

# **Sistemas de Recomendación: Modelos Basados en el Contenido**

Gestión del conocimiento en las  
organizaciones

Maya Santana Amador  
José Miguel Díaz González  
Déniz Isael Ipekel Castro



## Índice

Índice.....	1
Introducción.....	2
Modelos basados en Contenidos.....	3
Cálculos de TF-IDF.....	4
Similaridad del coseno.....	6
Análisis.....	7
Visualización y análisis de resultados.....	8
Conclusión.....	9



## Introducción

En este informe se presenta el desarrollo y funcionamiento de un sistema de recomendación basado en el contenido. Con el objetivo principal de analizar el comportamiento de un modelo capaz de identificar la similitud entre documentos en texto plano, utilizando técnicas de representación vectorial como TF-IDF y medidas de similitud como el coseno.

El programa (implementado en Java), recibe un conjunto de archivos de texto, un listado de parada (Stop words) y un fichero lematización, y genera como salida una serie de tablas y matrices que reflejan la importancia de los términos y el grado de relación entre los documentos.

Con ello, buscamos comprender cómo un sistema de recomendación puede identificar automáticamente la afinidad semántica entre diferentes piezas de información, sirviendo de base para aplicaciones como la recomendación de artículos, noticias o productos.



## 1. Modelos basados en contenidos

### Descripción general

El sistema de recomendación basado en el contenido utiliza la información interna de los ítems para generar recomendaciones. Cada documento se representa como un vector de características, donde cada término tiene un peso asociado que refleja su relevancia en el contexto del documento y del corpus completo.

El programa (implementado en Java), recibe un conjunto de archivos de texto, un listado de parada (Stop words) y un fichero lematización, y genera como salida una serie de tablas y matrices que reflejan la importancia de los términos y el grado de relación entre los documentos.

El proceso desarrollado consta de: una lectura y procesamiento de los documentos (.txt), eliminar las palabras de parada, para evitar términos poco significativos; lematización de los textos para unificar las variaciones de una misma, cálculo de las métricas TF-IDF para cada término relevante y el cálculo de la similitud coseno entre los vectores de los documentos, y, la generación de salidas en formato CSV con los documentos.



## Cálculo de TF-IDF

El **TF-IDF** es una medida estadística que evalúa la importancia de una palabra en un documento dentro de un conjunto de documentos(corpus). Este modelo combina dos factores:

**Frecuencia de término (TF):** mide cuántas veces aparece un término en un documento. Se calcula con la siguiente fórmula.

**Frecuencia inversa de documento (IDF):** reduce el peso de las palabras más frecuentes en todo el corpus. Se calcula con la siguiente fórmula.

$$IDF(t) = \log \left( \frac{N}{df_t} \right)$$

El peso final del término (**TF-IDF**) en el documento se obtiene multiplicando el IDF y el Tf (aunque esta vez el que aparece en la segunda fórmula):

$$TF-IDF(t, d) = TF(t, d) \times IDF(t)$$

$$w_{t,d} = \begin{cases} 1 + \log_{10} tf_{t,d}, & \text{if } tf_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Cada documento se convierte en un vector numérico, donde cada posición corresponde al TF-IDF de un término. Este vector permitirá comparar documentos entre sí o buscar los más parecidos a un documento dado.

La implementación a código es la siguiente:



```
public void compute() {  
    // Construir vocabulario  
    Set<String> allTerms = new HashSet<>();  
    corpus.values().forEach(allTerms::addAll);  
    vocab.addAll(allTerms);  
    vocab.sort(String::compareTo);  
  
    int N = 0;  
    for (List<String> f : corpus.values()) {  
        N += f.size();  
    }  
  
    // Calcular IDF = log(N / df)  
    for (String term : vocab) {  
        int df = 0;  
        for (List<String> docTokens : corpus.values()) {  
            for (String s : docTokens) {  
                if (s.equals(term)) { df++; }  
            }  
        }  
        idf.put(term, Math.log((double) N / df));  
    }  
  
    Map<String, Double> sumVec = new HashMap<>();  
  
    // Calcular TF y TF-IDF  
    for (String doc : docNames) {  
        List<String> tokens = corpus.get(doc);  
        Map<String, Long> freq = tokens.stream().collect(Collectors.groupingBy(t -> t, Collectors.counting()));  
  
        Map<String, Double> tf = new LinkedHashMap<>();  
        Map<String, Double> tfidf = new LinkedHashMap<>();  
  
        List<Double> tfValues = new ArrayList<>();  
  
        for (String term : vocab) {  
            double tfVal = Math.log10((double) freq.getOrDefault(term, 1L)) + 1;  
            if (tfVal == 1) { tfVal = 0; }  
            tfValues.add(tfVal);  
            tf.put(term, tfVal);  
        }  
  
        double sumCuad = 0;  
  
        for (double v : tfValues) {  
            sumCuad += v * v;  
        }  
  
        sumCuad = Math.sqrt(sumCuad);  
        sumVec.put(doc, sumCuad);  
  
        for (String term : vocab) {  
            double tfidfVal = tf.get(term) / sumVec.get(doc);  
            tfidf.put(term, tfidfVal);  
        }  
  
        tfVectors.put(doc, tf);  
        tfidfVectors.put(doc, tfidf);  
    }  
}
```

```
for (String term : vocab) {  
    double tfVal = Math.log10((double) freq.getOrDefault(term, 1L)) + 1;  
    if (tfVal == 1) { tfVal = 0; }  
    tfValues.add(tfVal);  
    tf.put(term, tfVal);  
}  
  
double sumCuad = 0;  
  
for (double v : tfValues) {  
    sumCuad += v * v;  
}  
  
sumCuad = Math.sqrt(sumCuad);  
sumVec.put(doc, sumCuad);  
  
for (String term : vocab) {  
    double tfidfVal = tf.get(term) / sumVec.get(doc);  
    tfidf.put(term, tfidfVal);  
}  
  
tfVectors.put(doc, tf);  
tfidfVectors.put(doc, tfidf);
```



## Similaridad de coseno

Para determinar qué tan similares son dos documentos, se aplica la similaridad coseno, que mide el ángulo entre sus características. Su valor oscila (sin similitud) y 1 (idénticos). Esta métrica se utiliza ampliamente en sistemas de recomendación textuales porque no depende de la longitud absoluta de los documentos, sino de su dirección semántica.

Su fórmula es la siguiente:

$$\text{sim}(\vec{a}, \vec{b}) = \frac{\sum_{u \in U} (r_{u,a} - \bar{r}_u)(r_{u,b} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,a} - \bar{r}_u)^2} \sqrt{\sum_{u \in U} (r_{u,b} - \bar{r}_u)^2}}$$

Y la implementación es la siguiente:

```
public static double cosine(Map<String, Double> a, Map<String, Double> b) {
    double dot = 0;
    for (String term : a.keySet()) {
        double va = a.get(term);
        double vb = b.get(term);
        dot += va * vb;
    }
    return dot;
}

public static double[][] computeCosineMatrix(Map<String, Map<String, Double>> vectors) {
    List<String> docs = new ArrayList<>(vectors.keySet());
    int n = docs.size();
    double[][] M = new double[n][n];

    for (int i = 0; i < n; i++) {
        M[i][i] = 1.0;
        for (int j = i + 1; j < n; j++) {
            double sim = cosine(vectors.get(docs.get(i)), vectors.get(docs.get(j)));
            M[i][j] = M[j][i] = sim;
        }
    }
    return M;
}
```



## Análisis

### Análisis con documentos originales

Para el análisis se utilizaron distintos conjuntos de documentos, incluyendo los que se encuentran disponibles en el repositorio dado, así como un conjunto adicional de diez textos elaborados para la prueba. Cada documento fue sometido al mismo proceso de pre procesamiento lingüístico: eliminación de stopwords, lematización de términos y cálculo de frecuencias.

Los resultados obtenidos mostraron que los términos con mayor peso TF-IDF corresponden a las palabras más específicas y relevantes de cada documento, mientras que los términos más comunes o repetidos en varios textos obtienen valores más bajos. Esto demuestra que el modelo logra identificar correctamente los conceptos clave que distinguen unos documentos de otros.

La matriz de similitud coseno generada refleja el grado de relación semántica entre los documentos. Se observó que los pares de documentos con temáticas afines alcanzaron valores de similitud superiores a 0.7, mientras que los documentos de temáticas diferentes se situaron en torno a 0.6.

Viendo estos resultados, el sistema consigue una representación discriminativa del contenido, permitiendo establecer relaciones significativas entre documentos sin depender de información externa.

Los documentos 5 y 6 muestran el valor más alto fuera de la diagonal (0.74), lo que implica que son los más parecidos léxica y temáticamente.

**Comparador directo de documentos**

Documento A

document-05.txt

Documento B

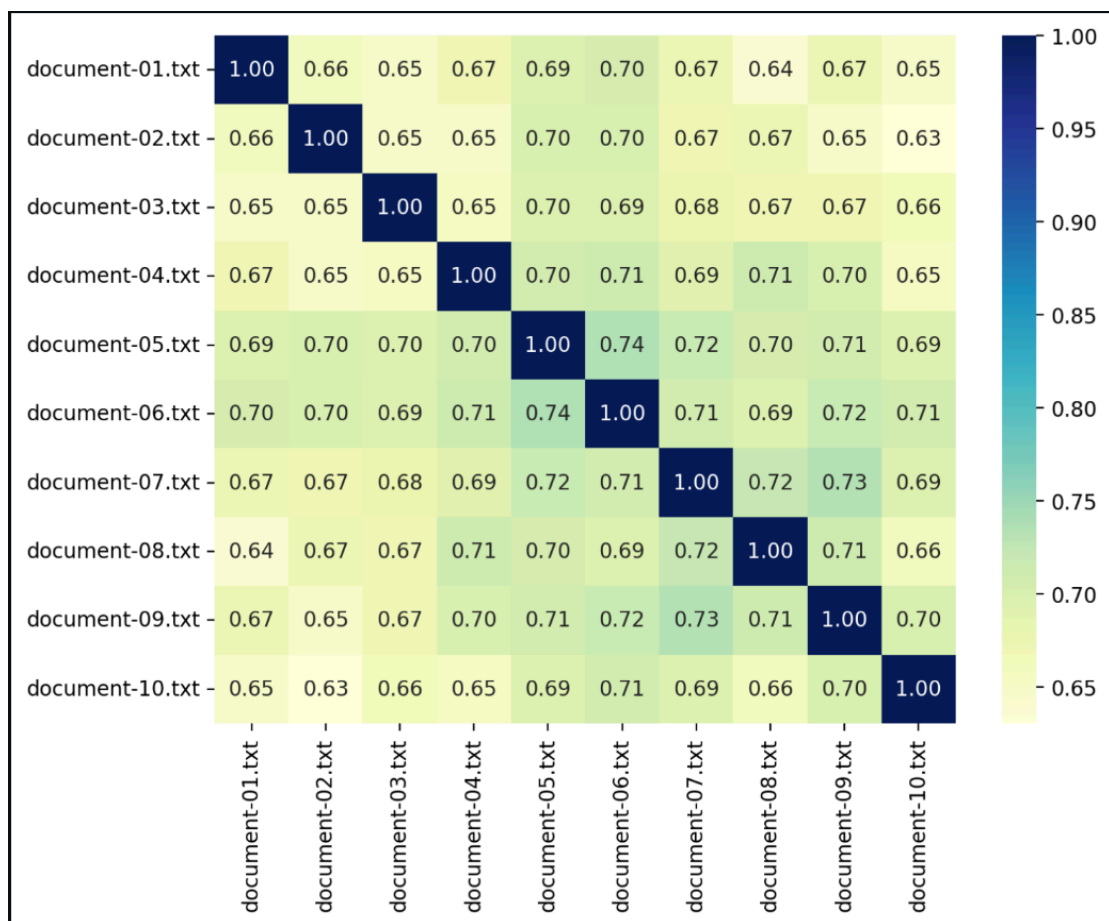
document-06.txt

Similitud del coseno entre document-05.txt y document-06.txt: 0.736

Términos comunes (152):

rise, edge, time, soil, gold, notebook, notice, balance, end, unseen, lock, write, light, air, glass, feel, quietly, small, gentler, sound, kind, trail, hear, sky, pine...





### Análisis con documentos nuevos

La matriz de similaridad coseno del segundo corpus muestra un rango de valores menor que en el conjunto original.

Las puntuaciones de similaridad varían entre 0.19 y 0.48, tienen valores medios alrededor de 0.30–0.35, por lo que podemos decir que vemos una menor coherencia léxica global entre los documentos del corpus original.

Los resultados son los esperados ya que escogimos los texto para que estuvieran algunos muy relacionados entre ellos y otros nada relacionados.



Los valores más altos (aproximadamente 0.48) los encontramos entre los documentos 2, 3, y 4. Mientras que los menos relacionados son los documentos que tienen valores aproximados de 0.19 como por ejemplo el 3 y 8.

Con esto confirmamos que el modelo TF-IDF que hemos desarrollado puede distinguir bien qué documentos son más afines y cuáles menos.

**Comparador directo de documentos**

Documento A  
doc-02.txt

Documento B  
doc-03.txt

Similitud del coseno entre doc-02.txt y doc-03.txt: 0.482

Términos comunes (22):  
time, notebook, sit, faint, move, a, light, air, water, feel, small, think, sound, path, look, long, thin, trees, begin, watch, beneath, world

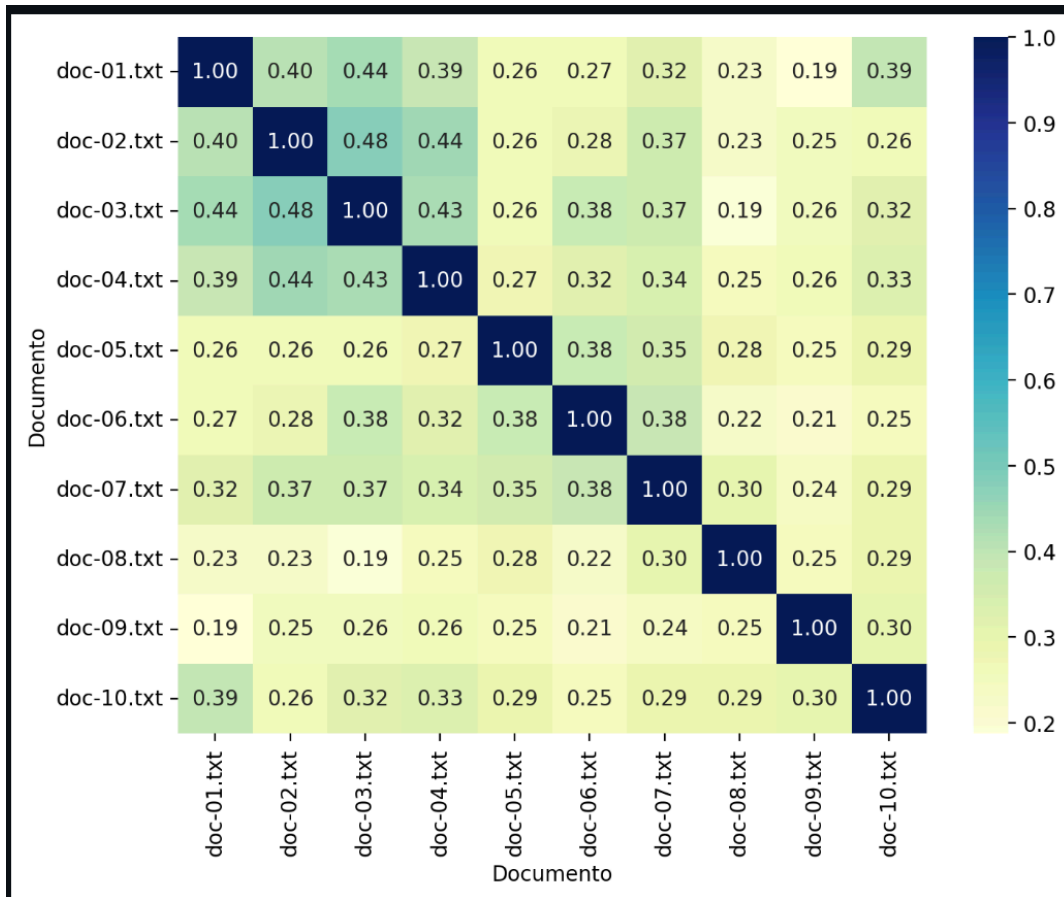
**Comparador directo de documentos**

Documento A  
doc-03.txt

Documento B  
doc-08.txt

Similitud del coseno entre doc-03.txt y doc-08.txt: 0.193

Términos comunes (5):  
feel, time, a, stay, light



## Visualización y análisis de resultados

Para facilitar la interpretación de los datos obtenidos, tenemos un dashboard interactivo en Python utilizando la librería Streamlit.

Este panel permite representar de forma visual los valores de TF-IDF y las relaciones de similitud coseno calculadas por el sistema Java.

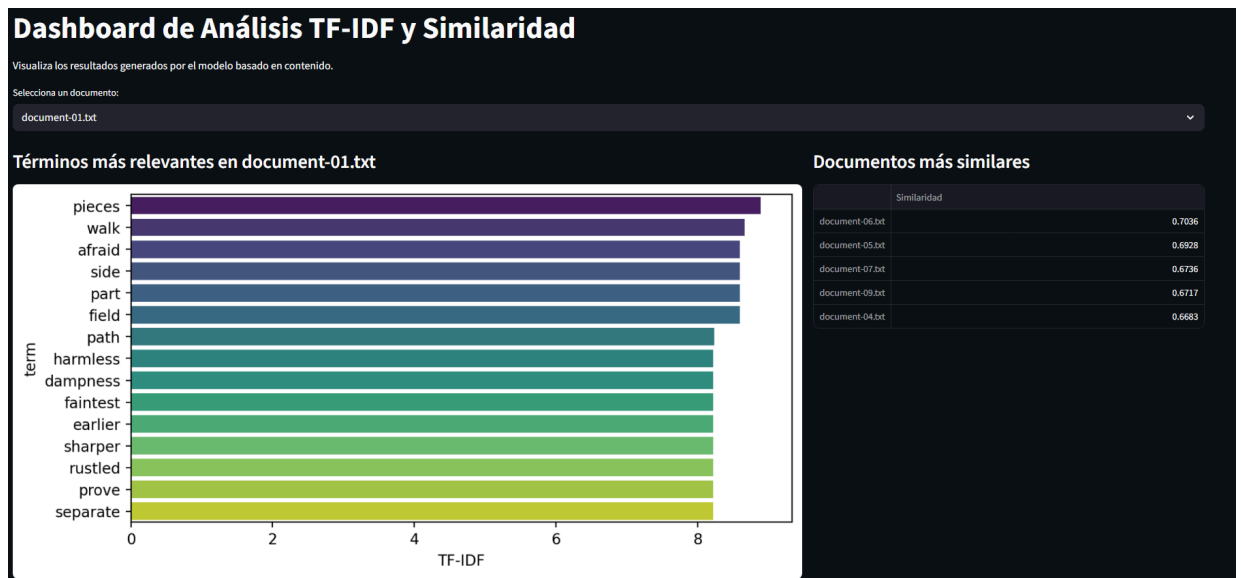
El dashboard utiliza automáticamente los ficheros CSV generados (output/per\_document/ y output/similarity.csv):

- **Top términos TF-IDF:** gráfico de barras que muestra las palabras más relevantes de cada documento.
- **Documentos más similares:** tabla que lista los textos con mayor afinidad semántica respecto al documento seleccionado.



- **Matriz de similitud (heatmap):** representación visual de los valores de similitud coseno, donde los tonos más oscuros indican una mayor similitud entre documentos.

Este módulo no forma parte del sistema de cálculo principal, sino que actúa como herramienta de análisis de una manera muy visual, permitiendo ver mejor el comportamiento del modelo y los resultados obtenidos.



## Conclusión

El desarrollo de este sistema de recomendación ha permitido comprender en profundidad el funcionamiento y las ventajas del modelo basado en el contenido. A través del cálculo de TF-IDF y la similitud coseno, se ha conseguido representar y comparar documentos de manera objetiva, obteniendo recomendaciones consistentes y relevantes.

Los experimentos realizados muestran que el método es eficaz para identificar similitudes temáticas entre textos y que su rendimiento es adecuado incluso con corpus de tamaño medio. Además, la combinación de técnicas de limpieza de texto, eliminación de stop words y lematización mejora de forma notable la calidad de las recomendaciones.