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.

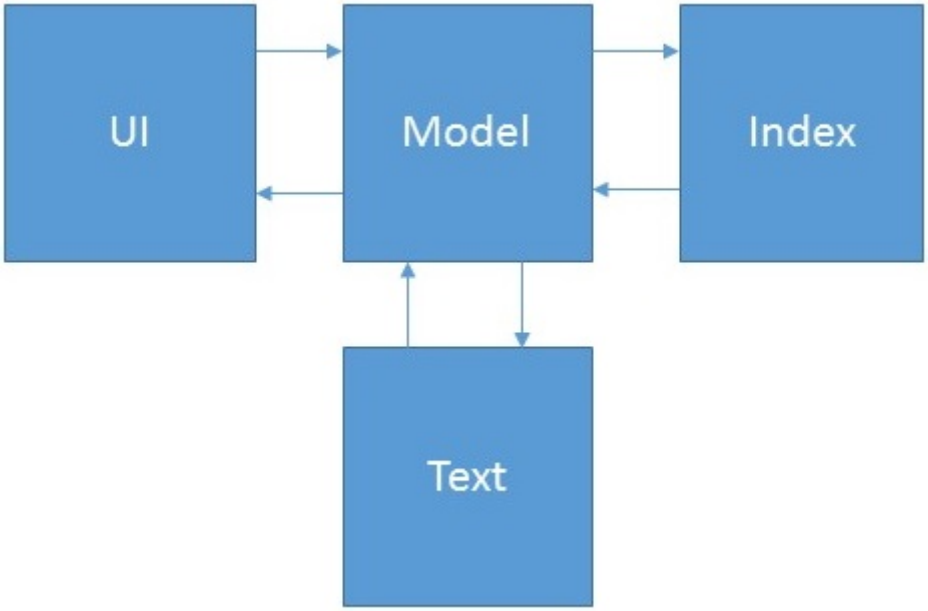

Fig 1. Estructura

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 entonces se requiere de

2.2 Perceptron

Entre los primeros modelos aplicables de redes neuronales tenemos el *perceptron*, el cual, en su visión simplista, definía que la función de activación neuronal, o actividad sináptica, era una función lineal donde su entrada, de manera general, serían números reales y las salidas se daban en binario. En la figura 1 se muestra la analogía de las tres neuronas del ejemplo de Hebb (y un *bias*).

Por definición está compuesto por dos capas de neuronas, una para la entrada y otra para la salida y se asumen completamente conectadas entre sí, mientras que los costos o pesos de las conexiones para la entrada ya están previamente fijados y luego a través de su algoritmo interno de aprendizaje calcula los pesos para su salida.

Un ejemplo de uso de *perceptron* es el de reconocer puntos en el plano bidimensional (x_1, x_2) donde existen dos posibles clases de soluciones, las cuales son linealmente separables y, por ende, luego de número finito de corridas a través del perceptron, este asegura su convergencia.

Sin embargo, exactamente ese fue uno de los problemas encontrados por Minsky y Papert en 1969, cuando lanzan un artículo demostrando las limitantes del perceptron. Si las clases de solución no eran linealmente separables entonces el modelo jamás convergerá.

2.3 Backpropagation

En consecuencia, para resolver la limitación de separabilidad lineal, surge el *perceptron* multicapas (MLP) el cual es, en principio, lo mismo que su propuesta anterior con la diferencia de que entre las capas de entrada y salida existen $n-$capas tal que juntas toman el nombre de capas ocultas.
Además, se pretende que las conexiones intermedias tengan el mayor número de conexiones posibles con el objetivo final de optimizar la respuesta, aunque en la práctica no siempre se pueda tomar de esa forma.

Sin embargo, este modelo no sería capaz de reutilizar el algoritmo original para el aprendizaje de un *perceptron*, al menos no hasta que en 1975 se lanza el algoritmo de *backpropagation*.

Para su aplicación, las neuronas del MLP se consideran en un estado constante de entradas y salidas, con una función no lineal de activación. Mientras se asume un error E para un ciclo determinado, se puede aplicar la regla del gradiente descendiente para encontrar la conexión óptima entre los pesos $w_{i,j}$ con lo cual se asegura que luego de número finito de ciclos el error E será suficientemente pequeño.

Entonces, para un ciclo $t+1$ la variación del peso de un nodo $\Delta w_{i,j}$ se calcula de la siguiente manera:

$$\Delta w_{i,j}(t+1) = lrate * Err_j * o_j * (1 - o_j) * o_i$$

Donde Err_j es el error entre el valor deseado en la salida y_j y el valor o_j producido por la neurona j, lo cual puede ser expresado como $Err_j = |y_j - o_j|$.

En cada ciclo (iteración, época) el algoritmo consta de dos fases:

1. Hacia adelante es cuando las entradas pasan a propagarse hasta llegar a las neuronas de salida y computar un respuesta.

2. Hacia atrás es cuando el error es calculado y no cumple con los requisitos de mínimo, entonces se propaga hacia los nodos intermedios provocando los cambios de peso $\Delta w_{i,j}$.

Mientras el error se propaga de vuelta, el Err_i para un nodo i intermedio es calculado multiplicando todos los errores Err_j de todas las neuronas j adyacentes a la neurona i mediante su peso correspondiente $w_{i,j}$. El procedimiento se repite hasta que el error global sea suficientemente pequeño.

3 Modelo

En todo sistema de recuperación de información derivado de los modelos algebraicos clásicos, los vectores de documentos son comparados con los vectores de consulta para calcular su relevancia y por ende los términos de índices en documentos y consultas deberán ser comparados y pesados para computar tal relevancia.

Una red neuronal es una representación simplificada del cerebro humano utilizando grafos, donde los nodos se comportan como neuronas y las aristas como las conexiones sinápticas. En cada instante, un nodo es definido por su nivel de activación (función de su estado inicial y las señales que recibe como parámetro). Entonces, dependiendo de su nivel de activación, un nodo A podría emitir una señal a otro nodo adyacente B. La fuerza de la señal en el nodo B depende directamente del peso asociado a la arista que une los nodos A y B.

En el proceso de hacia atrás descrito anteriormente, tales señales tienen relación directa con el error y por ende, mientras pasan épocas de ejecución tales señales se debilitan.

Para los nodos de consulta son asignados un valor de activación inicial de 1 (máximo), con los cuales pasan a la siguiente capa de nodos de la red neuronal y envían su señal de activación a los nodos correspondientes atenuadas según la normalización de los pesos de sus aristas $\overline{w}_{i,q}$.

$\overline{w}_{i,q} = \frac{w_{i,q}}{\sqrt{\sum_{i<t} w_{i,q}^2}}$

De la misma forma se calcularán los pesos para cada arista entre sus nodos correspondientes de las distintas capas de la red neuronal en cuestión. Finalmente, cuando la señal llega a un nodo documento d_j su nivel de activación está dado por:

$\sum_{i<t} \overline{w}_{i,q}\overline{w}_{i,j} = \frac{\sum_{i<t} w_{i,q}w_{i,j}}{\sqrt{\sum_{i<t} w_{i,q}^2}\sqrt{\sum_{i<t} w_{i,j}^2}}$

El cual coincide exactamente con la fórmula de relevancia del modelo clásico vectorial para la primera iteración de la red neuronal.

3.1 Definición Formal

Una red neuronal es un cuádruplo
$RN =< Q, D, F, R >$ donde:

$Q-$Conjunto de nodos de consulta donde todos tienen nivel de activación máxima y los pesos de las aristas correspondientes son las componentes del vector asociado a la consulta.

$D-$Conjunto de nodos de documentos con funciones de entrada y salida de señales en la cual los pesos de las aristas que les inciden son las componentes del vector asociado a la salida de la red neuronal.

$F-$Como derivado del modelo vectorial se enmarca en la teoría del Álgebra Lineal de vectores y sus operaciones sobre un espacio $t-$dimensional y, además, involucra la teoría de grafos, estos dirigidos y ponderados como parte de la propia teoría de redes neuronales.

$R-$Función de relevancia (ranking) enunciada anteriormente y la misma que en el modelo vectorial.

Como definición conjunta se puede definir como una función $F : \overline{x} \rightarrow R, F(x, \Theta)$ tal que Θ contiene los parámetros que permiten realizar un mapeo del objeto de entrada x y se devuelve otro objeto y.

3.2 Comparación

A continuación se presentan los principales aspectos que definen un sistema de recuperación puestos en comparación con el modelo de Redes Neuronales.

Framework Booleano - Teoría de Conjuntos y Álgebra Booleana
Vectorial - Espacio t-dimensional y Álgebra Lineal
Redes Neuronales - Teoría de Grafos, Espacio t-dimensional y Álgebra Lineal

Pesos Booleano - {0, 1}
Vectorial - $w_{i,j}$
Redes Neuronales - $\overline{w}_{i,j}$

Consultas Booleano - Expresiones Booleanas
Vectorial - Vectores de peso
Redes Neuronales - Nodos

Documentos Booleano - Vectores Binarios
Vectorial - Vectores de peso
Redes Neuronales - Nodos

Similitud Booleano - {0,1}
Vectorial - $\cos \alpha$
Redes Neuronales - $\cos \alpha$

Dependencia Booleano - No
Vectorial - No
Redes Neuronales - Si

Correspondencia Parcial Booleano - No
Vectorial - Si
Redes Neuronales - Si

Ranking Booleano - No
Vectorial - Si
Redes Neuronales - Si

4 Ejemplo

Para entender el funcionamiento de las redes neuronales, se presenta a continuación un ejemplo ilustrativo de un red neuronal dedicada al reconocimiento de vocales a partir de una imagen.

Primeramente, el sistema debe conocer las muestras de cada vocal parametrizadas:

A: 01110100011000111111100011000110001

E: 11111100001000011100100001000011111

I: 11111001000010000100001000010011111

O: 01110100011000110001100011000101110

U: 10001100011000110001100011000101110

Fijar como número de entradas de la red 35, puesto que son 35 los números de la matriz resultante de $7x5$ resultante de la parametrización y como número de salidas 5 ya que esa es la cantidad de vocales.

Note que en una primera instancia no es necesaria una capa oculta, de manera empírica se pueden ir añadiendo capas con el objetivo de optimizar resultados o simplemente disminuir el tiempo en ejecución antes de dar tales resultados. Además, se puede evitar el uso de neuronas auxiliares (bias) y se utiliza como función de activación la sigmoidal (aunque pueden ser muchas otras).
Se fijan los valores iniciales de $lrate$, M, E_{min} como marco de trabajo donde:
- $lrate$ puede tomar valores entre 0 y 1. Usualmente se les da un valor medio.
- M puede tomar valores entre 0 y 1. Se utiliza para asegurar que el algoritmo no se caiga en ciclos en ninguno de los pasos
- E_{min} se define como cota de ejecución y es usualmente un valor lo suficientemente pequeño tal que el resultado se considere una respuesta aceptable.

5 Conclusiones

El alza de nuevas de tareas de recuperación de información demanda replantear muchas de las métricas que consideramos correctas para evaluar la relevancia de un documento en un sistema dada una consulta y, a la vez, para otra situación ser totalmente lo contrario. Es, por ende, que los modelos de Redes Neuronales deberán no solo evolucionar desde el punto de vista computacional, es decir, en cuanto a temas de optimizacíon, sino que debe ser acompañado de numerosos aspectos que interactúan sobre ellos.
Está reflejado que a lo largo del tiempo, estos modelos han demostrado cuantiosos resultados para muchas ramas de la ciencia, con lo cual su importancia existe más allá del plano teórico, estando presentes en casi cada acción tecnológica que cada persona promedio realiza al menos una o dos veces al día y aún se encuentra en constante evolución.

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.