ESCUELA COLOMBIANA DE INGENIERÍA
JULIO GARAVITO

PROYECTO DE GRADO

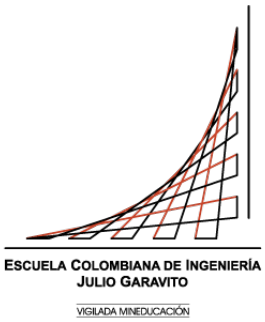
Advanced Natural Language Processing Techniques to Profile Cybercriminals

Autor:
Alejandro ANZOLA ÁVILA

Director:
Daniel Orlando DÍAZ LÓPEZ, *Ph.D.*

Programa de Ingeniería de Sistemas

BOGOTÁ, COLOMBIA



9 de julio de 2019

Índice general

Índice de figuras

Índice de cuadros

Escuela Colombiana de Ingeniería
Julio Garavito

Resumen

Programa de Ingeniería de Sistemas

Estudiante de Ingeniería de Sistemas

Advanced Natural Language Processing Techniques to Profile Cybercriminals

por Alejandro ANZOLA ÁVILA

Se identifican diferentes metodologías de *Data Science* usadas comúnmente para el análisis de datos estructurados y no–estructurados. Luego se realiza una exploración del estado del arte en técnicas actuales de perfilamiento de cibercriminales y técnicas de clasificación de datos. Finalmente se realizan 3 propuestas de modelos de *Natural Language Processing* para la ayuda de agentes de seguridad del Estado en sus tareas contra el crimen.

Escuela Colombiana de Ingeniería
Julio Garavito

Abstract

Programa de Ingeniería de Sistemas

Estudiante de Ingeniería de Sistemas

Advanced Natural Language Processing Techniques to Profile Cybercriminals

by Alejandro ANZOLA ÁVILA

Different commonly used *Data Science* methodologies are identified for structured and non-structured data analysis. After that a State of the Art exploration of current techniques for cybercriminal profiling and data classification is made. Lastly, 3 *Natural Language Processing* model proposals are made to aid State security agents in their tasks against crime.

Notación

Esta notación es una adaptación de la presentada en [1].

Números y Arreglos

a	Un escalar (entero o real)
$\boldsymbol{a}$	Un vector
$\boldsymbol{A}$	Una matriz
$\mathbf{A}$	Un tensor
$\boldsymbol{I}_n$	Matriz identidad con n filas y n columnas
$\boldsymbol{I}$	Matriz identidad con dimensionalidad implícita por el contexto
$\boldsymbol{e}^{(i)}$	Vector estándar base $[0, \dots, 0, 1, 0, \dots, 0]$ con un 1 en la posición i
a	Una variable aleatoria real
$\mathbf{a}$	Un vector de variables aleatorias
$\mathbf{A}$	Una matriz de variables aleatorias

Conjuntos y Grafos

$\mathbb{A}$	Un conjunto
$\mathbb{R}$	El conjunto de números reales
$\mathbb{N}$	El conjunto de números naturales
$\{0, 1\}$	El conjunto que contiene al 0 y el 1
$\{0, 1, \dots, n\}$	El conjunto que contiene todos lo números entre 0 y n
$[a, b]$	El intervalo real que incluye a y b
$(a, b]$	El intervalo real que no incluye a pero si b
$\mathbb{A} \backslash \mathbb{B}$	Substracción de conjuntos, e.g., el conjunto que contiene los elementos de $\mathbb{A}$ que no están en $\mathbb{B}$
$\mathcal{G}$	Un grafo
$Pa_{\mathcal{G}}(\mathrm{x}_i)$	El padre de x_i en $\mathcal{G}$

Índices

a_i	Elemento i del vector $\boldsymbol{a}$, con índice empezando en 1
a_{-i}	Todos los elementos del vector $\boldsymbol{a}$ a excepción del elemento i
$A_{i,j}$	Elemento i, j de la matriz $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Fila i de la matriz $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Columna i de la matriz $\boldsymbol{A}$
a_i	Elemento i del vector aleatorio $\mathbf{a}$

Operaciones de Álgebra Lineal

$\boldsymbol{A}^\top$	Traspuesta de la matriz $\boldsymbol{A}$
$\det(\boldsymbol{A})$	Determinante de la matriz $\boldsymbol{A}$

Calculo

$\frac{dy}{dx}$	Derivada de y con respecto a x
$\frac{\partial y}{\partial x}$	Derivada parcial de y con respecto a x
$\nabla_{\boldsymbol{x}} y$	Gradiente de y con respecto a $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matriz de derivadas de y con respecto a $\boldsymbol{X}$
$\int f(\boldsymbol{x}) d\boldsymbol{x}$	Integral definida sobre un dominio entero $\boldsymbol{x}$
$\int_{\mathbb{S}} f(\boldsymbol{x}) d\boldsymbol{x}$	Integral definida con respecto a $\boldsymbol{x}$ sobre el conjunto $\mathbb{S}$

Teoría de Probabilidad e Información

$\mathrm{a} \perp \mathrm{b}$	Las variables aleatorias a y b son independientes
$\mathrm{a} \perp \mathrm{b} \mid \mathrm{c}$	Son condicionalmente independientes dado c
$P(\mathrm{a})$	Una distribución de probabilidad de una variable aleatoria discreta
$p(\mathrm{a})$	Una distribución de probabilidad de una variable aleatoria continua
$\mathrm{a} \sim P$	La variable aleatoria a tiene distribución P
$\mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$	Distribución gaussiana sobre $\boldsymbol{x}$ con media $\boldsymbol{\mu}$ y varianza $\boldsymbol{\sigma}^2$

Funciones

$f : \mathbb{A} \to \mathbb{B}$	La función f con dominio $\mathbb{A}$ y rango $\mathbb{B}$
$f \circ g$	Composición de las funciones f y g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	Una función de $\boldsymbol{x}$ parametrizado por $\boldsymbol{\theta}$.
$\log x$	Logaritmo natural de x
$\log_b x$	Logaritmo base b de x
$\sigma(x)$	Función sigmoid logística, $\frac{1}{1+\exp(-x)}$
$\zeta(x)$	Función Softplus, $\log(1+\exp(x))$
$\mathrm{softmax}(\boldsymbol{z})_i$	Función Softmax, $\exp(\boldsymbol{z}_i) / \sum_{j=1} \exp(\boldsymbol{z}_j)$
$\|\boldsymbol{x}\|_p$	Norma L^p de $\boldsymbol{x}$
$\|\boldsymbol{x}\|$	Norma L^2 de $\boldsymbol{x}$
x^+	Parte positiva de x, e.g., $\mathrm{máx}(0, x)$
$\mathbf{1}_{\mathrm{condition}}$	Es 1 si $\mathrm{condition} \in \mathbb{B}$ es verdadero, 0 de lo contrario
$[\mathrm{condition}]$	Es 1 si $\mathrm{condition} \in \mathbb{B}$ es verdadero, 0 de lo contrario

Datasets

$\mathbb{X}$	Un conjunto de muestras de entrenamiento
$\boldsymbol{x}^{(i)}$	La i-ésima muestra (entrada) de un Dataset
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	El objetivo asociado con $\boldsymbol{x}^{(i)}$ para aprendizaje supervisado
$\boldsymbol{X}$	La matriz de $m \times n$ con muestra de entrada $\boldsymbol{x}^{(i)}$ en la fila $\boldsymbol{X}_{i,:}$

Capítulo 1

Introducción

1.1. Objetivo general

El objetivo de este proyecto es generar herramientas y estrategias para el perfilado de cibercriminales con ayuda de metodologías de Natural Language Processing (NLP) aplicado a datos recolectados de comunicaciones y redes sociales.

1.2. Objetivos específicos

- Diseñar e implementar una solución de lenguaje natural para realizar el perfilado de sospechosos.
- Identificar el estado del arte en sistemas que usan NLP para apoyar agencias de seguridad del Estado.
- Implementación de artefactos para la construcción de *Dataset*s con información recolectada de medios privados como de fuentes abiertas.
- Validar la solución desarrollada frente a un escenario real.
- Modelado de diferentes metodologías, heurísticas y meta–heurísticas para NLP.

Capítulo 2

Cronograma

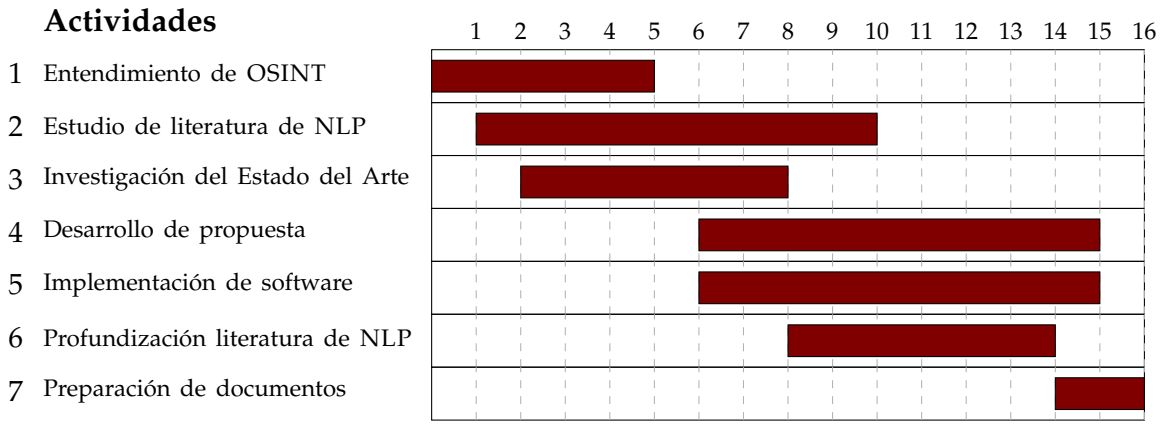


FIGURA 2.1: Diagrama Gantt de actividades de 1er periodo.

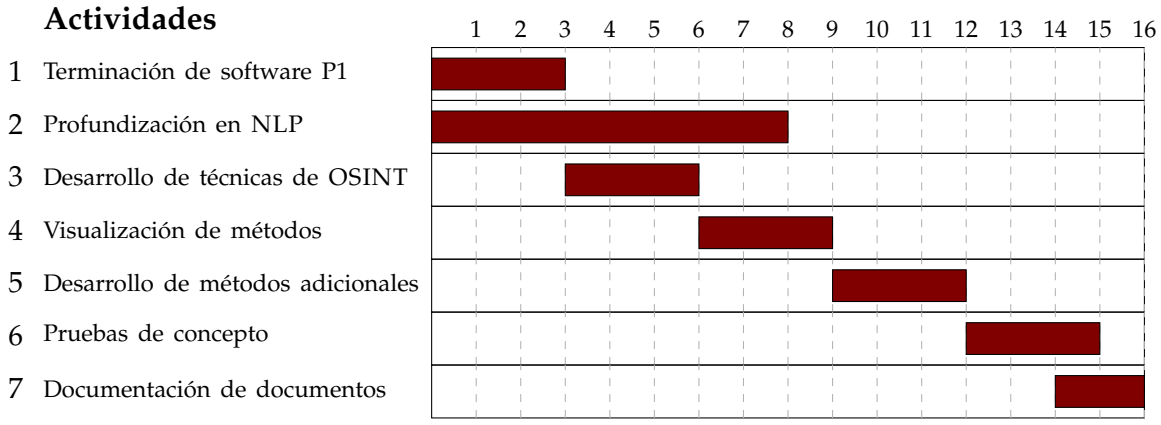


FIGURA 2.2: Diagrama Gantt de actividades de 2do periodo (Planeado).

#	**Descripción**
1.1	Entendimiento proyecto de Open Source Intelligence (OSINT).
1.2	Estudio de la literatura de NLP.
1.3	Investigación del Estado del Arte.
1.4	Desarrollo de la propuesta de investigación.
1.5	Desarrollo de la implementación y pruebas.
1.6	Realización de curso de NLP de *National Research University Higher School of Economics* en *Coursera*.
1.7	Preparación final de documento de libro de proyectos del primer periodo.
2.1	Terminación y mejora de software desarrollado en el primer periodo.
2.2	Periodización de metodologías aplicadas de NLP.
2.3	Desarrollo de nuevas técnicas para la obtención de información de fuentes OSINT como también de identificar nuevas fuentes de información que tengan que ver con terroristas.
2.4	Desarrollo de un *dashboard* detallado que ayuden al agente de seguridad realizar sus labores.
2.5	Desarrollo de métodos adicionales para el perfilamiento de cibercriminales.
2.6	Realizar pruebas demostrativas que permitan visualizar la obtención de información de las fuentes abiertas en tiempo real.
2.7	Documentación final de los documentos entregables.

CUADRO 2.1: Descripción del cronograma de actividades.

Capítulo 3

Logros y productos del proyecto

Entre los logros y productos obtenidos durante la ejecución de este proyecto de grado se encuentran los siguientes:

1. Entendimiento de las generalidades de *Data Science*:
   - Tipos de *Machine Learning*.
   - Sistemas de detección de anomalías (*Anomaly Detection Systems*).
   - Diferentes modalidades de *Clustering*.
2. Identificación de modelos de NLP aplicables para el perfilado de cibercriminales.
3. Entendimiento de los modelos de clasificación y Clustering:
   - Clasificador de Naïve Bayes.
   - Maquinas de soporte vectorial (*Support Vector Machine*).
   - Mapas autoorganizados (*Self-organizing Maps*).
4. Entendimiento de los modelos utilizados en NLP:
   - Predicción de etiquetas con modelos de regresión lineal.
   - Reconocimiento de *Named Entities*.
   - Uso de *embeddings* generados con *StarSpace* para los k textos mas similares.
5. Propuesta de modelos de NLP para el perfilado de cibercriminales:
   - Modelo de predicción de hashtags de Twitter con modelos lineales junto con su implementación.
   - Modelo de reconocimiento de Named Entities con redes *Long Short Term Memory*.
   - Modelo de Clustering en redes SOM con *embeddings* de StarSpace.

Capítulo 4

Marco teórico

La Web contiene una gran cantidad de opiniones respecto a productos, políticos, y mucho mas, expresado en forma de noticias, sitios de opinión, reseñas en tiendas online, redes sociales. Como resultado, el problema de "Minería de opinión" ha obtenido una atención creciente en las ultimas dos décadas y es un factor decisivo para las nuevas organizaciones (como es mencionado en [2]). De esto mismo partimos que el análisis de textos para extraer el significado y demás componentes extraíbles del texto componen un factor que debe considerarse al momento de realizar decisiones, de manera que los avances hechos hasta ahora tienen como meta una aplicación practica de lo que se conoce como *Natural Language Processing*.

Luego de los ataques terroristas del 11 de Septiembre de 2001 en Estados Unidos, se realizaron fuertes criticas respecto a la inteligencia, donde el director del FBI llamado *Robert S. Mueller* indico que el principal problema que la agencia tuvo fue que se enfocaba demasiado en lidiar con el crimen luego de que fue cometido y ponía muy poco énfasis en prevenirlo (adaptado de [3]). Es por esto que el uso de NLP para temas de seguridad como también de metodologías de *Machine Learning* y *Deep Learning* han sido ampliamente utilizadas en ámbito de seguridad luego de estos eventos.

Para obtener una mejor inteligencia se necesito de mejores tecnologías a las que se tenían entonces (véase [3, pág 2]):

- Integración de datos (o *Data Integration* en ingles).
- Análisis de vínculos (o *Link Analysis* en ingles).
- Agentes de software (o *Software Agents* en ingles).
- Minería de texto (o *Text Mining* en ingles).
- Redes neuronales (o Artificial Neural Network (ANN) en ingles).
- Algoritmos de *Machine Learning* (o *Machine Learning Algorithms* en ingles).

4.1. Análisis de vínculos (*Link Analysis*)

Es la visualización de asociaciones entre entidades y eventos, por lo general involucran una visualización por medio de una gráfica o un mapa que muestre las relaciones entre sospechosos y ubicaciones, sea por medio físico o por comunicaciones en la red.

4.2. Agentes de software (*Software Agents*)

Es el software que realiza tareas asignadas por el usuario de manera autónoma, donde sus habilidades básicas son:

- **Realización de tareas:** Hacen obtención de información, filtrado, monitoreo y reporte.
- **Conocimiento:** Pueden usar reglas programadas, o pueden aprender reglas nuevas (véase 4.5).
- **Habilidades de comunicación:** Reportar a humanos e interactuar con otros agentes.

4.3. Aprendizaje de maquina (*Machine Learning*)

De acuerdo con [4], se define como un conjunto de métodos que pueden detectar patrones automáticamente en datos, y luego usar los patrones descubiertos para predecir los datos futuros, o realizar otra clase de toma de decisiones con un grado de incertidumbre, por tal motivo es necesario el uso de teoría de probabilidad, que puede ser aplicada a cualquier tipo de problema que involucra incertidumbre.

4.3.1. Tipos de *Machine Learning*

Machine Learning (ML) esta principalmente dividida en dos tipos. El método predictivo o bien **aprendizaje supervisado** (*Supervised Learning*), donde el objetivo es aprender un mapeo de las entradas $\boldsymbol{x}$ a las salidas y, dado un conjunto de pares de etiquetas de entrada–salida $D = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$. D se le llama el conjunto de entrenamiento y N es el numero de muestras de entrenamiento.

En la forma mas sencilla, cada entrada de entrenamiento $\boldsymbol{x}_i$ es un vector D–dimensional de números, a estos se le llaman *características* o *atributos*.

De manera similar la forma de la salida puede ser en principio cualquier cosa, pero la mayoría de métodos asumen que y_i es una variable *categórica* o *nominal* de algún conjunto finito, $y_i \in \{1, \ldots, C\}$, o que y_i es un escalar real, $y_i \in \mathbb{R}$. Cuando la variable y_i es categórica, al problema se le reconoce como **clasificación** o **reconocimiento de patrones**, y cuando es un valor real se le conoce como un problema de **regresión**.

El segundo tipo principal de ML es el descriptivo o **aprendizaje no–supervisado** (*Unsupervised Learning*), en este solo están disponibles los datos de entrada $D = \{\boldsymbol{x}_i\}_{i=1}^N$, y la meta es encontrar "patrones interesantes" en los datos. Este es un problema mucho menos definido, debido a que no se conocen los tipos de patrones que se quieren encontrar, y no hay una métrica obvia de error (no como aprendizaje supervisado en la que se puede comparar nuestra predicción de y para un $\boldsymbol{x}$ con el valor observado).

Un tercer tipo de aprendizaje de maquina es conocido como ***Reinforcement Learning***, el cual es un tipo menos usado. Este es útil cuando se quiere aprender como actuar o comportarse cuando se recibe una recompensa ocasional o una señal de castigo.

4.3.2. Sistemas de Detección de Anomalías (*Anomaly Detection Systems*)

Existen diferentes aproximaciones para los sistemas de anomalías, sin embargo una similitud entre todos estos sistemas es que se intenta realiza una *detección de desviaciones*, y su tarea es detectar los datos *atípicos* en un sistema [5].

Uno de los que se pueden encontrar en la literatura son los sistemas de detección de anomalías basados en realizar un estimado probabilístico con alguna distribución de probabilidad de donde para una serie de características m se trata de estimar una distribución gaussiana $\mathbf{X} \sim \mathcal{N}(\mu, \sigma^2)$ que tiene media μ y varianza σ^2 por cada característica, por lo que existirán m diferentes distribuciones .

Para estimar cada una las medias y variaciones de cada característica j, μ se estima con la Ecuación 4.1 y σ^2 se estima con la Ecuación 4.2.

$$\mu_j = \frac{1}{m}\sum_{i=1}^{m} x_j^{(i)} \tag{4.1}$$

$$\sigma_j^2 = \frac{1}{m}\sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2 \tag{4.2}$$

De donde para calcular la probabilidad de que una muestra se trata de una anomalía se calcula la probabilidad $p(x)$, luego de que fueron estimadas las distribuciones de cada característica del conjunto de entrenamiento, por lo que se define un ϵ de manera heurística, de forma que se determina que una muestra anómala si $p(x) < \epsilon$, la Ecuación 4.3 representa cual es la probabilidad de una muestra x con una distribución gaussiana.

$$p(x) = \prod_{j=1}^{n} p(x_j, \mu_j, \sigma_j^2) = \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right) \tag{4.3}$$

Si el modelo esta dando como resultado muchos falsos positivos, lo que se debe hacer es reducir σ.

Este método de sistemas de detección de anomalía también es brevemente tratada en [6] con una representación de hiper-planos, que es equivalente con este descrito.

4.4. Minería de datos (*Data Mining*)

Según [5], la minería de datos se define como el proceso de descubrir información útil en repositorios grandes de datos. Las técnicas de minería de datos son desplegadas para limpiar grandes bases de datos para encontrar patrones nuevos y útiles que de lo contrario podrían permanecer desconocidos. También ofrecen capacidades para predecir la salida de observaciones futuras, tales como predecir si un cliente nuevo gastara mas de $100 en una tienda.

No todas las tareas de descubrimiento de información son considerados como *Data Mining*. Por ejemplo, realizar una consulta de campos individuales usando un sistema de base de datos o encontrar una pagina web por medio de una búsqueda en Internet son tareas relacionadas con *adquisición de información*.

4.4.1. Minería de texto (*Text Mining*)

Es un subcampo de Inteligencia Artificial conocida como *Natural Language Processing*, en donde las herramientas de minería de datos pueden capturar rasgos críticos del contenido de un documento basado en el análisis de sus características lingüísticas.

La mayoría de los crímenes son electrónicos por naturaleza, por lo que se dejan rastros textuales que investigadores pueden seguir y analizar. Estas se enfocan en el descubrimiento de relaciones en texto no–estructurado y pueden ser aplicados al problema de *búsqueda* y *localización de palabras clave*.

4.4.2. Clasificación

Clasificación es la tarea de asignarle una de varias categorías predefinidas a objetos, y es una tarea que tiene una variedad extensa de aplicaciones. Ejemplos de esto se encuentran la detección de correos no deseados en mensajes de e–mails basándose del encabezado o el cuerpo del mensaje, categorización de células benignas de malignas basándose en los resultados de escaneados MRI o incluso la clasificación de galaxias basado en su forma.

Definido formalmente, clasificación es la tarea de aprender una función objetivo f que mapea cada conjunto de atributos x a una clase predefinida de etiquetas y.

La función objetivo también se define informalmente como un *modelo de clasificación*.

Metodologías de clasificación

Existen muchos métodos para la clasificación de datos no–estructurados, entre los descritos aquí están:

- Clasificador basado en reglas (véase la Sección 4.5).
- Redes neuronales artificiales (véase la Sección 4.6).
- Maquinas de soporte vectorial (véase la Sección 4.7).
- Clasificador de Naïve Bayes (véase la Sección 4.8.2).

4.4.3. Clustering

El análisis de clusters agrupa objetos de datos basándose únicamente en la información encontrada en los datos que describen los objetos y sus relaciones. El objetivo es que objetos dentro de un grupo sean similares (o relacionados) el uno al otro, y que sean diferentes (o sin relación) a objetos en otros grupos. Entre mayor sea la similitud dentro de un grupo y entre mayor sea la diferencia entre grupos, sera mejor o mas distintivo el clustering.

Los métodos de clustering se hacen referencia comúnmente en ML como métodos no–supervisados, los cuales se describen en 4.3. Un método de estos se describe en 4.6.1 conocidos como mapas autoorganizados.

4.5. Sistemas Basados en Conocimiento (*Knowledge Based Systems*)

Según [7], los Knowledge Based Systems (KBS) son uno de los mayores miembros de la familia de Artificial Intelligence (AI). El KBS consiste de una Knowledge Base (KB) y un programa de búsqueda llamado Inference Engine (IE) representado en la Figura 4.1. La KB puede ser usado como un repositorio de conocimiento de varias formas.

Existen 5 tipos de KBS, donde uno de ellos es conocido como *Expert Systems*, usados como Rule–based Systems (RBS), donde su KB esta dado como reglas y el IE esta dado por algo llamado Working Memory (WM), que representa los hechos que se conocen inicialmente del sistema junto con los hechos que se van dando como inferencia de las reglas.

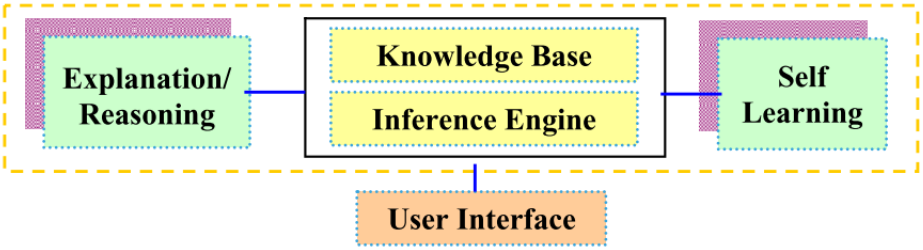


FIGURA 4.1: Arquitectura KBS. Tomado de [7]

Estas reglas pueden resumirse como una colección de condicionales de la forma **IF/ELSE** que se componen de un *antecedente* y un *consecuente*.

Existen dos tipos de RBS, definidos como *Deductive Systems* y *Reactive Systems*, donde el Deductive Systems tiene como objetivo realizar una conclusión en base a los hechos iniciales en la WM, por el otro lado se tienen los Reactive Systems, los cuales de igual manera a los Deductive Systems, toman los hechos de la WM y realizan sea una acción interactiva con su entorno o bien una modificación de los hechos que se encuentran en la WM tal como la adición o eliminación de hechos. Tómese el ejemplo de la Ecuación 4.4 tomada de [8], donde x es la temperatura y AC es aire acondicionado.

$$\begin{cases} \text{IF x es moderado,} & \text{THEN y = ajustar AC a bajo} \\ \text{IF x es alto,} & \text{THEN y = ajustar AC a moderado a alto} \\ \text{IF x es muy alto,} & \text{THEN y = ajustar AC a alto} \end{cases} \tag{4.4}$$

4.6. Redes Neuronales Artificiales (*Artificial Neural Network*)

El estudio de redes neuronales artificiales fue inspirado por los intentos de simular los sistemas biológicos de neuronas. El cerebro humano se compone principalmente de células nerviosas llamadas *neuronas*, enlazadas con otras neuronas por medio de hebras de fibra conocidas como *axones*. Los axones son usados para transmitir impulsos nerviosos de una neurona a otra cada vez que las neuronas son estimuladas. Una neurona esta

conectada a axones de otras neuronas por medio de *dendritas*, las cuales son extensiones desde el cuerpo de la neurona. El punto de contacto entre una dendrita y un axón se conoce como *sinapsis*. Los neurólogos han descubierto que el cerebro humano aprende por medio de cambiar la fuerza de la conexión sináptica entre las neuronas a través de estimulación repetitiva por el mismo impulso.

De manera análoga a la estructura del cerebro humano, una ANN se compone de una estructura interconectada de nodos y vínculos directos.

4.6.1. Mapa autoorganizado (*Self-organizing Maps*)

El objetivo principal de los Self-organizing Maps (SOM) es de transformar una patrón de entrada m–dimensional en un mapa discreto uni– o bi–dimensional, donde sus principales características es que es un algoritmo que se basa en Unsupervised Learning, es *Feedforward*, tiene una sola capa de neuronas donde su propósito es realizar Clustering y una reducción de dimensionalidad sobre los datos de una forma topológicamente ordenada.

Los SOM tienen tres características distintivas:

- **Competencia:** por cada patrón de entrada, las neuronas en la red competirán entre ellas para determinar un ganador.
- **Cooperación:** la neurona ganadora determina la ubicación espacial (vecinos) alrededor de donde otras vecinas también se verán estimuladas.
- **Adaptación:** la neurona ganadora como también sus vecinas tendrán sus pesos asociados actualizados, y se tiene que los vecinos entre mas cerca estén del ganador, mayor es el grado de adaptación.

El algoritmo de aprendizaje de SOM parte de primero inicializar los pesos de las o neuronas con pesos aleatorios pequeños de una distribución de probabilidad aleatoria o uniforme, donde cada vector de entrada se define como $\boldsymbol{x} = [x_1, \ldots, x_m]^\top \in \mathbb{R}^m$ y la entrada general de N patrones como $\boldsymbol{X}^{m \times N}$, el vector de pesos de la neurona i es $\boldsymbol{w}_i = [w_{i1}, \ldots, w_{im}] \in \mathbb{R}^{1 \times m}$, con la matriz de pesos $\boldsymbol{W}^{o \times m}$.

Para alcanzar el objetivo de *competencia*, se realiza por cada patrón de entrada x_i una comparación con cada uno de los pesos de las o neuronas y se establece la de menor distancia $\|x_i\|_p$ (típicamente la distancia Euclidiana o equivalentemente la norma L^2 e.g. $p = 2$), dejando un ganador winner, tal como en la Ecuación 4.5.

$$\mathrm{winner} = \arg\min_j \|x_i - w_j\|_p \, ; j = 1, \ldots, o \tag{4.5}$$

Luego de establecer la neurona ganadora, se realiza el paso para alcanzar la *cooperación*, que consiste en que por medio de una función kernel h (típicamente una una distribución gaussiana), que permite establecer un área de afectación de las otras neuronas según su ubicación física en el mapa, definidos como r_{winner} y r_j que son la ubicación de la neurona ganadora y la neurona vecina j, en el cual el grado de afectación de la neurona vecina depende de la distancia L^2 de la que esta de la neurona ganadora, definido en la Ecuación 4.6.

$$h_{j,\mathrm{winner}}(t) = \exp\left(\frac{-\|r_j - r_{\mathrm{winner}}\|^2}{2\sigma(t)^2}\right) \tag{4.6}$$

Parte importante del proceso de convergencia del SOM es que a medida que avanzan las iteraciones t del algoritmo el área de afectación se va reduciendo como parte del proceso de adaptación, por lo que definimos $\sigma(t) = \sigma_0 \exp(-t/\tau_1)$, donde τ_1 es una constante heurística y σ_0 la dimensión del mapa SOM.

Finalmente para alcanzar la *adaptación* se realiza una actualización de los pesos de la matriz $\boldsymbol{W}$ en base a la influencia de área $\sigma(t)$ y de una tasa de aprendizaje $\alpha(t) = \alpha_0 \exp(-t/\tau_2)$, donde τ_2 es otra constante heurística y α_0 es una constante de aprendizaje inicial, que debe ser $0 \leq \alpha_0 \leq 1$, la actualización se describe por la Ecuación 4.7 y el proceso puede ser visto gráficamente en la Figura 4.2, tanto de forma uni– como bi–dimensional.

$$w_j(t+1) = w_j(t) + \alpha(t) h_{j,\text{winner}}(t) \Big[x_i - w_j(t) \Big] \tag{4.7}$$

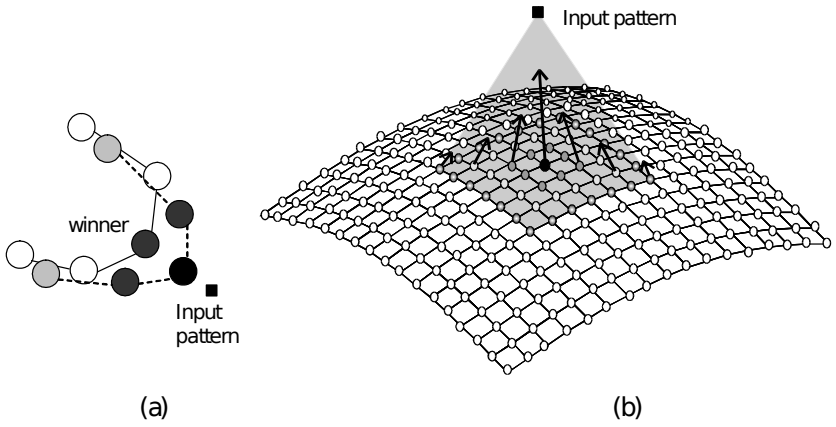


FIGURA 4.2: Proceso de adaptación de SOM, (a) uni–dimensional, (b) bi–dimensional. Tomado de [9]

Luego de que el algoritmo de aprendizaje termina de realizar las iteraciones, la salida de este es la matriz de pesos $\boldsymbol{W}$, en la Figura 4.3 se puede apreciar una aproximación del algoritmo con un mapa uni–dimensional tratando de aproximar una función polar con ruido adicionado en un gráfico 2D. Adicionalmente pueden verse los efectos de *High Bias* y *Overfitting* con diferentes cantidades de neuronas en la Figura 4.4.

En la Figura 4.5 se puede ver una aplicación de los SOM, en donde se realiza una clusterizacion de casos de homicidios donde los parámetros son características de los homicidios, según [3] este resultado da una buena aproximación para sospechar de que estos son cometidos por personas distintas o si bien están siendo perpetrados por un mismo individuo o grupo.

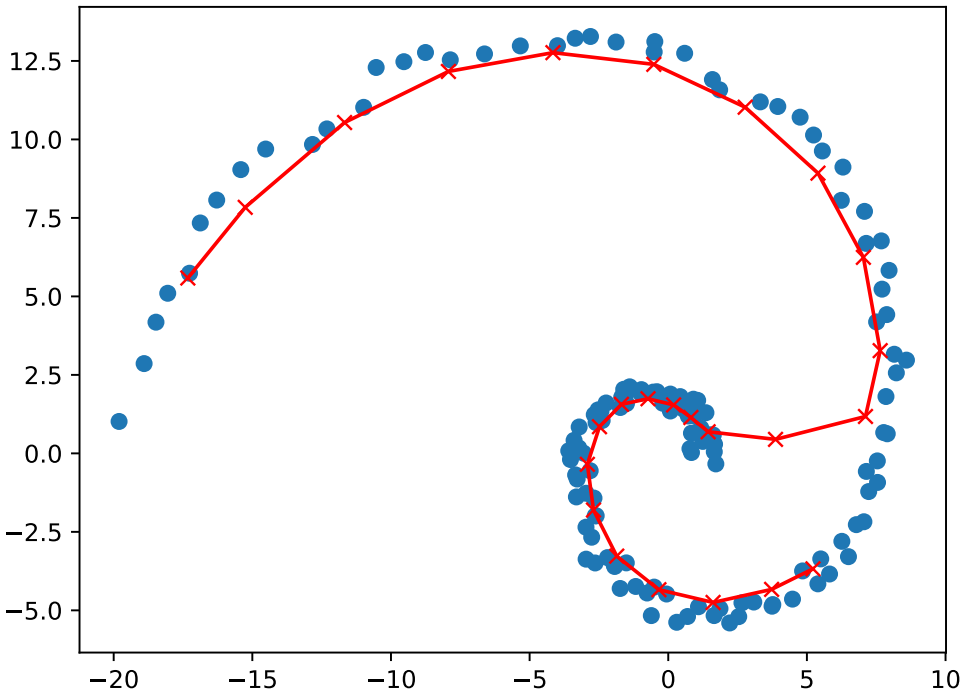
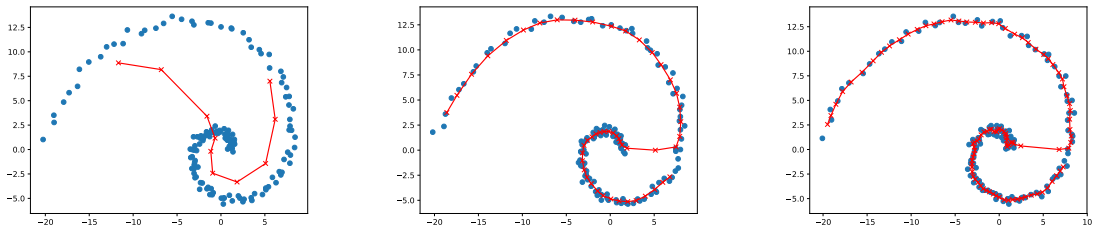


FIGURA 4.3: Ejemplo de salida de SOM uni-dimensional con 25 neuronas. Implementación propia.



(a) (b) (c)

FIGURA 4.4: Comparación de salidas de SOM uni-dimensional con (a) 10 neuronas (b) 50 neuronas y (c) 100 neuronas. Implementación propia.

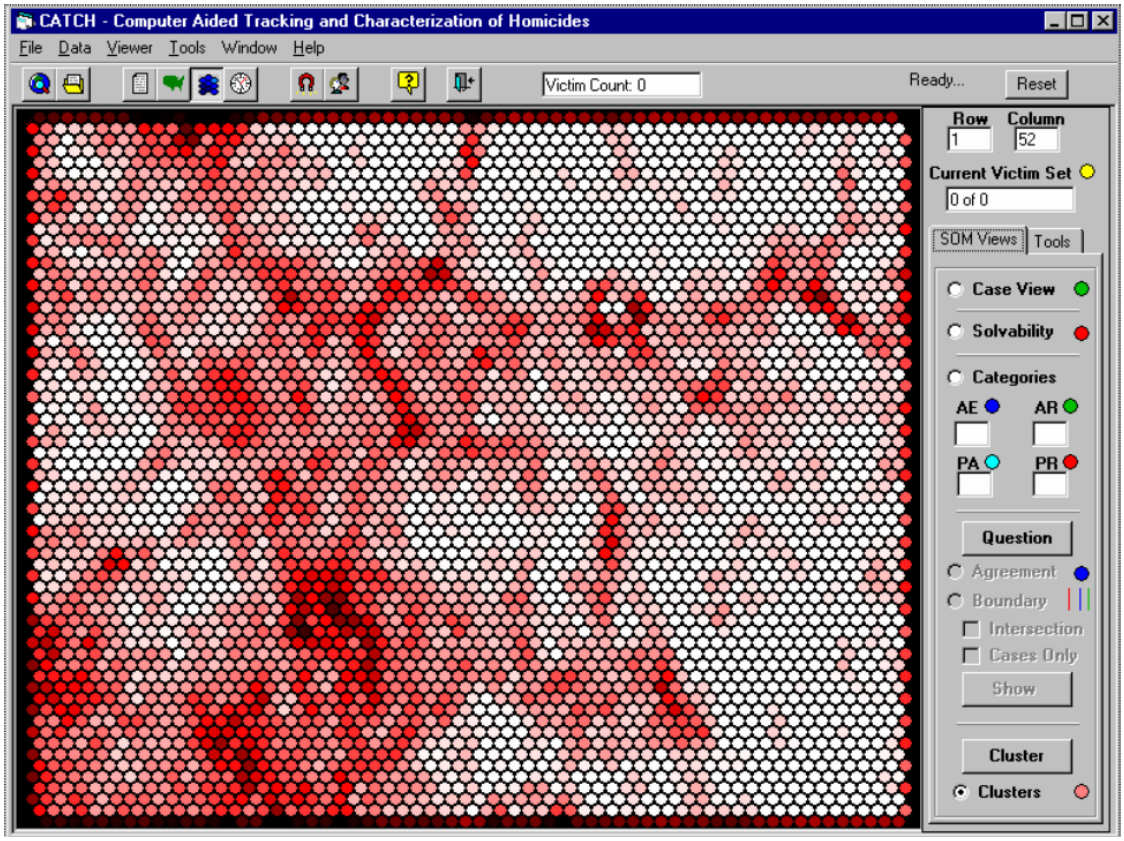


FIGURA 4.5: Ejemplo de uso de SOM en aplicaciones de perfilado. Tomado de [3]

4.7. Maquina de soporte vectorial (*Support Vector Machine*)

Support Vector Machine (SVM) es una técnica de clasificación que tiene sus raíces en la teoría de aprendizaje estadístico que ha mostrado resultados empíricos prometedores en muchas aplicaciones practicas, desde reconocimiento de dígitos escritos a mano a categorización de texto. SVM también funciona muy bien con datos de alta dimensionalidad. Otro aspecto destacable de esta aproximación es que representa la frontera de decisión usando un subconjunto de las muestras de entrenamiento, conocidos como los *support vectors*.

4.7.1. Maximum Margin Hyperplanes

Se puede entender a los *Maximum Margin Hyperplanes* como hiper-planos que ayudan a separar datos en un hiper-espacio y que poseen un margen de decisión entre los datos, como ejemplo tómese la Figura 4.6, donde el hiper-plano B_1 tiene un margen de decisión mas grande que el hiper-plano B_2.

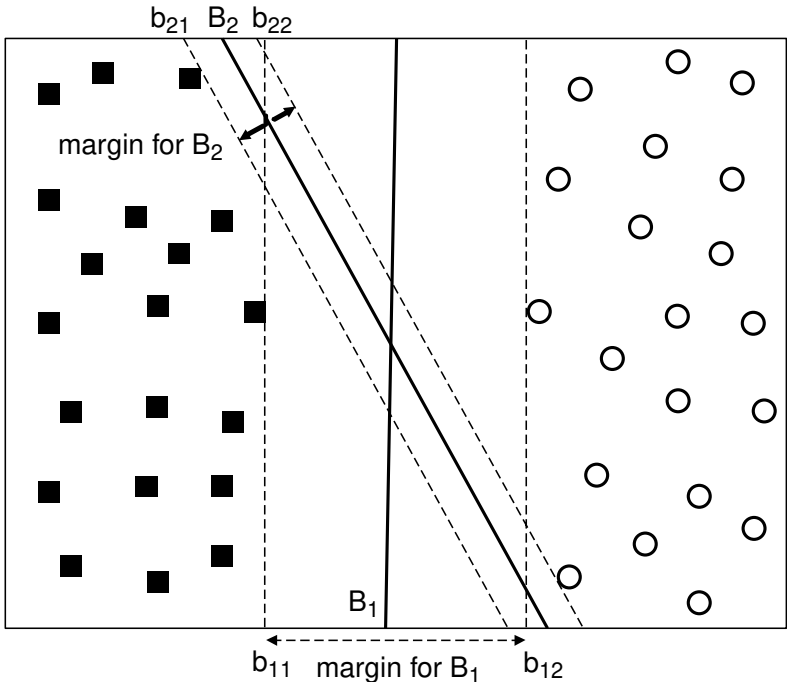


FIGURA 4.6: Maximum Margin Hyperplanes. Tomado de [5]

Finalmente, el objetivo final de los SVM es la búsqueda de un hiper-plano con el mayor margen de decisión. Existen dos tipos de SVM, el lineal y el no–lineal. El lineal realiza la separación de los datos con su hiper-plano a partir de los datos de entrada en su espacio vectorial original, mientras que el no–lineal consta de realizar una transformación de los espacios de los datos de entrada a uno en que sea linealmente separable (véase como ejemplo la Figura 4.7), sin embargo al realizar la transformación, el algoritmo de SVM se ve afectado por la dimensionalidad de la entrada, por lo que existe lo que se conoce como la función *kernel* para remediarlo.

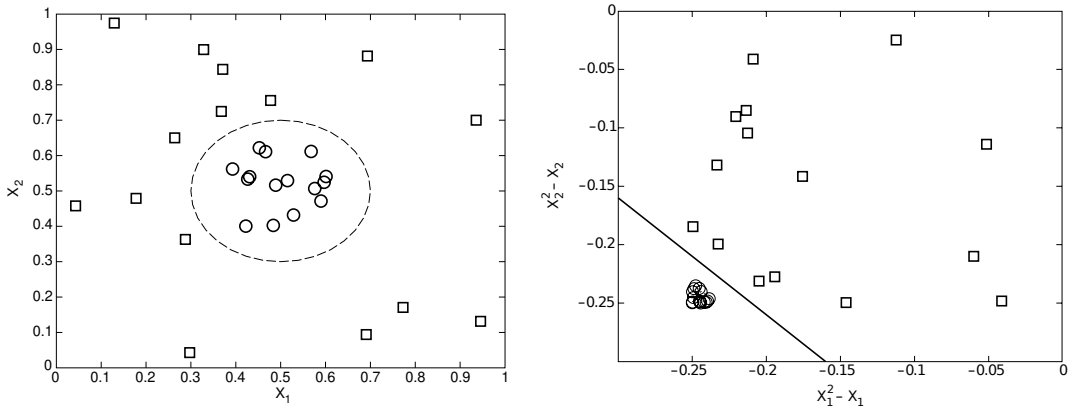


FIGURA 4.7: Transformación de espacios en *Support Vector Machine*. Tomado de [5]

4.7.2. Función Kernel

La función polinomial de similaridad, K, la cual es calculada en el espacio original de los datos de entrada, se le conoce como la **función Kernel**. En principio se asegura que la función kernel puede ser expresada siempre como el producto punto entre dos vectores de entrada en algún espacio de alta dimensionalidad, la función de kernel también tiene la particularidad de que el computo de los productos punto con la función toman considerablemente menos tiempo que realizar la transformación de espacios, dejando de lado la transformación, acelerando la tarea de clasificación.

4.8. Clasificadores Bayesianos

En muchas aplicaciones de relaciones entre el conjunto de atributos y la etiqueta es no–determinante. Es decir, la etiqueta de clase de un dato de un conjunto de prueba no puede ser determinado con certeza a pesar de ser un atributo idéntico a los atributos de entrenamiento. Esto puede ser producto de que los datos poseen ruido o la presencia de ciertos factores que afectan la clasificación pero no son incluidos en el análisis. Para esto es crucial el teorema de Bayes, el cual es un principio estadístico que combina el conocimiento previo de las clases con la nueva evidencia que se obtiene de los datos.

4.8.1. Teorema de Bayes

El teorema de Bayes dice que para un par de variables aleatorias x e y, donde $P(\mathrm{x} = x \mid \mathrm{y} = y)$ es la probabilidad de que la variable x tome el valor x dado que el valor de la variable y es y. Se tiene entonces la Ecuación 4.8.

$$P(\mathrm{y} \mid \mathrm{x}) = \frac{P(\mathrm{x} \mid \mathrm{y})P(\mathrm{y})}{P(\mathrm{x})} \tag{4.8}$$

Usando el teorema de Bayes para clasificación

Para denotar el problema de clasificación desde una perspectiva estadística se define a x como el conjunto de atributos y y como el conjunto de etiquetas de clase. Si la etiqueta de clase tiene una relación no–determinante con los atributos, entonces se pueden tomar a x y a y como variables aleatorias y capturar su relación probabilística con $P(\mathrm{y} \mid \mathrm{x})$, conocida como la probabilidad posterior para y, dada su probabilidad previa $P(\mathrm{y})$.

Durante la fase de entrenamiento, es necesario aprender las probabilidades posteriores $P(\mathrm{y} \mid \mathrm{x})$ para cualquier combinación de x y y basándose en la información recolectada de los datos de entrenamiento.

Dado que lo que se quiere realizar es una clasificación que represente la probabilidad de que dado un valor de $\mathrm{x} = x$ este relacionado con que $\mathrm{y} = y$, se puede reconocer primero que x se mantiene constante para lo que son los datos de entrenamiento, y que lo desconocido sea la clasificación $\mathrm{y} = y$ con probabilidad $P(\mathrm{y} \mid \mathrm{x})$, al conocer esta probabilidad, un valor de prueba x' puede ser clasificado por medio de encontrar la clase y' que maximice la probabilidad posterior $P(\mathrm{y}' \mid \mathrm{x}')$.

4.8.2. Clasificador Naïve Bayes

Un clasificador de Naïve Bayes estima la probabilidad condicional de las clases por medio de suponer que los atributos son condicionalmente independientes, dado la etiqueta de clasificación y. La suposición de independencia condicional se puede dar por la Ecuación 4.9.

$$P(\mathbb{X} \mid \mathrm{y} = y) = \prod_{i=1}^{d} P(\mathrm{x}_i \mid \mathrm{y} = y) \tag{4.9}$$

Donde cada conjunto de atributos $\mathbb{X} = \{x_1, \ldots, x_d\}$ que consiste de d atributos.

Como funciona el clasificador Naïve Bayes

Con la suposición de independencia condicional, en vez de computar la probabilidad condicional de clases para cada combinación de $\mathbb{X}$, solo se debe realizar para establecer la probabilidad condicional de cada x_i, dado y.

Para clasificar un dato de prueba, el clasificador computa la probabilidad posterior para cada clase y como se muestra en la Ecuación 4.10.

$$P(\mathrm{y} \mid \mathbb{X}) = \frac{P(\mathrm{y}) \prod_{i=1}^{d} P(x_i \mid \mathrm{y})}{P(\mathbb{X})} \Rightarrow P(\mathrm{y}) \prod_{i=1}^{d} P(x_i \mid \mathrm{y}) \tag{4.10}$$

Puede ignorarse $P(\mathbb{X})$ debido a que es un termino constante. Para esto se realiza una normalización de forma que $\sum_{\forall \mathrm{y} \in \mathbb{Y}} P(\mathrm{y} \mid \mathbb{X}) = 1$.

Capítulo 5

Estado del arte

Dentro de la literatura se encuentran diversos métodos específicamente para el perfilamiento de cibercriminales, como también métodos genéricos para el análisis de la información, aquí se exponen varios trabajos relacionados con estas tareas.

En [10] se propone un método de análisis automático de forums del bajo mundo. Ellos usan procesamiento de lenguaje natural y *Machine Learning* para generación de información de alto nivel sobre foros del bajo mundo, primero identificando publicaciones que involucran transacciones y luego extrayendo los productos y precios. También demuestran como un analista puede usar estas metodologías automatizadas para investigar otras categorías de productos y transacciones.

En [11] se introducen embeddings de palabras bilingües, es decir, representaciones de palabras asociadas entre dos lenguajes en el contexto de modelos de lenguaje neuronal. Ellos proponen un metodo para el aprendizaje de un gran corpus sin etiquetas. Estos nuevos embeddings mejoran significativamente en similaridad semántica.

En [12] desarrollan un método de minería de redes de cibercriminales poco supervisado para facilitar la forensia de cibercrimenes. El método propuesto es un modelo generativo probabilístico por un algoritmo de muestreo sensible al contexto y muestran un mejora significativa respecto a un método con *Latent Dirichlet Allocation* (LDA) y otro método basado en SVMs.

En [13] se realiza una visualización de categorías sacadas de Wikipedia de forma que se muestran sus relaciones con ayuda del meta-modelo SOM, ayudando así realizar una reducción de los espacios de búsqueda. Por medio de la selección de una neurona especifica era posible la obtención de categorías conceptualmente similares. La evaluación de las activaciones de neuronas individuales indicaban que formaban patrones de forma coherente que podrían ser útiles en la construcción de interfaces de usuario para la navegación sobre estructuras categóricas.

En [14] exploran diversas técnicas computacionales para realizar el Clustering de grupos de lenguajes con la categorización de participantes de foros de cibercriminales. Ellos hacen uso de un modelo de red neuronal de lenguaje para generar representaciones de vectores de tamaño fijo de mensajes publicados por los participantes de los foros. Ellos afirman que sus resultados muestran que Vectores de párrafos superan a las aproximaciones tradicionales de frecuencia de n–gramas para generar embeddings de documentos que son útiles en la clusterizacion de cibercriminales en grupos de lenguajes.

En [15] realizan clusterización de documentos HTML con lógica difusa, de donde

describen que en los resultados experimentales muestran una mejora significativa respecto a modelos de espacio vectorial con funciones tradicionales de pesado de parámetros en un dataset estándar de pruebas.

Capítulo 6

Propuesta de aplicación

La propuesta consta de a aplicación de modelos de NLP para el desarrollo de labores de inteligencia en escenarios de ciberterrorismo por medio de 3 modelos, pensados para el análisis de texto en redes sociales como Twitter en aras de realizar un perfilado de ciber-criminales potenciales, esto por medio de NLP, de donde se parte varias metodologías que hacen uso de tecnologías Estado-del-Arte.

Para esto se realiza el ciclo Cross Industry Standard Process for Data Mining (CRISP-DM), de donde se encuentran las fases de (se tomaron cuatro de las seis fases):

1. Business understanding
2. Data acquisition
3. Modelling
4. Deployment

6.1. Entendimiento del negocio (Business understanding)

En el proceso de perfilado una tarea muy importante se basa en la búsqueda de pistas (como diversos ejemplos dados en [3]), por lo que la mayor parte de este proceso confiere de tratar de encontrar patrones que confieran algún tipo de relación entre lo que es un actor y una acción, si bien no necesariamente de forma directa. Es por esto que técnicas como las que se encuentran en la minería de texto confieren una posibilidad de realizar muchas de estas labores, tales como el Clustering de elementos, como el caso de la Figura 4.5 en la Sección 4.6.1.

La necesidad cae en las agencias de seguridad que requieren de estar constantemente buscando posibles relaciones entre ataques pasados o futuros, y por tanto nuevas herramientas que permitan obtener nuevas habilidades en la esta labor son esenciales.

Finalmente, metodologías de análisis de textos a partir de fuentes abiertas, tales como en [6], permitirían no solo obtener la información, sino que es necesario tener una forma de analizarla e incluso predecir relaciones a partir del contenido que no es explícitamente dicho.

6.2. Adquisición de datos (Data acquisition)

Para esta labor de adquisición de datos, la intención es de recolectar datos de diversas fuentes, sin embargo para el alcance del proyecto se proponen dos metodologías de obtención de información de fuentes abiertas.

6.2.1. Obtención de datos con API de Twitter

Twitter permite el acceso de datos de su plataforma por medio de códigos de autorización provistos en ella, que luego son usados para obtener la información con criterios definidos por el usuario.

Petición de autorización en Twitter

Para realizar una petición de datos de acceso a la plataforma es necesario crear una "App" por medio de visitar la pagina de Administración de aplicaciones (o *Application Management* en ingles). Se necesita una cuenta de Twitter previamente creada.

Luego se procede a crear la App, para esto sera necesario proveer información respecto a que se hará con la aplicación y que datos requerirá esta. En este proceso es importante tener en cuenta que debido al uso masivo de información dentro de Twitter, este ha puesto restricciones al acceso por temas de privacidad e influencia, es por esto que el proceso para pedir autorización de crear una nueva aplicación debe incluir la mayor cantidad de información que provea la intención del uso. Luego de llevar a cabo la solicitud luego de aproximadamente un día le sera notificado si fue autorizado para utilizar la plataforma.

En este punto podrá crear la aplicación y generar los tokens de autorización, de los cuales requerirá de:

- `CONSUMER_KEY`
- `CONSUMER_SECRET`
- `ACCESS_TOKEN`
- `ACCESS_SECRET`

Esta petición de datos se realiza por medio de una consulta `GET` donde la URI provee los datos como los codigos de autorización.

Dentro del directorio de Software de este proyecto se incluye un programa que permite la obtención de tweets con todos meta-datos ofrecidos por la plataforma, de manera que obtiene tweets con temas relacionados especificados en un archivo de texto externo.

6.2.2. Obtención de datos con datasets públicos en Archive

Otra posibilidad de obtener datos en una plataforma abierta es la de la pagina web de ARCHIVE[1].

En esta plataforma se puede descargar meses de contenido obtenido de cualquier origen en Twitter en el tiempo de un mes, aunque este información no fue obtenida con niveles altos de precisión, es decir, no se obtuvieron todos los tweets que ocurrieron ese mes ni tampoco fueron obtenidos en un ambiente de alta disponibilidad, por lo que no todos los tweets en ese instante fueron recolectados.

Una forma recomendada de obtener los datos es por medio de *torrents*, que permite una descarga fiable del contenido comprimido de ese mes[2] ya que además provee comprobación de integridad y descarga de secciones de los archivos corruptas. Un motor

[1] https://archive.org/search.php?query=collection%3Atwitterstream&sort=-publicdate&page=2

[2] Cada mes pesa aproximadamente 50GB

de torrents Open Source es el de qBittorrent[3], y tiene implementaciones en Windows, MacOS y Linux.

Luego de haber descargado todo el archivo comprimido de tweets se puede proceder a descomprimirlo, de donde se encontraran archivos comprimidos mas pequeños divididos por día del mes, sin embargo téngase en cuenta que descomprimir estos archivos puede llegar a pesar 500GB, por lo que una alternativa mas conveniente es de realizar un filtrado de los datos relevantes que se necesiten de manera individual por cada uno de esos archivos, de esta forma solo se tendría la información requerida, en vez de ruido. Para esto existe una aplicación incluida en la carpeta de Software de este proyecto con un posible uso, donde también se hace uso de [16] para el procesamiento mas eficiente.

6.3. Modelamiento (Modelling)

Como parte de la propuesta se proponen 3 modelos para tratar diferentes aspectos en perfilado de donde se representan los diferentes modelos en la Figura 6.1.

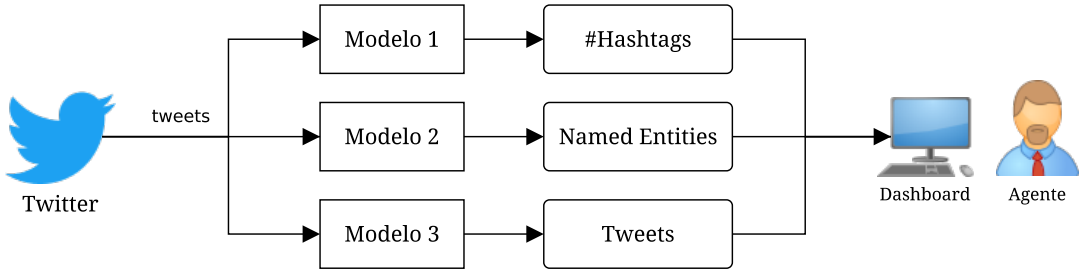


FIGURA 6.1: Arquitectura de propuesta.

6.3.1. Modelo 1: Predicción de etiquetas de Twitter con modelos lineales

En Twitter, las publicaciones que se realizan tienen la posibilidad de incluir menciones de temas de tendencia por el conocido *hashtag*, escrito como `#Tema`, y tiene la gran utilidad de realizar una mención explicita del tema que se quiere tratar y donde además la tarea de encontrar textos directamente relacionados con un tema son fácilmente localizables.

Así mismo, en la literatura de NLP es muy común el uso de diferentes representaciones de palabras o conjuntos de palabras. Una representación de palabras típicas es por medio de los Bag of Words (BOW), donde se establece un diccionario de palabras de tamaño N, y donde cada palabra tiene un vector que lo representa. A cada palabra se le asigna un identificador único en ese diccionario, por lo que existiría una traducción de palabra a identificador y una secuencia de palabras para poder ser recuperado por medio del índice, como se muestra en la Ecuación 6.1 y la Ecuación 6.2.

$$\text{word2idx} = \Big\{(\text{word}_i, i) : \forall i \in \{1, \ldots, N\}\Big\} \tag{6.1}$$

[3] https://www.qbittorrent.org/

$$\text{idx2word} = \Big[\text{word}_i\Big], \forall i \in \{1, \ldots, N\} \tag{6.2}$$

Otra representación común en NLP es la de Term Frequency – Inverse Document Frequency (TF-IDF), que se divide en dos partes, expresadas en las Ecuación 6.3, la Ecuación 6.4 y la composición de ambas en la Ecuación 6.5, D es el corpus de palabras. Este consiste en penalizar palabras que ocurren mucho en un documento pero no mucho en el corpus o bien de penalizar la palabras que se repiten poco en un documento pero se repiten mucho en el corpus, por lo que un punto medio entre ambos es recompensado.

$$\text{tf}(t, d) = \text{Frecuencia del termino (o n–grama) } t \text{ en el documento } d \tag{6.3}$$

Existen diferentes variaciones para realizar representar el conteo de términos **tf** de forma normalizada, como se representa en el Cuadro 6.1.

Esquema	**Peso de tf**
Binario	$0, 1$
Conteo directo	$f_{t,d}$
Frecuencia de términos	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
Normalización logarítmica	$1 + \log(f_{f,d})$

CUADRO 6.1: Variaciones de **tf**

$$\text{idf}(t, D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right); N = |D| \tag{6.4}$$

$$\text{tf-idf}(t, d, D) = \text{tf}(t, d) \cdot \text{idf}(t, D) \tag{6.5}$$

Creación del Corpus de palabras

Debido a que los textos que se encuentran en Twitter no tienen forma estructurada es necesario realizar un preprocesamiento de cada post recolectado para luego contar la frecuencia de cada palabra dentro del corpus y así establecer las primeras N palabras mas usadas que van a componer el corpus de palabras.

Para realizar el preprocesamiento de cada texto se hacen los siguientes pasos:

1. Convertir todas las palabras a minúscula (e.g. “LaTeX” → “latex”)
2. Reemplazar todos los caracteres especiales de texto a espacios en blanco (e.g. “`@;,:\n\t\r`” → “␣␣␣␣␣␣␣”)
3. Remover todos los símbolos extraños, es decir todo lo que no sean numeros, ni letras ni los simbolos que se encuentran normalmente en tweets (e.g. “ `%()&$!^` ” → “”)
4. Remover todas las *stopwords*, que son palabras que no añaden ningún valor semántico al texto
   (e.g. “las palabras son una forma de expresarnos” → “palabras forma expresarnos”)

Luego se realiza un conteo de todas la palabras presentes dentro del corpus de donde se sacan las N primeras palabras para incluirlas en el BOW.

Luego de esto se generan las etiquetas (o *tags* en ingles) que se toman directamente de los textos de entrenamiento, de estos también se les realiza un conteo, que servirán para la predicción de las etiquetas según el contenido del texto.

Conversión de textos a vectores

Para realizar la conversión de textos a vectores y así poder representar un texto como un vector se procede a primera realizar una generación de identificadores para cada palabra de manera como se describió en el inicio de la Sección 6.3.1.

La manera en que se representa un texto en forma de vector $\boldsymbol{s}$ es por medio de la sumatoria de los vectores que representan cada palabra como se representa en la Ecuación 6.6, recuérdese que $\boldsymbol{e}^{(i)}$ es un vector con un 1 en la posición i y ceros en el resto del vector.

$$\boldsymbol{s} = \sum_{(\text{word},i)\in\text{word2idx}} \boldsymbol{e}^{(i)}, \text{word} \in d \tag{6.6}$$

Generación del conjunto de entrenamiento, validación y pruebas

Para la generación de los conjuntos se toman las sumatorias generadas de cada tweet en su forma de vector y se coloca en una matriz de $\boldsymbol{T}^{m\times N}$, donde m son el numero de muestras de Twitter y N el tamaño del Corpus.

Clasificador de regresión logística

La regresion logistica hace parte de uno de los algoritmos mas importantes en AI, esta consiste en procesar la entrada de un modelo, que para el caso actual es lineal, que se procesa con una serie de hiper-parámetros que se denominara $\boldsymbol{\theta}$ y la entrada del modelo como $\boldsymbol{x}$. Se tiene que al realizar una regresión con este modelo se calcula $\boldsymbol{\theta}^\top \boldsymbol{x}$, que da como resultado un valor en $\mathbb{R}$ con rango indefinido. La regresión logística simbolizada como $\sigma(x)$, conocida como la función *sigmoide* que es una función que para una entrada x de dominio $(-\infty, \infty)$ se tenga un rango de $(0, 1)$, la Ecuación 6.7 define la función.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{6.7}$$

Equivalentemente, la regresión logística con modelos lineales se calculan como $\sigma(\boldsymbol{\theta}^\top \boldsymbol{x})$ y permite realizar un suavizado de la regresión de forma mitiga parte de los problemas de Overfitting y High Bias, como se muestra en la Figura 6.2.

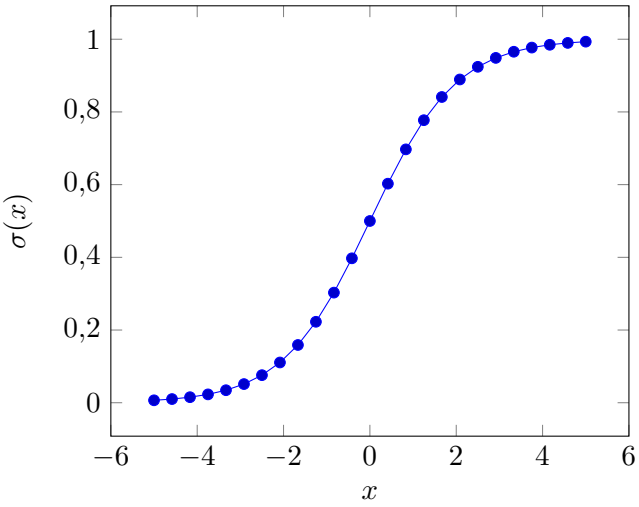


FIGURA 6.2: Gráfica de función sigmoide.

Multiclasificador de One vs Rest

La multiclasificacion como es brevemente introducido en la Sección 4.3, permite realizar una clasificación en C clases con una entrada $\boldsymbol{x}$. El multiclasificador de *One vs Rest* realiza la clasificación por medio de distinguir la separación de una muestra $\boldsymbol{x}_i$ en una clase $c \in \{1, \ldots, C\}$ respecto al resto de muestras, de forma que se calcula la pertenencia de la muestra i-ésima en esa clase con un estimador $\boldsymbol{\theta}_c$. De los resultados dados por cada estimador se estima cual es el k-ésimo estimador que da el máximo valor, de donde se estima finalmente que la clase k es donde pertenece la muestra.

El uso de la regresión logística se puede llevar a cabo para normalizar los resultados de los estimadores $\boldsymbol{\theta}$, de donde el proceso para calcular el estimador consta de realizar un proceso de gradiente descendente, que utiliza como función de costo la Ecuación 6.8 para uso de regularización con L^2 y la Ecuación 6.9 para el uso de L^1, basadas del manual de [17]. La función ζ es la función *softplus* (véase la Notación).

$$\min_{\boldsymbol{\theta},c} \frac{1}{2}\boldsymbol{\theta}^\top \boldsymbol{\theta} + C \sum_{i=1}^{n} \zeta(-y_i(\boldsymbol{X}_i^\top \boldsymbol{\theta} + c)) \tag{6.8}$$

$$\min_{\boldsymbol{\theta},c} \|\boldsymbol{\theta}\|_1 + C \sum_{i=1}^{n} \zeta(-y_i(\boldsymbol{X}_i^\top \boldsymbol{\theta} + c)) \tag{6.9}$$

En el caso para realizar una multiclasificación, la regresión logística solo puede realizar una clasificación binaria, esto es que para una entrada $\boldsymbol{x}$, esta solo puede realizar una clasificación con salidas $y \in [0, 1]$. Debido a esta restricción, existen una metodología de multiclasificacion llamada *One vs Rest*, que permite utilizar la regresión logística y realiza la multiclasificacion por medio de crear n estimadores que permiten estimar cada una de las C clases respecto al resto de la entrada, por cada una de los estimadores se estima cual es la clase con mayor valor en la regresión logística individual de ese estimador, la Figura 6.3 representa una versión simplificada del proceso.

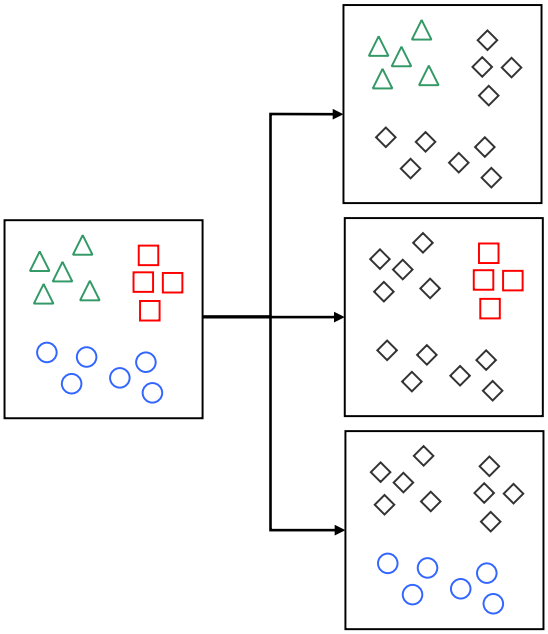


FIGURA 6.3: Algoritmo de One vs Rest.

La manera en que se entrena cada estimador $i \in \{1, \ldots, C\}$ es por medio de optimizar el valor de la función de predicción $\hat{y}(\boldsymbol{x}, \boldsymbol{\theta}_i)$, $\boldsymbol{x} \in \mathbb{X}$ (Ecuación 6.10), como se muestra en la Ecuación 6.11, y es el valor verdadero de la estimación y θ_0 se le conoce como el *bias* del modelo.

$$\hat{y}(\boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}^\top \boldsymbol{x} = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \tag{6.10}$$

$$\boldsymbol{\theta}_i = \arg\min_{\boldsymbol{\theta}_i} |\hat{y}(\boldsymbol{x}, \boldsymbol{\theta}_i) - y| \tag{6.11}$$

Luego de haberse entrenado cada estimador con sus valores óptimos, se puede realizar la predicción de las clases a las que pertenece una entrada $\boldsymbol{x}$ con la Ecuación 6.12.

$$c = \arg\max_i \ \sigma(\hat{y}(\boldsymbol{x}, \boldsymbol{\theta}_i)) \tag{6.12}$$

Predicción de hashtags

El uso que se le puede dar a todo lo mencionado anteriormente en esta sección es de predecir etiquetas de Twitter típicas de terrorismo a partir del texto que se encuentra en el tweet con ayuda de alguna de las representaciones de BOW o TF-IDF, tal como se muestra en el ejemplo de la Figura 6.4.

"Really excited to add @plaidavenger to my deathlist along with Italy and @Plaid_Obama after receiving that information."	⇒	**#deathlist, #KillEveryone, #ISIS**

FIGURA 6.4: Predicción de hashtags con modelos lineales.

De manera que se determinan los C hashtags de los datos de entrenamiento $\mathbb{X}$ para luego poder realizar la prediccion de los hashtags a partir un texto de entrada d que luego se convierte a representacion BOW o TF-IDF para pasarlo por los C estimadores del *One vs Rest* y estimar las etiquetas predecidas recuperandolas como la Ecuación 6.13 de manera de diccionario indexado por el numero de la clase $i \in \{1, \ldots, C\}$ como llave y el hashtag como el valor.

$$\{(i, h)\} \; ; \; h \in \text{hashtags} \tag{6.13}$$

6.3.2. Modelo 2: Reconocimiento de *Named Entities* con redes *Long Short Term Memory*

Igual que en el modelo presentado en la Sección 6.3.1, en Twitter como en cualquier otra red social se presentan en sus textos muchas veces la mención de las llamadas Named Entities. Son objetos del mundo real, tales como personas, ubicaciones, organizaciones, productos, entre otros que pueden ser denotados con nombre propio, y pueden ser abstractos o tener una existencia física.

En consecuencia, este modelo tiene como propósito reconocer Named Entities que puedan dar una mejor aproximación a entender el contexto de conversaciones en masa de cibercriminales identificados o bien realizar una identificación de quienes son en base de cuales son los temas que tienden a mencionar mas habitualmente.

Redes neuronales recurrentes (*Recurrent Neural Network*s)

Este tipo de redes tiene la característica de recordar salidas y entradas pasadas de datos, de forma que decisiones futuras respecto a una clasificación pueden tener una mejor forma de basarse. Las redes neuronales (ANNs) típicamente no pueden hacer estas operaciones ya que no tienen incorporada la habilidad de forma que recuerden entradas y salidas pasadas, sino que se les realiza un entrenamiento de una información de entrada y de salida que se conoce con anticipación, de forma que su salida solo se concentra en su entrada inmediata de datos.

Por este motivo, las *Recurrent Neural Networks* (RNNS) poseen una gran ventaja en situaciones puntuales, tales como el caso expuesto al inicio de esta sección, sin embargo, el modelo clásico de RNN es como la arquitectura presentada en la Figura 6.6, esta no provee resultados significativos debido a limitantes respecto al nivel de recordación que tienen originalmente.

La naturaleza recurrente de estas redes viene del hecho de representarse como redes donde su salida hace parte de la entrada a la siguiente iteración de la red, tal como se aprecia en la Figura 6.5.

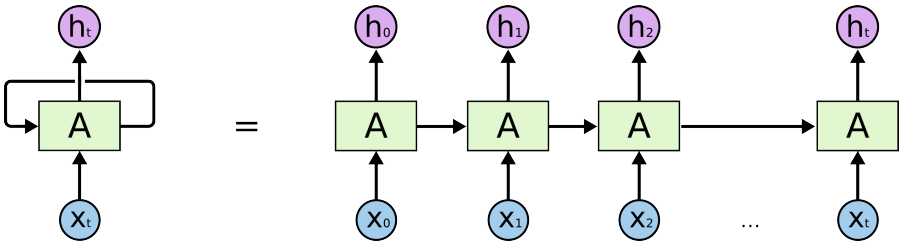


FIGURA 6.5: Red RNN simplificada. Tomado de [18].

Redes neuronales *Long Short Term Memory*

En la literatura se puede encontrar usualmente el uso de las redes Long Short Term Memory (LSTM), las cuales proveen por medio de una arquitectura mas compleja la posibilidad de tener recordación de largo y corto plazo, como se muestra en la Figura 6.7. Para que estas redes tengan la característica de tener recordación es necesario de una celda de memoria, donde esta es la entrada para la siguiente iteración de la red, junto con la entrada de datos a la red.

Téngase en cuenta que la arquitectura presentada en esta sección hace referencia a la implementación de LSTM mas utilizada, sin embargo existen muchas mas variaciones de esta con diferentes propiedades que son deseables en diferentes situaciones.

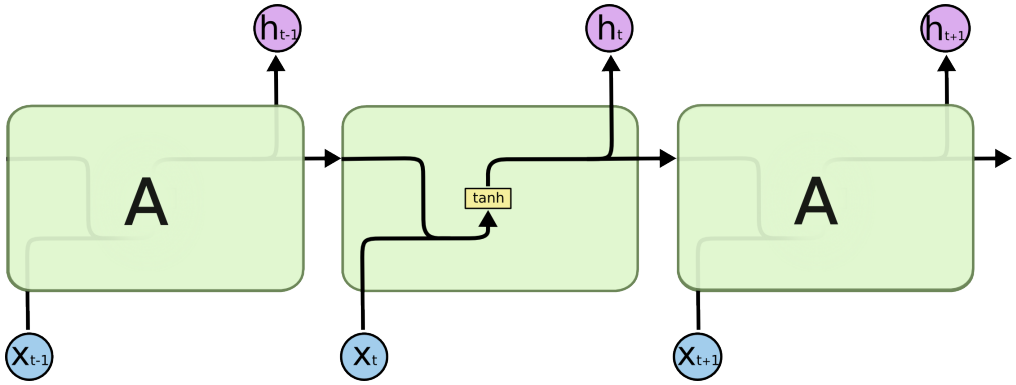


FIGURA 6.6: Red RNN clásica. Tomado de [18].

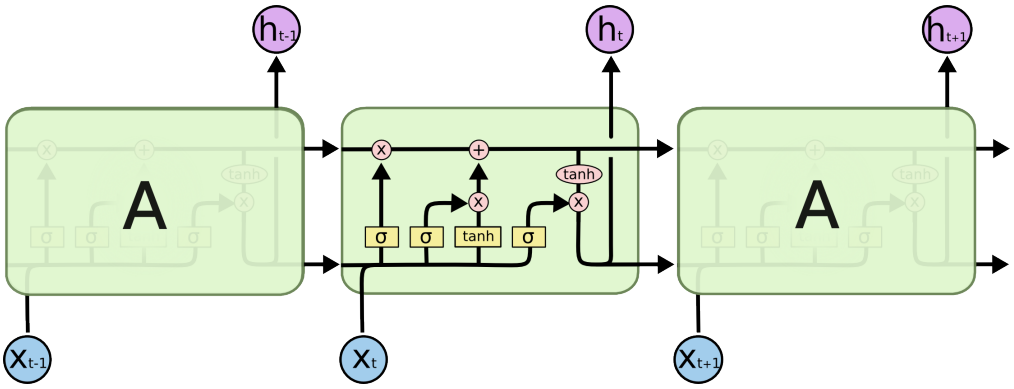


FIGURA 6.7: Red LSTM clásica. Tomado de [18].

Redes neuronales *Bidirectional Long Short Term Memory*

En la tarea de etiquetado de secuencias de texto las Bidirectional Long Short Term Memory (BI-LSTM) tienen acceso a muestras de entrada tanto de futuro como de pasado por una cierta cantidad de tiempo. El uso de esta característica se puede aprovechar a favor ir a estados en el futuro (por medio de estados hacia adelante) y hacia el pasado (por estados hacia atrás) para un momento particular en el tiempo [19].

Con el fin de que la red sea bidireccional es necesario de una celda de memoria para el caso de desplazarse hacia adelante, y una para desplazarse hacia atrás.

Reconocimiento de *Named Entities*

Para el reconocimiento de Named Entities tómese como ejemplo el Cuadro 6.2, donde las etiquetas asignadas corresponden al reconocimiento correspondiente de cada entidad. Las etiquetas (tags) de respuesta son *Otro* (O) o una de estas: *Persona* (PER), *Ubicación* (LOC), *Organización* (ORG) y *Misceláneo* (MISC), donde las partes de etiqueta B- y I- corresponden al inicio de una palabra o una posición intermedia de la entidad.

Texto	Donald	Trump	es	presidente	de	Estados	Unidos
Etiqueta	B-PER	I-PER	O	O	O	B-ORG	I-ORG

CUADRO 6.2: Ejemplo de reconocimiento de *Named Entities*.

La tarea de etiquetado visto para una LSTM y una BI-LSTM puede ser visto en la Figura 6.8 y la Figura 6.9.

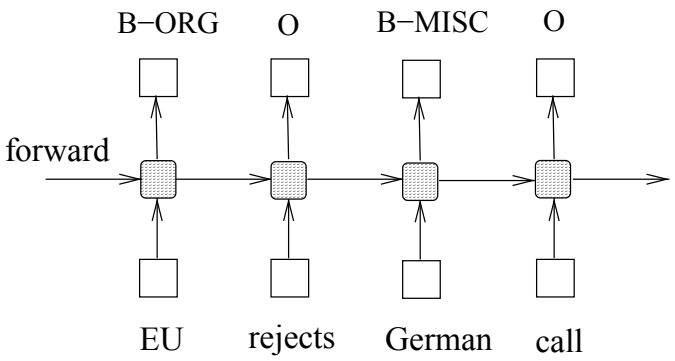


FIGURA 6.8: Etiquetado con una LSTM. Tomado de [19].

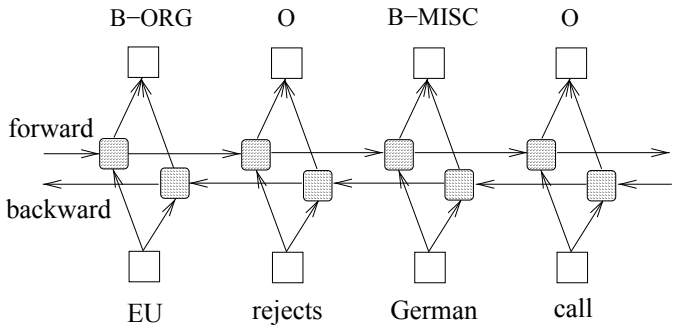


FIGURA 6.9: Etiquetado con una BI-LSTM. Tomado de [19].

Entrenamiento de *Bidirectional Long Short Term Memory*

El entrenamiento de las redes BI-LSTM se empieza de manera similar a la descrita en la Sección 6.3.1 con el procedimiento de asignar un identificador a cada *token* y a cada *tag* como en la Ecuación 6.14 y la Ecuación 6.15, donde N es el numero de tokens y T el numero de tags. De manera que el conjunto de entrenamiento $\mathbb{X}$ consta de tuplas $(\text{token}, \text{tag})$, de donde se agregan dos tipos especiales de tokens `<PAD>` y `<UNK>`, que representan relleno y algo desconocido respectivamente.

$$\text{token2id} = \Big\{(\text{token}_i, i) : \forall i \in \{1, \ldots, N\}\Big\} \tag{6.14}$$

$$\text{tag2id} = \Big\{(\text{tag}_i, i) : \forall i \in \{1, \ldots, T\}\Big\} \tag{6.15}$$

Generación de BATCHES para las redes BI-LSTM Las redes neuronales son comúnmente entrenadas por grupos (o *batches* en ingles) de entrada. Eso significa que las actualizaciones de los pesos dentro de la red se basan en secuencias en cada instante de tiempo. Pero estas secuencias dentro de un batch deben tener la misma longitud, así que estas se rellenaran con una etiqueta de relleno `<PAD>`. También es buena idea proveer la red con la longitud de las secuencias, para que así pueda saltarse computaciones innecesarias de las partes de relleno. A las partes de relleno se le asigna un tag de `O`.

Construcción de capas de la red neuronal Para esta construcción se procede a generar una matriz de *embeddings* aleatorios, que en el caso de *TensorFlow* son tensores aleatorios y se establecen las células de procesamiento *forward* y *backward* con valores de marginalización ξ (valor establecido heurísticamente) que es una forma de regularización importante en redes neuronales, de donde se especifica la probabilidad de retención de los valores de la iteración anterior.

La salida de las celdas de *forward* y *backward* serán dos tensores $\mathbf{F}$ y $\mathbf{B}$, que se concatenan en un tensor $\mathbf{O}$.

Computación de predicciones Se computa la función $\mathrm{softmax}$ sobre $\mathbf{O}$ para convertir las entradas en una forma de probabilidad, de forma que se estan calculando las probabilidades de cada una de las entradas para luego calcular cuales son las entrada de mayor probabilidad, estimando así la predicción P, como se muestra en la Ecuación 6.16.

$$P = \arg\max_i(\mathrm{softmax}(\sigma(\mathbf{O}_i))) \tag{6.16}$$

Sin embargo, durante el entrenamiento no necesitamos una predicción, sino una función de perdida (o *loss* en ingles), que se calcula como se muestra en la Ecuación 6.17, y en la Figura 6.10, donde $\hat{y}$ es la predicción y y es el valor verdadero.

$$\gamma(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{6.17}$$

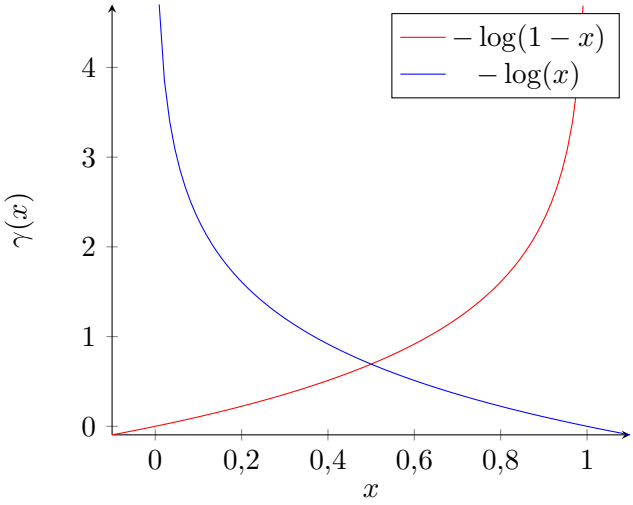


FIGURA 6.10: Gráficas de cross-entropy (azul) si $y = 1$ (rojo) si $y = 0$.

De manera que si se calcula una predicción de 1×10^{-8} cuando el valor verdadero es 1, dará como resultado un valor alto de perdida (e.g. $\gamma(1 \times 10^{-8}, 1) \approx 18.42$).

Se calcula un tensor de perdida $\mathbf{L}$ con $\mathrm{softmax}(\gamma(\sigma(\mathbf{O})))$, para luego calcular la media $\mu_{\mathbf{L}}$ del tensor $\mathbf{L}$. Y se pasa por una tensor mascara $\mathbf{M}$ (Ecuación 6.18) que multiplica por $\mathbf{L}$, donde $\mathcal{D}$ es el batch de entrenamiento. Entiéndase al tensor $\mathbf{M}$ como el tensor que representa las posiciones que no son `<PAD>` con valor de 1 y valor 0 si lo son dentro del batch.

$$\mathbf{M} = \sum_{i \in \{1,\ldots,|\mathcal{D}|\}} \boldsymbol{e}^{(i)}_{\text{tag}_i=\text{PAD}} \tag{6.18}$$

Para realizar el entrenamiento con esta función de perdida se realiza una optimización (el optimizador estocástico ADAM [20] es una opción eficiente) con una tasa de aprendizaje α.

6.3.3. Modelo 3: Búsqueda de tweets relacionados con *embeddings* de StarSpace

En este modelo, la intención es que a partir de una colección de textos guardados en una base de datos se puedan obtener los k textos mas similares a un texto de consulta q, de manera que se encuentren los textos de mayor similaridad en cuanto a las palabras usadas por medio del uso de *embeddings*.

¿Que son y como se usan los embeddings?

Un *embedding* se puede entender como vectores n–dimensionales que representan palabras dentro de un diccionario, para los modelos previamente vistos BOW y TF-IDF se pueden entender como *embeddings*, sin embargo para la aplicación en este modelo, es necesario otro *embedding* que permita encontrar la similaridad entre palabras de forma que palabras que tengan un significado parecido tengan una distancia menor que si se comparara con una palabra que fuera antónima, de forma que $\text{dist}(\boldsymbol{p}_{\text{asombroso}}, \boldsymbol{p}_{\text{genial}}) \gg \text{dist}(\boldsymbol{p}_{\text{asombroso}}, \boldsymbol{p}_{\text{terrible}})$, donde dist típicamente seria la distancia de similaridad coseno (Ecuación 6.19).

$$\text{similaridad}(a, b) = \cos(\theta) = \frac{\boldsymbol{p}_a \times \boldsymbol{p}_b}{\|\boldsymbol{p}_a\| \, \|\boldsymbol{p}_b\|}, \; a, b \in d \tag{6.19}$$

Ejemplos de uso popular en la industria son de *embeddings* previamente entrenados estan *GloVe* [21], *word2vec* [22], *fastText* [23], entre muchos otros.

Calculo de vectores de varias palabras

Una adición a las ventajas dados por estos *embeddings* es la capacidad de representar textos completos por medio de obtener la representación de cada termino $t \in d$ por medio de obtener el promedio los vectores representativos de cada palabra (Ecuación 6.20).

$$\boldsymbol{s} = \frac{1}{|d|} \sum_{t \in d} \boldsymbol{p}_t \tag{6.20}$$

Evaluación de similaridad entre textos

Nos podemos imaginar que usando un buen *embedding* la similaridad coseno entre frases duplicadas[4] va a ser menor que para las frases aleatorias que generemos. Sobretodo que para cada par de frases duplicadas podemos generar R ejemplos de frases negativas y encontrar la posición del texto duplicado correcto.

[4] Entiéndase a una frase duplicada como textos que sean equivalentes "semánticamente" pero no necesariamente iguales sintácticamente

Sin embargo, no es buena idea considerar que el primer candidato sera siempre el primer resultado que dio la mayor similaridad, por lo que se tomara como una lista ordenada y formular una métrica de comparación. Podemos tomar a K como un numero razonable de mejores elementos y N el numero de consultas (tamaño de la muestra).

Hits@K La primera métrica que se usaría seria el numero de "hits" correctos para algún K, vea la Ecuación 6.21.

$$\text{Hits@K} = \frac{1}{N}\sum_{i=1}^{N}[\text{dup}_i \in \text{topK}(q_i)] \tag{6.21}$$

Donde q_i es la i–ésima consulta, dup_i es su duplicado y $\text{topK}(q_i)$ son los K elementos mas similares provistos por nuestro modelo donde $[\text{condition}]$ obtiene valor de 1 si la condición es verdadera y 0 de lo contrario.

DCG@K La segunda métrica es conocida como *Discounted cumulative gain* (versión simplificada), dada por la Ecuación 6.22.

$$\text{DCG@K} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\log_2(1+\text{rank}_{\text{dup}_i})}[\text{rank}_{\text{dup}_i} \leq K] \tag{6.22}$$

Donde $\text{rank}_{\text{dup}_i}$ es la posición del duplicado en la lista ordenada de frases mas cercanas al texto de la consulta q_i. De acuerdo a esta métrica, el modelo obtiene una mayor recompensa por una posición mas alta de la respuesta correcta. Si la respuesta no aparece en topK en absoluto, entonces la recompensa es 0.

StarSpace

StarSpace [24] es un modelo neuronal de propósito general para el aprendizaje eficiente de *embedding* de entidades para resolver una gran variedad de problemas, entre estas están:

- Aprendizaje de *embeddings* a nivel de palabras, frases y documentos.
- Obtención de información: "ranking" de conjuntos de entidades/documentos u objetos.
- Clasificación de textos.
- Aprendizaje de similaridad de textos.
- Entre otras.

¿Como funciona y cual es la principal diferencia con *word2vec*? StarSpace puede ser entrenado específicamente para algunas tareas. En contraste con *word2vec*, el cual intenta entrenar *embeddings* similares para palabras en contextos similares, StarSpace usa *embeddings* para las frases completas (tal como la suma de los *embeddings* para palabras y frases). A pesar de que se obtienen *embeddings* en ambos casos como resultado del entrenamiento. Los *embeddings* de StarSpace son entrenados usando datos supervisados.

Uso de *embeddings* creados con *StarSpace*

En este caso, StarSpace debería usar dos tipos de pares de secuencias para el entrenamiento: "positivas" y "negativas". Las muestras "positivas" son extraídas del conjunto de entrenamiento $\mathcal{D}$ (duplicados, alta similaridad) y las muestras "negativas" que son generadas aleatoriamente (se asume que tienen una baja similaridad).

De forma que se pueda usar StarSpace es necesario primero compilar el código fuente provisto del repositorio de código de GitHub[5] (Solo funciona MacOS o Linux). Se puede clonar el codigo fuente por medio de introducir los comandos en la Terminal de la Figura 6.11.

```
1 git clone https://github.com/facebookresearch/Starspace.git
2 cd Starspace
3 make
```

FIGURA 6.11: Ejemplo en compilación de *StarSpace* en la Terminal.

Para luego ejecutarlo con diferentes parámetros de entrada y un conjunto de entrenamiento en un archivo `.tsv`, como se muestra en la Figura 6.12.

```
1 starspace train -trainFile prepared_train.tsv -model modelSaveFile \
2   -trainMode 3 -adagrad 1 -ngrams 1 -epoch 5 -dim 100 \
3   -similarity cosine -minCount 2 -verbose 1 -fileFormat labelDoc \
4   -negSearchLimit 10 -lr 0.05
```

FIGURA 6.12: Ejemplo en ejecución de *StarSpace* en la Terminal.

Donde se ejecuta a StarSpace en modo de entrenamiento, tiene el archivo de entrenamiento (`prepared_train.tsv`), el modo de entrenamiento para exploración de similaridad de textos (`-trainMode 3`), uso de optimización ADAGRAD (`-adagrad 1`), con n-gramas de tamaño 1 (`-nrams 1`), 5 iteraciones (`-epoch 5`), vectores resultantes en $\mathbb{R}^{100}$ (`-dim 100`), uso de similaridad coseno (`-similarity cosine`), conteo mínimo de 2 términos (útil si no se quieren obtener *embeddings* de palabras extremadamente raras, `-minCount 2`), numero mínimo de ejemplos negativos como 10 usado durante el entrenamiento (`-negSearchLimit 10`) y una tasa de aprendizaje de 0,05 (`-lr 0.05`).

Luego de esto el resultado es un archivo de *embeddings*, donde cada linea esta compuesta de una palabra y los 100 números que componen al vector de esa palabra.

Estos *embeddings* resultantes permiten luego realizar la obtención de los k textos mas similares a un tweet de consulta q_i obtenido de *pool* de textos obtenidos de Twitter de fuentes relevantes de forma que se encuentren textos relacionados el perfilado de individuos que tuvieran relación con grupos terroristas.

6.4. Despliegue (Deployment)

A la fecha, los productos disponibles para el despliegue son el primer modelo propuesto (Sección 6.3.1), una implementación del meta-modelo SOM (Sección 4.6.1) y la implementación de la recolección de información de tweets (Sección 6.2).

[5] https://github.com/facebookresearch/StarSpace

Capítulo 7

Conclusiones y trabajos futuros

7.1. Conclusiones

- Se investigaron diferentes metodologías de NLP y Data Science para la tarea de perfilado de cibercriminales por medio de información de fuentes abiertas.
- Es necesario probar las metodologías propuestas con información obtenida de fuentes abiertas que este validada de forma que el entrenamiento de ellos sean efectivos en la tarea.
- Se logro un obtener una vista amplia de las posibles aplicaciones de NLP aplicados a contextos de ciber espacio.

7.2. Trabajos futuros

- Utilizar el meta-modelo de SOM (Sección 4.6.1) como un cuarto modelo para el clustering de textos representados con embeddings pre-entrenados o generados con StarSpace.
- Utilizar metodologías de visualización de las redes BI-LSTM como la visualización expuesta en [25] con el fin de depurar su ejecución.
- Desarrollo de un *dashboard* para la ayuda de visualización de un agente de seguridad del estado a desempeñar su tarea de perfilado de cibercriminales.
- Desarrollo de técnicas avanzadas de obtención de información de redes sociales, tanto de redes sociales populares como impopulares, de manera similar al desarrollado con OSINT en [6].

Glosario

A

Artificial Intelligence (AI)

Inteligencia Artificial, área de las ciencia de la computación que se encarga de la creación de maquinas inteligentes que tienen un comportamiento que se asemeja a tener inteligencia. 9, 23

Artificial Neural Network (ANN)

Para predecir la probabilidad de crímenes y nuevos ataques terroristas. 5, 10, 26

B

Bag of Words (BoW)

Es una representación simplificada usada en procesamiento de lenguaje natural. En este modelo, un texto es representado como una bolsa (multiconjunto) de sus palabras. 21, 22, 25, 26, 31

Bidirectional Long Short Term Memory (Bi-LSTM)

Red neuronal LSTM bidireccional, que permite el uso de secuencias de palabras futuras y pasadas para una mejor clasificacion de la actual. 28, 29, 34

C

Clustering

Es la tarea de agrupar conjuntos de objetos de manera que este en grupos de mismo tipo. 4, 10, 17, 19

Coursera

Sitio web de cursos de aprendizaje en https://www.coursera.org. 3

Cross Industry Standard Process for Data Mining (CRISP-DM)

Proporciona una descripción normalizada del ciclo de vida de un proyecto estándar de análisis de datos. El modelo CRISP-DM cubre las fases de un proyecto, sus tareas respectivas, y las relaciones entre estas tareas. 19

D

Data Integration

Para acceder a múltiples y diversas fuentes de información. 5

Data Mining

Proceso de descubrir automáticamente información útil en repositorios grandes de datos. 7

Data Science

Es una campo multidisciplinario que usa metodos, procesos, algoritmos y sistemas cientificos para extraer conocimiento y revelaciones de datos estructurados y no estructurados. 4, 34

Dataset

Conjunto de datos, en el campo de Inteligencia Artificial generalmente se refiere a conjuntos públicos o de acceso limitado de donde es posible obtener información útil con sistemas de computo. 1

Deductive Systems

Sistemas deductivos, son sistemas basados en conocimiento que en base al estado de la memoria de trabajo y a las reglas presentes en la base de conocimiento, agregan nuevas inferencias a la memoria de trabajo. 9

E

Expert Systems

Sistemas expertos, son sistemas de computo que emulan la abeldad de toma de decisiones de un humano experto. 9

F

Feedforward

Es un termino que describe típicamente un elemento o un camino dentro de un sistema de control que pasa una señal de control de una fuente externa. En la inteligencia artificial este termino no difiere en significado. 10

H

High Bias

El modelo esta poco condicionado a la entrada por lo que para un modelo predictivo dará resultados demasiado generales para cualquier conjunto de datos. 11, 23

I

Inference Engine (IE)

Motor de inferencia, es un componente del sistema que aplica reglas lógicas a la base de conocimiento para deducir información nueva. 9

K

Knowledge Base (KB)

Base de conocimiento, es una tecnología usada para almacenar información estructurada y no-estructurada usada por un sistema de computo. 9

Knowledge Based Systems (KBS)

Sistemas Basados en Conocimiento, su objetivo es abstraer conocimiento de un experto de un área en una representación dentro de un computador. 9

L

Link Analysis

Para visualizar asociaciones y relaciones criminales y terroristas. 5

Long Short Term Memory (LSTM)

Red neuronal recurrent que tiene memoria Largo y Corto Plazo, son un tipo de redes neuronales recurrentes que son capaces de aprender dependencias de largo plazo. 27, 28

M

Machine Learning Algorithms

Para extraer perfiles de perpetradores y mapas gráficos de crímenes. 5

Machine Learning (ML)

Informalmente ha sido definido como "El campo de estudio que le da a computadores la habilidad de aprender sin ser explícitamente programados", este tiene tres tipos de algoritmos de aprendizaje: aprendizaje supervisado, aprendizaje no–supervisado, y aprendizaje por refuerzo. 6, 8

Maximum Margin Hyperplanes

Hiperplanos que permiten separar datos en espacios de alta dimensionalidad con un margen asociado para separarlos. 13

N

Named Entities

Son típicamente palabras que denotan personas particulares u organizaciones, pero no son sustantivos. 4, 26, 28

Natural Language Processing (NLP)

Rama de la inteligencia artificial que lidia con la interacción entre computadores y humanos usando el lenguaje natural. 1–5, 19, 21, 22, 34

O

Open Source Intelligence (OSINT)

Disciplina responsable de la adquisición, procesamiento y posterior transformación en inteligencia de información obtenida de fuentes públicas como prensa, radio, televisión, internet, informes de diferentes sectores y, en general, cualquier recurso de acceso público (Tomado de [6]). 3, 34

Overfitting

El modelo esta demasiado acoplado a la entrada por lo que para un modelo predictivo dará resultados solo condicionados para la entrada de entrenamiento, pero tendrá mal desempeño en cualquier otro conjunto de prueba. En modelos de clustering, se vera que la salida esta muy condicionada a la entrada por lo que no tendrá un buen desempeño tampoco. 11, 23

R

Reactive Systems

Sistemas reactivos, son sistemas basados en conocimiento que en base al estado de la memoria de trabajo y a las reglas presentes en la base de conocimiento, agregan o eliminan inferencias de la memoria de trabajo. 9

Recurrent Neural Network (RNN)

Redes que recuerdan salidas y entradas pasadas de datos, de forma que decisiones futuras respecto a una clasificación pueden tener un mejor resultado. 26

Reinforcement Learning

Aprendizaje de refuerzo, es el aprendizaje de que acción se debe tomar de forma que se logre una señal de recompensa lo mas alto posible. 6

Rule–based Systems (RBS)

Sistemas basados en reglas, son sistemas expertos que en su base de conocimiento contienen reglas para crear nuevas inferencias. 9

S

Self-organizing Maps (SOM)

Mapas auto-organizados, son redes neuronales especializadas para el clustering de datos semejantes según una función de distancia, da como resultado un mapa discreto uni- o bi-dimensional. 10, 11, 17, 33, 34

Software Agents

Para el monitoreo, obtención, análisis y actuación sobre la información. 5

StarSpace

Es un modelo neuronal de propósito general para el aprendizaje eficiente de embeddings. 4, 32–34

Supervised Learning

Aprendizaje supervisado, su objetivo es aprender un mapeo de unas entradas a unas salidas definidas. 6

Support Vector Machine (SVM)

Maquinas de Soporte Vectorial, son modelos de aprendizaje supervisado usados para el problema de clasificación y análisis de regresión. 13, 14, 17

T

TensorFlow

TensorFlow es una plataforma de codigo abierto para machine learning. Tiene un ecosistema comprensible y flexible de herramientas, librerias y recursos de la comunidad que le permite a investigadores impulsar el estado del arte en machine learning y desarrolladores puede construir y desplegar facilmente aplicaciones con Machine Learning incorporado. 30

Term Frequency – Inverse Document Frequency (TF-IDF)

Frecuencia de términos – Frecuencia inversa de documento, es una estadística numérica que esta diseñada para reflejar que tan importante es una palabra a un documento dentro de un corpus. 22, 25, 26, 31

Text Mining

Búsqueda sobre terabytes de información en documentos, paginas web y correos electrónicos. 5

U

Unsupervised Learning

Aprendizaje no-supervisado, su objetivo es obtener patrones estructurados de una serie de datos no estructurados. 6, 10

W

Working Memory (WM)

Memoria de trabajo, es donde se encuentra el estado actual del sistema antes y después de haber ejecutado reglas de un sistema basado en reglas, y es donde se encuentran las inferencias hechas por el motor de inferencia. 9

Bibliografía

[1] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[2] Ana Maria Popescu y Orena Etzioni. «Extracting product features and opinions from reviews». En: *Natural Language Processing and Text Mining* (2007), págs. 9-28. ISSN: 08247935. DOI: 10.1007/978-1-84628-754-1_2. arXiv: 0309034 [cs].

[3] Jesús Mena. *Investigative Data Mining for Security and Criminal Detection*. Elsevier Science, 2003. ISBN: 9780080509389. URL: https://books.google.com.co/books?id=3mDlrtJuZv4C.

[4] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020. URL: https://www.amazon.com/Machine-Learning-Probabilistic-Perspective-Computation/dp/0262018020/ref=sr_1_2?ie=UTF8&qid=1336857747&sr=8-2.

[5] Pang-Ning Tan, Michael Steinbach y Vipin Kumar. *Introduction to Data Mining*. US ed. Addison Wesley, mayo de 2005. ISBN: 0321321367. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0321321367.

[6] Martín José Hernández Medina, Ricardo Andrés Pinto Rico y Cristian Camilo Pinzón Hernández. *Inteligencia de fuentes abiertas para el contexto colombiano*. Inf. téc. Escuela Colombiana de Ingeniería Julio Garavito, 2018.

[7] Priti Srinivas Sajja y Rajendra Akerkar. «Knowledge-based systems for development». En: *Advanced Knowledge Based Systems: Model, Applications & Research* 1 (2010), págs. 1-11.

[8] Jerry M Mendel. *Uncertain Rule-Based Fuzzy Systems*. ISBN: 9783319513690. DOI: 10.1007/978-3-319-51370-6.

[9] Leandro Nunes De Castro. *Fundamentals of natural computing: basic concepts, algorithms, and applications*. Chapman y Hall/CRC, 2006.

[10] Rebecca S. Portnoff y col. «Tools for Automated Analysis of Cybercriminal Markets». En: (2017), págs. 657-666. DOI: 10.1145/3038912.3052600.

[11] Will Y Zou y col. «Bilingual Word Embeddings for Phrase-Based Machine Translation». En: *Proceedings ofthe 2013 Conference on Empirical Methods in Natural Language Processing* (2013), págs. 1393-1398. URL: http://www.aclweb.org/anthology/D13-1141.

[12] Raymond Y.K. Lau, Yunqing Xia y Yunming Ye. «A probabilistic generative model for mining cybercriminal networks from online social media». En: *IEEE Computational Intelligence Magazine* 9.1 (2014), págs. 31-43. ISSN: 1556603X. DOI: 10.1109/MCI.2013.2291689.

[13] Julian Szymański y Włodzisław Duch. «Self Organizing Maps for Visualization of Categories». En: *Neural Information Processing*. Ed. por Tingwen Huang y col. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, págs. 160-167. ISBN: 978-3-642-34475-6.

[14] Victor Benjamin y Hsinchun Chen. «Identifying language groups within multilingual cybercriminal forums». En: *IEEE International Conference on Intelligence and Security Informatics: Cybersecurity and Big Data, ISI 2016* (2016), págs. 205-207. DOI: 10.1109/ISI.2016.7745471.

[15] Alberto P. García-Plaza, Víctor Fresno y Raquel Martínez. «Web page clustering using a fuzzy logic based representation and Self-Organizing Maps». En: *Proceedings - 2008 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2008* (2008), págs. 851-854. DOI: 10.1109/WIIAT.2008.249.

[16] O. Tange. «GNU Parallel - The Command-Line Power Tool». En: *;login: The USENIX Magazine* 36.1 (2011), págs. 42-47. DOI: http://dx.doi.org/10.5281/zenodo.16303. URL: http://www.gnu.org/s/parallel.

[17] Lars Buitinck y col. «API design for machine learning software: experiences from the scikit-learn project». En: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. 2013, págs. 108-122.

[18] *Understanding LSTM Networks*. URL: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[19] Zhiheng Huang, Wei Xu y Kai Yu. «Bidirectional LSTM-CRF Models for Sequence Tagging». En: (2015). arXiv: 1508.01991. URL: http://arxiv.org/abs/1508.01991.

[20] Diederik P. Kingma y Jimmy Ba. «Adam: A Method for Stochastic Optimization». En: (2014), págs. 1-15. arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[21] Jeffrey Pennington, Richard Socher y Christopher Manning. «Glove: Global vectors for word representation». En: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, págs. 1532-1543.

[22] Tomas Mikolov y col. «Efficient Estimation of Word Representations in Vector Space». En: *Intelligent Systems Reference Library* 58 (2014), págs. 1-12. ISSN: 18684408. DOI: 10.1007/978-3-319-01967-3_1. arXiv: arXiv:1301.3781v3.

[23] Armand Joulin y col. «FastText.zip: Compressing text classification models». En: *arXiv preprint arXiv:1612.03651* (2016).

[24] L. Wu y col. «StarSpace: Embed All The Things!» En: *arXiv preprint arXiv:1709.03856* (2017).

[25] Andreas Madsen. «Visualizing memorization in RNNs». En: *Distill* (2019). DOI: 10.23915/distill.00016. URL: https://distill.pub/2019/memorization-in-rnns.

[26] Patrick Henry Winston. *Artificial Intelligence (3rd Ed.)* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992. ISBN: 0-201-53377-4.

[27] Steven Bird, Ewan Klein y Edward Loper. *Natural Language Processing with Python*. 1st. O'Reilly Media, Inc., 2009. ISBN: 0596516495, 9780596516499.