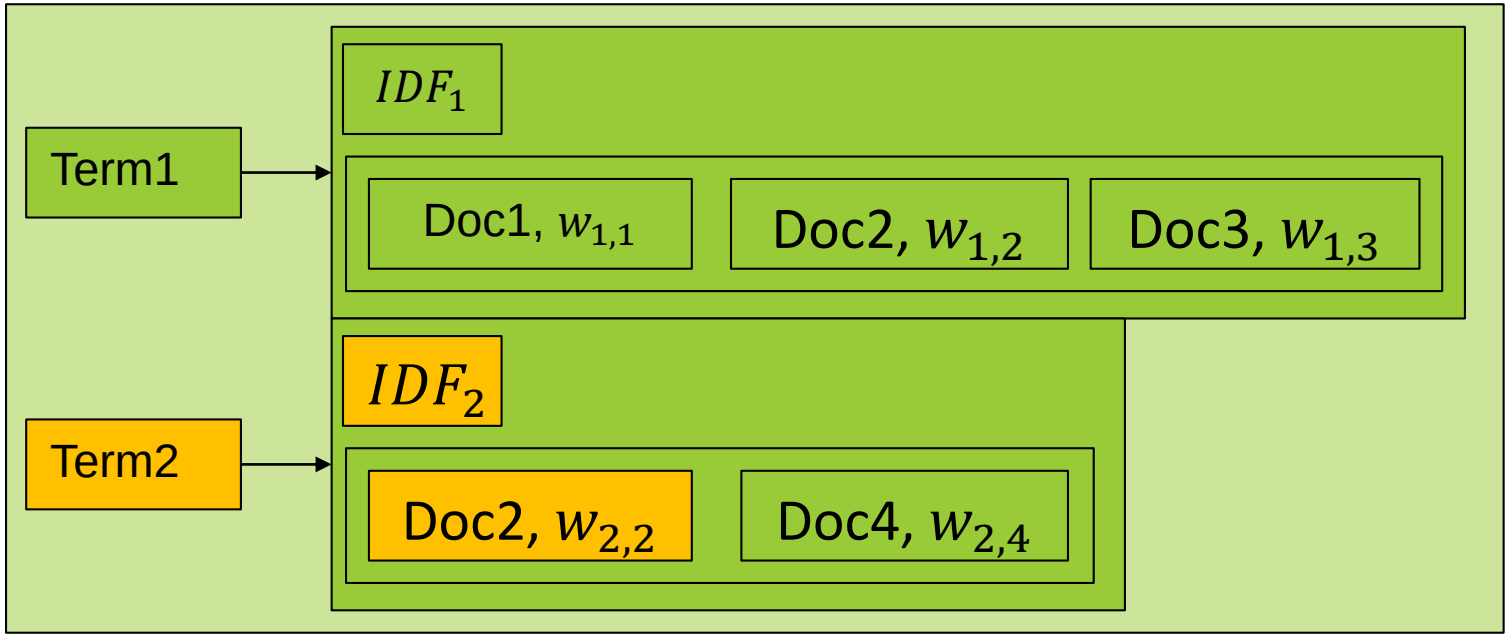
$Doc_1 \rightarrow w_{1,1} \cdot IDF_1 \cdot IDF_1$

$Doc_2 \rightarrow w_{1,2} \cdot IDF_1 \cdot IDF_1$

Esquema del Sistema Recuperación



Esquema del Sistema Recuperación

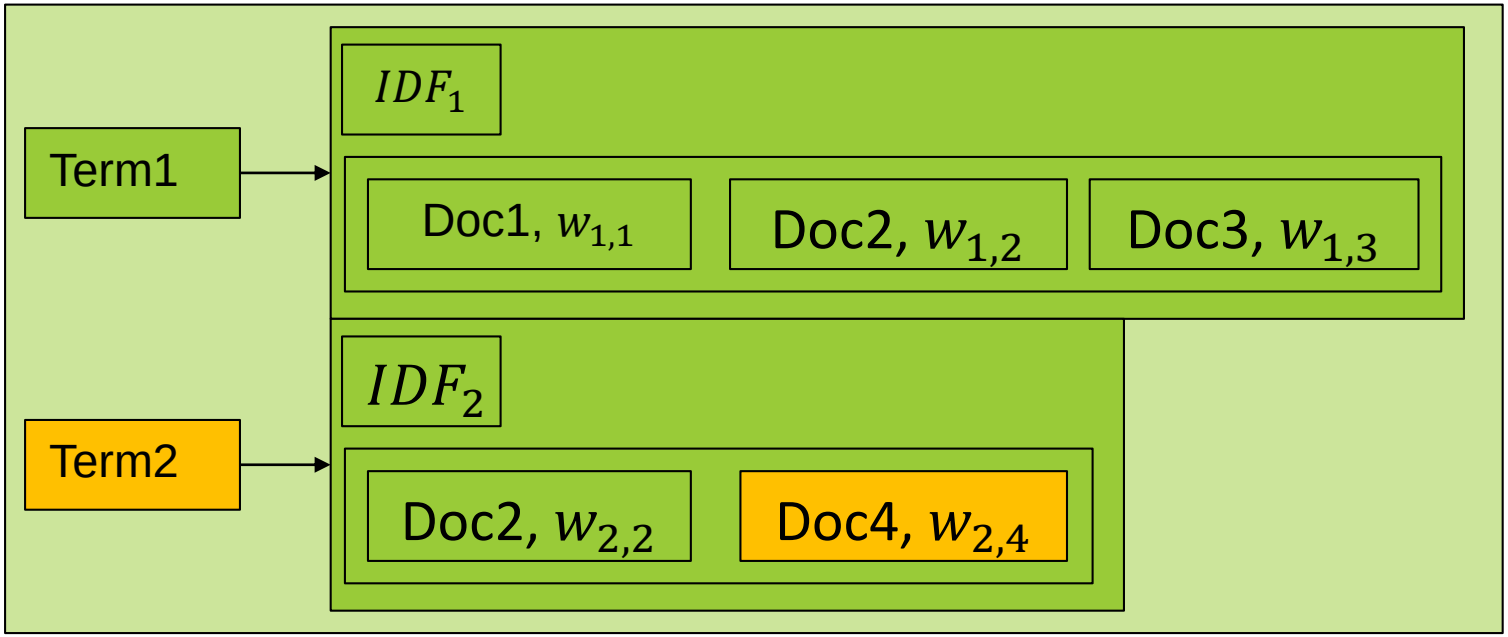


$Doc_1 \rightarrow w_{1,1} \cdot IDF_1 \cdot IDF_1$

$Doc_2 \rightarrow w_{1,2} \cdot IDF_1 \cdot IDF_1 + w_{2,2} \cdot IDF_2 \cdot IDF_2$

$Doc_3 \rightarrow w_{1,3} \cdot IDF_1 \cdot IDF_1$

Esquema del Sistema Recuperación



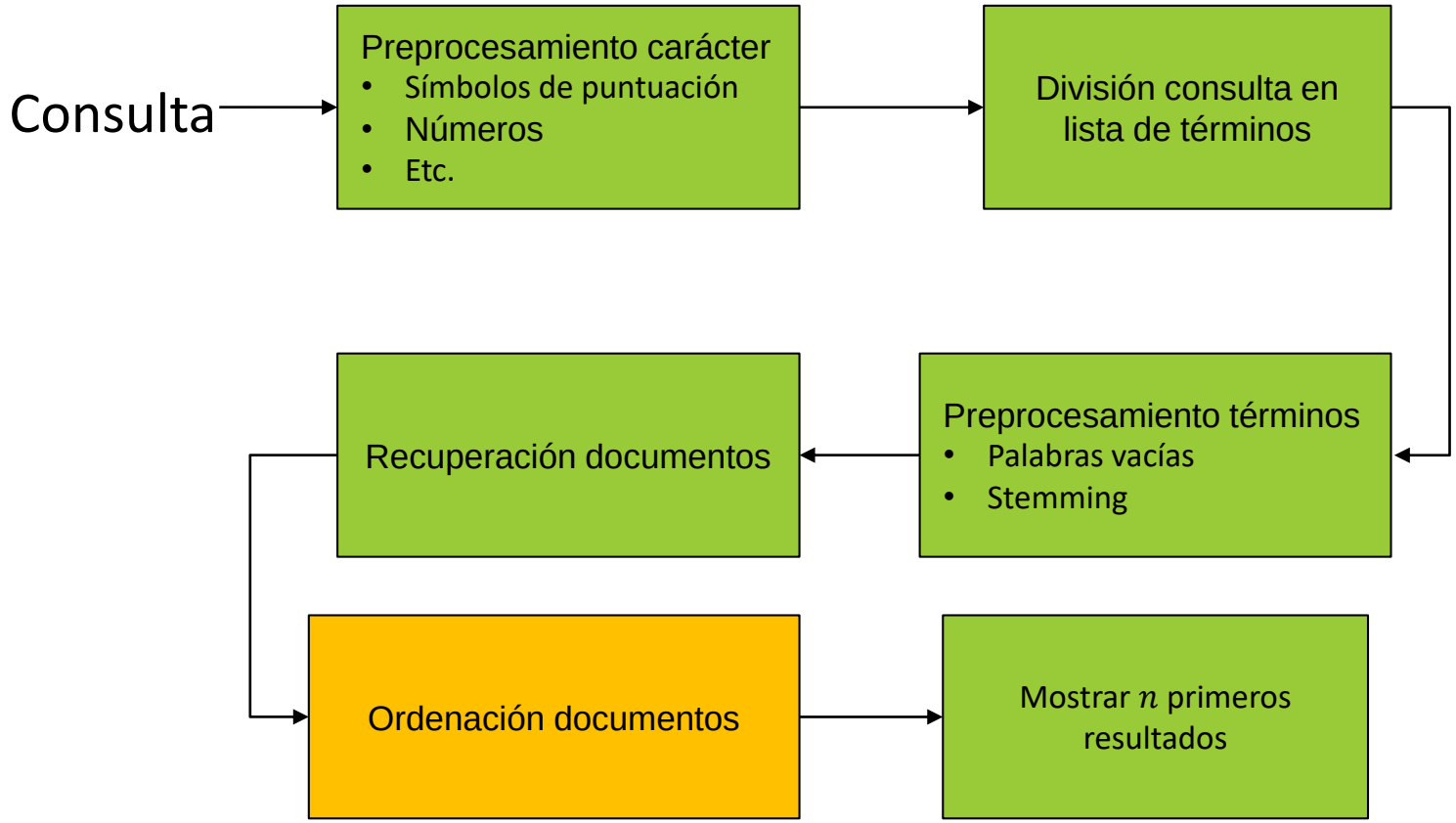
Doc_1 → $w_{1,1} \cdot IDF_1 \cdot IDF_1$

Doc_2 → $w_{1,2} \cdot IDF_1 \cdot IDF_1 + w_{2,2} \cdot IDF_2 \cdot IDF_2$

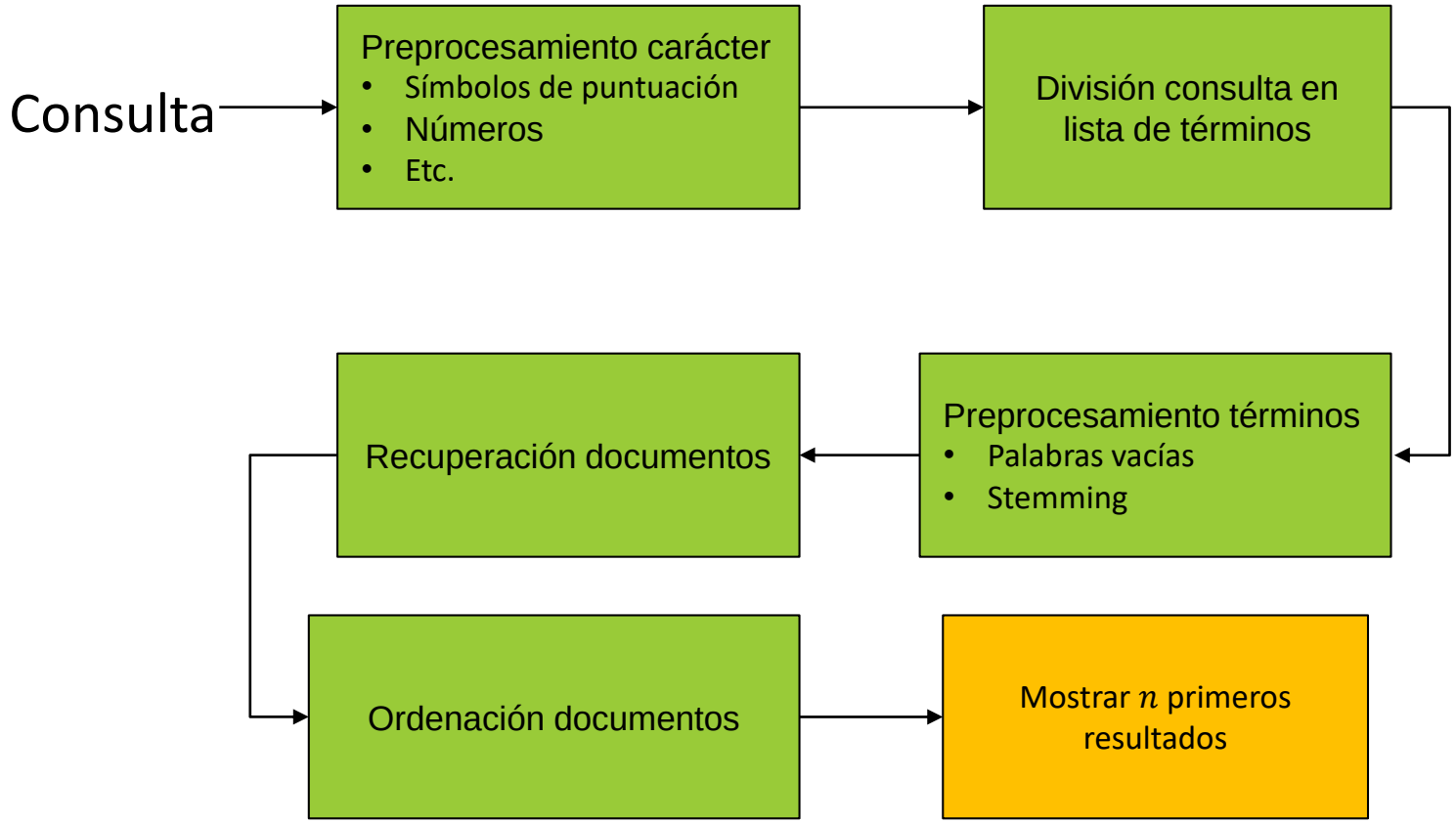
Doc_3 → $w_{1,3} \cdot IDF_1 \cdot IDF_1$

Doc_4 → $w_{2,4} \cdot IDF_2 \cdot IDF_2$

Esquema del Sistema Recuperación



Esquema del Sistema Recuperación



Esquema del Sistema Recuperación

```
Introduce the query terms (enter to finish): cancer
  Applying the characters filter.
  Applying the terms filter.
Document ID:000087503800030     (weight: 4.1250968154778285)
Summary: Cancer - Preventing cervical cancer
Document ID:000265193600031     (weight: 2.8217745309972484)
 All cancers arise as a result of changes that have occurred in the DNA sequence of the genomes of
cancer cells. Over the past quart[...]
Document ID:000168862900052     (weight: 2.76638738521863)
Summary: Diagnosing cancer in vivo
Document ID:000181965400029     (weight: 2.7502178709296565)
Summary: Developmental predisposition to cancer
Document ID:000286636300028     (weight: 2.701803385652045)
Summary: Mitochondrial Capture by a Transmissible Cancer
Document ID:000311031600033     (weight: 2.547076443387621)
 Cancer metabolism has received a substantial amount of interest over th[...]
Document ID:000168710000055     (weight: 2.4744158997219863)
 Cancer genetics has for many years focused on mutational events that have their primary effect
within the cancer cell. Recently that [...]
Document ID:000225161400038     (weight: 2.47305691873641)
 Disruption of the normal regulation of cell-cycle progression and division lies at the heart of
the events leading to cancer.[...]
Document ID:000288754500043     (weight: 2.322599919133862)
 The description and interpretation of genomic abnormalities in cancer cells have been at [...]
Document ID:000324545700031     (weight: 2.319051273187568)
 Phenotypic and functional heterogeneity arise among cancer cells within the same tumour as a
conseque[...]
```