Similitud Coseno

Iván Molina Rebolledo

Benemérita Universidad Autónoma de Puebla

Primavera 2022
3 de mayo de 2022

Motivación

En el analisís de documentos, la comparación de los mismos es muy importante para extraer información relevante que puede ser usada en consultas. o extracción de datos.

Motivación

En el analisís de documentos, la comparación de los mismos es muy importante para extraer información relevante que puede ser usada en consultas. o extracción de datos. Por eso resulta necesario buscar maneras de poder comparar documentos, y que nos permita tener un valor de similitud preciso para poder describir a detalle los documentos que se comparan.

Similitud Coseno

Una de las formas de comparar documentos es la similitud coseno. La similitud coseno es una medida de similitud entre dos documentos, que se calcula a partir de la suma de productos de los vectores de pesos de cada uno de los documentos.

Similitud Coseno

Una de las formas de comparar documentos es la similitud coseno. La similitud coseno es una medida de similitud entre dos documentos, que se calcula a partir de la suma de productos de los vectores de pesos de cada uno de los documentos.
Esto es dividido entre el producto de la raiz de la suma de los cuadrados de cada uno de los elementos de los vectores de pesos.

Similitud Coseno

Una de las formas de comparar documentos es la similitud coseno. La similitud coseno es una medida de similitud entre dos documentos, que se calcula a partir de la suma de productos de los vectores de pesos de cada uno de los documentos.
Esto es dividido entre el producto de la raiz de la suma de los cuadrados de cada uno de los elementos de los vectores de pesos.
Más adelante veremos a detalle cómo se implementa esta similitud.

Diseño del proyecto

Este proyecto está dividido en diversas etapas internas que nos permiten tener un control total de lo que se está haciendo en cada paso.

Diseño del proyecto

Este proyecto está dividido en diversas etapas internas que nos permiten tener un control total de lo que se está haciendo en cada paso.
Estas etapas se explican a continuación en las siguientes diapositivas.

Parsing

En esta etapa se realiza el parsing de los documentos, y se obtiene una lista de los documentos que se van a analizar.

Parsing

En esta etapa se realiza el parsing de los documentos, y se obtiene una lista de los documentos que se van a analizar.
Aquí es también donde los documentos son procesados. Pasamos cada uno de los textos que tokenizan y elimnan simbolos innecesarios.

Parsing

En esta etapa se realiza el parsing de los documentos, y se obtiene una lista de los documentos que se van a analizar.
Aquí es también donde los documentos son procesados. Pasamos cada uno de los textos que tokenizan y elimnan simbolos innecesarios.
Como último paso se realiza el stemming de los documentos.

Parsing

En esta etapa se realiza el parsing de los documentos, y se obtiene una lista de los documentos que se van a analizar.
Aquí es también donde los documentos son procesados. Pasamos cada uno de los textos que tokenizan y elimnan simbolos innecesarios.
Como último paso se realiza el stemming de los documentos. Esto es hecho a partir de una librería de steaming programada en C.

Parsing

En esta etapa se realiza el parsing de los documentos, y se obtiene una lista de los documentos que se van a analizar.
Aquí es también donde los documentos son procesados. Pasamos cada uno de los textos que tokenizan y elimnan simbolos innecesarios.
Como último paso se realiza el stemming de los documentos. Esto es hecho a partir de una librería de steaming programada en C. Sin embargo, es un poco antigua, así que puede no ser tan eficiente como sus contrapartes en lenguajes como Python.

Stage 0 (Estructura de datos)

Es importante definir una estructura de datos para los documentos. Nosotros la establecimos como se muestra a continuación:

Stage 0 (Estructura de datos)

Es importante definir una estructura de datos para los documentos. Nosotros la establecimos como se muestra a continuación:

```
data Documents = Documents {
  __docsN :: Int,
  __docs :: M.Map Int [(Int, Int)],
  __weights :: M.Map Int [(Int, Float)],
  __words :: [T.Text]
} deriving (Show, Generic)
```

Además en esta etapa se incializa la estructura de datos.

Stage 0 (Estructura de datos)

Es importante definir una estructura de datos para los documentos. Nosotros la establecimos como se muestra a continuación:

```
data Documents = Documents {
  __docsN :: Int,
  __docs :: M.Map Int [(Int, Int)],
  __weights :: M.Map Int [(Int, Float)],
  __words :: [T.Text]
} deriving (Show, Generic)
```

Además en esta etapa se incializa la estructura de datos. Establemos las palabras y el número de documentos, mientras que lo demás lo incializamos vacío.

Stage 1

En esta etapa no sucede mucho.

Stage 1

En esta etapa no sucede mucho. Sólo se realiza el proceso de conteo de palabras y se guarda en la estructura de datos.

Stage 1

En esta etapa no sucede mucho. Sólo se realiza el proceso de conteo de palabras y se guarda en la estructura de datos.
Al finalizar, tendremos una estructura de datos actualizada.

Stage 2 I

Esta etapa se encarga de calcular los pesos para cada uno de los terminos. Esto está dado por la siguientes funciones:

```
idf :: Int -> Documents -> Float
idf word doc = let
  n = fromIntegral $ doc ^. _docsN
  d = fromIntegral $ length $ doc ^. _docs &
    M.lookup word & fromMaybe []
  in logBase 2 $ n / d
```

Stage 2 II

```
idf :: Int -> Documents -> Float
idf word doc = let
  n = fromIntegral $ doc ^. _docsN
  d = fromIntegral $ length $ doc ^. _docs &
    M.lookup word & fromMaybe []
  in logBase 2 $ n / d
```

Stage 2 III

```
tfidf :: Int -> Int -> Documents -> Float
tfidf word doc docs =
  tf word doc docs * idf word docs
```

Stage 2 IV

Basta con proveer de los indices adecuados (así como de la estructura de datos) para que se pueda realizar el cálculo de los pesos.

Similitud Coseno

Está defincíon es el centro de nuestro proyecto.

Similitud Coseno

Está definción es el centro de nuestro proyecto.

```
cosineSimilarity :: [Float] -> [Float] -> Float
cosineSimilarity v1 v2 = let
  t = sum $ zipWith (*) v1 v2
  a b = sqrt $ sum $ map (^2) b in
  t / (a v1) * (a v2)
```

Sin embargo la función no es invocable directamente, sino que se debe invocar desde una función externa que se encarga de realizar el cálculo de la similitud coseno para todos los pares de documentos.

Ensamblaje (*main*)]

En esta etapa se realiza el ensamblaje del proyecto.

Ensamblaje (*main*)]

En esta etapa se realiza el ensamblaje del proyecto. Definimos cada una de las funciones que nos permiten generar las tablas de datos.

Ensamblaje (*main*)]

En esta etapa se realiza el ensamblaje del proyecto. Definimos cada una de las funciones que nos permiten generar las tablas de datos.
Las tablas generadas son exportadas a archivos LaTeX, para que puedan ser visualizadas en el documento principal de este proyecto.

Datos

Por desgracia, no se puede visualizar todos los datos que se generan en una sola diapositiva.

Datos

Por desgracia, no se puede visualizar todos los datos que se generan en una sola diapositiva.
Tan sólo una de las tablas tiene más de cuatrocientosmil registros.

Datos

Por desgracia, no se puede visualizar todos los datos que se generan en una sola diapositiva.
Tan sólo una de las tablas tiene más de cuatrocientosmil registros.
Por suerte LaTeX es capaz de visualizar una tabla de datos tan grande, y es la manera en la que se muestran los datos en el documento.

Conclusiones

El proceso de indexación es lento y puede ser demorado.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande. Sin embargo, existen alternativas que pueden ser muy eficientes.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande. Sin embargo, existen alternativas que pueden ser muy eficientes.
En cuanto a la similitud coseno, calcular todos los pares de documentos es una tarea muy grande.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande. Sin embargo, existen alternativas que pueden ser muy eficientes.
En cuanto a la similitud coseno, calcular todos los pares de documentos es una tarea muy grande. De hecho, es el paso más lento y complicado de computar.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande. Sin embargo, existen alternativas que pueden ser muy eficientes.
En cuanto a la similitud coseno, calcular todos los pares de documentos es una tarea muy grande. De hecho, es el paso más lento y complicado de computar. Aunque esto se puede deber a la cantidad de documentos que estamos tratando.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande. Sin embargo, existen alternativas que pueden ser muy eficientes.
En cuanto a la similitud coseno, calcular todos los pares de documentos es una tarea muy grande. De hecho, es el paso más lento y complicado de computar. Aunque esto se puede deber a la cantidad de documentos que estamos tratando.
Sin embargo los tiempos de ejecución mejoran al compilarse a código nativo con GHC.

Conclusiones

El proceso de indexación es lento y puede ser demorado. Esto se debe a la forma que manejamos los datos para ser procesados.
Aparte Haskell es un lenguaje inmutable, por lo que el costo de modificar una estructura de datos es muy grande. Sin embargo, existen alternativas que pueden ser muy eficientes.
En cuanto a la similitud coseno, calcular todos los pares de documentos es una tarea muy grande. De hecho, es el paso más lento y complicado de computar. Aunque esto se puede deber a la cantidad de documentos que estamos tratando.
Sin embargo los tiempos de ejecución mejoran al compilarse a código nativo con GHC. Mientras que el cálculo con GHCI es mucho más lento.

Muchas gracias

Este ha sido un proyecto de investigación de la asignatura de Recuperación de Información.
«Espacio para dudas y comentarios, demostración del código»