Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

Proyecto Terminal

SISTEMA CLASIFICADOR DE DOCUMENTOS DE PROYECTOS TERMINALES USANDO EL CONCEPTO DE MEMORIA ASOCIATIVA

Trimestre: 11-I

Alumno:

Juan Luis Ugalde Anaya
Matrícula: 204206008

Asesor:

Dr. Oscar Herrera Alcántara

México D.F., Diciembre de 2011

INDICE

Antecedentes

En las ciencias de la computación nos interesa crear modelos matemáticos que simulen la inteligencia humana, esto más que nada para facilitar ciertas funciones. Con base en esto, nos interesa diseñar y operar sistemas (software o hardware) que sean capaces de aprender y recordar conceptos e ideas abstractas [1]. La memoria humana se basa principalmente en las asociaciones; cuando intentamos recuperar una información, una cosa nos recuerda otra que nos recuerda todavía a otra, y así sucesivamente. Por ejemplo, los seres humanos somos capaces de reconocer el rostro de una persona, basta con sólo ver una parte de la cara. De esta manera, podemos recordar su nombre, la forma en la que la conocimos, etc.

En el proceso de memorización siempre intervienen dos fases:

Fase de asociación. Se produce la asociación de los objetos en cuestión el que queremos recordar (objetivo) y el que nos estimula al recuerdo (estimulante).

Fase de recuperación. Basta con alimentar a nuestra memoria con el objeto estimulante y el objetivo aparecerá automáticamente [2].

Lo que en realidad hace nuestra memoria es reconocer ciertas características mostradas por un objeto, lo que en el área de la Inteligencia Artificial se llama Reconocimiento de Patrones. Sin embargo, nuestra memoria no sólo sirve para recordar objetos, también nos ayuda en la clasificación de los mismos. Basta con aprender las características que distinguen a ciertos objetos de naturaleza similar y cuando uno de ellos se nos presenta, automáticamente sabremos a que clase pertenece [3].

Estos modelos son conocidos como memorias asociativas y se manejan en el área de inteligencia artificial. El concepto de memoria asociativa es un elemento cuyo propósito fundamental es recuperar patrones completos a partir de patrones de entrada que pueden ser alterados con ruido sustractivo, auditivo o mixto [3].Permite almacenar y recuperar información cuando sea necesario es decir es una red retroalimentada. A esto también se le llama memoria de direccionamiento por contenido.

El presente proyecto tiene como objetivo describir la metodología a través de la cual se realiza el reconocimiento de archivos de texto almacenados en formato PDF. El formato PDF es el tipo de extensión de las propuestas o proyectos terminales de computación ya que este formato brinda mejor presentación. Al momento de interpretar la información se usó un cambio de formato a **.txt** y escogíamos las palabras claves de cada documento que nos daban un vector de n-dimensiones que se utiliza como entrada de una red neuronal usada como memoria asociativa.

Las pruebas realizadas en este proyecto muestran la posibilidad de realizar el reconocimiento de archivos de texto usando bloques de $n = 164$ coeficientes representando cada uno de estos coeficientes a la entrada que alimentara a la red neuronal.

1. Introducción

La obtención de los archivos en formato PDF y pasarlos a TXT lo realice con una API del lenguaje java llamada "pdfbox-app-1.3.1" [4], cuando el archivo ya se extensión ahora si podemos manipular la información. Ya que la búsqueda de documentos se puede realizar de varias formas ya sea por el título, por el contenido por formato. En mi caso lo realice por contenido, ya que de un PT se le extrae los datos más importantes que abarque la mayor información posible. Una vez adquiridos los rasgos característicos del archivo de textos se requiere hacer búsquedas con empates parciales. El uso de una red neuronal artificial nos permite realizar este tipo de reconocimientos parciales. Lo que nos indica que son datos muy fáciles de aprender para dicha red, ya que las dimensiones son razonables.

En base a las consideraciones anteriores, puede establecerse el siguiente sistema de búsqueda (figura 1.1).

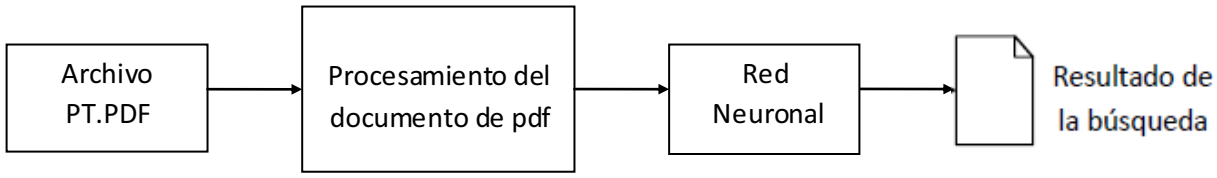


Figura1.1: Diagrama de bloques de la arquitectura del sistema de búsqueda de documentos mediante el uso de memoria asociativa.

La primera etapa consiste en extraer el texto de los Pts.de formato PDF. La segunda etapa consiste en convertir el formato de PDF a TXT para poder procesarlo y extrayendo un conjunto de rasgos característicos. En la siguiente etapa se utiliza una arquitectura de una red neuronal que tiene como entrada el conjunto de rasgos característicos obtenidos en la etapa anterior. Finalmente es sistema devuelve una lista de los documentos ordenados según su similitud con la consulta (documento ingresado).

1.1 Organización

A continuación se detalla la organización de este proyecto.

Capitulo 1. Adquisición de datos. Se creó una interfaz gráfica usando la biblioteca de clase Swing1 de Java para poder capturar el contenido del proyecto terminal y poder desglosarlo en cadenas de palabras más pequeñas para hacer un manejo más fácil.

Capitulo 2. Procesamiento de información. En esta fase la información (texto) se extrae los archivos en formato PDF, aplicando el anti-diccionario el cual contiene palabras claves que hablen de computación, almacenando la información filtrada en archivos planos de texto en .TXT .

Capitulo3. Red Neuronal, se implementara una red neuronal para la fase de entrenamiento con los datos obtenidos en la etapa anterior. Para implementar la red neuronal artificial se ha utilizado la herramienta Weka. Weka es una extensa colección de algoritmos de Máquinas de conocimiento desarrollados por la universidad de Waikato (Nueva Zelanda) implementados en Java. La licencia de Weka es GPL (GNU PublicLicense), lo que significa que este programa es de libre distribución y difusión. De esta manera, obtendremos una arquitectura de una red neuronal adecuada para lograr una buena capacidad de generalización.

Capitulo4. Evaluación, Se presenta los experimentos realizados para la evaluación del sistema y sus resultados.

Capitulo5. Creación de aplicación Software, en la que se representan todos los resultados de las consultas y en la que se podrá seleccionar un archivo de texto para poderlo visualizarlo o bien hacer más consultas.

Capitulo 6. Conclusiones y trabajo futuros, se realizara un análisis crítico al proyecto realizado y se sugieren futuros métodos de optimización para mejores resultados.

2. Prepocesamiento de archivos de texto

En algunos sistemas de búsqueda de archivos de texto se hace referencia al título del documento y no al contenido. Una manera simple de obtener este contorno es mediante el uso de valores que describen los intervalos del texto o documento.

Los cuales son:

- Quitar los acentos al texto.
- Aplicarle un documento llamado anti-diccionario.
- Convertir todas las palabras a minúsculas.
- Quitar los signos matemáticos ó todos los símbolos.

3. Red neuronal

Una red neuronal artificial (RNA por sus siglas), está constituida por elementos que se comportan de forma similar a una neurona biológica (neurona artificial). Estos elementos están organizados de una forma parecida a la que presenta el cerebro humano. Las tres partes fundamentales de una neurona son su entrada, su salida y su capacidad de procesamiento. Por lo tanto una neurona artificial es un modelo matemático que simula el comportamiento de una neurona biológica.

Uno de los modelos más conocidos es el que propuso Rosenblat el cual se muestra a continuación (Figura 3.1)

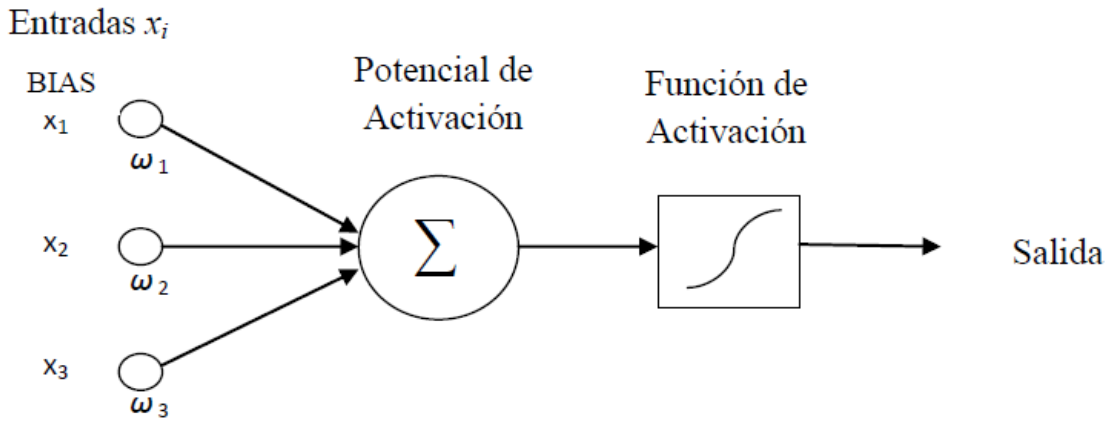


Figura 3.1. Modelo de Perceptrón propuesto por Rosenblat

A la conexión de varias salidas perceptrones usadas como entradas de otros se le conoce como red de perceptrones. Por lo que la salida de esta red neuronal está dada por la evaluación en cascada de la salida de varios perceptrones, dando lugar a un perceptron que dará la capa de salida (Figura 3.2).

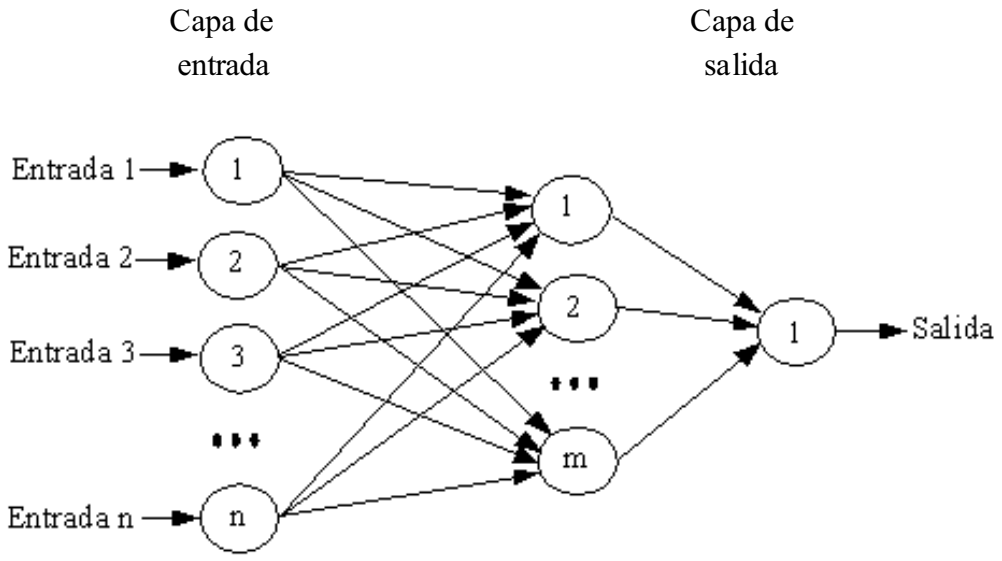


Figura 3.2. Modelo de red de Perceptrones con una salida.

Para trabajar con una red neuronal existen varios procesos.

- Entrenamiento. Consiste en colocar nuevos valores en la entrada de la red neuronal para dar lugar a un ajuste de los pesos sinápticos y de esa manera aprender esos nuevos valores.
- Aprendizaje. Consiste en minimizar el error entre la salida deseada y la obtenida ajustando el valor de los pesos sinápticos.
- Prueba. Consiste en realizar pruebas con valores que no causan ajuste en los pesos sinápticos y así poder comprobar una salida optima
- Generalización. Consiste en colocar valores en la entrada de la red neuronal que no se usaron para el entrenamiento y obtener salidas aceptables.

4. Evaluación

Esta etapa se divide en varias fases, las cuales se describen a continuación.

Fase I. Obtención de rasgos característicos.

1 Creación de una base de datos llamada proyectosBD que contenga una tabla llamada pt_diccionario en la que se almacenaran los resultados obtenidos (ver sección instalación de software).

2 Iniciamos el proceso leyendo el un archivo en formato .PDF y convertirlo en formato .TXT.

3 Extraemos la información del directorio PTPrimas en el cual los archivos ya están en .txt y aquí ya se les aplica el anti-diccionario el cual es quitarle al documento todos los (adjetivos, vocales, preposiciones), se cambian a minúsculas todas las palabras, se les quita los acentos.

4 Aquí se les aplica la función tokenizer que hace que se separen todas las palabras para almacenarlas en un archivo llamado concatenación ahí se almacenan los n documentos para poderles aplicar el diccionario.

5 El diccionario no es más que las palabras claves que se pueden repetir en todos los PTs y que hable de computación que en mi caso fue del tamaño de 164 palabras.

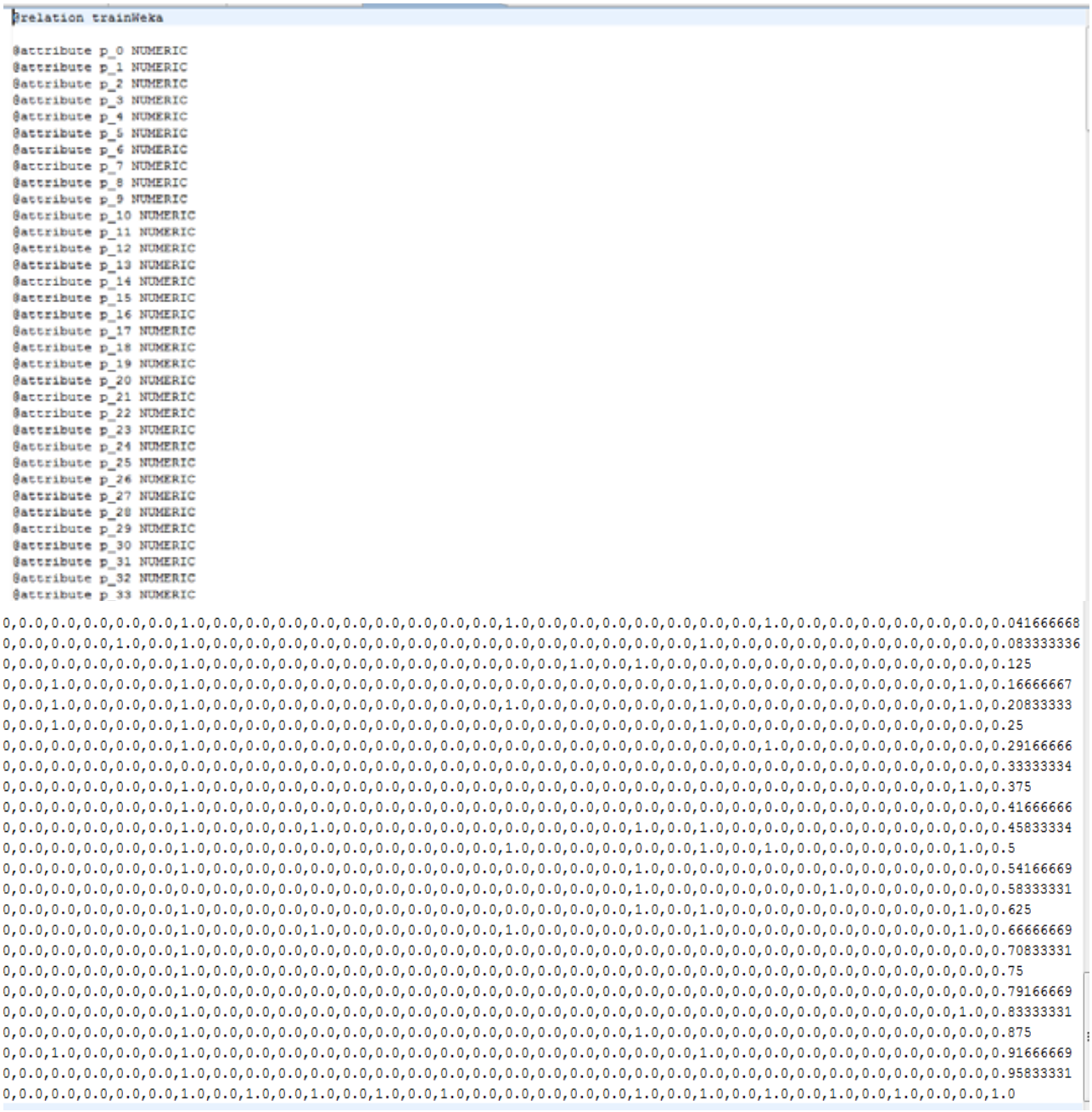
6 Aquí el archivo final se llama matriz y aquí se almacenan puros 1's y 0's y es el que será enviado a la base de datos, los vectores son de tamaño 164 más un **id** que nos va a servir como identificador de cada archivo de texto.

Ver código 1, 2, 3 en la sección códigos fuente. En el que se muestra el código completo correspondiente a esta fase.

Fase II. Recuperación de información

1. Accesar a la base de datos para obtener el máximo y el mínimo por columna de la tabla potencias de cada $\omega = 165$ coeficientes calculados, esto con el objetivo de poder realizar el proceso de normalización.
2. Se crea la cabecera de un archivo con extensión arff (Formato de Archivo de Atributo- Relación) llamado trainWeka.arff en cual usaremos para el entrenamiento de la red neuronal.

Y de esta manera poder completar la sección correspondiente a los datos en el archivo trainWeka.arff (Ver figura 4.1).



```
@relation trainWeka

@attribute p_0 NUMERIC
@attribute p_1 NUMERIC
@attribute p_2 NUMERIC
@attribute p_3 NUMERIC
@attribute p_4 NUMERIC
@attribute p_5 NUMERIC
@attribute p_6 NUMERIC
@attribute p_7 NUMERIC
@attribute p_8 NUMERIC
@attribute p_9 NUMERIC
@attribute p_10 NUMERIC
@attribute p_11 NUMERIC
@attribute p_12 NUMERIC
@attribute p_13 NUMERIC
@attribute p_14 NUMERIC
@attribute p_15 NUMERIC
@attribute p_16 NUMERIC
@attribute p_17 NUMERIC
@attribute p_18 NUMERIC
@attribute p_19 NUMERIC
@attribute p_20 NUMERIC
@attribute p_21 NUMERIC
@attribute p_22 NUMERIC
@attribute p_23 NUMERIC
@attribute p_24 NUMERIC
@attribute p_25 NUMERIC
@attribute p_26 NUMERIC
@attribute p_27 NUMERIC
@attribute p_28 NUMERIC
@attribute p_29 NUMERIC
@attribute p_30 NUMERIC
@attribute p_31 NUMERIC
@attribute p_32 NUMERIC
@attribute p_33 NUMERIC
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.041666668
0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.083333336
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.125
0,0.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.16666667
0,0.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.20833333
0,0.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.25
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.29166666
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.33333334
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.375
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.41666666
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.45833334
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.5
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.54166669
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.58333331
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.625
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.66666669
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.70833331
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.75
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.79166669
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.83333331
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.875
0,0.0,1.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.91666669
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.95833331
0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,1.0,0.0,0.0,1.0
```

Figura 4.1. Formato de archivo trainWeka.arff generado.

Ver código 4 en la sección códigos fuente. En el que se muestra el código completo correspondiente a esta fase.

Fase III. Implementación de red neuronal

1. Para la implementación de la red neuronal se utilizo la API (interfaz de programación de aplicaciones) WEKA, esta API contiene un modelo de red neuronal (MultilayerPerceptron), que nos permite realizar el entrenamiento de la misma. Para realizar el entrenamiento de la red neuronal, se necesita el archivo generado en la fase anterior (trainWeka.arff) y definir una arquitectura adecuada para un óptimo entrenamiento.

   En este caso con 500 épocas se logro minimizar el error de entrenamiento. Los datos obtenidos por WEKA se muestran en la tabla II:

Correlation coefficient	1
Mean absolute error	0
Root mean squared error	0
Relative absolute error	0.0152 %
Root relative squared error	0.0165 %
Total Number of Instances	24

Tabla II. Resultados generados por WEKA al finalizar el entrenamiento para 24 PTS.

En la tablaIII se muestra la arquitectura de la red neuronal entrenada así como sus parámetros de entrenamiento y en la figura 4.2 se muestra la red neuronal generada por WEKA.

Capa	Neuronas	Función de activación	Tasa de aprendizaje	Momentum
Entrada	164	Lineal	0.1	0.1
Oculta 1	5	Sigmoidal	0.1	0.1
Salida	1	Lineal	0.1	0.1

Tabla III. Arquitectura de red neuronal

2. Al finalizar el entrenamiento de la red neuronal se crea un archivo llamado ReporteWeka.txt el cual contiene el modelo de la red neuronal indicando los pesos correspondientes a cada conexión entre las neuronas, en seguida en otro archivo llamado PesosWeka.txt se almacenan únicamente el valor de los pesos que se encuentran el ReporteWeka.txt.

Ver código 5 en la sección códigos fuente. En el que se muestra el código completo correspondiente a esta fase.

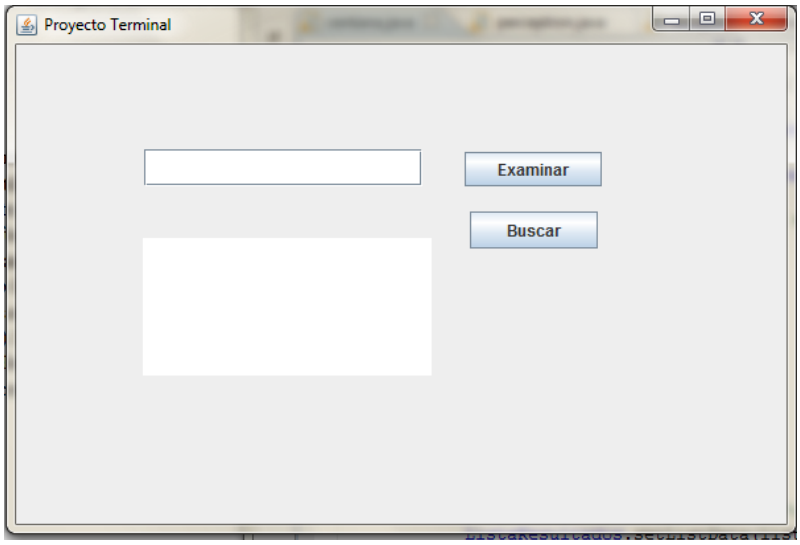
Fase IV. Identificación de arquitectura de la red neuronal

1. Se realiza la interpretación de los pesos almacenados en el archivo PesosWeka.txt para tener una arquitectura correcta y lograr una buena generalización. Para esto se hicieron pruebas usando datos aleatorios del archivo de entrenamiento de la red neuronal obteniendo como salida una buena generalización. La lectura de los datos se realiza mediante un archivo en el que se simula, únicamente para esta fase y a modo de prueba, una propuesta.txt. El nombre de este archivo es propuesta.txt.

Ver código 6 en la sección códigos fuente. En el que se muestra el código completo correspondiente a esta fase.

5. Aplicación software

Se creó una aplicación software que permite realizar una selección de un archivo en f**ormato .PDF**. Este archivo seleccionado es el que se utiliza en la fase IV del capítulo anterior con la diferencia que ahora se trata de un archivo diferente a los utilizados para el entrenamiento de la red neuronal. Para la realización de esta aplicación fue necesario utilizar 3 clases implementadas en lenguaje JAVA, la primera de ellas se llama ventana.java (Ver código 7 en la sección códigos fuente), en la que se crea una interfaz gráfica usando la biblioteca de clase *swing*, la cual está integrada por 2 botones, un campo de selección del archivo a procesar y un área en la que se mostraran los resultados de la consulta (ver figura 5.1). La función del botón Examinares abrir un explorador de archivos para que el usuario pueda seleccionar el archivo en formato **.PDF** a procesar. El botón Buscar, realiza el procesamiento necesario al archivo seleccionado para poder mostrar los resultados en el área correspondiente.



Una segunda clase usada para el funcionamiento de la aplicación es invocada desde la clase ventana.java una vez seleccionado el archivo y pulsando el botonBuscar, esta clase es llamada procesar.java, en esta clase se realiza el procedimiento descrito en el Capítulo 4 en la Fase I siguiendo los pasos 2, 3, 4, 5 y 6, en este último paso se genera un archivo llamado propuesta.txt en el que se almacenan los resultados obtenidos al finalizar el procesamiento del archivo en cuestión.

Ver código 7 en la sección códigos fuente. En el que se muestra el código completo correspondiente a esta fase.

Por último esta clase invoca a la clase pesosWeka.java utilizada en la Fase IV del capítulo 4, la cual nos regresa una lista de 5 archivos en formato .PDF ordenadas según su similitud con la consulta.

6. Conclusiones y trabajos futuros

En este trabajo se presento una metodología en la que se procesan archivos de texto en formato PDF y a partir de este se realiza un reconocimiento del mismo. Se observo que al convertir a textos planos (TXT) se puede hacer un manejo más fácil de la información ya que se manipula y con los diferentes métodos que elabore, se pudo obtener los rasgos más específicos de un documento, de esta manera se hizo más pequeño, sin perder la información del PDF. Con esto se puede hacer la consulta y mostrar los documentos más parecidos.

Algunas pruebas que se realizaron fueron con los mismos PT´s que están almacenados en nuestra base de datos, en la tabla IV siguiente se muestra los resultados obtenidos en cada prueba.

PT´s	Resultados	PT´s	Resultados
20090-01a	1,2,3,4,5	20090-14b	14,15,16,17,19
20090-02a	2,3,4,5,6	20090-15	15,16,17,19,20
20090-03a	4,5,6,7,8	20090-16a	16,17,19,20,21
20090-04a	5,6,7,,8,9	20090-17a	17,19,20,21,22
20090-05a	5,6,7,8,9	20090-19a	19,20,21,22,23
20090-06a	6,7,8,9,10	20090-20a	20,21,22,23,24
20090-07a	7,8,9,10,11	20090-21a	21.22.23.24.25
20090-08a	8,9,10,11,13	20090-22a	22,23,24,25,PTTarareo
20090-09	9,10,11,13,14	20090-23a	23,24,25,PTTarareo,1
20090-10a	10,11,13,14,15	20090-24a	24,25,PTTarareo,1,2
20090-11	11,13,14,15,16	20090-25a	25,PTTarareo,1,2,3
20090-13a	13,14,15,16,17	PTTarareo	PTTarareo,25,24,1,2

Tabla IV Resultados obtenidos en la prueba del software

Cabe mencionar que los en color amarillo fui por que al ejecutarlos la red no predijo el PT original y nos mostro los mas parecidos.

Realice otra prueba mas con algunos PT´s (4) modificados y también incluí el mío para saber si mi aplicación funcionaba los resultados se muestran en la tabla V.

PT´s - Modificados	Resultados
25	24,25,1,2,3
09	8,9,10,11,13
10	9,10,11,13,14
PTFinal	PTTarareo,25,24,1,2
Proyect	16,17,19,20,21

Tabla V. Resultados obtenidos con PT´s modificados.

Por eso concluyo que se cumplieron los objetivos planteados en mi propuesta y el mejor resultado es el que se muestra en color amarillo ya que al ingresar mi PTFinal me mostro como resultado, el PTTarareo ya que son muy similares .

En trabajos futuros se sugiere que el diccionario lo realicen más sofisticado para poder quitar pronombres, similitud de antónimos y poner más términos que hablen de computación para poder tener un manejo más viable de la información y aplicarle sus vecinos más cercanos y poder elaborar una matriz de correlación.

Las clases a continuación se muestra la clase principal llama a los métodos de otras clases que más adelante se muestran sus códigos.

Códigos fuentes

1. **Clase Principal**

Clase que realiza la conversión de formato de PDF a TXT de la carpeta PTs y los almacena en PTPrimas, para su fácil manipulación.

```
//import java.io.BufferedReader;
import java.io.File;
import java.io.FileWriter;
//import java.io.IOException;
//import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.StringTokenizer;



public class Principal {
```

```
        /**
         * Autor Juan Luis Ugalde Anaya
         * Función principal que invoca los métodos
         *
         */
        public static void main(String[] args) {

                String texto = null;
                pdf archivoPDF = new pdf();

                File dirpdf = new File("PTs");
                File dirtxt = new File("PTPrimas");
                String[] ficheros = dirpdf.list();
                String[] ficherostxt = dirtxt.list();
      File txt = new File("Concatenacion/union.txt");

      txt.delete();//Borra el archivo union.txt

                Procesar archivo = new Procesar();
                //Carga el antidiccionario
                String antiDiccionario[] = archivo.cargarAntiDiccionario();

                if (ficheros == null)
                System.out.println("No hay ficheros en el directorio especificado");
       else {
           for (int i = 0; i < ficheros.length; i++) {
                   String file =  ficheros[i];
                        System.out.println("Archivo a procesar: "+file);
                        // Se Extrae el texto del archivos pdf

                        StringTokenizer st = new StringTokenizer(file, ".");

                        String nombreArchivo = st.nextToken();
                        nombreArchivo = "PTPrimas/"+ nombreArchivo+".txt";  // nombre
de nuevo archivo txt
                        //System.out.println("nombre: "+nombreArchivo);

                        //Se extrae el texto del pdf y se almacena en texto
                        texto = archivoPDF.getTexto("PTs/"+ficheros[i] );
                        System.out.println("Texto extraido exitosamente ");

                        //System.out.println(texto); //Imprimir texto de pdf


                        /* Extraer cada archivo PDF a procesar separando la extension
cambiandosela
```

```
                        * por txt para el almacenamiento de cada PDF en su propio txt con el
mismo nombre
                        * */

                       // Carga el texto extraido del archivo PDF
                              String documentoBase[] =
archivo.cargarTexto(texto,texto.length());

        // Aplicar anti-diccionario
archivo.aplicarAntidiccionario( documentoBase, archivo.nPalabrasTexto, antiDiccionario,
archivo.nPalabras);

// ====================     PT´s Primas  ================================


                String TotalPalabras1[] =   archivo.cuentaPalabras(); // Total de palabras
filtradas para el archivo base

                              guarda (nombreArchivo,TotalPalabras1,0,0);
                              guardatxt(TotalPalabras1);

                              System.out.println("Archivo  "+nombreArchivo+"
Procesado");
                              System.out.println();
          }// fin for

          // Concatenar archivos txt

          if (ficherostxt == null)
                System.out.println("No hay ficheros en el directorio especificado");
        else {
                for (int i = 0; i < ficherostxt.length; i++) {
                        System.out.println(ficherostxt[i]);


                        }//Fin for
        }
     }//fin else

        cuentaUnion totalPalabrasUnion = new cuentaUnion();

String palUnion[] = totalPalabrasUnion.nPalabrasUnion("Concatenacion/union.txt");
                        // Cuenta cocurrencias de archivo union.txt

        guarda("Concatenacion/unionFiltrado.txt",palUnion,1,totalPalabrasUnion.cont);
```

```
}

        private static void guardatxt(String[] palabras) {

                FileWriter fichero = null;
        PrintWriter pw = null;
        File directorio = new File("Concatenacion");

                try
                 {
                  fichero = new FileWriter("Concatenacion/union.txt",true);
                  pw = new PrintWriter(fichero);
                    for(int j=0; j < (palabras.length);j++){

                                pw.println(palabras[j]);
                    }
                    //pw.println(b[8]);
                   fichero.close();
                 } catch (Exception e) {
                   e.printStackTrace();
                }
}

        private static void guarda(String nombre, String palabras[],int op,int cont) {

        FileWriter fichero = null;
        PrintWriter pw = null;

        try
          {

           fichero = new FileWriter(nombre);
           pw = new PrintWriter(fichero);

           if(op == 0){
             for(int j=0; j < (palabras.length);j++){
                      pw.println(palabras[j]);
             }
           }
           else if(op == 1){
                    for(int j=0; j < (cont);j = (j+2)){
                                pw.println(palabras[j]);//+":"+palabras[j+1]);
                    }
           }

          fichero.close();
         } catch (Exception e) {
```

```
                e.printStackTrace();
            }


        }

}
```

2. Clase Procesar

Clase que realiza todos los métodos que son llamados en la clase principal tales como: anti-diccionario, el cambio de mayúsculas a minúsculas, eliminando todos los signos y los separamos en palabras (tokens).

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.util.StringTokenizer;



public class Procesar {

        static int nPalabras = 0;
        static int nPalabrasTexto = 0; //# de palabras en el doc
        static int totalPalabras = 0;
        static int cont = 0;
        static String [] c = null;
        static String [] contPalabras = null;// Almacena # de veces que se repiten las
palabras
        static String [] pSeleccionadas = null;// Palabras seleccionadas despues de aplicar
filtros
        static String anti_dic = "anti-diccionario.txt";
        static File antiDiccionario = null;


        public  String[] cargarAntiDiccionario(){

                String [] diccionario = null;

                        antiDiccionario = abrir(anti_dic);

                diccionario = new String[(int)antiDiccionario.length()];

                diccionario = separar(antiDiccionario,(int)antiDiccionario.length());
```

```
        return diccionario;
    }


    /*****************************************************************
     Aquí es el método que separa, cambia a minúsculas, quita puntuaciones
          del documento
    *****************************************************************/

        public static String [] separar(File archivo,int l){
            FileReader fr = null;
          BufferedReader br = null;
          String []temporal = new String[l];
          nPalabras = 0;
            try {
           // Apertura del fichero y creacion de BufferedReader para poder
           // hacer una lectura comoda (disponer del metodo readLine()).

           fr = new FileReader (archivo);
           br = new BufferedReader(fr);

           // Lectura del fichero
           String linea;
           String Minuscula;
           while((linea=br.readLine())!=null){
              Minuscula = linea.toLowerCase();
              StringTokenizer tokens=new StringTokenizer(Minuscula);
              while(tokens.hasMoreTokens()){
                            String str = tokens.nextToken(", . -");
                           temporal [nPalabras] = str;
                           nPalabras++;

              }
           }
          }
          catch(Exception e){
           e.printStackTrace();
          }
            return temporal;
        }
/*****************************************************************
                    Apertura del archivo
*****************************************************************/

public static File abrir(String a){
```

```
File archivo = null;

 try {
    // Apertura del fichero y creacion de BufferedReader para poder
    // hacer una lectura comoda (disponer del metodo readLine()).
    archivo = new File (a);
              }
 catch(Exception e){
    e.printStackTrace();
 }

return archivo;
}

/*
 * Separa el texto extraido del pdf en tokens
 * */

public String[] cargarTexto(String texto, int l) {

          String temporal [] =  new String [l];
          StringTokenizer tokens=new StringTokenizer(texto);
           nPalabrasTexto = 0; // inicialización de variable
          while(tokens.hasMoreTokens()){
                    String str = tokens.nextToken(", . - \" /“ /” ? º  1 2 3 4 5 6 7 8 9 0 _ \t ( )
     [ ] : ¹? „ ~ …… ? ; • # * & @ – / \\ $  ' = @ % ^ + { } += ? ¿" + " < > ! ");
                    temporal [nPalabrasTexto] = str.toLowerCase();

                    nPalabrasTexto++;

           }
          return temporal;
}


public void aplicarAntidiccionario(String a[],int i,String b[],int j){
          int x=0,l=0,k=0;
          totalPalabras = 0;
          c = new String[nPalabrasTexto];//Mover  si no alcanza
           for(l=0 ; l<i ;l++){
                    for(k=0 ; k<b.length ;k++){
                              if(a[l].equals(b[k]))
                                        x=1;
                    }
                    if(x==1){
                              //        Si se encuentra la palabra analizada en el anti-diccionario
```

```
                        x=0;
                }
                else{
                        //  No se encuentra la palabra analizada en el anti-diccionario

                  quitarAcentos(a[l]);
                }
         }
}

private void quitarAcentos(String palabra) {
        char letras[]=new char[palabra.length()];
        letras = palabra.toCharArray();
        String p = "";

        for (int i=0; i<letras.length; i++){
                //System.out.println("letra: "+letras[i]);
                if (letras[i] == 'á')
                        letras [i] = 'a';
                else if (letras[i] == 'é')
                        letras [i] = 'e';
                else if (letras[i] == 'í')
                        letras [i] = 'i';
                else if (letras[i] == 'ó')
                        letras [i] = 'o';
                else if (letras[i] == 'ú')
                        letras [i] = 'u';

                 p += Character.toString(letras[i]);

        }
   c[totalPalabras]=p;
   //System.out.println("c: "+c[totalPalabras]);
        totalPalabras++;
        }

//*************************************************************************

public String[] cuentaPalabras(){
        contPalabras = new String[(totalPalabras*2)];

        for (int j =0;j<totalPalabras; j++){
          if (buscar(c[j]) == 0){//sino esta repetida, agregala
                  agrega(c[j]);
          }
          else { // si esta repetida incrementa
                  incrementa(c[j]);
```

```
            }
        }

        //  SEPARAR PALABRAS DE NUMERO DE CONCURRENCIAS

        pSeleccionadas = new String [cont/2];  // Inicialización de arreglo
        int i=0,k=1;
        for (int j = 0; j < cont; j=j+2){

            pSeleccionadas [i]= contPalabras[j];
            // PARA HISTOGRAMA, USAR CONTpALABRAS CON cont de tope
                    //System.out.println(" "+contPalabras[j]+" "+contPalabras[k]);
                    //k+=2;
                    i++;
                   }
        cont=0;
        return pSeleccionadas;
        }

//**********************************************************************

public String[] cuentaPalabrasUnion(String union[]){
        String []PalabrasUnion = new String[union.length];

        for (int j =0;j<union.length; j++){
          if (buscar(c[j]) == 0){//sino esta repetida, agregala
                    agrega(c[j]);
          }
          else { // si esta repetida incrementa
                    incrementa(c[j]);
          }
        }

        //  SEPARAR PALABRAS DE NUMERO DE CONCURRENCIAS

        pSeleccionadas = new String [cont/2];  // Inicialización de arreglo
        int i=0,k=1;
        for (int j = 0; j < cont; j=j+2){

            pSeleccionadas [i]= contPalabras[j];
            // PARA HISTOGRAMA, USAR CONTpALABRAS CON cont de tope
                    //System.out.println(" "+contPalabras[j]+" "+contPalabras[k]);
                    //k+=2;
                    i++;
                   }
        cont=0;
        return pSeleccionadas;
```

```
        }


//*************************************************************************
//*************************************************************************

public static void incrementa(String palabra){
        for (int i=0; i<totalPalabras; i++){
                if (palabra.equals(contPalabras[i])){// busca la posicion en la cual se
encuentra la palabra
                        //System.out.println("Repetida: "+contPalabras[i]+ "
"+contPalabras[i+1]);
                        int n = Integer.parseInt( contPalabras[i+1]);//extrae el número de
veces que se ha encontrado
                        n++;//incrementa en 1
                        contPalabras[i+1] = n+"";// actualiza el valor
                        //System.out.println("Actualizada: "+contPalabras[i]+ "
"+contPalabras[i+1]);
                }
        }
}


//*************************************************************************

//*************************************************************************

public static int buscar(String palabra){
        int n=0;
 for (int i=0; i<contPalabras.length; i++){

        if (palabra.equals(contPalabras[i])){// si se encuentra palabra dentro de contPalabra,
n = 1
                n = 1;
                }
        }
        return n;
}

//*************************************************************************
  //para agregar cuantas veces se repite la palabra
//*************************************************************************

public static void agrega(String p){

        contPalabras [cont] = p;
        contPalabras [cont+1] = "1";
```

```
            cont = cont+2;

        }
}
```

3. Clase PDF

Clase que permite realizar la manipulación de archivos en formato .PDF en el lenguaje JAVA.

```
import org.apache.pdfbox.pdmodel.PDDocument;
import org.apache.pdfbox.util.PDFTextStripper;
import java.io.*;


/**
 * @author Ugalde Anaya Juan Luis
 **/
public class pdf
{

  public String getTexto( String nombrePDF)
  {
    String texto = null;
    try
    {
      int nPaginas = 0;
      // Se carga el documento PDF
      PDDocument DocPDF=PDDocument.load(new File(nombrePDF));

      // # de paginas del documento PDF
                  nPaginas = DocPDF.getNumberOfPages();
                  //          System.out.println("# de paginas: "+nPaginas);

      // Crear el Stripper de Texto
                  PDFTextStripper stripper = new PDFTextStripper();

      // Setear parÃ¡metros del PDFTextStripper
                  stripper.setSortByPosition(false);
                  stripper.setStartPage(1);// # Pagina Inicio de PDF
                  stripper.setEndPage(nPaginas);// # Pagina Fin de PDF

      // Transformar contenido del PDFTextStripper a una String
                  texto = stripper.getText(DocPDF);
              // System.out.println(texto);
       DocPDF.close();
    }
```

```
   catch(Exception ex)
   {
    ex.printStackTrace();
   }
        return texto;
  }

}
```

4. Clase de Recuperación de datos.

Clase que crea el archivo trainWeka.arff leyendo los datos almacenados en la base de datos BD proyectos para facilitar su ubicación.

```
 import java.io.FileWriter;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;


public class recuperacionDatos {
        static int t= 164; //Numero de entrada
        public static int []max= new int[165];
   public static int []min= new int[165];

        public static void main(String[] args) {

                double x[] = new double[8];
                String url = "jdbc:postgresql://localhost:5432/proyectosBD";
       String user = "postgres";
       String password = "root";
       Statement s;
       ResultSet rs;
       String consulta=null;


       double []nivel= new double[170];/* arreglo en que se almacenarán los datos
                                                                leídos desde la base de
datos tarareo*/


       Connection conn = null;
              try{
                     Class.forName("org.postgresql.Driver");
```

```
                System.out.println("Conexion exitosa");


        } catch (ClassNotFoundException cnfe){
          System.out.println("No se pudo encontrar el controlador JDBC");
          System.exit(1);
        }

      try {
         conn = DriverManager.getConnection
                    (url, user, password);

        int i,w=0;
        s = conn.createStatement();
        for ( i = 0; i<=(t);i++){
           consulta="";
                consulta = "SELECT max(p_"+i+") FROM pt_diccionario";
                //System.out.println();
                rs = s.executeQuery(consulta);
                        while (rs.next())
                        {
                //max[i] =rs.getInt(1);/*Almacenamiento del valor máximo en variable*/
                //System.out.println("maximo: "+rs.getInt(1));
                w++;
                        }
                        consulta="";
                consulta = "SELECT min(p_"+i+") FROM pt_diccionario";
                rs = s.executeQuery(consulta);
                while (rs.next())
                {
                //min[i] =rs.getInt(1);  /*Almacenamiento del valor mínimo en variable
                                        //
         min                                                    */
                //System.out.println("minimo: "+rs.getInt(1));
                }
        }
        //System.out.println(w);
        System.out.println(" Creando cabecera de archivo train.arff");
        crearCabecera();  // crear cabecera de archivo con formato arff
        System.out.println(" Creacion de cabecera exitosa");

//=================================================================
============


                consulta="";
         consulta = "SELECT * FROM pt_diccionario";
```

```
        rs = s.executeQuery(consulta);
        /*
         * Lectura de los datos almacenados en la base de datos
         *                                                      */
                while (rs.next()){
                        for ( i = 0; i<=t;i++){
                                nivel[i] = rs.getInt(i+1);
                                //System.out.println("id "+(i));
                                }
                        //System.out.println(rs.getDouble(166));

                        nivel[165] = rs.getDouble(166);
                                //System.out.println("id "+nivel[164]);
                                normalizar(nivel); //Normalización de los datos leídos desde
la base de datos

                }
        s.close();
        conn.close();
        System.out.println(" Archivo arff creado exitosamente");
      } catch (SQLException sqle) {
        System.out.println("Conexión fallida");
        System.exit(1);
       }
        }
        /*
         * Método que crea archivo trainWeka.arff junto con la cabecera
         *
         * */
        private static void crearCabecera() {
         FileWriter fichero = null;
    PrintWriter pw = null;

    try
      {
       fichero = new FileWriter("trainWeka.arff");
       pw = new PrintWriter(fichero);
        pw.println("@relation trainWeka\n");
        for (int i = 0; i <= t; i++) {
                pw.println("@attribute p_"+i+" NUMERIC");
                        }

        pw.println("@attribute id_pt NUMERIC");
        pw.println("@data\n");
      fichero.close();
     } catch (Exception e) {
        e.printStackTrace();
```

```
    }


        }
        /*Método que normaliza los datos almacenados en la variable x
         * y manda a gurdarlos en el archivo trainWeka.arff
         * */
        private static void normalizar(double[] x) {

                double resultados[] = new double [170];

                for(int i=0; i< t; i++){
                        resultados[i] = (x[i] - min[i])/(max[i] - min[i]);
                }
                  resultados[164] = x[166];
                // guarda resulatados
    guarda(x);

        }

        /*
         * Metodo que agrega al final del archivo trainWeka.arff los datos normalizados
         * */
        private static void guarda(double[] b) {

    FileWriter fichero = null;
    PrintWriter pw = null;

    try
      {
        fichero = new FileWriter("trainWeka.arff",true);
        pw = new PrintWriter(fichero);
          for(int j=0; j <= t;j++){

                   pw.print(b[j]+",");
                  // System.out.println("j: "+j+" "+b[j]);
          }
          pw.println(b[165]);
          //System.out.println(b[164]);
          //System.out.println();

       fichero.close();
     } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

5. Clase perceptron

Clase que implementa la red neuronal (Perceptron Multilayer) que lee el archivo trainWeka.arff para su entrenamiento y genera un archivo llamado ReporteWeka.txt en el que se muestra la arquitectura de la red neuronal indicando los pesos y las conexiones entre neuronas (nodos). También genera un archivo llamado PesosWeka.txt en el que únicamente se almacenan los pesos a partir del archivo ReporteWeka.txt.

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
//import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
//import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

//import weka.core.Instance;
import weka.core.Instances;
import weka.core.Utils;
//import weka.core.converters.ArffLoader;
import weka.classifiers.Evaluation;
//import weka.classifiers.bayes.NaiveBayesUpdateable;
import weka.classifiers.functions.MultilayerPerceptron;
//import weka.gui.beans.Classifier;
public class perceptron {

	/**
	 * Opciones para definir una arquitectura con WEKA

	-L. Define la tasa de aprendizaje (default 0.3)

	-M. Define el momentum (default 0.2)

	-N. Específica el número de épocas para parar el entrenamiento (default 500)

	-V  default 0. No se realiza validación, mientras se especifique un número de épocas

	-S Establecer la semilla para el generador de números aleatorios. (default 0)

	-E Ajuste del umbral para el número de errores consecutivos permitidos durante la prueba de validación. (default 20)

	-G Permite visualizar la arquitectura de la red neuronal.
```

-B No reprocesa los casos automáticamente usando un valor nominal de filtro binario .

-H Establece el numero de nodos que se utilizaran en la arquitectura. Cada numero deberá ir separado por comas.(default 4)

-C No normaliza los datos de entrenamiento

-I No normaliza los datos de los atributos.

-R No permite a la red reseteo automatico
 *
 */
```
public static void main(String[] args) throws Exception {
        try{
                FileReader trainreader = new FileReader("trainWeka.arff");

                Instances train = new Instances(trainreader);

                train.setClassIndex(train.numAttributes() - 1);


                MultilayerPerceptron mlp = new MultilayerPerceptron();//
                mlp.setOptions(Utils.splitOptions("-L 0.1 -M 0.1 -N 500 -V 0 -S 0 -E 1 -H
5 -G -B -C -I -R"));

                mlp.buildClassifier(train);

                System.out.println(mlp);// Muestra la arquitectura indicando los pesos y
las conexiones
                                                                        // entre neuronas
(nodos)

                Evaluation eval = new Evaluation(train);
                eval.evaluateModel(mlp, train);// EvaluaciÃ³n del entrenamiento


               System.out.println(eval.toSummaryString("\nResultados\n", false));

                //CreaciÃ³n de archivo en el que se almacenan los resultados
               BufferedWriter writer = new BufferedWriter(new
FileWriter("ReporteWeka.txt"));
              writer.write(""+mlp);
              writer.flush();
              writer.close();
              trainreader.close();


               StringTokenizer tokens;
               FileReader fr = null;
               FileWriter fw = null;
               PrintWriter pw = null;
```

```
                    fw = new FileWriter("PesosWeka.txt");
                    pw = new PrintWriter(fw);
                //ExtracciÃ³n de los pesos
                fr = new FileReader("ReporteWeka.txt");
                BufferedReader bf = new BufferedReader(fr);
                String linea;
                String palabra;
                double num = 0.0,d=0.0;
                int i;

                while((linea = bf.readLine())!= null){
                    tokens = new StringTokenizer(linea," ");
                    while(tokens.hasMoreTokens()){
                            palabra = tokens.nextToken();

                            if( verifica(palabra) == true){
                                    num = Double.parseDouble(palabra);
                                    i = (int)num;
                                    d = num - i;
                                    if (d != 0.0)
                                            pw.println(num);
                            }
                    }
                }
                pw.close();
                fw.close();
                fr.close();

                System.out.println("Archivo Creado Exitosamente");
                    } catch(Exception ex){
                            ex.printStackTrace();
                    }
        }

        private static boolean verifica(String palabra) {
                double num = 0.0;
                try{
                        num = Double.parseDouble(palabra);
                }catch (Exception e) {
                        return false;
                }
                return true;
        }
}
```

6. Clase pesosWeka.

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.text.DecimalFormat;
import java.util.StringTokenizer;
import java.util.logging.Level;
import java.util.logging.Logger;


public class pesosWeka {

        public static int intput = 166;//165 entradas + BIAS
        public static int nC1 = 5;// # neuronas de la capa 1 30
        //public static int nC2 = 2;// # neuronas de la capa 2 26
        public static double [][] wij = new double [nC1+1][intput+1] ;// Almacenamiento
de Pesos de Entrada a Capa 1
       // public static double [][] wjk = new double [nC2+1][nC1+1] ; // Almacenamiento
de Pesos de Pesos de Capa 1 a Capa 2
        public static double [] s = new double [nC1+1]; // Salida
        public static double [] in = new double [intput];// Datos de entrada
        public static double [] sumasC1 = new double [nC1+1];// Calculo tanh en C1
       // public static double [] sumasC2 = new double [nC2+1];// Calculo sigmoidal en C2


        public static void main(String[] args)  throws IOException{
//      public static int recupera() throws IOException{
                        cargarPesos();
                                System.out.println("Pesos Cargados");
                        leerTarareo();
                System.out.println("Tarareo leido");
              int i = recuperarPT();
              System.out.println("quisiste decir este:   "+i);
            //return i;
        }

        private static int recuperarPT() {

                DecimalFormat df = new DecimalFormat("0.00");
            double r, entero, decimal;
```

```
            sumatoriaC1();//Sumatoria cada peso por entrada
            sigmoidalC1();
            //sumatoriaC2();
            //sigmoidalC2();


            r = salida();

          System.out.println("\nsalida red: "+df.format(r));
           r = (r*24); // Recuperar PT
           entero=(long)r;
           decimal=r-entero;
        // devuelve el numero dentro del rango de PT´s
          if(decimal >= 0.5){
                entero = (int)(entero+1);
          }
          if(entero >= 24) entero = 9.0;
            return (int)entero%24;
    }


    private static double salida() {
            double suma = 0.0;
          for (int l= 0; l<= nC1; l++){
            suma += sumasC1[l]*s[l];
            }
          return suma;
    }


    private static void sigmoidalC1() {
            double e = 0.0;
for (int i= 1; i<= nC1; i++){
    e = Math.exp(-(sumasC1[i]));
    sumasC1[i] = (1/(1+e));
    }

    }

    private static void sumatoriaC1() {
             double suma = 0.0;
           System.out.println();
           // Sumatoria de datos de entrada por su respectivo peso
           sumasC1[0]=1;///bias
             for (int j = 1; j<= nC1; j++){
               for (int i=0;i<intput; i++){
```

```
                    suma += in[i]*wij[j][i];
                    }
                    sumasC1[j] = suma;
                    suma = 0.0;
                }// fin sumatoria

        }

        private static void leerTarareo()throws IOException {
                 try {
                int i=1;
                FileReader fr = null;
                fr = new FileReader("propuesta.txt");
                BufferedReader bf = new BufferedReader(fr);
                String sCadena;
                sCadena = bf.readLine();
                StringTokenizer st = new StringTokenizer(sCadena);
                    in[0]=1; // Bias
                        while(st.hasMoreTokens()){
                    String str = st.nextToken(",");
                    in[i] = Double.parseDouble(str);
                    i++;
                        }
            } catch (FileNotFoundException ex) {
                Logger.getLogger(pesosWeka.class.getName()).log(Level.SEVERE, null,
ex);
            }

        }

        private static void cargarPesos() throws NumberFormatException, IOException {
                double n=0;
    int i=1,cont=1,linea;
    int j = 1, k = 1, l = 1;
    FileReader fr = null;
    fr = new FileReader("PesosWeka.txt");
    BufferedReader bf = new BufferedReader(fr);

    //   Lectura de los pesos asociados al Bias a las diferentes capas
    System.out.println("Cargando pesos S para la salida ");
       for (linea = 0; linea <= (nC1); linea ++){
          s[linea] = Double.parseDouble(bf.readLine());
          //System.out.println(s[linea]);
          }

        //Se cargan los pesos correspondientes a cada entrada en wij
```

```
        for (i = 1 ; i<=nC1; i++){
                System.out.println(" Cargando pesos para Nodo "+i);
           for(j = 0; j<intput;j++){

              wij[i][j] = Double.parseDouble(bf.readLine());
            //System.out.println(wij[i][j]);
           }
        }
   // fin de carga de pesos
      fr.close();
        }
}
```

7. Clase ventana principal.

Clase que implementa la interfaz gráfica que permite realizar una búsqueda del documento o proyecto terminal a comparar o a realizar la comparación si se encuentran otros similares donde se muestran dos botones los cuales es uno examinar el cual busca en cualquier ruta un archivo en formato PDF si no está en este formato no realizara nada, ya seleccionado se le da clic en el botón buscar el cual nos mostrara los 5 más parecidos.

```
import javax.swing.SwingUtilities;
import java.awt.BorderLayout;

import javax.swing.JFileChooser;
import javax.swing.JPanel;
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.SwingConstants;
import java.awt.ComponentOrientation;
import java.awt.Rectangle;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

import javax.swing.JTextField;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.JList;

public class ventana extends JFrame {

        private static final long serialVersionUID = 1L;
```

```
	private JPanel jContentPane = null;
	private JButton BotonExaminar = null;
	private JTextField ArchivoSeleccionado = null;
	String fileName = "";  // @jve:decl-index=0:
	String RutaAbsolutaFile = "";  // @jve:decl-index=0:
	String texto = null;  // @jve:decl-index=0:
	static int ndic = 0;
	private static int resultado[];
	static String PalabrasDiccionario [] = new String[200];  // @jve:decl-index=0:
	private JButton BotonBuscar = null;
	private JList ListaResultados = null;

	/**
	 * This method initializes ButonExaminar
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getButonExaminar() {
		if (BotonExaminar == null) {
			BotonExaminar = new JButton();
			BotonExaminar.setText("Examinar");
			BotonExaminar.setHorizontalAlignment(SwingConstants.CENTER);

	BotonExaminar.setComponentOrientation(ComponentOrientation.RIGHT_TO_LE
FT);
			BotonExaminar.setRolloverEnabled(false);
			BotonExaminar.setBounds(340, 81, 104, 26);
			BotonExaminar.addActionListener(new
java.awt.event.ActionListener() {
				public void actionPerformed(java.awt.event.ActionEvent e) {
					System.out.println("Examinando ..."); // TODO Auto-
generated Event stub actionPerformed()
					JFileChooser chooser = new JFileChooser();
				chooser.setCurrentDirectory(new File("C:/"));
				chooser.setSelectedFile(new File(fileName));

chooser.setFileSelectionMode(JFileChooser.FILES_ONLY);


				//Object cmd = e.getSource();
				int code = chooser.showOpenDialog(jContentPane);
					if (code == JFileChooser.APPROVE_OPTION) {
						File selectedFile = chooser.getSelectedFile();
						fileName = selectedFile.getName();
						RutaAbsolutaFile =
selectedFile.getAbsolutePath();
						if(filtro(fileName)){
```

```
                                                System.out.println("Archivo
Seleccionado: "+fileName);

        ArchivoSeleccionado.setText(fileName);

                                                }
                                                else {
                                                        System.out.println("Formato no
soportado");
                                                }

                                        }
                                    }
                                }
                            );

                }
                return BotonExaminar;
        }

                private boolean filtro(String fileName) {
                        String archivo = fileName.toLowerCase();
                        if(archivo != null){
                        if(archivo.endsWith(".pdf")){
                                return true;
                        }

                }
                        return false;
        }


        /**
         * This method initializes ArchivoSeleccionado
         *
         * @return javax.swing.JTextField
         */
        private JTextField getArchivoSeleccionado() {
                if (ArchivoSeleccionado == null) {
                        ArchivoSeleccionado = new JTextField();
                        ArchivoSeleccionado.setBounds(new Rectangle(97, 79, 211, 28));

                }
                return ArchivoSeleccionado;
        }

        /**
```

```
	 * This method initializes BotonBuscar
	 *
	 * @return javax.swing.JButton
	 */
	private JButton getBotonBuscar() {
		if (BotonBuscar == null) {
			BotonBuscar = new JButton();
			BotonBuscar.setBounds(new Rectangle(344, 126, 97, 28));
			BotonBuscar.setText("Buscar");
			BotonBuscar.addActionListener(new java.awt.event.ActionListener()
{
				public void actionPerformed(java.awt.event.ActionEvent e) {
					System.out.println("Buscar archivo:
"+RutaAbsolutaFile); // TODO Auto-generated Event stub actionPerformed()
					buscar();


				}
			});
		}
		return BotonBuscar;
	}

	private void buscar() {

		FileWriter fichero = null;
      PrintWriter pw = null;
		Procesar archivo = new Procesar();
		//Carga el antidiccionario
		String antiDiccionario[] = archivo.cargarAntiDiccionario();

		pdf archivoPDF = new pdf(); // instancia a la clase pdf
		//Se extrae el texto del pdf y se almacena en texto
		texto = archivoPDF.getTexto(RutaAbsolutaFile );
		System.out.println("Texto extraido exitosamente ");
		//System.out.println(texto);

		String documentoBase[] = archivo.cargarTexto(texto,texto.length());

		// Aplicar anti-diccionario
			archivo.aplicarAntidiccionario( documentoBase,
archivo.nPalabrasTexto, antiDiccionario, archivo.nPalabras);

		// Se almacenan el total de palabras filtradas
			String PalabrasPT[] =   archivo.cuentaPalabras(); // Total de palabras
filtradas para el archivo base
```

```
                try
            {

             fichero = new FileWriter("PT.txt");
             pw = new PrintWriter(fichero);


              for(int j=0; j < (PalabrasPT.length);j++){
                    pw.println(PalabrasPT[j]);
              }
              fichero.close();
                  } catch (Exception e) {
                    e.printStackTrace();
                  }

                    // Cargar el archivo Diccionario.txt y llenar arreglo resultados
                    // para crear archivo propuesta.txt que sirve de entrada a la red
neuronal
                    cargarDiccionario();
                    entradaRed(PalabrasPT);// Crea el archivo propuesta.txt



        }



        private void entradaRed(String[] PalabrasPT) {

                int i=0;
                FileReader fr = null;
                BufferedReader br = null;
                resultado = new int[(PalabrasDiccionario.length)];// Aqui va el # de datos en
Diccionario.txt
                for ( i = 0; i < PalabrasDiccionario.length; i++) {
                        resultado[i] = 0;
                }

                try {
                        fr = new FileReader ("PT.txt");// Carga el archivo PT.txt que
contiene las palabras filtradas del
                         br = new BufferedReader(fr);  // archivo pdf a analizar
                      String linea;
                      while((linea=br.readLine())!=null){
                         StringTokenizer tokens=new StringTokenizer(linea);
                         while(tokens.hasMoreTokens()){
```

```
                                        String str = tokens.nextToken("  ");//Para quitar lineas
vacias
                                        //System.out.println(str);

                                        busca(str);
                                    }
                                }
                        } catch (FileNotFoundException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        } catch (IOException e) {
                                // TODO Auto-generated catch block
                                e.printStackTrace();
                        }

                        guardar(resultado);             // guarda el arreglo resultado el que contine
ceros y unos
                        recuperarPT();          // invoca la clase pesosWeka que interpreta la
arquitectura de la red
                                                // previamente entrenada para recuperar los
archivos mas parecidos
                }

                private static void busca(String str) {

                        for (int i=1; i < PalabrasDiccionario.length; i++){

                                if (str.equals(PalabrasDiccionario[i])){// si se encuentra palabra
dentro de total, n = 1
                                        resultado[i+1] = 1;
                                        break;
                                        }

                                }
                        }

                private void recuperarPT() {

                        pesosWeka pt = new pesosWeka();
                        String [] lista;
                        try {
                                lista = pesosWeka.recupera();// regresa en lista los 5 pts mas
parecidos

                                ListaResultados.setListData(lista);// Muestra la lista en el area de la
interfaz
```

```
            } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
            }
    }

    private void guardar(int[] resultado) {
             FileWriter fichero = null;
        PrintWriter pw = null;
        int i;
        try
          {// guarda en propuesta.txt los 0´s y 1´s almacenados en resultados[]
           fichero = new FileWriter("propuesta.txt");
           pw = new PrintWriter(fichero);

            for (i = 0; i< (ndic); i++) {
                    pw.print(resultado[i]+",");
                            }
           pw.println(resultado[i]);
          fichero.close();
         } catch (Exception e) {
           e.printStackTrace();
        }




    }

    private void cargarDiccionario() {
             File txt = new File("Diccionario/diccionario.txt");
             FileReader fr = null;
             BufferedReader br = null;
             ndic = 0;

   try {

     fr = new FileReader (txt);
             br = new BufferedReader(fr);
     String linea;
                    while((linea=br.readLine())!=null){
                            StringTokenizer tokens=new StringTokenizer(linea);

                            while(tokens.hasMoreTokens()){
                                    String str = tokens.nextToken("  ");//Para quitar lineas
vacias
```

```
                                        PalabrasDiccionario[ndic] = str;  // Se almacenan las
palabras de diccionario.txt
                                         //System.out.println(Palabras[ndic]);

                                           ndic++;
                                }

                         }
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

        }

        /**
         * This method initializes ListaResultados
         *
         * @return javax.swing.JList
         */
        private JList getListaResultados() {
                if (ListaResultados == null) {
                        ListaResultados = new JList();
                        ListaResultados.setBounds(new Rectangle(96, 146, 219, 104));
                }
                return ListaResultados;
        }

        /**
         * @param args
         */
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                SwingUtilities.invokeLater(new Runnable() {
                        public void run() {
                                ventana thisClass = new ventana();

        thisClass.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                                thisClass.setVisible(true);
                        }
                });
        }

        /**
         * This is the default constructor
         */
        public ventana() {
```

```
            super();
            initialize();
        }

        /**
         * This method initializes this
         *
         * @return void
         */
        private void initialize() {
            this.setSize(600, 400);
            this.setContentPane(getJContentPane());
            this.setTitle("Proyecto Terminal ");
        }

        /**
         * This method initializes jContentPane
         *
         * @return javax.swing.JPanel
         */
        private JPanel getJContentPane() {
            if (jContentPane == null) {
                    jContentPane = new JPanel();
                    jContentPane.setLayout(null);
                    jContentPane.setEnabled(false);
                    jContentPane.add(getButonExaminar(), BorderLayout.NORTH);
                    jContentPane.add(getArchivoSeleccionado(), null);
                    jContentPane.add(getBotonBuscar(), null);
                    jContentPane.add(getListaResultados(), null);
            }
            return jContentPane;
        }

}
```

Ejecución del Software

Requisitos

- Tener instalada la maquina virtual de java JRE V. 6.
  (Sino cuenta con JRE instalada en su equipo dirigirse a la siguiente liga http://www.java.com/es/ en donde encontrara la versión adecuada así como las instrucciones de instalación para diferentes sistemas operativos).

Ejecución.

Para usuarios Linux copiar la carpeta en el escritorio, después abrir una consola y ubicarse en la ruta en donde se encuentra el directorio tarareo, una vez dentro del directorio ejecutar el siguiente comando:

Java –jar PT.jar

Para usuarios Windows, simplemente ubicarse en la carpeta PT y hacer doble click en el archivo PT.jar

Creación de la Base de Datos

Para la realización de este trabajo se utilizo el manejador PostgreSQL 8.4 [5] creando una base de datos llamada proyectosBD y una tabla llamada pt_diccionario. A continuación se muestran las sentencias SQL para la creación tanto para la base de datos como para la tabla.

Sentencias SQL para la creación de base datos proyectosBD.

```
CREATE DATABASE proyectosBD
  WITH OWNER = postgres
  ENCODING = 'UTF8'
  TABLESPACE = pg_default
  LC_COLLATE = 'es_MX.UTF-8'
  LC_CTYPE = 'es_MX.UTF-8'
  CONNECTION LIMIT = -1;
```

Sentencias SQL para la creación de tabla pt_diccionario en base datos proyectosBD.

```
CREATE TABLE pt_diccionario
(
  p_0 real,
  p_1 real,
  p_2 real,
  p_3 real,
  p_4 real,
  p_5 real,
  p_6 real,
  p_7 real,
```

p_8 real,
p_9 real,
p_10 real,
p_11 real,
p_12 real,
p_13 real,
p_14 real,
p_15 real,
p_16 real,
p_17 real,
p_18 real,
p_19 real,
p_20 real,
p_21 real,
p_22 real,
p_23 real,
p_24 real,
p_25 real,
p_26 real,
p_27 real,
p_28 real,
p_29 real,
p_30 real,
p_31 real,
p_32 real,
p_33 real,
p_34 real,
p_35 real,
p_36 real,
p_37 real,
p_38 real,
p_39 real,
p_40 real,
p_41 real,
p_42 real,
p_43 real,
p_44 real,
p_45 real,
p_46 real,
p_47 real,
p_48 real,

p_49 real,
p_50 real,
p_51 real,
p_52 real,
p_53 real,
p_54 real,
p_55 real,
p_56 real,
p_57 real,
p_58 real,
p_59 real,
p_60 real,
p_61 real,
p_62 real,
p_63 real,
p_64 real,
p_65 real,
p_66 real,
p_67 real,
p_68 real,
p_69 real,
p_70 real,
p_71 real,
p_72 real,
p_73 real,
p_74 real,
p_75 real,
p_76 real,
p_77 real,
p_78 real,
p_79 real,
p_80 real,
p_81 real,
p_82 real,
p_83 real,
p_84 real,
p_85 real,
p_86 real,
p_87 real,
p_88 real,
p_89 real,

p_90 real,
p_91 real,
p_92 real,
p_93 real,
p_94 real,
p_95 real,
p_96 real,
p_97 real,
p_98 real,
p_99 real,
p_100 real,
p_101 real,
p_102 real,
p_103 real,
p_104 real,
p_105 real,
p_106 real,
p_107 real,
p_108 real,
p_109 real,
p_110 real,
p_111 real,
p_112 real,
p_113 real,
p_114 real,
p_115 real,
p_116 real,
p_117 real,
p_118 real,
p_119 real,
p_120 real,
p_121 real,
p_122 real,
p_123 real,
p_124 real,
p_125 real,
p_126 real,
p_127 real,
p_128 real,
p_129 real,
p_130 real,

```
  p_131 real,
  p_132 real,
  p_133 real,
  p_134 real,
  p_135 real,
  p_136 real,
  p_137 real,
  p_138 real,
  p_139 real,
  p_140 real,
  p_141 real,
  p_142 real,
  p_143 real,
  p_144 real,
  p_145 real,
  p_146 real,
  p_147 real,
  p_148 real,
  p_149 real,
  p_150 real,
  p_151 real,
  p_152 real,
  p_153 real,
  p_154 real,
  p_155 real,
  p_156 real,
  p_157 real,
  p_158 real,
  p_159 real,
  p_160 real,
  p_161 real,
  p_162 real,
  p_163 real,
  p_164 real
  id_pt double precision
)
WITH (OIDS=FALSE);
ALTER TABLE pt_diccionario OWNER TO postgres;
```

Bibliografía:

[1]. Kohonen, T. (1972) Correlation matriz memories, IEEE TransactionsonComputers, C-21, 4, 353-359

[2]. SoumenChakrabarti. Mining the Web : discovering knowledge from hypertxt dat. Morgan

[3]. Kaufmann, San Francisco, CA, cop. 2003. ISBN: 1558607544

YAÑEZ, C. "Memorias Asociativas αβ". Tesis Doctoral. Instituto Politecnico Nacional, Centro de Investigación en Computación. (2002).

[4]. API para poder leer documentosde formato PDF a TXThttp://pdfbox.apache.org/

[5] PostgreSQL, revisado en Noviembre de 2010. *http://www.postgresql.org/download/*

[6] PostgreSQL Driver JDBC, revisado en Noviembre de 2010, *http://jdbc.postgresql.org/*

[7] HERRERA, ALCANTARA, O. Reconocimiento de archivos de sonido con redes neuronales y wavelets. Avances en Sistemas Inteligentes en México. Editores Miguel González y Oscar Herrera. Sociedad Mexicana de Inteligencia Artificial. ISBN 9786079536725.Primera edición, (2010)

Recursos:

- Computadora con procesador Celeron Dual Core y 1 GB de RAM
- Sistema operativo Linux distribución Ubuntu 9.40 para cual dispone licencia libre
- Entorno de desarrollo de lenguaje Java Eclipse Classic 3.5.1. Licencia gratis