UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

Detección y Clasificación de Malware usando técnicas inteligentes

Rubén Donate Serrano

Mayo de 2018

UNIVERSIDAD DE CASTILLA-LA MANCHA

ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÁFICA DE COMPUTACIÓN

TRABAJO FIN DE GRADO

Detección y Clasificación de Malware usando técnicas inteligentes

Autor: Rubén Donate Serrano

Directores: José Luis Martínez Martínez

José Miguel Puerta Callejón

Mayo de 2018

Declaración de Autoría

Yo, pepito perez con DNI..... , declaro que soy el único autor del trabajo fin de grado titulado ..... y que el citado trabajo no infringe las leyes en vigo sobre propiedad intelectual y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a.....

Fdo: pepito perez

Resumen

Este documento pretende ser un manual sobre el uso del paquete de LaTeX 2_ε `tfg-esiiab.sty`. Este paquete de estilo da el formato al documento según las normas de estilo establecidas por la Escuela Superior de Ingeniería Informática de Albacete, perteneciente a la Universidad de Castilla-la Mancha.

A mis compañeros y alumnos.

Agradecimientos

Agradecer a todas aquellas personas que me han ayudado a aprender LaTeX 2_ε, bien sea por lo que me enseñaron, o por las preguntas que me transmitieron y que yo intenté solucionar.

Índice general

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1

INTRODUCCIÓN

1.1. Motivación

1.2. Objetivos

1.3. Estructura de la memoria

Capítulo 2

TÉCNICAS UTILIZADAS

Capítulo 3

ANTECEDENTES Y ESTADO DE LA CUESTIÓN

Para el proyecto de Kaggle que se ha seleccionado para este Trabajo de Final de Grado, se he realizado una búsqueda para intentar encontrar propuestas presentadas para éste. De esta, se ha conseguido encontrar varias propuestas que se presentadas de este mismo reto.

A continuación, se explicará el proceso que llevaron a cabo para desarrollar estas soluciones.

3.1. Solucción 1

Esta primera solución fue realizada por Vishnu Chevli [7]. Además, es la mas sencilla de las soluciones que se van a comentar.

Esta solución se ha desarrollado para poder ser ejecutada de manera indistinta en Python 2 y Python 3. Consiste en descomprimir las dos bases de datos proporcionadas para después los ficheros con extensión .asm descártalos y los que poseen la extensión .bytes comprimirlos con gzip. Esto se realiza para reducir el espacio que ocupa las bases de datos en el disco duro, ya que el espacio de estas sin comprimir es de aproximadamente 1 Tb. A continuación, se recorre y se accede a todos estos ficheros comprimidos para analizar en cada uno de ellos el fichero binario en formato hexadecimal que contiene. Este análisis consiste en contar las apariciones en éste de dos elementos en hexadecimal para terminar guardándolos en un fichero csv dentro de un archivo comprimido. Este análisis se realiza de manera paralela para las dos bases de datos terminando así la extracción de características de estas.

El siguiente paso, consiste en construir el modelo y obtener la predicción para el mismo. Para ello se comienza leyendo el fichero proporcionado con las etiquetas para cada uno de los ficheros que forman la base de datos de train. A continuación, se recupera las características que se han extraído en el paso anterior y se crea un array de numpy de ceros cuyo índice es una tupla formada por el id del fichero en primer lugar y como segundo valor el número de posibles características mas una posición adicional para almacenar la etiqueta obtenida anteriormente.

Capítulo 4

METODOLOGÍA Y DESARROLLO

Capítulo 5

EXPERIMENTOS Y RESULTADOS

Capítulo 6

CONCLUSIONES Y PROPUESTAS

6.1. Conclusiones

6.2. Trabajo futuro

Bibliografía

[1] R. Montiel Soto, Y. Ledeneva, R. A. García-Hernández, and R. Cruz Reyes, "Comparación de tres modelos de texto para la generación automática de resúmenes," *Procesamiento del Lenguaje Natural*, no. 43, 2009.

[2] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[3] G. Cobo, X. Sevillano, F. Alías, and J. C. Socoró, "Técnicas de representación de textos para clasificación no supervisada de documentos." *Procesamiento del lenguaje natural*, vol. 37, 2006.

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LibLinear: A library for large linear classification," *Journal of machine learning research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[5] Distributed (Deep) Machine Learning Community. Documentación de parametros de xgboost. [Online]. Available: https://xgboost.readthedocs.io/en/latest/parameter.html#parameters-for-tree-booster

[6] I. Singer, "A general theory of dual optimization problems," *Journal of Mathematical Analysis and Applications*, vol. 116, no. 1, pp. 77–130, 1986.

[7] V. Chevli. Beating the benchmark for Microsoft Malware Classification Challenge (BIG 2015). [Online]. Available: https://github.com/vrajs5/Microsoft-Malware-Classification-Challenge 5

[8] M. Trofimov, D. Ulyanov, and S. Semenov. Microsoft Malware Classification Challenge third place solution. [Online]. Available: https://github.com/geffy/kaggle-malware

[9] [Online]. Available: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?Name=Adware:Win32/Lollipop

[10] [Online]. Available: https://www.microsoft.com/en-us/wdsi/threats/malware-encyclopedia-description?name=Win32/Gatak

[11] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," *CoRR*, vol. abs/1802.10135, 2018. [Online]. Available: http://arxiv.org/abs/1802.10135

CONTENIDO DEL CD