

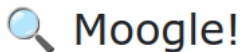
# Mooglee!

Eduardo Brito Labrada


Facultad de Matemática y Computación

21 de julio de 2023

# Una breve introducción



Introduzca su búsqueda

 Buscar

Moog! es una aplicación *totalmente original* cuyo propósito es buscar inteligentemente un texto en un conjunto de documentos.

# Una breve introducción

Es una aplicación web, desarrollada con tecnología .NET Core 6.0, específicamente usando Blazor como *framework* web para la interfaz gráfica, y en el lenguaje C#.

# Una breve introducción

La aplicación está dividida en dos componentes fundamentales:

- **Moog1eServer** es un servidor web que renderiza la interfaz gráfica y sirve los resultados.

# Una breve introducción

La aplicación está dividida en dos componentes fundamentales:

- `Moogleserver` es un servidor web que renderiza la interfaz gráfica y sirve los resultados.
- `Moogleserver` es una biblioteca de clases donde está...ehem...casi implementada la lógica del algoritmo de búsqueda.

# ¿Para qué sirve?

La idea original del proyecto es buscar en un conjunto de archivos de texto (con extensión `.txt`) que estén en la carpeta `Content`.

Primeramente, se aconseja a quien use esta aplicación tener instalado Linux, ya que no se garantiza la misma eficiencia si esta en un dispositivo que use Windows.

En caso de tener instalado Windows, puede optar por instalar Windows Subsystem for Linux (WSL) que añade funcionalidades de Linux en Windows.



# Instrucciones

Lo primero que el usuario debe hacer para poder usar este proyecto es instalar .NET Core 6.0.

# Instrucciones

Lo primero que el usuario debe hacer para poder usar este proyecto es instalar .NET Core 6.0.

Luego, debe pararse en la carpeta del proyecto y dependiendo de su sistema operativo hacer lo siguiente:

- **Linux o WSL:** Debe tener instalado `make`. Puede instalarlo ejecutando el siguiente comando en el terminal `sudo apt update && sudo apt install make`. Luego deberá ejecutar `make dev`

# Instrucciones

Lo primero que el usuario debe hacer para poder usar este proyecto es instalar .NET Core 6.0.

Luego, debe pararse en la carpeta del proyecto y dependiendo de su sistema operativo hacer lo siguiente:

- **Linux o WSL:** Debe tener instalado `make`. Puede instalarlo ejecutando el siguiente comando en el terminal `sudo apt update && sudo apt install make`. Luego deberá ejecutar `make dev`
- **Windows:** Debería poder ejecutar este proyecto usando `dotnet watch run --project Moogleserver`

# Instrucciones

- Abra en su navegador `http://localhost:5000`
- Introduzca su búsqueda en la “entrada” y luego presionando el botón “Buscar”

# Motor de búsqueda

El motor de búsqueda usa un modelo vectorial que computa para una *query* dada qué tan relevante es un documento determinado.

# Motor de búsqueda

Este modelo vectorial usa *TF-IDF* con *Cosine Similarity* para computar la relevancia.

# Term Frequency and Inverse Document Frequency

Para computar el vector *TF-IDF* se hace uso de la fórmula:

$$TFIDF = \left(\frac{tf}{tw}\right) \times \ln\left(\frac{td}{dt}\right)$$

- *tf* es la frecuencia del término en el documento actual.
- *tw* es la cantidad de palabras totales en el documento actual.
- *td* es la cantidad total de documentos a analizar.
- *dt* es la cantidad de documentos que contienen el término.

# Term Frequency and Inverse Document Frequency

Para computar el vector *TF-IDF* se hace uso de la fórmula:

$$TFIDF = \left(\frac{tf}{tw}\right) \times \ln\left(\frac{td}{dt}\right)$$

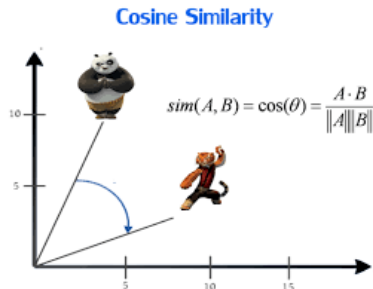
- *tf* es la frecuencia del término en el documento actual.
- *tw* es la cantidad de palabras totales en el documento actual.
- *td* es la cantidad total de documentos a analizar.
- *dt* es la cantidad de documentos que contienen el término.

**Nota:** dado que  $\frac{td}{dt}$  puede causar problemas por la división entre 0 decidí que si  $dt = 0$  luego  $TFIDF = 0$ , tiene sentido hacer esto ya que si  $dt = 0$  el término no aparece en ningún documento.



# Cosine Similarity

Después de lo anterior necesitamos calcular la “similitud” entre el vector *document* y el vector *query* para lo cual se hace uso de *Cosine Similarity*.



# Cosine Similarity

La idea es intentar estimar el “ángulo” comprendido entre el vector *query* y el vector *document*: mientras menor sea este ángulo, mayor “similitud” tendrán estos vectores. Para lo anterior se hace uso de la fórmula:

$$\cos \alpha = \frac{v_d \cdot v_q}{||v_d|| ||v_q||}$$

- $v_d$  el vector de *document*
- $v_q$  el vector de *query*
- $||v||$  es la magnitud del vector  $v$

# Funcionalidades adicionales

Están implementados y se pueden usar sin problemas de ningún tipo los siguientes operadores:

- !: delante de una palabra este símbolo indica que esa palabra **no debe aparecer** en ningún documento que sea devuelto.

# Funcionalidades adicionales

Están implementados y se pueden usar sin problemas de ningún tipo los siguientes operadores:

- !: delante de una palabra este símbolo indica que esa palabra **no debe aparecer** en ningún documento que sea devuelto.
- ^: delante de una palabra este símbolo indica que esa palabra **debe aparecer** en cualquier documento que sea devuelto.

# Funcionalidades adicionales

Están implementados y se pueden usar sin problemas de ningún tipo los siguientes operadores:

- **!**: delante de una palabra este símbolo indica que esa palabra **no debe aparecer** en ningún documento que sea devuelto.
- **^**: delante de una palabra este símbolo indica que esa palabra **debe aparecer** en cualquier documento que sea devuelto.
- **~**: entre dos o más términos este símbolo indica que esos términos deben aparecer cerca, o sea, que mientras más cercanos estén en el documento mayor será la relevancia.

# Funcionalidades adicionales

Están implementados y se pueden usar sin problemas de ningún tipo los siguientes operadores:

- **!**: delante de una palabra este símbolo indica que esa palabra **no debe aparecer** en ningún documento que sea devuelto.
- **^**: delante de una palabra este símbolo indica que esa palabra **debe aparecer** en cualquier documento que sea devuelto.
- **~**: entre dos o más términos este símbolo indica que esos términos deben aparecer cerca, o sea, que mientras más cercanos estén en el documento mayor será la relevancia.
- **\***: cualquier cantidad de símbolos **\*** delante de un término indican que ese término es más importante, por lo que su influencia en el score debe ser mayor que la tendría normalmente (este efecto será acumulativo por cada **\***)