Charlotte
Un buscador amigable

Wendy Díaz w.diaz@estudiantes.matcom.uh.cu[1]
Daniel de la Osa d.osa@estudiantes.matcom.uh.cu[1]
Jose Luis Alvarez j.alvarez@estudiantes.matcom.uh.cu[1].

Facultad de Matemática y Computación (MATCOM),
Universidad de la Habana (UH), Cuba.

Resumen Se presenta una aplicación que permite construir un sistema de recuperación de información sobre un directorio especificado, basado en el modelo de Semantica Latente. Este sistema se implementó con mecanismos de retroalimentación a traves de la interacción con el usuario y también la capacidad de anotar un corpus en cuanto a *querys* contra documentos relevantes.

Palabras Claves: LSI, LSA, SVD, RNN

1. Introducción

Los sistemas de recuperación de información son herramientas de gran importancia hoy en día, sobre todo con el aumento exponencial de la información generada a diario. Estos permiten de forma eficiente localizar dentro de miles de documentos lo que se busca de forma rápida e eficiente superando a cualquier bibliotecario. Además permiten retroalimentarse de la información que brinda el usuario "aprendiendo" a recuperar mejor cada vez. Estos sistemas se basan en modelos matemáticos, dentro de estos modelos encontramos el **LSI** o modelo de *Latent Semantic Indexing*. Las ventajas de este modelo [1,2] lo hacen un candidato para su implementación en una aplicacíon que permita la creación de un **SRI** sobre un directorio especificado.

2. Descripción

Charlotte es un Sistema de Recuperación de Información que trabaja de manera local sobre una carpeta del sistema hospedero. Es capaz de cargar todos los archivos en formato **txt** y **pdf** y a partir de estos generar un sistema de recuperación de información basado en un modelo especificado, en este caso **LSI** o Vectorial, y de esta manera se podrán hacer consultas sobre este directorio y anotarlo.

Como ya mencionamos se pueden elegir entre dos modelos para realizar búsquedas, el modelo Vectorial y el modelo de **Semántica Latente**. El usuario puede escoger construir una instancia de cualquiera de estos modelos; o incluso

seleccionar ambos, con el fin de realizar comparaciones entre ellos gracias a las facilidades que brinda el sistema. Además también se puede elegir los valores para el parametro **k** del modelo **LSI** así como la cantidad de elementos que se quiere que se muestren dado una búsqueda.

Sobre la elección del **k** como plantea [1] en su artículo, es empírico por lo que se debería probar con distintos valores para un directorio en específico para alcanzar los mejores resultados.

La implementación de los modelos se apoyaron en las bibliotecas de *Sklearn*, *Gensim* para el vectorial y **LSI** respectivamente, *Numpy* para el trabajo con las matrices, muy importante para estos modelos ya que el **Álgebra Vectorial** constituye su *framework*

Ya generado el modelo podemos hacer diferentes búsquedas en lenguaje natural, participar en un proceso de retroalimentción y ver estadísticas sobre diferentes métricas que miden el desempeño del sistema. Además podemos anotar el directorio y y de esta forma crear un corpus de prueba para evaluar el sistema.

3. Uso

Para empezar a utilizar la aplicación lo primero será iniciar el servidor de flask ejecutando el archivo **start.bat** ubicado en la raíz del proyecto. Copie la dirección http://localhost:5000 en su buscador para acceder a la interfaz gráfica de esta.

En la página inicial seleccione primero el modelo deseado y después haga *click* en el botón *Seleccionar directorio*. Esto es necesario para ejecutar aquellas funcionalidades que dependen de un modelo ya inicializado.

Entonces aparecerá la vista de búsqueda como valor predeterminado, pero se abrirán muchas más opciones en la barra de navegación. Empecemos por la vista inmediata.

3.1. Pestaña *Search*

En la vista de búsqueda se tiene un **text-box** en el cual el usuario escribirá la consulta a realizar. Justo al lado se encuentra el botón '*Look for it*' que realiza el pedido al servidor.

Los resultados después se muestran en forma de tabla; mostrando para cada documento, su nombre (con un *link* al mismo), la similitud alcanzada con el vector consulta, y un *checkbox* que indica si el documento es relevante dada la consulta actual. Esto último es lo que permite la retroalimentación del sistema.

3.2. Pestaña *Compare*

Es esta pestaña se podrá realizar consultas y ver los resultados de cada modelo (Vectorial y Semántico) *documento a documento*. Mostrará una tabla donde la fila $i-$ésima contendrá los elementos $i-$ésimos del *ranking* para ambos modelos asi como su valor de similitud. O sea, la primera fila tendrá el elemento más importante de cada modelo respectivamente.

3.3. Pestaña *Relevants*

Aqui el usuario podrá marcar de una manera muy cómoda los documentos relacionados con una consulta de su elección. Este tipo de información, si bien es muy tedioso para introducirla en el sistema, es de gran utilidad. Permite medir y comparar los resultados del modelo con el objetivo de ajustar los parámetros del mismo para obtener un mejor desempeño.

Para que el usuario sea capaz de realizar esta ardua tarea de la mejor forma posible, le mostramos al usuario parte del texto para que se familiarize con el contexto rápidamente.

3.4. Pestaña *Statistics*

En esta sección se muestran los valores de las métricas para cada consulta guardada en disco. Estas son:

1. *Precision*
2. *Recobrado*
3. $Medida{-}F_1$
4. $Medida{-}F$
5. $R - precision$

Una consulta se guarda en disco si tiene *feedback* o conjunto de documentos relevantes. Ambas informaciones solo pueden ser otorgadas por el usuario.

3.5. Pestaña *Test case*

En esta pestaña se muestran los resultados de correr el sistema en un corpus de 1000 documentos de *eHealth* con consultas anotadas.

Se muestra el texto de cada consulta, el modelo utilizado, así como las metricas asociadas a esta tupla.

Estas consultas ya traen etiquetados los documentos relevantes asociados. Es por eso que han servido para medir la eficacia y la correctitud del sistema.

4. Funcionamiento

El sistema basa su funcionamiento en los modelos Vectorial y **LSI**. Para crear el sistema para un directorio se realizan diferentes pasos de los cúales se hablará a continuación, así como lo que hay detrás de cada funcionalidad que presenta la aplicación.

4.1. Preprocesamiento

El preprocesamiento es una etapa clave para el fucionamiento del **SRI**. Este se realiza tanto a los documentos indexados como a las diferentes *querys*, ya que la realización de este ha demostrado tener efectos muy positivos a la hora de la recuperación.

En este preprocesamiento se realizan tres tareas importantes:

- *Stopwords*
- Eliminación de *Stopwords*
- Lematización

EL primero consiste en tokenizar cada elemnto que está en nuestro directirio ya sean palabras, signos de puntuación, es decir simbolos del idioma. Esto es útil ya que permite remover elementos innecesarios para la recuperanción. En este punto también se usan expresiones regulares para limpiar el texto. Todo esto con el objetivo de queadarse solo con palabras.

Luego se pasaa a la eliminación de stopwords para esto se cargan dos listas de stopwords, una en español y otra en inglés ya que el sistema es independiente del idioma dado los modelos implementados. Dado estas listas simplemente se hace un filtro a las palabras y se eliminan las que estan en estas.

Finalmente se pasa a la lematización, este proceso también se realiza para ambos idiomas, usando dos diccionarios donde para cada palabra se reduce a su signieficado más basico es decir las formas verbales a sus verbos, o los plurales a singulares, en general analiza la morfología de las palabras llevándolas a su lexema.

Ya realizado este proceso los documentos están listos para comenzar a crear el modelo.

4.2. Creación del Modelo

El modelo principal implementado es el **LSI**, usando la biblioteca *Gensim*. Esta biblioteca sigue forma de crear el modelo mostrada en [2], pero de forma óptima ya que se trabajan con matrices de grandes dimensiones. Para lograr esto *gemsim* calcula una matrix de **TF-Idf**, que es una matrix término-documento, y luego mediante la descomposicion SVD reduce dimensiones de esta captando la información latente. Esto se explica en la teoría mostrada en [1,2].

Esto ocurre en el metodo *savelsimodel*, aquí también se salvan en disco los modelos, para cargarlos si se vuelve a elegir este directorio. Si se quiere generar un modelo nuevo de be eliminar los archivos del modelo creados en ese directorio.

El modelo Vectorial no es más que la matriz que hablamos anteriormente de **Tf-Idf**, que puede tener enormes dimensiones dado el vocabulario del conjunto de documentos. Este tambien se salva en disco. Junto con la el modelo en si que es una clase de *sklearn* que se llama *TFidfVetoraizer* que contiene toda la información necesaria para transformar las consultas echas por el usuario al espacio de los documentos indexados.

4.3. Búsquedas

Ya teniendo los modelos cada ves que el usuario realiza una consulta este texto pasa al método *searchquery* donde se preprocesa la consulta por los mismos pasos que se procesaron los documentos y luego estase lleva al espacio del modelo, es decir de los documentos indexados. Ya en el mismo espacio se usa el coseno como medida de similitud para realizar poder generar un ranking con respecto a qué documentos son más parecidos a la consulta y así poder devolverlos al usuario.

4.4. Retroalimentación

El proceso de retroalimentación se realiza si el usuario lo desea marcando dado el resultado de una consulta los documentos recuperados relevantes para él. Esto pasa al método de *addretrofeeddata*, donde se recalcula los pesos para el vector consulta siguiendo el algorítmo de *Roccio*. Posteriormente se guarda un registro en forma de diccionario donde para cada consulta se guarda el vector consulta optimisada dado la selección del usuario. Lego cuando el usuario realiza de nuevo la misma consulta se busca la similitud con este vector optimisado guardado lo cúal mejora la recuperación ya que los pesos son ajustados de manera que se acerque más a los vectores relevantes y se aleje de los irrelevantes permitiendo la entrada al ranking de nuevos vectores que pueden ser relevantes.

4.5. Anotación

Para lograr la anotación, cada ves que el usuario en guarda un cojunto de documentos relevantes para una consulta específica, esto se guarda como un *json* llamado *relevantes*. Así se va creando un conjunto de prueba que luego en la vista de búsqueda se pueden hacer las consultas que ya esten en este diccionario y así dado lo que recupere el sistema poder calcular las diferentes métricas y evaluar el sistema.

5. Resultados

En la carpeta **screenshots/** se muestran una serie de capturas de pantalla que muestra la aplicación funcionando. En la imagen *3.png* se muestra el resultado de la consulta 'perro' con el modelo de **Semántica Latente**.

Como se puede apreciar, en primer lugar aparece un documento llamado *Canis Lupus Familiaris*, este es el nombre científico de esa raza, así que es un muy buen resultado.

El resto de los resultados tratan temas muy relacionados con la consulta, esto es así cuando se mencionan razas de perros, categorías de animales o elementos relacionados con el cine y el animal.

En la imagen *4.png* se muestra una comparación de ambos modelos con la misma consulta. Es fácil percatarse de que hay más ruido en los resultados del

modelo vectorial. La banda 'Buldog', ni 'Caballeros de la Quema' ni 'Enviroment Canada' son resultados relevantes para esta consulta.

Con respecto al corpus de prueba se obtubo un promedio de precisión de hasta 45 % en el caso de **LSI** y 49 % en el caso del modelo vectorial. Este es un gran resultado, teniendo en cuenta que estos modelos no son adecuados para este tipo de corpus donde hay tan pocos documentos, tan parecidos y con un vocabulario tan rico. Sobre todo en el caso de **LSI**, se ve afectado por la longitud de las consultas, ya que al ser tan largas no queda claro para el sistema el concepto al que esta asociada, sino que se ve como una mezcla de varios de estos conceptos y queda como resultado una consulta borrosa que en realidad no pregunta nada en concreto sino un poco de varios conceptos.

Referencias

1. *Latent Semantic Analysis. De la Osa, Daniel* 2019.
2. *Latent Semantic Indexing. Alvarez, Jose L.* 2019.