Preentrega Proyecto Final de SRI:  
Modelo Vectorial

Lauren Guerra Hernández

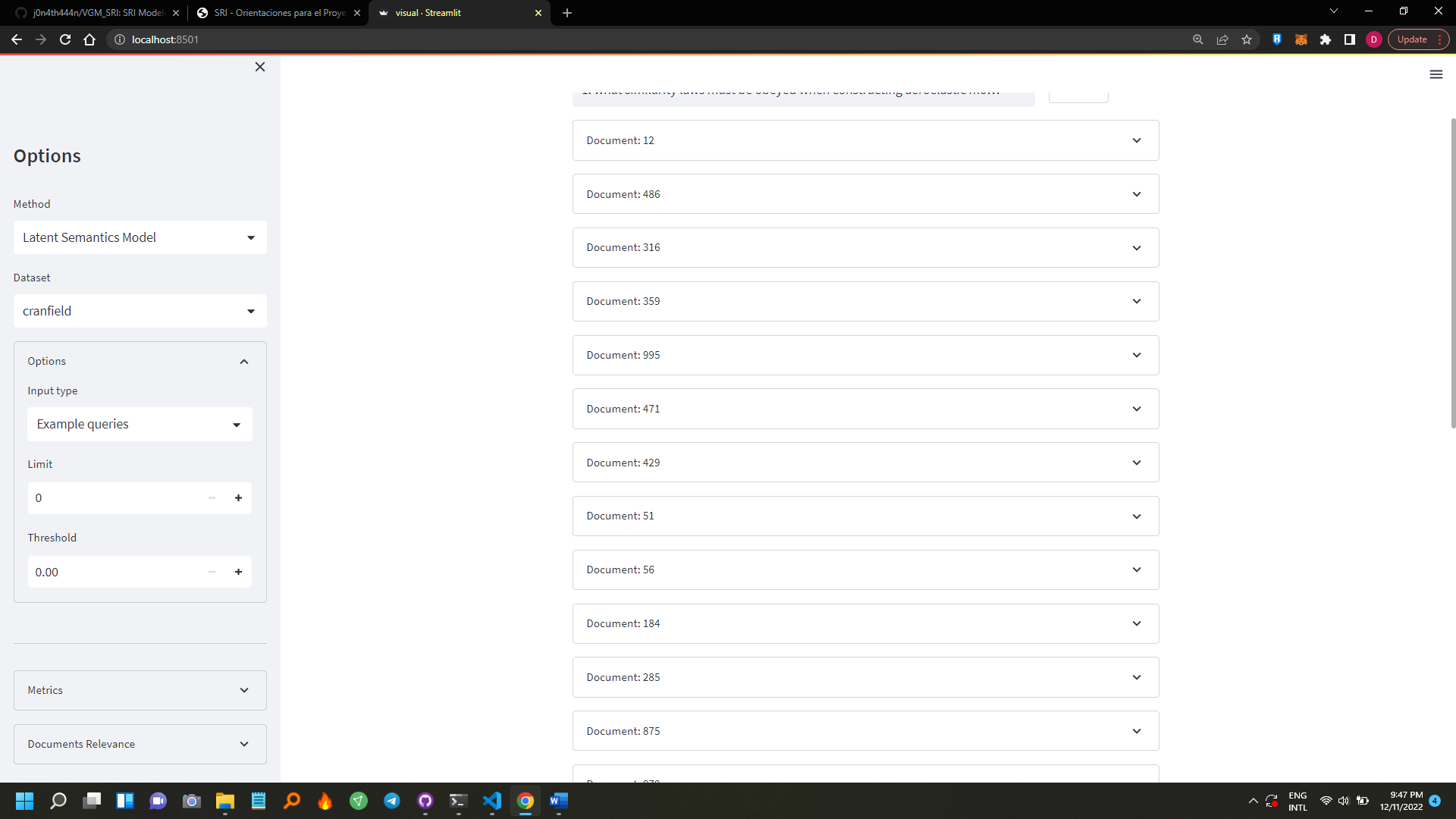
Paula Rodríguez Pérez

Dennis Fiallo Muñoz

**Presentación Visual:**

La interfaz visual se implementa usando las facilidades del módulo de Python streamlit. De esta se presentan a continuación sus funcionalidades.

Presenta a su izquierda una barra de opciones como se ve en la imagen.

Aparece primeramente un selectbox para alternar entre los modelos de recuperación de información presentados.

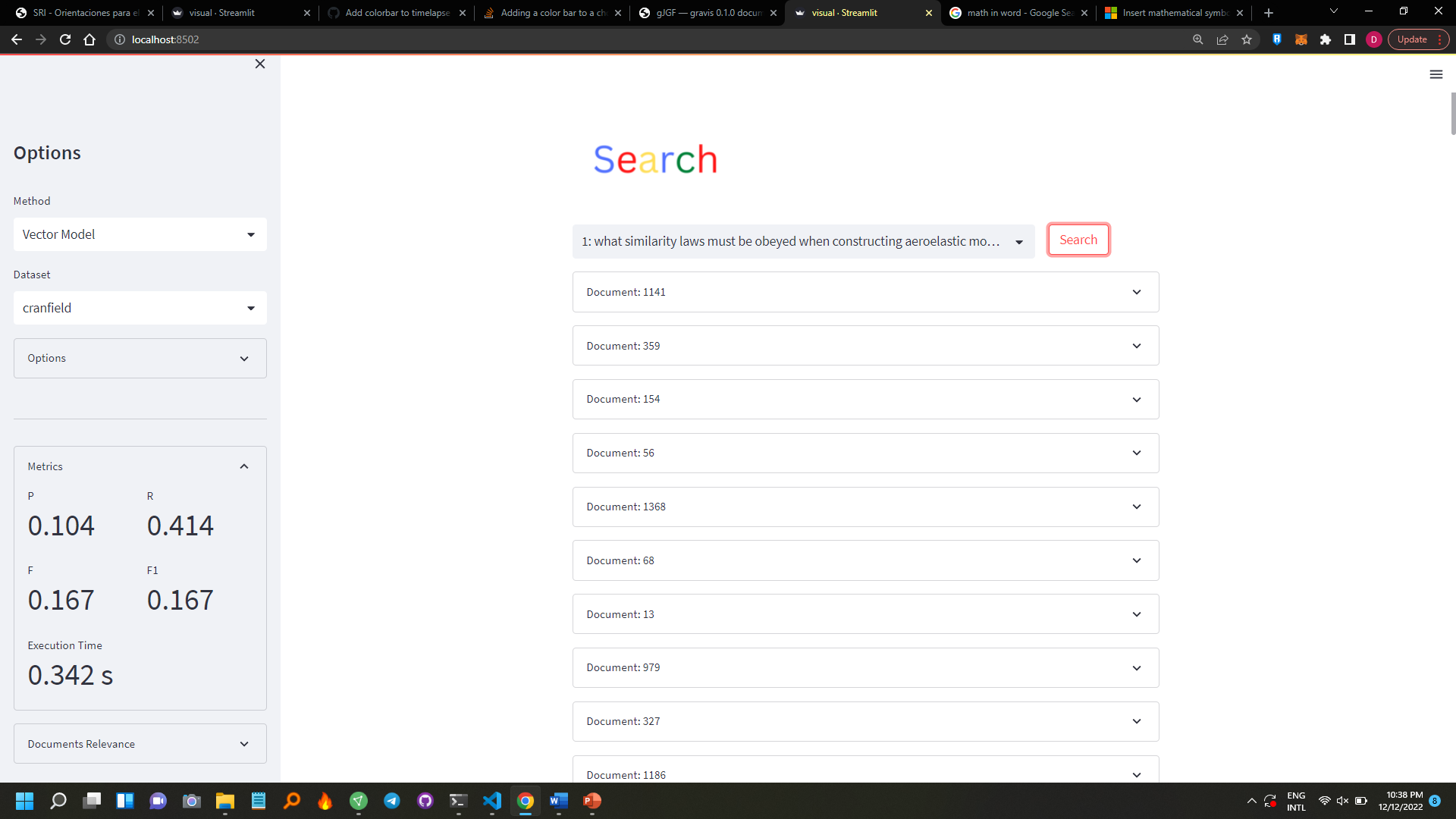
Luego aparece otro selectbox para seleccionar el set de datos del cual se quiere hacer la consulta.

Más abajo se presenta una ventada desplegable con las opciones de alternar el método para recibir la entrada de la consulta (entre una entrada de texto clásica o un conjunto de consultas predeterminado). Además, se puede cambiar tanto el límite como el umbral de los resultados que se devuelven.

Bajo la línea va a estar una ventana desplegable que cuando se tengan los resultados de la consulta nos dará los resultados de las métricas.

Mientras que más abajo se podrán ver de la consulta actual los documentos que son realmente relevantes, cuan relevantes con en realidad además de decir si este fue recuperado por el modelo seleccionado o no.

Mientras, del lado derecho de esta barra se mostrará la selección de consulta y los resultados de esta en ventanas desplegables en las cuales se da es título del documento, su contenido y su similitud con la consulta.



Modelo Base:

La implementación de los tres sistemas que presentamos van a seguir los mismos pasos en la recuperación de información pues heredan de la clase base Model. Por tanto, los sistemas van a seguir las etapas que se presentan a continuación:

**Etapa 1:** Verificar si ya se tienen almacenada la información recuperada del set de datos actual, en caso de que no se hayan recuperado los datos de estos documentos con anterioridad se pasa a la Etapa 2 para extraer los datos, sin embargo, si ya se tienen los datos recuperados se pasa a la Etapa 5 para lograr mayor velocidad al mostrar los resultados ya que se cuenta con la información necesaria de los documentos.

**Etapa 2:** Tomar los documentos del set de datos.

**Etapa 3:** Separar los términos de cada documento y eliminar los signos de puntuación y las stopwords, luego se lexemizan los términos para quedarnos con su forma canónica. Después de tener la lista de terminos se pasa a aplicar la expansión de consulta buscando sinónimos del término y agregándolo a la consulta.

**Etapa 4:** Calcular y almacenar la relación entre término con los documentos en que que aparece (cada modelo define su modo de hacer esta etapa).

**Etapa 5:** Calcular y almacenar los pesos de las consultas en los documentos del siguiente modo:

{término: w, ...}

**Etapa 6:** Calcular la similitud de la consulta con los documentos usando los datos recopilados anteriormente.

**Etapa 7:** Devolver ranking de documentos.

**Implementación del código y herramientas usadas:**

El código va a tener separado en diferentes archivos las clases principales de la ejecución que serían:

Model: Este se encuentra en base\_model.py y será la base de todos los modelos de recuperación de información que se presentan.

Datasets: Aparece en dataset.py, este se va a inicializar desde Model. Se encarga de extraer la información de los datasets; desde los términos en los documentos, hasta las consultas de ejemplo y la relevancia real de los documentos con respecto a las consultas. Además, tiene un conjunto de propiedades y métodos estáticos para tomar datos necesarios en diferentes formas según el uso que se le va a dar. Se apoya en el módulo de python ir\_datasets para facilitar el acceso a la información de los datasets de ejemplo.

Lexemizer: Este se encuentra en lexemizer.py. Apoyándose de las facilidades del módulo nltk, este está destinado a normalizar los terminos de los documentos y de las consultas; separando las palabras en tokens independientes, eliminado las stopwords, eliminado las mayúsculas y lematizando los términos, y para el caso de las consultas también se le aplica una expansión de consulta usando wordnet para agregar sinónimos de los términos.

Visual: Se encuentra en visual.py y se encarga de llevar todo el proceso visual del proyecto.

**Modelo Vectorial:**

Como ya vimos anteriormente, la implementación del Modelo Vectorial va a heredar de la clase Model, siguiendo pasos similares al resto de modelos. En un principio, al inicializar el modelo se crean 3 dict vacíos, los cuales van a ser usados más adelante, siendo estos:

docterms: un diccionario que tendrá como llaves los términos de todos los documentos, teniendo como valid almacenado cada uno los documentos en que aparecen y de estos la frecuencia, tf, idf y w. Se usa esto pues la mayor parte de los términos aparecen en una pequeña parte de los documentos, por tanto, al eliminar los elementos que no son necesarios se puede lograr ahorrar algo de tiempo de procesamiento.

queryterms: diccionario de términos de la query con su respectivo peso.

querysim: diccionario de documentos que tienen como valor su similitud con la consulta.

Luego se siguen una serie de etapas que en general se siguen en todos los modelos. Se comienza verificando si ya se encuentran recopilados los datos de los términos de los documentos, en este caso en docterms, si se encuentra vacío se extrae la información apoyándose en la clase Datasets. Para los cálculos de estos datos se usan las formulas clásicas de:

Mientras que para la consulta se aplica un suavizado variable con valor predeterminado de 0.5:

De estos se tiene que:

* sería la freqcuencia del término en el documento .
* Se calcula el máximo de los documentos.
* Si no aparece el término en el documento entonces
* es la cantidad de documentos en el sistema.
* es es la cantidad de documentos en que aparece el término .

Luego de tener los datos de los términos en los documentos y de la query se pasa a calcular la similitud entre estos usando:

Luego se devuelve un ranking organizado por similitud y eliminando los que sean cero, o en caso de existir un umbral los que estén por encima de este.

**Modelo Probabilístico:**

La implementación del Modelo Probabilístico hereda de la clase Model. En la inicialización del modelo se tendrán 3 diccionarios (dict) y un conjunto (set):

term\_docs: diccionario para almacenar los términos de la colección y su respectiva lista de documentos en los que aparece.

queryterms: diccionario para almacenar los términos de la consulta.

query\_doc\_sim: diccionario para almacenar por cada documento de la colección su similitud con la consulta

term\_p\_r: diccionario para almacenar las probabilidades de ocurrencia de un término en un documento relevante o no relevante a la consulta. Estas probabilidades serán almacenadas en una tupla donde el primer valor corresponde a la probabilidad de ocurrencia del término en un documento relevante a la consulta y en la segunda posición la probabilidad de que ocurra en un documento no relevante a la consulta.

Al igual que en los otros modelos implementados se verifica si ya se analizó la ocurrencia de los términos de la colección en los documentos, en este caso en term\_docs es donde se almacena dicha información, si se encuentra vacío se extrae la información apoyándose en la clase Datasets.

Para la implementación de nuestro modelo inicialmente la probabilidad de ocurrencia de un término en un documento relevante será tomada como una constante (0.5) y la probabilidad de ocurrencia de un término en un documento no relevante como:

Donde:

* es la cantidad de documentos en la colección.
* es es la cantidad de documentos en que aparece el término .

Posteriormente con estos valores de probabilidades iniciales se procede a calcular la similitud entre un documento y la consulta a través de la siguiente fórmula:

Donde:

* son la ocurrencia del término en el documento y la consulta respectivamente.
* serán las probabilidades de que el término ocurra en un término relevante o no relevante respectivamente.

Una vez obtenidos estos valores de similitud se realizará un ranking con los mismos.

Luego se procederá a realizar una pseudo-retroalimentación y para ello los valores de serán calculados de la siguiente forma:

Donde:

* es el conjunto de documentos recuperados.
* es el conjunto de los documentos recuperados en los que aparece el término .

Serán aplicadas constantes de suavizado en cada una de las fórmulas (0.5 en el numerador y 1 en el denominador) para evitar división por cero o logaritmos indeterminados.

Una vez obtenidos los nuevos valores de la similitud será computada nuevamente y se retornará un ranking en base a esta.

Este proceso donde se recalculan las probabilidades de ocurrencia de los términos en documentos relevantes o no, se analiza la similitud y se devuelve un ranking se realizará en tres ocasiones. El último ranking calculado será retornado como respuesta del algoritmo y se tendrá en cuenta para este si es establecido un umbral para la similitud.