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.

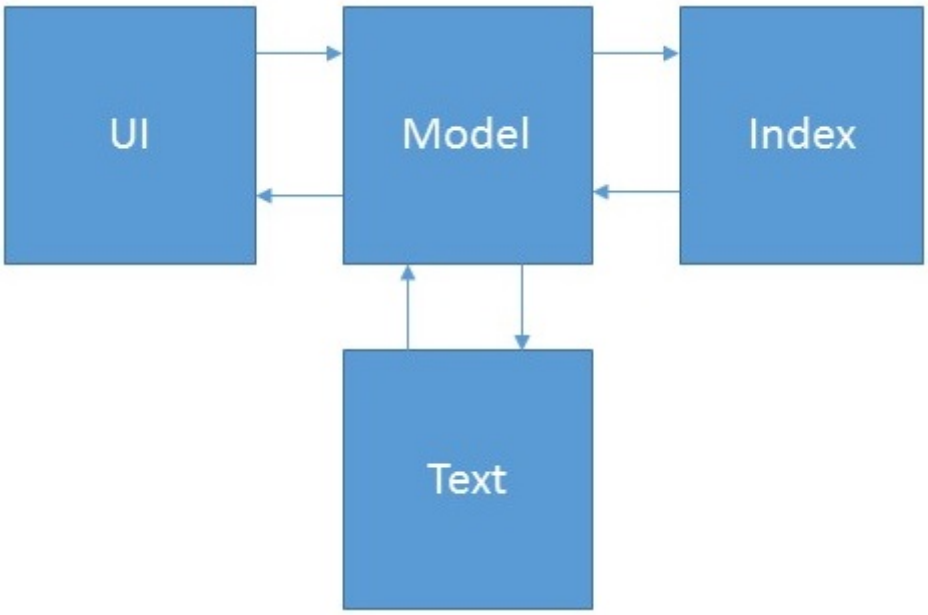

Fig 1. Interrelación de los módulos

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.

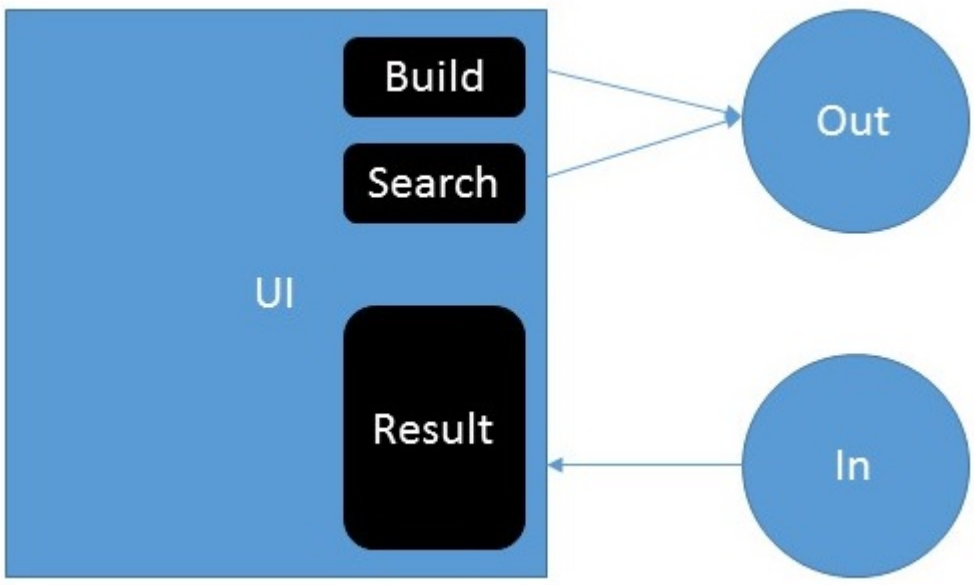

Fig 2. Entrada y salida de la interfaz de usuario

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 Modelado

Es el encargado de controlar el tráfico de datos necesarios para su computación. En primera instancia, hacia el módulo de procesamiento de texto realiza la conversión de contenidos de ficheros a texto plano o una simple cadena de caracteres. Al tener la lista de todos los términos a indizar los vectoriza en valores entre 0 y 1 y construye el modelo de redes neuronales.

Se define una red neuronal como una lista de matrices de pesos que representan las aristas entre nodos de capas adyacentes. Como propósito de generalización, como se verá adelante, se definen cuatro capas como se indica en la siguiente figura.

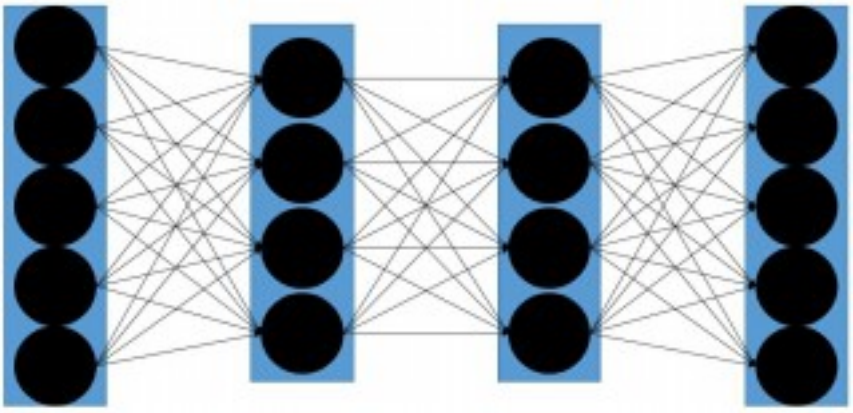

Fig 3. Ejemplo de definición del modelo

En el vector de la capa de entrada están los posibles términos de consulta, esos son todos los posibles términos a indizar y en el vector de la capa de salida se encuentran los documentos posibles. Entonces para el momento de creación las matrices de pesos constarán de vectores normalizados de números aleatorios y se mantiene, además, los valores reales de *tf* e *idf* que serán utilizados para obtener relaciones de pesos de aristas más exactas.
Una vez finalizada la creación del modelo también cumplirá con funciones vitales para el módulo de índices, como por ejemplo, la consulta no se realiza directamente del término sino que se hace una conversión de este hacia un vector de 0 y 1 que indican si la palabra existe en dicha consulta o no. Este vector será el parámetro de entrada de la red neuronal.
Finalmente, 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:

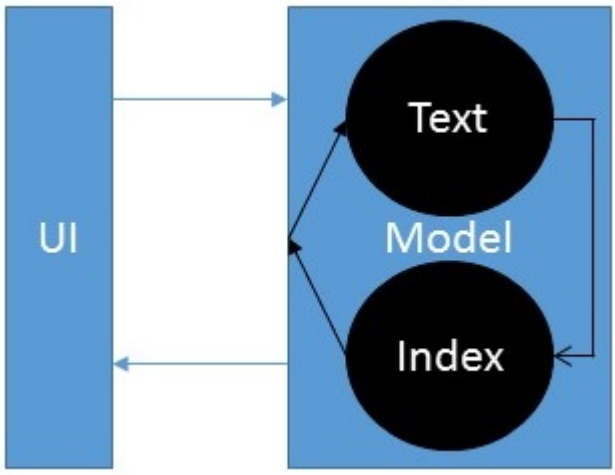

Fig 4. Apariencia en ejecución del modelo

2.4 Módulo de Índices

Este modelo tiene dos instancias fundamentales, cuando le es encomendada la tarea de finalizar la creación del modelo y cuando se realiza una consulta. Se les dice fundamentales ya que el resto depende directamente del funcionamiento de estas dos.
Cuando construye la red neuronal se tiene la matriz de pesos descrita en la sección anterior y se tienen los valores de tf e idf, entonces se quiere buscar la convergencia de la red, es decir, que dada una consulta sea capaz de retornar la probabilidad de relevancia a un documento. Es, de tal forma, que se asume que los pesos aleatorios predefinidos no son buenos valores para tal objetivo, entonces dados los otros dos parámetros del módulo, se obtendrán la relevancia sobre los documentos, por tanto, para toda consulta unitaria, es decir, el vector de entrada que contiene solo un 1 y el resto 0, se conocen sus valores de salida y se comienza la evaluación de nuestra red.
Se realiza el algoritmo hacia adelante, se evalúa el error y de acuerdo a la dirección del gradiente negativo se recalculan los pesos y se propaga el error en la red. Es importante destacar que la función de activación utilizada es la sigmoidal, tal que su derivada es la utilizada para la propagación. Esta es:
$\Delta w^{(t+1)}{}_{i,j} = lrate * Err_j * o_j * (1 - o_j) * o_i$
Luego de evaluar un número finito de iteraciones, se establece el modelo de redes neuronales y permite analizar consultas. Para este caso, no se calcula el error y se propaga para dar mejores resultados, sino que de manera directa se corre el algoritmo hacia adelante y se emite una salida. Las salidas del módulo son vectores de números entre 0 y 1, que representan la activación del documento con respecto a la consulta. Posteriormente, estas son procesadas y englobadas con los datos requeridos a la salida real del sistema como puede ser el nombre del documento.

Limitaciones El costo en memoria depende directamente de la cantidad de términos a indizar, ya que las matrices tienen medidas variables respecto a ello por ende para un gran número de términos puede ser altamente costoso y no solo en memoria ya que las multiplicaciones matriciales conllevan un gran uso de CPU y por tanto, de tiempo computacional.

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

3 Conclusiones

Un número estratosférico de aplicaciones tienen las redes neuronales y constantemente se suman nuevas. Mientras, de manera rudimentaria, creamos nuestra primera red neuronal con la cual obtenemos resultados satisfactorios. Se ha comprobado la teorá del álgebra lineal aplicada a este modelo y su similaridad con el modelo vectorial para inicializar sus valores de cómputo, además de la razón por la cual es llamada un paradigma propio dentro de la teoría de la recuperación de información.
Es importante destacar que este trabajo no termina aún, ya que como meta inmediata se encuentran temas como la retroalimentación dada las consultas para extender su propia definición de red neuronal.
En conjunto a la evaluación se ha comprobado que para mayores cantidades de términos y documentos el sistema obtiene mejores resultados en cuanto a su exactitud, y por tanto para valores pequeños es más propenso a retornar documentos que no son relevantes dada cualquier consulta.

References

1. Grus, J: Data Science from Scratch. O'Reilly Media Inc., 2015 Sebastopol, CA, USA.
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, 1999 Longman Publishing Co., Inc., Boston, MA, USA
3. Kasabov, Nikola.: Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering. MIT Press, 2002, Cambridge, MA, USA
4. 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.
5. Mitra, B., Craswell, N.: An Introduction to Neural Information Retrieval. 2018. Foundations and Trends in Information Retrieval, Cambridge, UK.
6. 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.
7. Valle Vidal, C.: Lectures on Artificial Neural Networks. 2009. Universidad Técnica Federico Santa María, Santiago, Chile
8. Lopez, D., Navas, G.: Diseño y Construcción de una red neuronal de propósito general. 2007, Universidad Técnica Salesiana, Quito, Ecuador.