Implementación de un Sistema Recuperación de Información utilizando Redes Neuronales

Andy González Peña, Juan Eray Álvarez Hernández

MATCOM, Universidad de la Habana

Abstract. Este trabajo recopila los puntos claves que conforman la implementación de un sistema de recuperación de información aunque no pretende proponer la mejor solución pero si una funcional para la aplicación de los modelos de redes neuronales en la teoría de los sistemas de recuperación de información.

1 Introducción

Comenzaremos describiendo la estructura de la implementación, enumerando sus dependencias y definiendo la entrada y salida del programa, con ello se va explicando su funcionamiento de manera general.
Luego para profundizar, subdividiremos las distintas tareas en cuatro módulos de los cuales se argumentan, de igual manera, su funcionamiento y estructura demostrando el cumplimiento de los puntos requeridos en cada uno de ellos.
Finalmente, se enumeran ejemplos de acuerdo con los resultados obtenidos que muestran los puntos fuertes y débiles de la implementación con el propósito de darle continuidad al proceso de refinamiento en el que actualmente se encuentra.

2 Estructura

El proyecto se prefiere ver como un todo y aun así consta de cuatro módulos: interfaz de usuario, procesamiento de texto, índice y modelado, todos implementados utilizando *Python* (v3.6) con entradas y salidas sobre documentos de tipo *JSON* que definen su comportamiento en cada instancia de la ejecución.
Cada módulo existe en proceso diferente del sistema operativo y, únicamente, se relacionan mediante accesos al disco duro para consultar los archivos de entrada y salida, para ello se creó un sistema de notificaciones.
Todo fichero *JSON*, como es conocido, son básicamente diccionarios de llave-valor, entonces cada módulo cuando emita una salida escribirá la estampilla de tiempo que le corresponde logrando de esta forma que el módulo que utilice ese fichero como entrada sepa que ha ocurrido un cambio al mismo y sea capaz entonces de efectuar sus operaciones.
El orden de la comunicación es definido según su necesidad, en primera instancia se encuentra el módulo de la interfaz de usuario que emite como salida la propia entrada del módulo de modelado y espera como entrada la salida del mismo. Mientras los módulos de procesamiento de textos e indizado se comportan de igual manera para la modelación no se comporta así como en un proceso plano, sino que se divide en tres instancias enviando y recibiendo de los otros módulos hasta que logra emitir la salida concisa que fue requerida por la interfaz de usuario.
En las secciones correspondientes a cada uno de ellos se discutirán los aspectos de su implementación y se definirán, de manera explícita como se comportan para emitir una salida de

datos consecuente a su situación.
Mientras, de manera general, se define la arquitectura del software según la interrelación existente presente en cada par como indica siguiente figura.

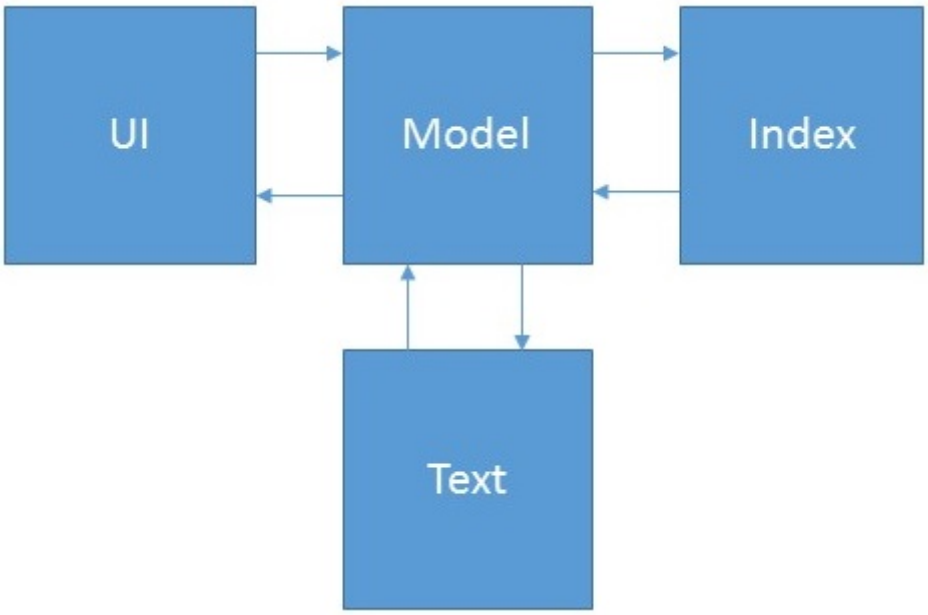

Fig 1. Interrelación de los módulos

2.1 Módulo Interfaz de Usuario

Es una interfaz web minimalista que cumple con los estándares de diseño para el cual el usuario solamente podrá interactuar según el sistema indique.

Dependencias Creada en *Python* utilizando *Flask* como *framework* en el que las plantillas renderizadas deberán ser interpretadas debido al uso de *cookies* en su definición, entonces se requiere del módulo denominado *jinja* para que actúe como intérprete.
Mientras, el diseño visual utiliza las clases definidas en *Bootstrap* y los *scripts* correspondientes, donde a ellos se le ha añadido uno propio para definir un reloj en la esquina superior derecha que ayuda a visualizar el tiempo transcurrido entre procesos.

Estructura Consta de cuatro rutas solamente y un enlace externo. En la barra de navegación al tope de la página están enumeradas. Estas son: ruta raíz que, en caso de ser iniciada la aplicación, iniciarár las *cookies* y otras variables necesarias para la aplicación, y en otro caso, simplemente redireccionará hacia la ruta **index**, en ella se encuentra el grueso de la aplicación de lo cual hablaremos a continuación, otra ruta es el "acerca" del proyecto, en el se encontrarán documentos relacionados con el tema como este y, por último, la ruta de evaluación del sistema en el que se ha descrito casos de prueba con las medidas definidas según la teoría.

Funcionamiento En la página principal se encontrarán los botones que indican las posibles opciones, es decir, si por ejemplo, aún no se ha construido un modelo, solamente se podrá esperar una dirección del sistema de archivos para construir uno y entonces si ya existe, se podrá realizar una consulta.
Cada botón funciona según su nombre lo indica, pero dos de ellos destacan por encima de todos ya que en ellos es donde se define la salida vía *JSON* del sistema, los botones de construcción y consulta se definen como indica la siguiente figura.

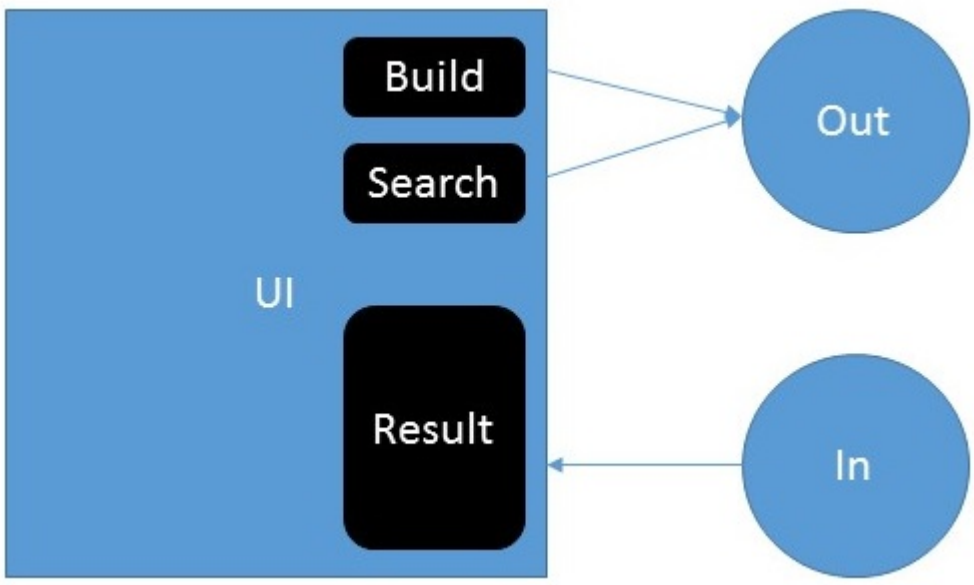

Fig 2. Entrada y salida de la interfaz de usuario

2.2 Módulo Procesamiento de Texto

Módulo sencillo que recibe texto plano y realiza las operaciones requeridas para devolver una lista de términos válidos para, posteriormente, ser indizados.

Metas alcanzadas El módulo realiza las operaciones requeridas para el procesamiento de texto. Estas son: *stemming* y la eliminación de *stopwords* y se logran con la aplicación de la librería especificada en la sección de dependencias.

Dependencias Creada en *Python* se apoya en su módulo *NLTK* que significa *Natural Language ToolKit* especializado en el procesamiento de lenguaje natural y, por consecuencia de texto, en él se encuentran distintos submódulos, entre ellos los necesarios son: *corpus* y *stem*.

2.3 Módulo de Índices

Es, básicamente, un diccionario en forma de árbol en el que cada llave es un nodo que apunta hacia otra llave en el diccionario hasta que encuentra un nodo hoja que apunta hacia un documento. Esta estructura de datos se crea en memoria y se crea según indica el módulo de modelado de tal forma que evita los cálculos matriciales como su definición exhibe y la propagación se realiza nodo a nodo según sus aristas de peso incidiendo sobre la activación de los mismos.

Limitaciones El costo en memoria asciende en gran pendiente debido a la definición combinatoria de los nodos y por tanto, está limitado a indizar relativamente pocos términos.

Dependencias Creada en *Python* no tiene ningún requerimento especial.

2.4 Módulo de Modelado

Está creado con la intención de modelar una red neuronal para la recuperación de información, siendo un grafo conexo y acíclico tal que a los efectos de implementación es un árbol.
Entonces sean X_1, X_2 ... X_n los términos a indizar en nuestra red neuronal, luego decimos que las combinaciones de, a lo sumo, cinco términos conforman los nodos de la primera capa, la siguiente serán las combinaciones de cuatro términos y así sucesivamente hasta llegar a los términos unitarios que representarán la última capa de la capa oculta hasta llegar a los nodos documentos.
En la siguiente figura se puede observar lo que sucede para tres términos.

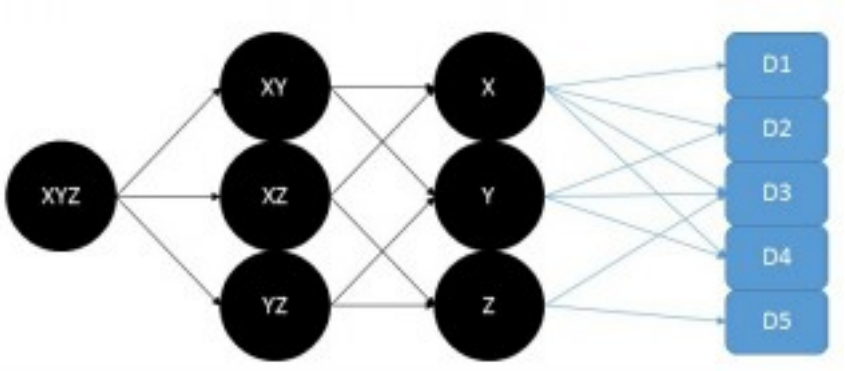

Fig 3. Ejemplo de definición del modelo

El modelo está implementado de tal forma que desde el punto de vista de la interfaz de usuario es un proceso unitario, es decir, en un ciclo de ejecución obtiene su entrada, envía y recibe lo necesario de los siguientes módulos, primero procesa textos, luego crea o busca índices y, finalmente, emite su salida hacia la interfaz. Esto significa que su apariencia es la siguiente:

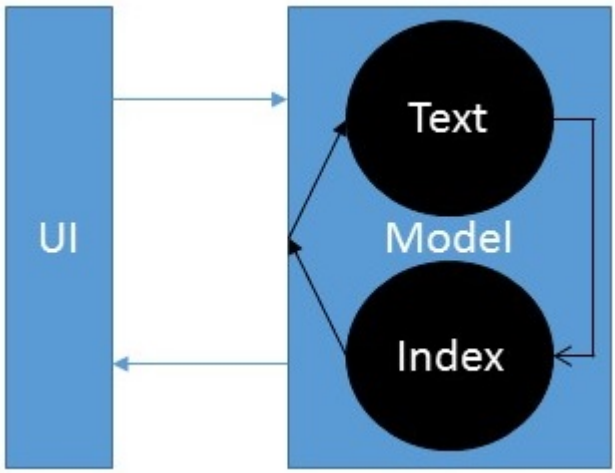

Fig 4. Apariencia en ejecución del modelo

3 Resultados

Adyacente al proyecto se encuentra un directorio de pruebas con pocos términos con el cual el sistema obtiene la máxima precisión tanto en recuperación como en el ordenamiento por relevancia, sin embargo a medida que aumenta la cantidad de términos a indizar por documentos tanto el tiempo en ejecución como el costo en memoria se disparan exponencialmente debido al costo factorial de la combinatoria de términos para construir los nodos.
Se encuentra que el sistema no es práctico debido a la gran limitante computacional sin embargo, es teóricamente exacto ya que recubre todos los posibles casos sobre los que un usuario promedio ejercería una acción mientras que se mantiene funcional.
La interfaz de usuario cumple con los principios de diseño antes mencionados y el procesamiento de texto utiliza herramientas altamente optimizadas, mientras la implementación del diccionario de índices se define según el modelo le indique, en lo que a funciones de diccionario implique es, también, altamente optimizada.

4 Conclusiones

Se encuentra un gran potencial en los sistemas de estas características, especialmente aplicando el modelo de redes neuronales, con lo cual queda pendiente optimizar la combinación de nodos o la subdivisión de consultas para obtener tales resultados para índices de términos mayores mientras que se ha cumplido con los requisitos de implementación a la vez que es teóricamente correcto y exacto con lo cual se obtienen los mejores resultados.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, 1999 Longman Publishing Co., Inc., Boston, MA, USA
2. Kasabov, Nikola.: Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering. MIT Press, 2002, Cambridge, MA, USA
3. Wilkinson, R., Hingston, P.: Using the cosine measure in a neural network for document retrieval. 1991. In Proc of the ACM SIGIR Conference on Research and Development. Chicago, USA.
4. Mitra, B., Craswell, N.: An Introduction to Neural Information Retrieval. 2018. Foundations and Trends in Information Retrieval, Cambridge, UK.
5. Kenter, T., Borisov, A., Van Gysel, C., Dehghani, M., de Rijke, M., Azarbonyad, H.: Lectures on Neural Networks for Information Retrieval. WSDN2018, University of Amsterdam, Netherlands.
6. Valle Vidal, C.: Lectures on Artificial Neural Networks. 2009. Universidad Técnica Federico Santa María, Santiago, Chile
7. Lopez, D., Navas, G.: Diseño y Construcción de una red neuronal de propósito general. 2007, Universidad Técnica Salesiana, Quito, Ecuador.