UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Detección de género en Twitter basado en características multimodales

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Marcos Esteve Casademunt

Tutor: Carlos David Martínez Hinarejos
Francisco Manuel Rangel Pardo

Curso 2018-2019

Resum

El *Autor profiling* consisteix a analitzar un conjunt de textos de forma automàtica, per tal d'obtenir informació en relació a les característiques estilístiques de l'autor basant-se únicament en el seu discurs. Algunes d'aquestes característiques són, per exemple: el gènere, l'edat, la personalitat i la varietat lingüística.

En aquest treball s'ha tractat d'extreure informació del gènere dels autors a Twitter utilitzant el corpus proporcionat pel PAN 2018. S'han explorat diferents representacions, com les borses de paraules o els *word embeddings*, per a l'extracció de característiques en informació textual. D'altra banda, s'han utilitzat tècniques basades en aprenentatge profund per a la detecció de gènere en les imatges.

Finalment s'han utilitzat tècniques de combinació de classificadors i s'han implementat els models desenvolupats en una aplicació pràctica del camp del màrqueting

Paraules clau: *Author profiling*, aprenentatge automàtic, detecció de gènere, PAN, Twitter, Word2Vec

Resumen

El *Author profiling* consiste en analizar un conjunto de textos de forma automática, con el fin de obtener información acerca de las características estilísticas del autor basándose únicamente en su discurso. Algunas de estas características son, por ejemplo: el género, la edad, la personalidad y la variedad lingüística.

En este trabajo se ha tratado de extraer información acerca del género de los autores en Twitter utilizando el corpus proporcionado por el PAN 2018. Se han explorado distintas representaciones como las bolsas de palabras o los *word embeddings* para la extracción de características en información textual. Por otra parte, se han utilizado técnicas basadas en aprendizaje profundo para la detección de género en las imágenes.

Por último, se han utilizado técnicas de combinación de clasificadores y se han implementado los modelos desarrollados en una aplicación práctica del campo del marketing

Palabras clave: *Author profiling*, aprendizaje automático, detección de género, PAN, Twitter, Word2Vec

Abstract

Author profiling consists of analyzing a set of texts automatically with the objective of obtain information about the stylistic characteristics of the author, based solely on his discourse. Some of these characteristics are, for example: gender, age, personality, and linguistic variety.

On this work, we tried to extract information about the gender of the authors on Twitter using the corpus provided by PAN 2018. Different representations have been explored, such as bag of words or word embeddings, for the extraction of features in textual information. On the other hand, techniques based on deep learning have been used for the detection of gender in images.

Finally, classifier combination techniques have been used and the models were included in a practical application of the marketing field.

Key words: Author profiling, machine learning, gender detection, PAN, Twitter, Word2Vec

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

A Carlos por su implicación en la docencia y confiar en el desarrollo de este proyecto.

A Kico por hablarme del *Author profiling* y junto con Autoritas permitirme implantar en su producto los modelos desarrollados en este trabajo.

A mis padres, por hacer todo lo que estaba en sus manos para que consiguiera llegar hasta este punto y enseñarme que con esfuerzo y constancia se consiguen las cosas.

A todos vosotros, gracias.

Data is the new oil Clive Humby

CAPÍTULO 1
Introducción

1.1 Motivación

La popularización de las redes sociales y la gran cantidad de información que generan, así como la proliferación de nuevas técnicas algorítmicas y nuevo *hardware* capaz de procesar grandes volúmenes de datos en tiempos aceptables, ha permitido un mayor desarrollo de campos como el procesamiento de lenguaje natural o la visión por computador.

La detección del género de los usuarios puede ser de gran importancia en campos como el marketing, donde se podría aplicar para segmentar mejor a los usuarios y determinar con más exactitud el *target* al que la empresa quiere dirigirse. También puede ser de gran relevancia para la seguridad ciudadana, donde permitiría, por ejemplo, detectar si existe una posible usurpación de identidad.

1.2 Objetivos

El objetivo principal de este trabajo consiste en la predicción del género de los usuarios en Twitter, utilizando para ello la información proporcionada por los mismos usuarios, ya sea imagen o texto. Para ello se plantean los siguientes subobjetivos:

- Explorar distintas técnicas para la clasificación del texto.
- Desarrollar modelos predictivos para la clasificación de imágenes.
- Indagar en el uso de técnicas de combinación de clasificadores, mezclando para ello los modelos desarrollados en el primer y segundo punto.
- Incorporar los modelos desarrollados en una aplicación práctica.

1.3 Estructura de la memoria

El presente documento se estructura en un total de 6 capítulos, los cuales se detallan a continuación:

En el **capítulo 2** se dará una visión general de las distintas técnicas utilizadas para la representación de la información, así como algoritmos de aprendizaje automático utilizados en el desarrollo de la memoria para el reconocimiento de patrones en texto e imagen.

En el **capítulo 3** se hablará acerca del desarrollo del *Author Profiling* y se comentará el problema a resolver, así como el conjunto de datos utilizados.

En el **capítulo 4** se comentará una posible solución que permita resolver el problema de la identificación del género en Twitter. También se explicarán las tecnologías empleadas en dicha solución.

En el **capítulo 5** se detallará la solución propuesta en el capítulo 4, profundizando, para ello, en el desarrollo de las distintas fases del sistema.

En el **capítulo 6** se incorporarán los modelos desarrollados en una aplicación práctica en el campo del marketing.

Por último, en el **capítulo 7** se comentarán las conclusiones extraídas y en el **capítulo 8** se detallará la relación del trabajo realizado con los estudios cursados.

CAPÍTULO 2
Reconocimiento de formas

En la última década, gracias a una mejora sustancial en la velocidad de los procesadores, así como la proliferación de las unidades de procesamiento gráfico (GPU), se han podido abordar problemas de gran complejidad computacional. Gracias a esto, ha sido posible tratar con éxito problemas de reconocimiento de patrones en distintas áreas, desde el reconocimiento de voz (como es el caso de los asistentes de Google o Alexa) a la detección de tumores y tejidos cancerosos por imagen de resonancia magnética.

El reconocimiento de formas se encarga de encontrar patrones en distintos aspectos perceptivos, como el habla, la visión, la escritura, etc.

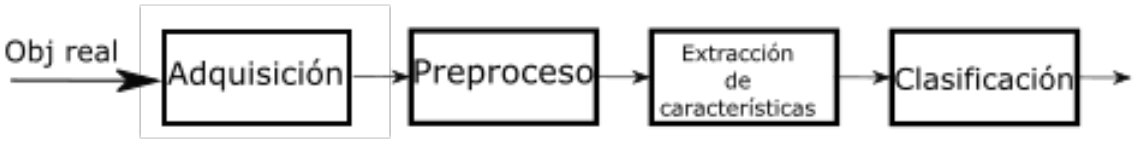

Figura 2.1: Sistema de reconocimiento de formas

Como se puede observar en la imagen de la Figura 2.1, un sistema de reconocimiento de formas consta de las siguientes partes: en primer lugar, tenemos el objeto del mundo real, del que mediante un proceso de adquisición con el uso de sensores, se obtiene su información en formato digital; una vez obtenida la información, se preprocesa y se realiza una representación de la misma; por último, se utilizan distintos algoritmos de aprendizaje automático con el fin de obtener modelos predictivos que permitan clasificar la muestra entre las distintas clases existentes. Estos modelos deben de ser capaces de generalizar, es decir, que ante una nueva muestra desconocida por el sistema el modelo usado sea capaz de discernir a que clase pertenece.

2.1 Extracción de características textuales

Para poder explotar la información textual es necesario realizar una representación de cada documento. Existen dos vertientes diferenciadas. Por una parte, las representaciones basadas en bolsa de palabras; en ella cada documento se representa mediante un vector, donde para cada una de sus posiciones se indica la frecuencia de aparición de una determinada palabra en dicho documento. Por otra parte, las representaciones basadas en *Word embeddings*, que se basan en las teorías de la semántica distribucional [9]. Estas teorías sostienen que elementos lingüísticos con distribuciones similares tienen significa-

dos similares. Es decir, palabras como peine y pelo tendrán distribuciones más similares que las palabras peine y elefante.

2.1.1. *Bag of Words*

La bolsa de palabras (*Bag of Words*) es un modelo utilizado en procesamiento de lenguaje natural. Este modelo permite realizar una representación en forma vectorial donde para cada *token* se especifica la frecuencia de aparición en el documento. Un *token* es una secuencia de carácteres que constituye una unidad semántica. En nuestro caso se ha escogido como *token* cada una de las palabras que conforman la oración. En la Tabla 2.1 se muestra un ejemplo con las frases "el perro tiene patas y el gato también" (Frase 1) y "el pájaro tiene garras" (Frase 2).

Tabla 2.1: Ejemplo de *Bag of Words* ponderada por frecuencia

Frase	el	garras	gato	pajaro	patas	perro	también	tiene	y
Frase 1	2	0	1	0	1	1	1	1	1
Frase 2	1	1	0	1	0	0	0	1	0

Existen diversas variantes de este modelo; una de las más frecuentes es utilizar la bolsa de palabras ponderada por Tf-idf:

$$tf(t,d) = 1 + \log\ f(t,d)\ si\ f(t,d) > 0$$

$$idf(t,d) = \log(\frac{N}{\sum_{f(t,d)>0} 1})$$

$$tfidf(t,d) = tf(t,d) + idf(t,d)$$

Donde N es el número total de documentos, y *f(t,d)* es la frecuencia de aparición del termino t en el documento d. Gracias a esta ponderación se consigue atenuar aquellos *tokens* con presencia en muchos documentos.

Uno de los principales problemas que tiene este modelo es la pérdida del contexto, ya que no captura relaciones entre *tokens*. Una forma de solucionarlo consiste en realizar una bolsa de palabras utilizando secuencias de *n-tokens* (n-gramas). Este modelo permite capturar relaciones entre distintos *tokens* consecutivos, aunque también puede presentar problemas de dimensionalidad y dispersión.

2.1.2. Word2Vec

Word2Vec es un algoritmo desarrollado por Tomas Mikolov en Google. Este algoritmo trata de realizar una representación basada en vectores que capture una gran cantidad de las relaciones sintácticas y semánticas entre palabras [9].

Este algoritmo consta de las siguientes partes:

- Una red neuronal
- Una codificación de cada palabra en formato *One Hot Vector*
- Un *Skip Gram* que capture los pares entrada y salida de la red neuronal

El *Skip Gram* consiste en obtener pares entrada-salida a partir de un conjunto de frases. Para ello, se determina un tamaño de ventana, se recorre las frases y, por cada palabra, se aplica dicha ventana para extraer los pares. La Figura 2.2 muestra un ejemplo de extracción de pares entrada-salida con un *Skip-gram* de grado 2.

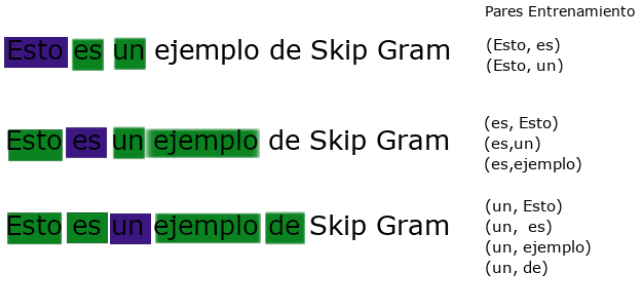

Figura 2.2: *Skip Gram* con tamaño de ventana 2 [15]

Una vez extraídos los pares es necesario realizar una codificación de dichas palabras, ya que las redes neuronales no pueden trabajar directamente con texto. Para ello se utiliza la representación en formato *One Hot Vector*, que consiste en codificar cada palabra como un vector, con un 1 en la posición de la palabra y 0 en el resto de posiciones. Por ejemplo, imaginemos que tenemos un vocabulario formado por las palabras: perro, gato, pájaro y pato; la codificación en formato *One Hot Vector* sería la presentada en la Tabla 2.2.

Tabla 2.2: Ejemplo *One Hot Vector* con V = {perro, gato, pájaro, pato}

perro	1	0	0	0
gato	0	1	0	0
pájaro	0	0	1	0
pato	0	0	0	1

Por último, se entrena un modelo basado en redes neuronales donde la capa de entrada y la capa de salida de la red tienen el mismo tamaño (talla del vocabulario), y la capa oculta un tamaño inferior, habitualmente 300 características [9]. Este modelo se representa en la Figura 2.3.

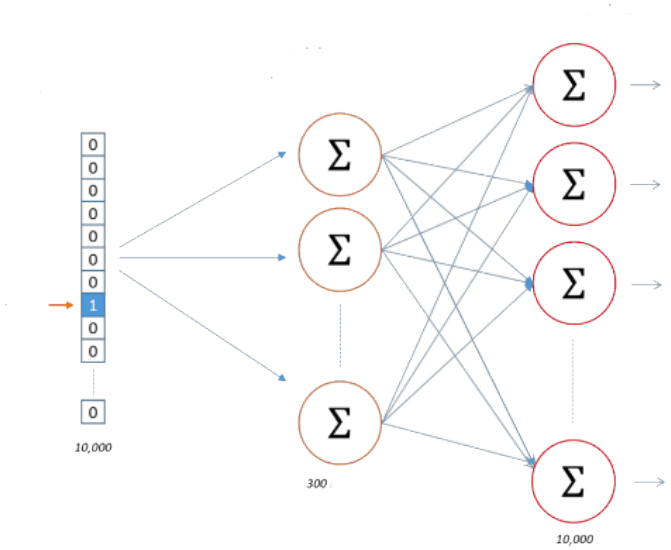

Figura 2.3: Red neuronal con talla del vocabulario 10000 elementos y talla de la capa oculta 300

Una vez entrenada, la capa oculta representa los *embeddings* para cada palabra. De esta forma se puede representar cada palabra como un vector de un tamaño inferior y que captura las relaciones semánticas y sintácticas entre distintas palabras. Palabras que tengan contextos similares deberán tener por tanto vectores similares [9].

2.2 Extracción de características en imágenes

Para trabajar con imágenes es necesario realizar una representación de la mismas. Existen distintos tipos de representaciones, desde una representación por histograma hasta representaciones por características locales donde una imagen se representa por varias partes de la misma. Una nueva tendencia ha surgido con la aparición de las redes neuronales profundas, consistente en representar las imágenes como etiquetas que las describan.

2.2.1. Representación por histograma

Una de las más utilizadas en la extracción de características en imágenes es la representación por histograma, bien sea por canales RGB o escala de grises. Esta representación consiste en representar la imagen como un vector de tantas componentes como niveles de gris. En la componente i-esima del vector se almacenará el número de píxeles con nivel de gris i.

Por ejemplo, supongamos que tenemos la siguiente matriz de pixeles 4*4:

1	100	256	256
10	100	1	1
2	1	2	256
1	256	2	100

La representación mediante histograma de escala de grises tendrá la siguiente forma:

0	1	2	...	10	...	100	...	256
0	5	3	...	1	...	3	...	4

Se puede calcular el tamaño que ocupará esta representación en memoria como:

$$l * \left\lceil \frac{\log_2(n+1)}{8} \right\rceil \text{bytes}$$

donde l son los niveles de gris que estamos contemplando y n el número de píxeles que tiene la imagen. En nuestro ejemplo, el tamaño de la representación es de 257 bytes.

2.2.2. Representaciones locales: el algoritmo Haar Cascade

Uno de los métodos más usados para realizar una representación de una figura es el uso de representaciones locales. Estas representaciones permiten realizar una representación a partir de distintas ventanas representativas de la imagen. Por ejemplo, se puede realizar una representación de una cara como una concatenación de las ventanas de los ojos, cejas, boca, barbilla, nariz y contorno.

El algoritmo *Haar Cascade* fue propuesto por Paul Viola y Michael Jones en el artículo *Rapid Object Detection using a Boosted Cascade of Simple Features* [3]. Este algoritmo consiste en un conjunto de clasificadores básicos (basados en *Adaboost*) que se combinan con el objetivo de detectar si en una determinada posición existe un determinado objeto. Para el entrenamiento de este algoritmo es necesario proporcionar un conjunto de imágenes positivas, así como un conjunto de negativas.

Para el entrenamiento del algoritmo, en primer lugar es necesario realizar una extracción de características. Para ello, se recorre la imagen aplicando distintas ventanas de *Haar* (Fig. 2.4) y variando la escala. Este método generará una gran cantidad de características, por lo que será necesario aplicar distintas técnicas para reducir la cantidad de cómputo necesario a la hora de detectar un objeto.

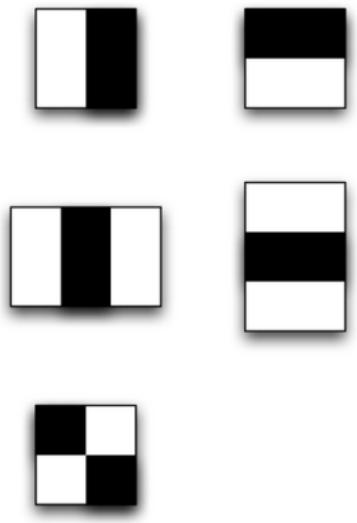

Figura 2.4: Características de Haar

Una vez extraídas las características se propone entrenar un clasificador en cascada. De esta forma para que un objeto resida en la posición de estudio tendrá que ser aceptado por todos los clasificadores. Es decir, si uno de los clasificadores rechaza la región,

(Fig. 2.5) dicha región será descartada y, por tanto, no considerada por el resto de los clasificadores.

Los primeros clasificadores serán muy simples y permitirán de forma muy eficiente eliminar una gran cantidad de ventanas. De esta forma se consigue reducir la cantidad de cómputo necesario a la hora de detectar una región, ya que no todas las ventanas serán evaluadas por todos los clasificadores ni todos los clasificadores tendrán el mismo coste computacional.

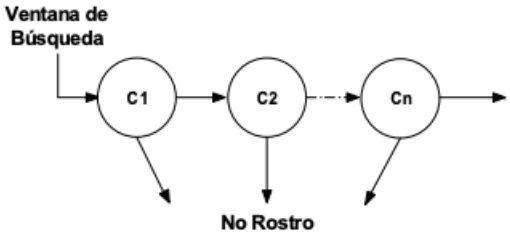

Figura 2.5: Clasificador en cascada

Este algoritmo, presentado en la Figura 2.6, se trata de una modificación de *Boosting*.

- Dado un conjunto de imagenes $(x_1, y_1), ..., (x_n, y_n)$ donde $y_i \in \{0, 1\}$ para ejemplos negativos o positivos respectivamente
- Inicializar los pesos $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ para $i = 0, 1$ donde m y l son el número de negativos y positivos, respectivamente
- for t = 1, ..., T:
  1. Normalizar los pesos como $w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$ convirtiendo w_t en una distribución de probabilidad
  2. Para cada caracteristica j, entrenar un clasificador h_j cuya única restricción sea utilizar una sola característica. El error se evaluará con respecto w_t como $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$
  3. Escoger el clasificador h_t con el menor error ϵ_t
  4. Actualizar los pesos: $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$ donde $e_i = 0$ si x_i se clasifica correctamente, en otro caso $e_i = 1$ y además $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
- El clasificador final será

$$h(x) = \begin{cases} 1 \sum_{t=1}^{T} \alpha_t & h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & h_t(x) < \frac{1}{2} \sum_{t=1}^{T} \alpha_t \end{cases}$$

donde $a_t = \log \frac{1}{\beta_t}$

Figura 2.6: Modificación de *Boosting* presentado para HaarCascade en [3]

2.2.3. Redes neuronales profundas: VGG16

La popularización de las redes neuronales, y en particular su uso en el aprendizaje profundo, ha marcado un barrera distintiva a la hora de realizar una tarea de extracción

de características en imágenes. Una de las arquitecturas más utilizadas en la actualidad para la extracción de características es VGG.

VGG es una arquitectura basada en redes neuronales convolucionales (ver apartado 2.3.3) presentada por Karen Simonyan y Andrew Zisserman en el artículo *Very Deep Convolutional Networks for Large-Scale Image Recognition* [4]. El objetivo del artículo consistía en enfrentarse a un problema de reconocimiento de imágenes sobre un *dataset* con gran variedad de casos como el de la competición propuesta por ImageNet en 2014[1].

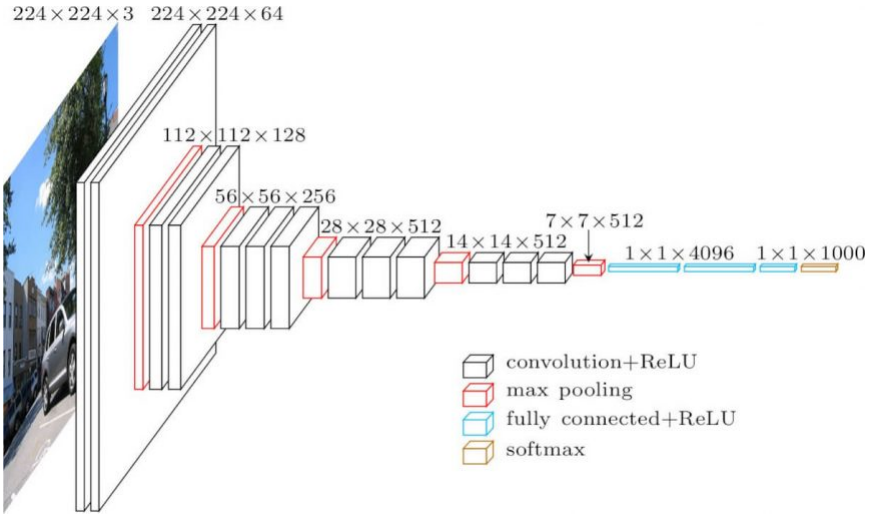

Figura 2.7: Arquitectura VGG

La arquitectura presentada en la Figura 2.7 esta formada por 16 capas donde la entrada de la red tiene un tamaño de 224 x 224 x 3 y la salida posee 1000 neuronas correspondientes a cada una de las clases que debían ser distinguidas por la red.

La red presentada, con alrededor de 144 millones de parámetros, se entrenó sobre tarjetas gráficas NVIDIA Titan Black durante dos semanas para resolver la tarea. Los resultados obtenidos por dicha red fijaron el actual estado del arte en detección de objetos, permitiendo distinguir entre 1000 clases de objetos distintos.

2.3 Aprendizaje

Una vez se ha realizado el preproceso y la extracción de características es necesario el uso de técnicas de aprendizaje automático que permitan extraer conocimiento a partir de los datos. En este trabajo se han utilizado diversas técnicas: las máquinas de vectores soporte (*Support Vector Machine* en inglés), la regresión logística (*Logistic Regression* en inglés) y las redes neuronales.

2.3.1. Máquinas de vectores soporte

Las máquinas de vectores soporte o SVM por sus siglas en ingles, constituye uno de los algoritmos de aprendizaje automático más usados en la actualidad, con aplicaciones tan diversas como la visión por computador, el procesamiento de lenguaje natural o la bioinformática. Este algoritmo fue presentado por Vladimir Vapnik en 1992 en la conferencia COLT y explicado con más detalle en el año 1998 [6] .

[1] http://image-net.org/challenges/LSVRC/2014/

Se trata de un algoritmo supervisado, perteneciente a la familia de los clasificadores lineales, ya que determina hiperplanos separadores en espacios de características con una dimensionalidad muy alta. El objetivo del algoritmo consiste en determinar un hiperplano separador tal que maximice el margen de separación entre las clases [11].

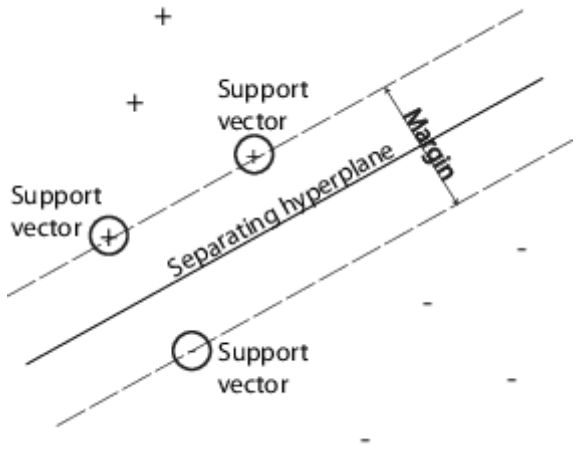

Figura 2.8: Frontera de decisión de margen máximo determinada por la SVM

En ocasiones no es posible encontrar un hiperplano que permita separar linealmente las muestras. El uso de las máquinas de vectores soporte de margen máximo provoca que las soluciones obtenidas tengan problemas generalizando. Ante este problema surge una extensión del algoritmo (*Soft Margin* SVM), que trata de producir soluciones más robustas al introducir variables de holgura. Estas variables de holgura se conocen en la práctica como **C** y constituye un parámetro del entrenamiento [11].

Existen conjuntos de datos en los cuales no se puede determinar un hiperplano separador para las muestras. Con el fin de solucionar este problema surge la idea de Kernel. Un Kernel es una función que, al aplicarla a un conjunto de datos que no poseen separabilidad lineal, trata de obtener un conjunto de datos que sean linealmente separables (Fig 2.9). Esta función de Kernel determina un nuevo parámetro en el aprendizaje de la SVM.

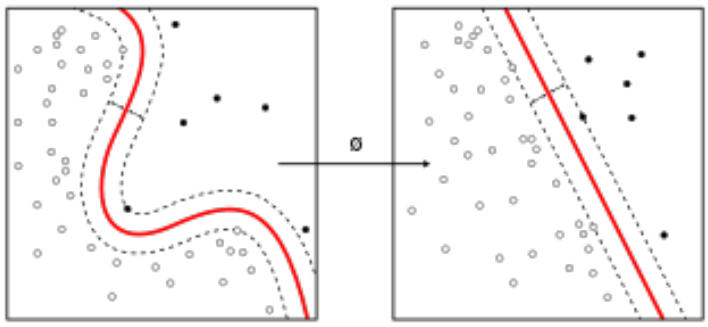

Figura 2.9: Aplicación de una función Kernel a un espacio no linealmente separable

2.3.2. Regresión logística

Otro de los algoritmos más conocido utilizado para resolver tareas de clasificación es la regresión logística. Se trata de un algoritmo similar a la regresión lineal pero utilizando una función Logit (Fig. 2.10) en lugar de una función lineal simple.

$$\text{logit(p)} = \log\left(\frac{p}{1-p}\right) = \log(p) - \log(1-p)$$

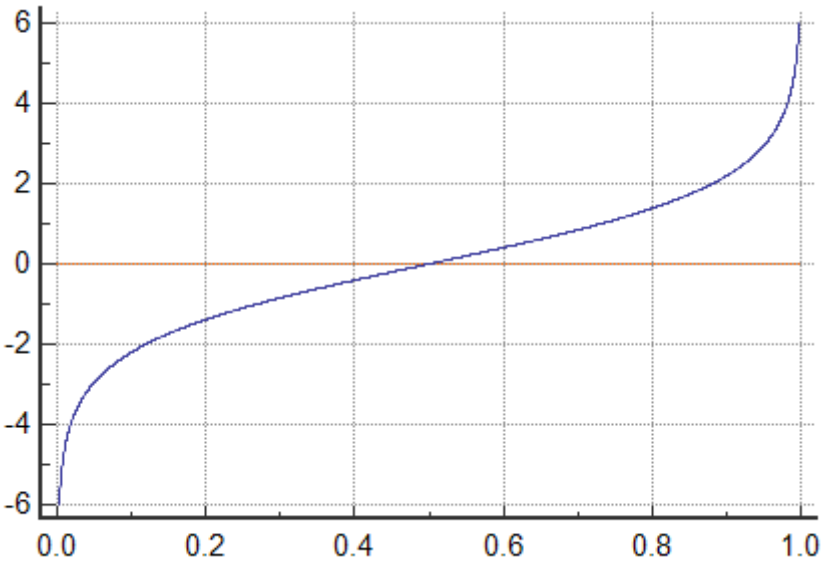

Figura 2.10: Función Logit

Gracias al uso de la función Logit el modelo es capaz de obtener fronteras de decisión que se adapten mejor a los datos, disminuyendo en muchos casos el error en tareas de clasificación.

2.3.3. Redes neuronales

Las redes neuronales son un sistema conexionista dentro del campo de la inteligencia artificial. Este modelo surge con la iniciativa de imitar la capacidad de las neuronas humanas a la hora de procesar información.

Contexto

El desarrollo de esta técnica comenzó en 1943 cuando McCulloch y Pitts presentaron un modelo matemático para una neurona (Fig. 2.11).

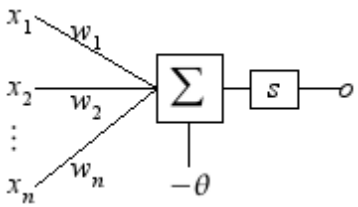

Figura 2.11: Modelo de una neurona presentado por McCulloch-Pitts [21]

En este modelo:

- x_i es la entrada i-esima de la neurona.

- w_i es el peso i-esimo entre la entrada y la neurona
- θ es un valor umbral
- s es una funcion no lineal (función de activación)
- o es la salida de la neurona

Este modelo fue desarrollándose en los siguientes años. En 1957 Rosenblatt presentó el algoritmo Perceptrón, pero poseía una limitación, ya que solo podía implementar funciones discriminantes lineales. Posteriormente, en 1986, Rumelhart, Hinton y Williams presentarían la técnica de retropropagación del error, con lo que las redes neuronales multicapa comenzaron a ser aplicables a problemas reales. Posteriormente, en 2006, Hinton presentó las redes profundas; gracias a esta técnica y a una mejora sustancial del *hardware* ha sido posible ver aplicaciones en distintos ámbitos con precisiones que marcan el actual estado del arte.

Perceptrón multicapa

Un perceptrón multicapa es un modelo estimado por aprendizaje automático supervisado, consistente en un conjunto de neuronas distribuidas en capas e interconectadas entre sí. Existe una capa de entrada, la cual percibe los datos y los propaga por la red hasta llegar a la capa de salida.

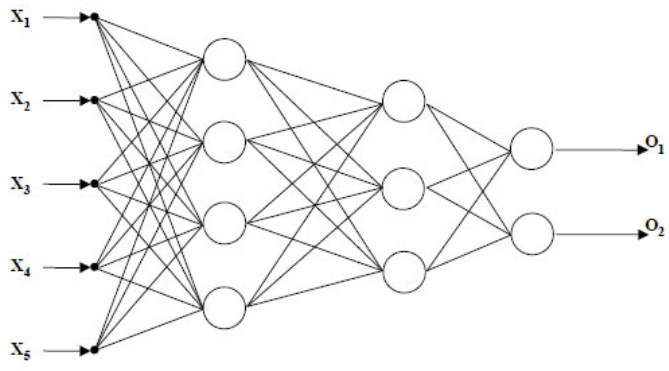

Figura 2.12: Perceptrón multicapa con 2 capas ocultas

En la Figura 2.12 se detalla una red cuya entrada es un vector con 5 componentes, la primera capa oculta posee 4 unidades, la segunda 3 y la capa de salida 2 neuronas.

Una vez definida la arquitectura de la red es necesario entrenar el modelo; para ello, se le proporciona un conjunto de muestras etiquetadas y se aprenden los pesos mediante el algoritmo BackPropagation (Figura 2.13).

Entrada: Topología, pesos iniciales $\theta_{ij}^{l} \quad 1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, factor de aprendizaje p, condiciones de convergencia, N datos de entrenamiento S.
Salidas: Pesos de las conexiones que minimizan el error cuadrático medio de S

Mientras no se cumplan las condiciones de convergencia:
Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$, inicializar $\Delta\theta_{ij}^{l} = 0$

Para cada muestra de entrenamiento (x,t) $\in$ S
Desde la capa de entrada a la de salida(l=1 , ... , L)
Para $1 \leq i \leq M_l$ calcular : ϕ_i^l y $s_i^l = g(\phi_i^l)$

Desde la capa de salida a la de entrada (l = L , ..., 1)
Para cada nodo $(1 \leq i \leq M_l,)$
Calcular $\delta_i^l = g^{'}(\phi_i^l)(t_{ni-s_i^L})$ si $l == L$ o $g^{'}(\phi_i^l)(\sum_r \delta_r^{l+1}\theta_{ri}^{l+1})$
Para cada peso $\theta_{ij}^{l}(0 \leq j \leq M_{l-1})$ calcular : $\Delta\theta_{ij}^{l} = \Delta\theta_{ij}^{l} + p\delta_i^l\, s_j^{l-1}$

Para $1 \leq l \leq L, 1 \leq i \leq M_l, 0 \leq j \leq M_{l-1}$ Actualizar pesos: $\theta_{ij}^{l} = \theta_{ij}^{l} + \frac{1}{N}\Delta\theta_{ij}^{l}$

Figura 2.13: Algoritmo BackPropagation

Redes convolucionales

Las redes neuronales convolucionales son un tipo de redes neuronales basadas en aprendizaje supervisado que tratan de realizar un procesado similar al que utiliza el córtex visual del ojo para identificar patrones.

El aspecto distintivo de este tipo de redes con respecto al resto es la operación de convolución. Esta operación consiste en la multiplicación mediante producto escalar de una matriz, llamada Kernel, contra la matriz de entrada. La matriz Kernel debe tener una dimensión menor a la imagen de entrada, por lo que al aplicarla sobre la imagen de entrada dará como resultado una matriz de un tamaño inferior (Fig. 2.14).

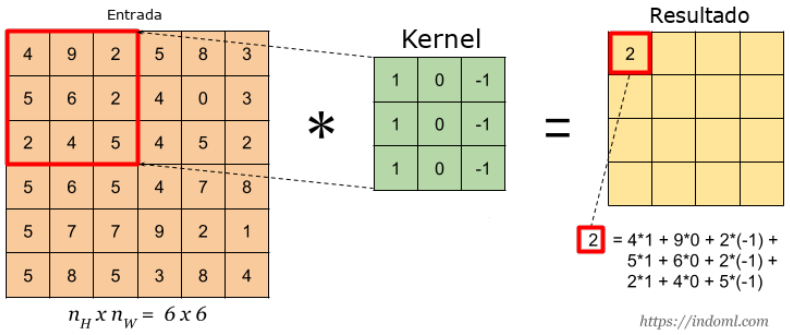

Figura 2.14: Ejemplo de operación de convolución [22]

En primera instancia esta nueva matriz Kernel se inicializa a valores aleatorios y se irá adaptando tras sucesivas iteraciones del algoritmo BackPropagation.

Otra de las capas muy utilizadas en redes convolucionales es la capa Max-Pool o Min-Pool. Este tipo de capa consiste en definir una matriz de dimensiones reducidas, habitualmente 2x2 o 3x3, y aplicarla sobre una matriz, quedándonos con el valor máximo o mínimo contemplado por la ventana (Fig. 2.15).

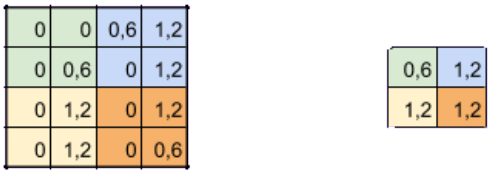

Figura 2.15: Ejemplo de aplicar un max-pool con una ventana de 2x2 [22]

CAPÍTULO 3
Author profiling

3.1 Contexto

El *Author profiling* consiste en el analísis del contenido compartido en distintos medios con el fin de predecir distintos atributos del autor, como puede ser el género, la edad o la orientación política, entre otros.

Las primeras investigaciones en *Author Profiling*, desarrolladas en el año 2003, se centraron en textos formales [1]. Trataron de establecer diferencias entre la escritura de las mujeres y la de los hombres utilizando para ello un total de 604 documentos del British National Corpus (BNC).

Actualmente, las investigaciones se centran en redes sociales, ya que el contenido es mucho más espontaneo y menos formal. Gracias a estas investigaciones se han conseguido desarrollar tecnologías en distintos ámbitos como la detección de pederastas [2] o potenciales terroristas.

3.2 Conjunto de datos: PAN 2018

El PAN es una serie de eventos científicos acerca de análisis forense de texto y recursos digitales. Algunas de las competiciones que se desarrollan son:

- Determinar si un usuario es un *bot* o un humano a partir del *feed* de Twitter
- Determinar el autor de un texto de entre una lista de candidatos
- Determinar el grado de fama, ocupación, edad y género dado el *feed* de una celebridad

El corpus proporcionado en la tarea del PAN 2018 para la tarea de detección de género consta de un conjunto de usuarios de Twitter etiquetados según su género. Para cada autor se proporcionan un total de 100 tuits y 10 imágenes. Además, cada autor se agrupa según la lengua de sus tuits: inglés, árabe o español. Cabe destacar que el *dataset* está equilibrado en cuanto al género. La distribución de los autores se puede apreciar en la Tabla 3.1.

Tabla 3.1: Distribución de autores por idioma y subconjunto de entrenamiento o test [8]

	Árabe	Inglés	Español	Total
Entrenamiento	1500	3000	3000	7500
Test	1000	1900	2200	5100
Total	2500	4900	5200	12600

3.2.1. Estructura

El corpus proporcionado se distribuye utilizando una carpeta para cada idioma. Por cada idioma existe una carpeta llamada `train` y otra `test`. Dentro de cada una de estas carpetas se disponen de tres ficheros:

- Una carpeta `text` con los ficheros estructurados en formato xml y referentes a los 100 tuits publicados por cada autor
- Una carpeta `photo` que incluye una subcarpeta por cada autor con las 10 imágenes de su *TimeLine*
- Un fichero llamado `<es|en|ar>.txt` que contiene para cada autor una etiqueta que indica si es hombre o mujer

La Fig 3.1 muestra la distribución de carpetas en el caso de idioma español.

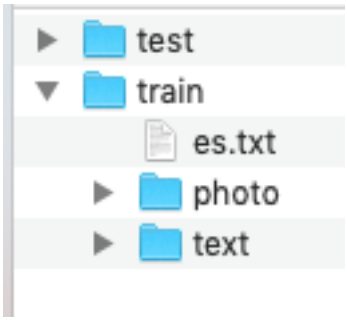

Figura 3.1: Distribución de las carpetas para el español

Como se ha comentado, la parte textual se representa en un fichero .xml por autor. La estructura de estos ficheros se puede apreciar en la Figura 3.2:

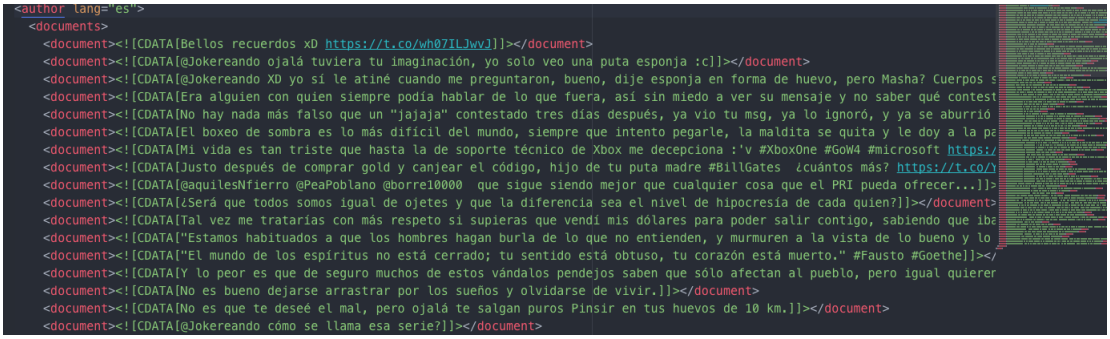

Figura 3.2: Distribución de 100 tuits en formato xml

En cuanto a las imágenes, se dispone de 10 imágenes por cada autor. En las Figuras 3.3 y 3.4 se pueden apreciar 10 autores (filas) y las 10 imágenes publicadas en su *TimeLine* (columnas).

Figura 3.3: Imágenes publicadas por mujeres

Figura 3.4: Imágenes publicadas por hombres

3.2.2. Métricas

El objetivo de la competición consiste en obtener la menor tasa de error posible. Para ello, es necesario proporcionar tres predicciones: basada en texto, basada en imagen y una combinación de ambas. El ranking se calculará para cada modalidad y para cada idioma siguiendo la siguiente ecuación:

$$ranking = \frac{acc_{ar} + acc_{es} + acc_{en}}{3} \tag{3.1}$$

Donde $acc_{ar}, acc_{es}, acc_{en}$ son las precisiones obtenidas para los tres idiomas de la competición.

3.2.3. Resultados

El mejor resultado obtenido en la **modalidad de texto** en español lo obtuvo el equipo Daneshvar con un 82.00 % de precisión [5]. Para ello realizaron un preproceso eliminando los *hashtags*, menciones y direcciones url. Además, reemplazaron cadenas con más de tres caracteres iguales por una cadena de exactamente tres caracteres y pasaron a minúsculas todo el texto.

Seguidamente realizaron una representación basada en bolsa de palabras con distintos tamaños de n-gramas, ponderada por TF-IDF, y eliminaron aquellos términos muy frecuentes. Por último entrenaron una máquina de vectores soporte con un Kernel lineal.

En cuanto a la **modalidad de imagen**, el mejor resultado obtenido en español lo ha conseguido el equipo Takashashi, con un 77.32 % de precisión [7]. Para ello utilizaron redes neuronales convolucionales basadas en ImageNet. El modelo presentado por los autores está basado en tres pasos:

- Extracción de características utilizando una arquitectura pre-entrenada basada en redes neuronales convolucionales (CNN) basadas en VGG16
- Fusión de las características extraídas
- Procesado de la fusión mediante una red neuronal fuertemente conectada

En primer lugar, cada imagen se procesa mediante redes neuronales convolucionales con el objetivo de realizar una extracción de características. Una vez extraídas las características de las 10 imágenes se combinan utilizando la media o el máximo. Por último, la salida combinada se procesa mediante una red neuronal fuertemente conectada para obtener la representación de cada autor.

Además, este equipo ha conseguido obtener la mayor precisión en la **modalidad combinada** de texto e imagen, alcanzando una precisión del 81.59 % para el español. Para ello en la modalidad de texto utilizaron una representación basada en *word embeddings* preentrenados con 3.17 millones de tuits en español y entrenaron un modelo basado en redes neuronales recurrentes. Por último, utilizaron un módulo de fusión para combinar las relaciones entre las imágenes y el texto de cada autor. La estructura del modelo se puede ver en la imagen de la Figura 3.5.

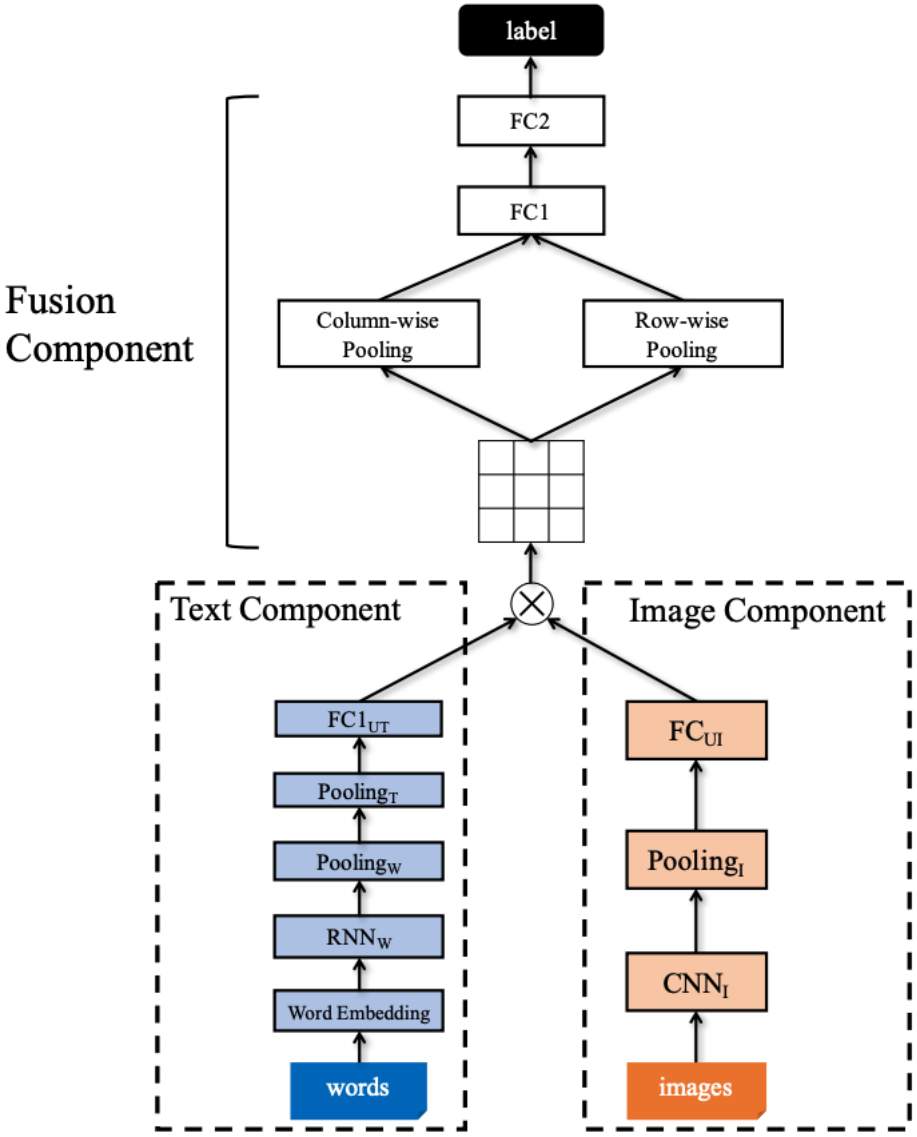

Figura 3.5: Modelo basado en redes neuronales propuesto por los autores en [7]

CAPÍTULO 4
Propuesta de solución

El objetivo principal de este capítulo es comentar, a grandes rasgos, el diseño de la solución al problema de detección de género en Twitter, así como comentar las tecnologías que se han empleado para su resolución.

4.1 Diseño de la propuesta

Para resolver el problema propuesto se propone un sistema de reconocimiento de patrones multimodal que combine las imágenes y los tuits publicados para determinar el género del usuario (Fig. 4.1). A continuación se detallará cada uno de los módulos del mismo.

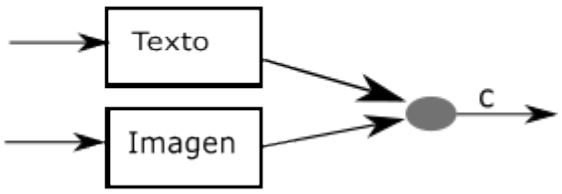

Figura 4.1: Esquema de combinación de clasificadores de texto e imagen

4.1.1. Reconocimiento textual

Para el reconocimiento del género en tuits se propone un sistema que utiliza dos tecnologías para la representación de la información textual. Por una parte *Bag of Words* (Apartado 2.1.1) y por otra *Word embeddings* (Apartado 2.1.2). El esquema del módulo textual puede verse en la Figura 4.2.

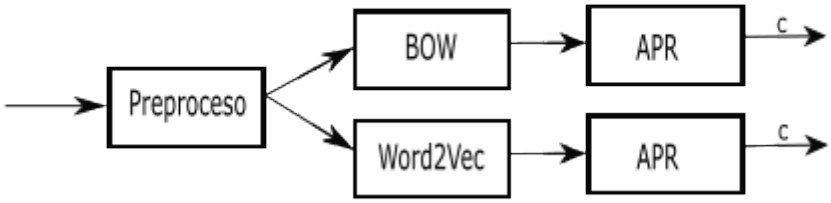

Figura 4.2: Arquitectura textual propuesta

En primer lugar se realiza un preproceso del texto con el objetivo de normalizar la entrada y, de esta forma, reducir el tamaño del vocabulario. Seguidamente se realizan las representaciones: por una parte se realizará una bolsa de palabras y por otra una representación basada en un Word2Vec preentrenado sobre 1 millón de tuits. Por último se entrenarán los clasificadores que distingan los tuits en base al género.

4.1.2. Reconocimiento en imágenes

Por otra parte, para el reconocimiento del género en imágenes inicialmente se propuso un sistema basado en tres tecnologías para la representación de imágenes: en primer lugar el uso de histogramas RGB (Apartado 2.2.1), el uso de HaarCascade (Apartado 2.2.2) y redes neuronales convolucionales (Apartado 2.3.3) para la detección de género en caras. También se probó el uso de VGG (Apartado 2.2.3) para realizar una representación basada en descriptores. Dada la baja precisión que obtuvo el modelo basado en histogramas RGB, terminó por ser descartado de la solución final propuesta. El esquema final de este sistema se presenta en la Figura 4.3.

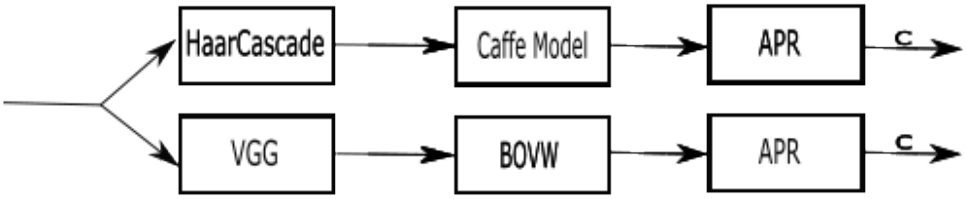

Figura 4.3: Arquitectura propuesta para el reconocimiento del género en imágenes

4.2 Tecnología empleada

Como lenguaje de programación se ha escogido Python debido a la gran cantidad de bibliotecas y utilidades como Scikit Learn [12], NLTK [13] o Tensorflow [14].

La gran mayoría de los modelos desarrollados se han realizado con la libreria Scikit-Learn. Scikit-Learn [12] es un *framework Open source* construido sobre NumPY[1], SciPy[2] y matplotlib[3]. Este *framework* permite, de forma eficiente y sencilla, realizar operaciones de análisis de datos.

[1] https://www.numpy.org/
[2] https://www.scipy.org/
[3] https://matplotlib.org/

Entre las operaciones que permite realizar están: el pre-proceso y extracción de características, técnicas de reducción de dimensionalidad y técnicas de clasificación y regresión (como las máquinas de vectores soporte, etc.). También dispone de distintas técnicas de selección de modelos como búsqueda en *grid* o validación cruzada.

Para el tratamiento de imágenes se ha utilizado OpenCV [16]. OpenCV es una biblioteca escrita en C/C++ para tareas de visión por computador. Posee interfaces para Java, C++ y Python, y es soportada por Windows, Linux y MacOS. Permite realizar operaciones con imágenes y posee distintos algoritmos para el procesado y la extracción de características.

Para la detección del género en fotos se han utilizado dos *frameworks*. Por una parte, Caffe [17], un *famework* basado en *deep learning* pensado para la clasificación utilizando redes neuronales profundas en tareas con imágenes. Mediante el uso conjunto con OpenCV se pueden cargar modelos pre-entrenados especificando un fichero con la arquitectura de la red y otro con los pesos de las neuronas.

Por otra parte se ha utilizado una red neuronal basada en VGG 16 (Apartado 2.2.3) y entrenada sobre ImageNet. Esta red ha permitido la extracción de etiquetas descriptivas sobre las imágenes. Esta arquitectura se puede ejecutar sobre Tensorflow + Keras[4].

[4] https://keras.io/

CAPÍTULO 5
Desarrollo de la solución

En este capítulo se detallarán las distintas fases que componen la solución propuesta anteriormente. Para ello se ahondará en las distintas fases del sistema de reconocimiento propuesto.

5.1 Preproceso

El objetivo principal de la fase de preproceso consiste en adaptar los datos de entrada para que la tarea de representación y aprendizaje sea menos ardua. A continuación se detalla el preproceso seguido tanto para la parte textual como para la de imágenes.

5.1.1. Preproceso textual

El objetivo principal del preproceso textual pasa por normalizar el texto. En la propuesta presentada cada autor se representa como un conjunto de tuits unidos por la etiqueta `<FinTweet>`. El preproceso seguido para cada tuits consiste en:

- Reemplazar las direcciones url por la etiqueta `<url>`
- Reemplazar los *hashtags* por la etiqueta `<hashtag>`
- Sustituir las menciones por la etiqueta `<mencion>`
- Eliminar los números sustituyéndolos por la etiqueta `<numero>`
- Convertir el texto a minúsculas

5.1.2. Preproceso en imágenes

Por otra parte, el preproceso seguido con las imágenes, en el caso de realizar la representación por histograma, ha consistido unicamente en reescalar todas las imágenes a un tamaño fijo.

Además, se ha utilizado el algoritmo HaarCascade comentado en el apartado 2.2.2 para extraer, si existen, las caras de las imágenes. Para el uso de este modelo es necesario dos cosas:

- Importar el módulo `CascadeClassifier` de openCV
- Cargar un fichero xml [20] con el modelo pre-entrenado

Al usar el algoritmo obtenemos una lista con las posiciones de inicio en (x,y), así como el ancho y alto de los rectángulos que contienen las caras.

5.2 Representación

Una vez realizado la fase de preproceso es necesario realizar una representación de la información textual y gráfica. Dicha representación será la que después se utilizará para realizar el aprendizaje.

5.2.1. Representación textual

En cuanto a la representación del contenido textual se han seguido dos estrategias: por una parte la representación basada en bolsa de palabras y por otra parte una representación basada en *word embeddings*.

Bag of Words

Se proponen dos tipos de representaciones basadas en *Bag of Words*. Por una parte una bolsa de palabras por n-gramas y por otra una bolsa de palabras por n-gramas de carácteres (char-n-gramas). Las características de cada una se detallan a continuación:

n-gramas:

- n $\in$ [1, 2]
- Ponderación TF-IDF
- Borrado de términos que aparecen en más del 95 % de los documentos
- Borrado de términos que aparecen en un único documento
- Borrado de *StopWords*

char-ngramas:

- n $\in$ [3,5]
- Ponderación TF-IDF
- Borrado de términos que aparecen en más del 95 % de los documentos
- Borrado de términos que aparecen en un único documento

Se ha considerado el borrado de las *StopWords* ya que, por lo general, son palabras muy repetidas y no suelen aportar información acerca del discurso del autor.

Este proceso se ha realizado utilizando las bibliotecas NLTK para la obtención de las *StopWords* en español y Scikit-Learn para realizar la representación utilizando la clase `TfidfVectorizer`.

Word embeddings: Word2Vec

Por otra parte se ha realizado una representación basada en *word embeddings*. Mediante la biblioteca Gensim [18] se ha generado un modelo basado en Word2Vec [9] con un tamaño de ventana de 5 y un tamaño del *embedding* de 300.

El modelo se ha entrenado con un total de un millón de tuits extraídos de una combinación de *streamings* de Twitter y datasets del PAN de años anteriores.

Una vez entrenado el modelo se obtienen los *embeddings* y, por cada documento, se recorren las palabras obteniendo la respectiva representación (ignorándola en caso de que no exista) y almacenándola para, posteriormente, realizar una media por columnas. De esta forma cada autor se representará como un vector de 300 características fruto de la media de los *embeddings* de las palabras utilizadas en sus *tweets*.

5.2.2. Representación en imágenes

En este apartado se detalla los distintos métodos utilizados para la representación de imágenes: la representación por histograma RGB, una aplicación del algoritmo HaarCascade con redes neuronales para extraer el género en caso de existir caras y VGG para la extracción de descriptores.

Representación por histograma

Para realizar esta representación, tal como se ha comentado en el apartado 2.2.1, se ha computado el histograma RGB de las 10 imágenes publicadas por el autor. De esta forma, un autor se representa con la media de cada columna de los 10 histogramas RGB de sus imágenes.

Porcentaje de hombres en fotos

Para la extracción de esta característica se ha realizado el preproceso de las 10 imágenes de cada autor como se ha comentado en el punto 5.1.2. Una vez extraídas las listas con las caras de las personas, se ha utilizado un modelo pre-entrenado y basado en redes neuronales para detectar si una cara pertenece a un hombre o a una mujer.

Por tanto, para cada autor se ha extraído una única característica, la cual se detalla a continuación:

$$p = \frac{n^{o}carashombre}{n^{o}carashombre + n^{o}carasmujer}$$

VGG

Se ha utilizado la arquitectura VGG (Apartado 2.2.3) preentrenada sobre Imagenet para extraer descriptores de las imágenes.

Seguidamente se ha realizado una bolsa de palabras visual donde cada autor se representa como la suma de las frecuencias de aparición de los descriptores en las 10 imágenes publicadas en su *TimeLine*. Por ejemplo, pongamos dos autores donde cada uno publica las siguientes imágenes:

(a) Autor 1 **(b)** Autor 2

La bolsa de palabras visual para cada autor sería la que se muestra en la Tabla 5.1.

Tabla 5.1: Ejemplo de *Bag of Words* visual para 2 autores

Característica	Autor1	Autor2
perro	1	0
bicicleta	1	0
casa	1	0
árbol	1	0
coche	0	1
avión	0	1

De esta forma, una imagen se puede representar como un vector de etiquetas que describan el contenido de la figura. En el ejemplo presentado el tamaño del vocabulario es de 6, pero en nuestro caso el vector tendrá una dimensión de 1000 componentes, ya que la red VGG preentrenada sobre Imagenet es capaz de distinguir entre 1000 tipos distintos de imágenes.

5.3 Experimentos

Para el desarrollo de los experimentos se han explorado las distintas representaciones comentadas en la Sección 5.2. Para evaluar las representaciones de los modelos se ha procedido de la siguiente forma:

- Realizar validación cruzada sobre el conjunto de entrenamiento para determinar el modelo que mejor se comporta en esta fase. Similar a lo realizado por los autores en [5]

- Entrenar un modelo utilizando SVM variando el Kernel y el parámetro de regularización C realizando de esta forma una búsqueda en *grid*

- Entrenar un modelo utilizando regresión logística variando el parámetro de regularización C

5.3.1. Texto

Como se ha comentado anteriormente, las dos representaciones explotadas en cuanto a información textual son las distintas bolsas de palabras y los *word embeddings*.

Validación cruzada

Los resultados por validación cruzada sobre el conjunto de entrenamiento se pueden visualizar en la Tabla 5.2.

Tabla 5.2: Intervalos al 95 % realizando validación cruzada C = 10 y Kernel Lineal en los modelos basados en BOW y Word2Vec

Representación	SVM	Regresión Logística
n-gramas	[0.73, 0.79]	[0.75,0.79]
char-n-gramas	[0.74, 0.78]	**[0.74,0.80]**
word2vec	[0.64, 0.72]	[0.62,0.74]

A la vista de los resultados se observa que el mejor modelo obtenido con validación cruzada se consigue utilizando char-n-gramas y regresión logística con C = 10.

Bag of Words

Tabla 5.3: Intervalos al 95 % utilizando SVM y *Bag of Words* como representación

Kernel	BOW	C=1	c=10	c=100
Linear	n-gramas	**[0.75,0.79]**	**[0.75,0.79]**	[0.75,0.78]
	char-n-gramas	**[0.77,0.80]**	[0.76,0.79]	[0.74,0.78]
RBF	n-gramas	[0.64,0.68]	[0.64,0.68]	[0.64,0.68]
	char-n-gramas	[0.62,0.67]	[0.62,0.67]	[0.62,0.67]

Tabla 5.4: Intervalos al 95 % utilizando regresión logística y *Bag of Words* como representación

	C=1	C=10	C=100	C=1000
n-grams	[0.73 , 0.77]	**[0.76 , 0.80]**	**[0.76 , 0.80]**	**[0.76 , 0.80]**
char-ngramas	[0.75 , 0.78]	**[0.77 , 0.81]**	**[0.77 , 0.81]**	[0.76 , 0.80]

Los resultados para la representación *Bag Of Words*, con intervalos de confianza del 95 %, se representan en las Tablas 5.3 y 5.4.

A la vista de los resultados, el algoritmo de regresión logística basado en una extracción de características de n-gramas de caracteres obtiene mejores predicciones disminuyendo el error en test. Además, este algoritmo tiene un menor coste computacional (O(n) frente al O(n^2) del algoritmo SVM) por lo que en caso de realizar un sistema que necesitara analizar una gran cantidad de datos resultaría más interesante implementar la solución basada en este algoritmo. Cabe destacar, a la vista de las precisiones obtenidas, que las diferencias entre ambos clasificadores no son significativas.

Word2Vec

Tabla 5.5: Intervalos al 95 % utilizando SVM y *word embeddings* como representación

Kernel	C=1	C=10	C=100
Linear	[0.63, 0.67]	[0.65, 0.69]	**[0.68, 0.72]**
RBF	[0.52, 0.56]	[0.60, 0.65]	[0.65, 0.69]

Tabla 5.6: Intervalos al 95 % utilizando Regresión logística y *word embeddings* como representación

C=1	C=10	C=100	C=1000
[0.63, 0.67]	[0.66, 0.70]	**[0.67, 0.71]**	**[0.67, 0.71]**

Los resultados para las representaciones por *word embeddings* con intervalos de confianza al 95 %, se presentan en las Tablas 5.5 y 5.6.

En este caso las máquinas de vectores soporte con un Kernel lineal y C = 100, obtienen un 70 %, fijando de esta manera la mayor precisión para la representación basada en *word embeddings*. Las soluciones obtenidas por el algoritmo de regresión logística obtienen soluciones similares con un coste computacional inferior. Se observa que no existen diferencias significativas entre ambos algoritmos.

Como se puede observar, estos modelos no consiguen alcanzar la precisión que obtienen los modelos basados en bolsa de palabras, por lo que parece que los sistemas basados en *word embeddings* no son los más adecuados para esta tarea. Puede que los algoritmos SVM o regresión logística no sean capaces de captar la expresividad que aporta utilizar este tipo de representaciones y, por tanto, resultaría más interesante utilizar otro tipo de algoritmos como las redes neuronales recurrentes propuestas por ejemplo por el equipo de Takashashi [7].

A la vista de los resultados, se observa que en ambos algoritmos existe una tendencia a aumentar la precisión al incrementar el parámetro de regularización; se observa por tanto, que un incremento en el parámetro C permite obtener fronteras de decisión con una mejor generalización.

5.3.2. Imagen

Como se ha comentado en el apartado 5.2.2, las representaciones propuestas para solucionar la tarea de reconocer el género en imágenes son tres: un histograma RGB, el porcentaje de hombres en fotos (% de H) y la aplicación de VGG para realizar una representación basada en descriptores.

Validación cruzada

Los resultados obtenidos con intervalos de confianza al 95 % utilizando validación cruzada sobre el conjunto de entrenamiento se observan en la Tabla 5.7.

Tabla 5.7: Intervalos al 95 % realizando validación cruzada en los modelos basados en BOVW y porcentaje de hombres en fotos

Representación	SVM	Regresión Logística
% de H	[0.55, 0.67]	[0.56, 0.66]
BOVW	**[0.62, 0.72]**	[0.58, 0.70]

En este caso, las bolsas de palabras visuales con Kernel RBF y C =1 obtiene los mejores resultados en el caso realizar una validación cruzada sobre los datos de entrenamiento.

Histograma RGB

Para esta representación se han realizado experimentos con distintos algoritmos de aprendizaje, pero los resultados han sido negativos. En el mejor de los casos se alcanzaba una precisión del 55 %.

Estos resultados tan bajos se puede deber a la gran variedad de imágenes que existen. Por lo tanto, una representación que no tiene en cuenta la geometría de los objetos que aparecen en la escena (como es esta) no parece ser adecuada para esta tarea.

Porcentaje de hombres en fotos

Tabla 5.8: Intervalos al 95 % utilizando SVM y una representación basada en el porcentaje de hombres en fotos

Kernel	C=1	C=10	C=100
Linear	[0.59,0.64]	[0.59,0.64]	[0.59,0.64]
RBF	**[0.61,0.66]**	[0.61,0.65]	[0.61,0.65]

Tabla 5.9: Intervalos al 95 % utilizando regresión logística y una representación basada en el porcentaje de hombres en fotos

C=1	C=10	C=100	C=1000
[0.61 , 0.65]	[0.61 , 0.65]	[0.61,0.65]	[0.61, 0.65]

Los resultados obtenidos usando el porcentaje de hombres se presentan en las Tablas 5.8 y 5.9.

A la vista de los resultados podemos observar que las SVM con Kernel RBF y C = 1 obtienen la mejor precisión. Cabe destacar que la regresión logística es capaz de adaptarse de forma similar y con un menor coste computacional.

Bag of Visual Words

Tabla 5.10: Intervalos al 95 % utilizando SVM y una representación basada en descriptores de la imagen

Kernel	C=1	C=10	C=100
Linear	[0.59, 0.63]	[0.57, 0.61]	[0.57, 0.61]
RBF	**[0.65, 0.69]**	[0.64, 0.68]	[0.61, 0.65]

Tabla 5.11: Intervalos al 95 % utilizando regresión logística y una representación basada en descriptores de la imagen

C=1	C=10	C=100	C=1000
[0.61 , 0.65]	[0.58 , 0.62]	[0.58 , 0.62]	[0.58 , 0.62]

A la vista de los resultados, presentados en las Tablas 5.10 y 5.11, se observa que la mejor precisión se obtiene con las máquinas de vectores soporte con un Kernel RBF y C = 1.

Como se ha comentado anteriormente, el modelo VGG preentrenado sobre Imagenet es capaz de distinguir entre mil objetos distintos. Esto supone una alta dimensionalidad que en muchas ocasiones acaba generando ruido y, por tanto, siendo una dificultad en las etapas de aprendizaje. Una posible forma de solucionar este problema consiste en aplicar técnicas de reducción de dimensionalidad a los datos. En nuestro caso se ha utilizado PCA.

Principal Component Analysis o PCA por sus siglas en inglés es una técnica de reducción de dimensionalidad no supervisada cuyo objetivo principal es encontrar una matriz de proyección que minimice el error de reconstrucción.

En la tarea se ha tratado de mejorar la tasa de acierto de los algoritmos al ser entrenados con un conjunto de datos de una dimensionalidad con un tamaño menor al original. Para realizar el estudio se ha tomado como punto de partida una SVM con Kernel RBF y C = 1 y un modelo basado en regresión logística con C = 1.

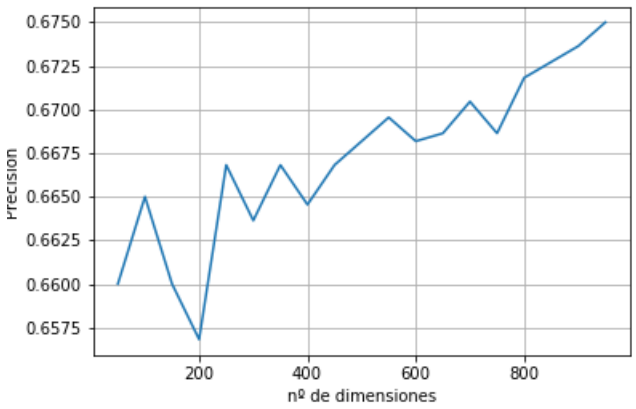

Figura 5.2: Evolución de la precisión al utilizar SVM como modelo y variar la dimensionalidad

Como se observa en el gráfico de la Figura 5.2, el modelo basado en máquinas de vectores soporte se comporta mejor cuanto mayor es el número de dimensiones, por lo que aplicar PCA no aporta ningún beneficio a la hora de disminuir la tasa de error.

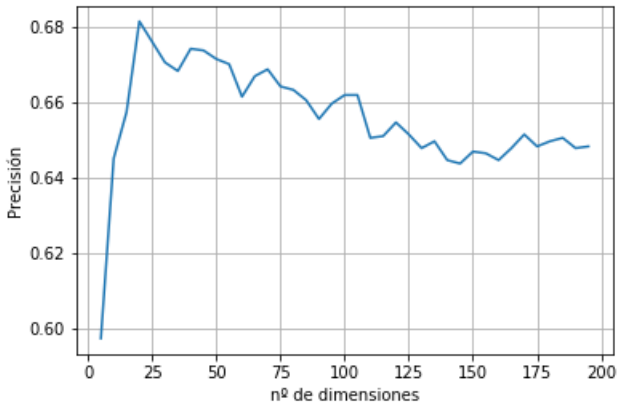

Figura 5.3: Evolución de la precisión en el modelo de regresión logística al variar la dimensionalidad

En cambio, como se muestra en la Figura 5.3, el modelo basado en regresión logística sí que se comporta mejor cuanto menor es la dimensionalidad de los datos, logrando alcanzar una precisión del 68.1 % cuando los datos poseen 20 dimensiones. El intervalo de confianza al 95 % asociado para dicha precisión es [0.66 , 0.70], que mejora, aunque no significativamente lo obtenido usando SVM.

5.4 Combinación de clasificadores

Una vez entrenados los distintos modelos para texto e imagen es necesario combinarlos para obtener una salida única al problema de detección del género en Twitter. Para ello se proponen distintos métodos de combinación como la votación mayoritaria y *Stacking*.

Para la construcción de los distintos sistemas de combinación se han escogido aquellos clasificadores que tienen una mayor precisión de manera individual. Estos son:

- Char-ngrams + regresión logística con C = 10
- Word2Vec + SVM con Kernel Lineal y C = 100
- Porcentaje de hombres en fotos + SVM con Kernel RBF y C = 1
- BOVW + PCA (20 dimensiones) + regresión logística con C = 1

5.4.1. Votación mayoritaria

El método de votación mayoritaria consiste en combinar un conjunto de clasificadores de forma que la clase establecida se determine como la mayoritaria predicha por todos los clasificadores.

Mientras que los clasificadores de forma individual presentan una precisión en el rango [0.63 , 0.80] el clasificador que combina la salidas mediante votación mayoritaria

es capaz de obtener una precisión del **0.76**. El intervalo de confianza al 95 % asociado a esta precisión es de [0.74 , 0.77].

Uno de los principales problemas que puede presentar la votación mayoritaria es que para que el sistema funcione relativamente bien es necesario que todos los modelos tengan una precisión aceptable, pero, por lo general, esto no se puede asegurar para todos los clasificadores. Por tanto, puede resultar interesante realizar una combinación de los clasificadores que tenga en cuenta cuáles de ellos se comportan de mejor forma a la hora de predecir el género.

5.4.2. *Stacking*

El algoritmo de Stacking [10] se trata de un método híbrido [11], es decir, es un algoritmo que intenta combinar distintos clasificadores con el fin de construir uno nuevo mucho más robusto.

El algoritmo se compone de dos fases. Inicialmente se entrena un conjunto de modelos de aprendizaje que deben ser capaces de distinguir entre las distintas clases del problema. Una vez entrenados los modelos iniciales se entrena un meta-modelo a partir de las clases predichas por los mismos. De esta forma, se busca que el meta-modelo sea capaz de discernir cuándo un determinado modelo se va a comportar mejor para darle mayor prioridad. Un esquema de la fase de entrenamiento del algoritmo se puede apreciar en la Figura 5.4

Entrada: Conjunto de datos etiquetados **D** = $\{(x_1,y_1), (x_2,y_2), \ldots , (x_i,y_i)\}$
Salida: Un clasificador compuesto **H**

Fase 1: Entrenar el primer nivel de clasificadores:
    Para todo clasificador h_t entrenarlo sobre D
Fase 2: Crear el nuevo conjunto de datos
    $\{(x'_i,y_i)\}$ donde $x'_i = \{ h_1(x_i), h_2(x_i), \ldots, h_t(x_i)\}$
Fase 3: Aprendizaje del meta-clasificador
    entrenar h' basado en el nuevo conjunto de entrenamiento $\{(x'_i,y_i)\}$
H(x) = h'(h_1(x),h_2(x), ... ,h_t(x))

Figura 5.4: Algoritmo de *Stacking*

Para nuestra tarea se propone un sistema basado en *Stacking* que combine las distintas tecnologías empleadas en ambas modalidades. El esquema propuesto se detalla en la Figura 5.5.

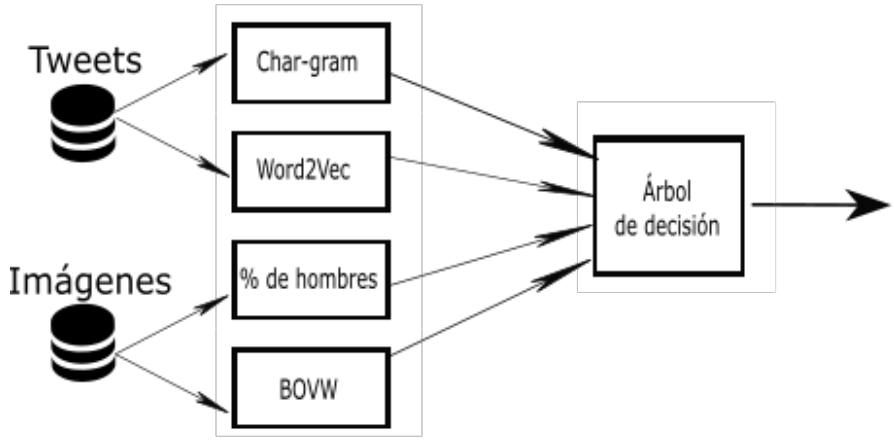

Figura 5.5: Esquema en *Stacking* propuesto para solucionar la tarea multimodal

Al realizar el entrenamiento del sistema se observa que la precisión alcanzada por el nuevo modelo es del **79.5 %**, muy similar a la obtenida por el sistema basado en char-ngramas. Esto se puede deber a que al utilizar los mismos datos para entrenar los modelos que para entrenar el metamodelo se ha producido un sobre-ajuste [11]. Una posible forma de subsanar este error consiste en utilizar un conjunto de datos distinto para adaptar el meta-modelo. El nuevo esquema algorítmico para esta nueva solución sería el presentado en la Figura 5.6.

Entrada: Conjunto de datos etiquetados **D** = $\{(x_1,y_1), (x_2,y_2), \ldots , (x_i,y_i)\}$, **P** = $\{(z_1,o_1), (z_2,o_2), \ldots , (z_i,o_i)\}$
Salida: Un clasificador compuesto **H**

Fase 1: Entrenar el primer nivel de clasificadores:
Para todo clasificador h_t entrenarlo sobre D
Fase 2: Crear el nuevo conjunto de datos
$\{(x'_i,o_i)\}$ donde $x'_i = \{ h_1(z_i),h_2(z_i), ..., h_t(z_i)\}$
Fase 3: Aprendizaje del meta-clasificador
entrenar h' basado en el nuevo conjunto de entrenamiento $\{(x'_i,o_i)\}$
H(x) = h'(h_1(x),h_2(x), ... ,h_t(x))

Figura 5.6: Algoritmo de *Stacking* con un nuevo conjunto de datos para adaptar el meta-modelo

Para subsanar el error de sobre-ajuste se ha optado por dividir el conjunto de test en dos subconjuntos:

- 20 % para adaptar el meta-modelo
- 80 % para realizar la validación del algoritmo

De esta forma se consigue una precisión del **77,5 %** con un intervalo de confianza al 95 % de [0.76 , 0.79]. El resumen de los resultados obtenidos se presenta en la Tabla 5.12.

Tabla 5.12: Precisiones de los modelos

Char-ngramas	Word2Vec	% de Hombres	BOVW	Stacking
79.6 %	69.1 %	63.3 %	68.1 %	77.5 %

Aunque este nuevo modelo no es capaz de lograr una precisión mayor que el modelo basado en n-gramas de caracteres, sí que consigue mejorar las precisiones del modelo basado en *word embeddings*, así como las de los modelos basados en detección de género en imágenes.

CAPÍTULO 6
Cosmos Profiling Twitter

El objetivo principal de este capítulo es detallar la implementación de uno de los modelos desarrollados en una aplicación práctica en el campo del marketing.

6.1 Contexto

Cosmos Profiling Twitter es una aplicación desarrollada por Autoritas Consulting [23] como parte de la *suite* de herramientas cosmos, con el objetivo de realizar censos en Twitter y aprovechar la potencia de técnicas de *Author profiling* y *machine learning* para segmentar a los usuarios.

La aplicación actualmente cuenta con módulos capaces de segmentar a los usuarios en base al discurso de los mismos. Algunos de estos módulos permiten, a partir de los tuits del usuario de Twitter:

- Detección del sexo del usuario
- Segmentación en base a la personalidad del usuario
- Segmentación del usuario en base a su edad
- Detección del lenguaje del usuario

6.2 Objetivos

El objetivo principal consiste en implementar un modelo de detección de género en caras, para de esta manera detectar el género en las fotos de perfil de los usuarios. Para el desarrollo del proyecto se propusieron los siguientes objetivos:

- La instalación debe ser sencilla, es decir, un usuario no técnico debe poder instalar la aplicación
- La aplicación debe seguir siendo multiplataforma (principalmente Windows y Mac)
- La aplicación debe seguir siendo local, es decir, no debe necesitar ejecutar tareas en un servidor *Backend* externo

6.3 Tecnología

En primer lugar se utiliza Electron [24], un *framework* para crear aplicaciónes multiplataforma de escritorio utilizando tecnologías web como Javascript, HTML y CSS. Con el uso de esta tecnología se diseña la interfaz o capa de vista de la aplicación.

En segundo lugar, para la capa lógica de la aplicación se utiliza Java. En nuestro caso será necesario también hacer uso del *framework* OpenCV, pero gracias a la facilidad de instalación y a la gran cantidad de *bindings* ha sido posible vincularlo con Java sin modificar la condición de ser multiplatforma.

Por último, para la capa de datos se utiliza Lucene [25]. Lucene es un índice de documentos almacenados mediante clave-valor. Mediante estos índices se consigue almacenar en memoria los censos de Twitter. Un posible ejemplo del sistema de índices se puede apreciar en la Figura 6.1 mediante Luke [26].

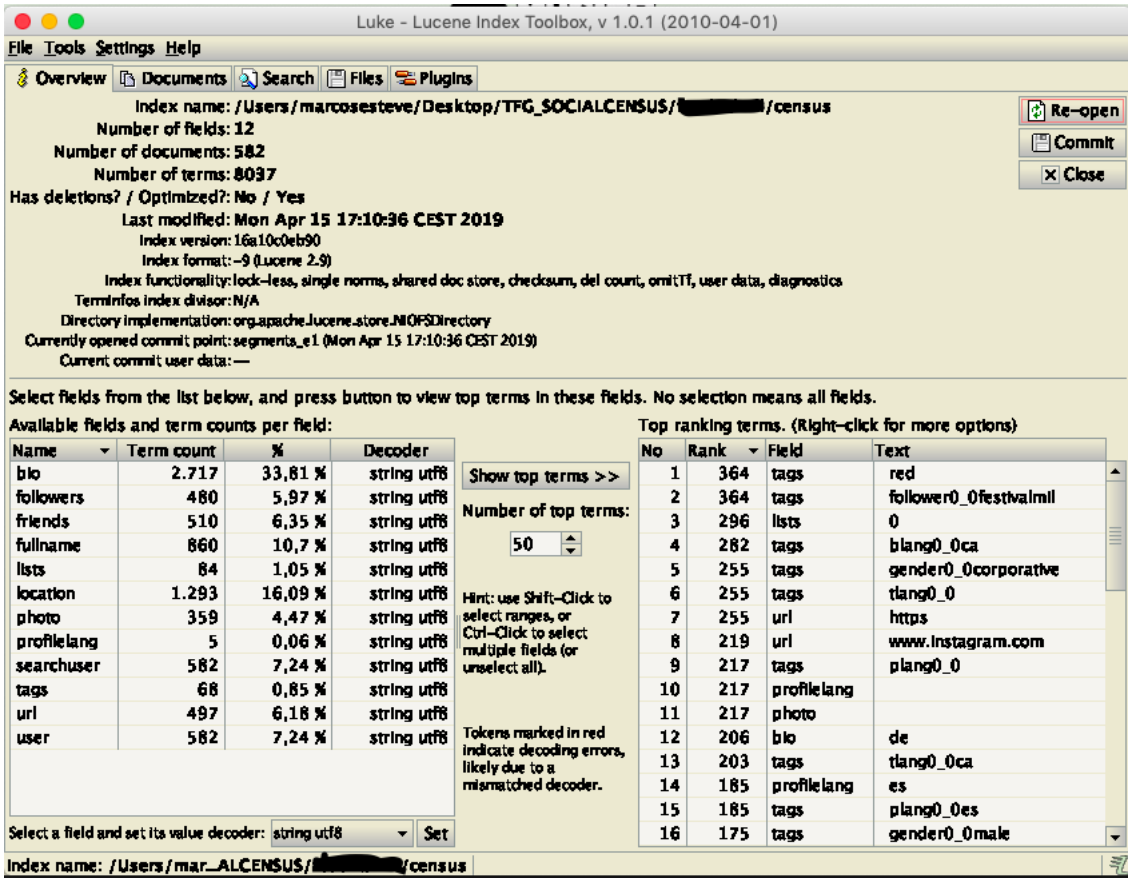

Figura 6.1: Ejemplo de índice en Luke

6.4 Desarrollo

Para el desarrollo del algoritmo detector de género mediante fotografías se ha implementado una clase `GenderByPhotoPredictor.java`.

En primer lugar se ha implementado el constructor de la clase.

```
1 public GenderByPhotoPredictor()
```

Este constructor permite realizar la carga de la biblioteca OpenCV así como la instanciación de las variables que hacen referencia al modelo de detección de caras basado en

HaarCascade y comentado en el Apartado 2.2.2, así como el modelo basado en redes neuronales para la detección del sexo al proporcionar una fotografía de una cara.

Seguidamente, se ha implementado una función `Predict`.

```
1 public String Predict(String photoURL)
```

Esta función se encarga de, dada la dirección URL de una fotografía, devolver un *String* formado por la predicción, la confianza y el nombre del fichero con la cara detectada almacenado en memoria. Para ello la función debe realizar distintos pasos. En primer lugar se debe obtener la imagen asociada a la dirección URL y transformarla a un formato tratable por OpenCV. Seguidamente se pasa a un módulo HaarCascade que, en caso de detectar una cara, esta es recortada y transmitida al módulo basado en redes neuronales Caffe para detectar el género. Un esquema del funcionamiento de la función `Predict` se puede apreciar en la Figura 6.2.

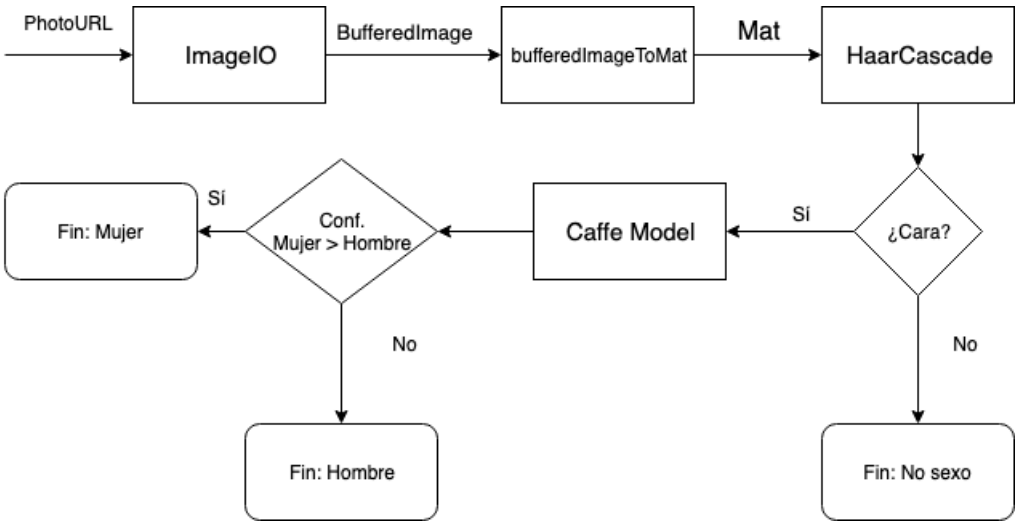

Figura 6.2: Flujo de ejecución de la función `Predict`

Además, se ha dejado el proyecto preparado para futuras tareas de visualización, ya que todas las imágenes que contienen un usuario etiquetado como hombre o mujer son guardadas en disco con su etiqueta y la predicción, así como la región de la cara analizada.

6.5 Pruebas

Para probar el sistema se ha escogido un censo proporcionado por la empresa. En este caso se trata de un censo formado por 582 usuarios. En algunos casos no se ha podido obtener la imagen de perfil asociada al usuario, debido a que esta no estaba disponible, por lo que al final se han conseguido recuperar 314 imágenes distintas. Algunas de las imágenes se pueden apreciar en la Figura 6.3.

Figura 6.3: Ejemplos de imágenes de perfil de Twitter en el censo empleado

Cabe destacar que las imágenes asociadas a los perfiles de usuarios tienen una gran cantidad de temáticas, por lo que la complejidad de escenas es muy alta. Además, no todos los usuarios tienen fotos de sus caras, por lo que no todos ellos podrán ser perfilados mediante su foto de perfil.

6.5.1. Resultados positivos

En nuestro censo se ha observado que el sistema ha sido capaz de detectar con exactitud un gran número de casos, muchos de ellos con una complejidad de escena elevada. A continuación se comentan algunos de estos resultados.

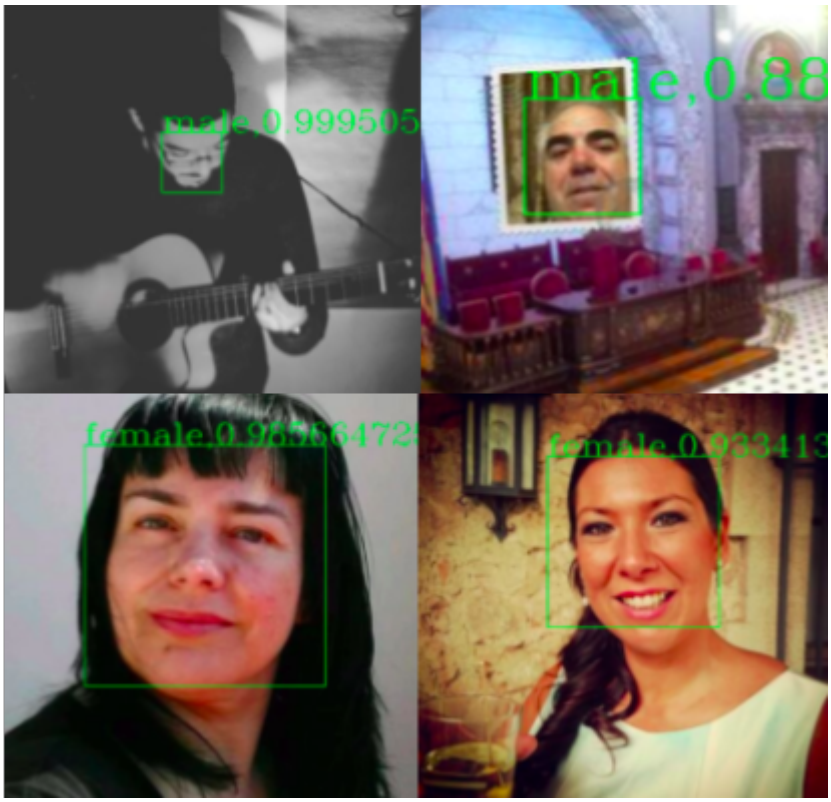

Figura 6.4: Casos de acierto en el sistema de detección de género en caras

Tal y como se muestra en la Figura 6.4, el sistema formado por HaarCascade y redes neuronales es capaz de detectar correctamente el género en una gran variedad de imágenes, aunque el rostro no se muestre completamente. Además el sistema ha sido capaz de distinguir el género en escenas muy complicadas como son las mostradas en la Figu-

ra 6.5. En estos casos se observa que el sistema de detección de caras HaarCascade ha fallado, pero la red neuronal ha sido capaz de trabajar con información imprecisa.

Figura 6.5: Casos de acierto con fallos en el sistema HaarCascade

6.5.2. Resultados negativos

Al igual que el sistema es capaz de detectar las caras, existen muchos casos en los cuales el sistema de reconocimiento de formas no es capaz de acertar el género del usuario. En este ámbito distinguimos dos errores. Por una parte un fallo en el sistema de detección de caras, lo que induce a que la red neuronal trabaje con información errónea. Por otra parte, un fallo en la red neuronal provocando que la imagen se etiquete mal.

Error en HaarCascade

En este caso, el error se debe a que el sistema de detección de caras HaarCascade detecta una cara donde no hay y, por tanto, la información que le pasa al modelo basado en redes neuronales es errónea y no se corresponde con una cara. Algunos ejemplos se pueden apreciar en la Figura 6.6.

Figura 6.6: Error en la detección de caras

Error en la red neuronal

En este último caso, el sistema de detección de caras sí que trabaja adecuadamente pero el sistema de redes neuronales no es capaz de acertar el género de la foto. En la Figura 6.7 se pueden apreciar algunos de estos errores.

Figura 6.7: Error en la detección del género con la red neuronal

6.6 Ejemplo de visualización

Entre los múltiples modos de los que dispone la aplicación desarrollada por Autoritas a continuación se muestran dos capturas donde la funcionalidad implementada ayuda a realizar la segmentación de los usuarios.

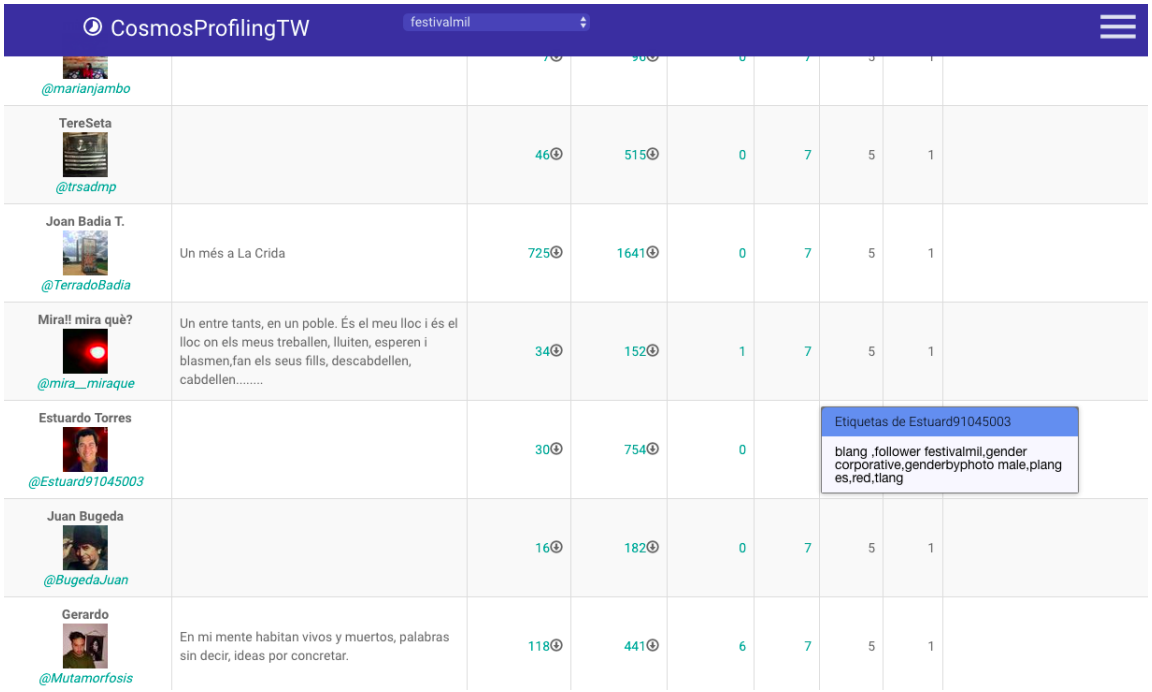

Figura 6.8: Etiquetas asignadas al usuario Estuardo Torres

En la Figura 6.8 se puede observar que el usuario Estuardo Torres ha sido etiquetado como un hombre gracias al uso del algoritmo de detección de caras en fotos. Esto se puede confirmar si observamos la foto de perfil que tiene asociada el usuario.

La potencia que aporta esta aplicación es la capacidad de realizar segmentaciones a una población analizando las características como conjunto. Esto se puede observar en la Figura 6.9, donde se observa la distribución de etiquetas asociada al género de los usuarios. En particular, se observa que de los 582 usuarios analizados el 11.32 % de los usuarios analizados por la foto han sido etiquetados como hombres, el 6.53 % como mujeres y el 35.74 % no ha podido ser etiquetado por el género de su foto. Es interesante destacar que la suma no es igual al 100 %. Esto se debe a que una gran cantidad de usuarios no disponen de fotos de perfil o éstas son erróneas.

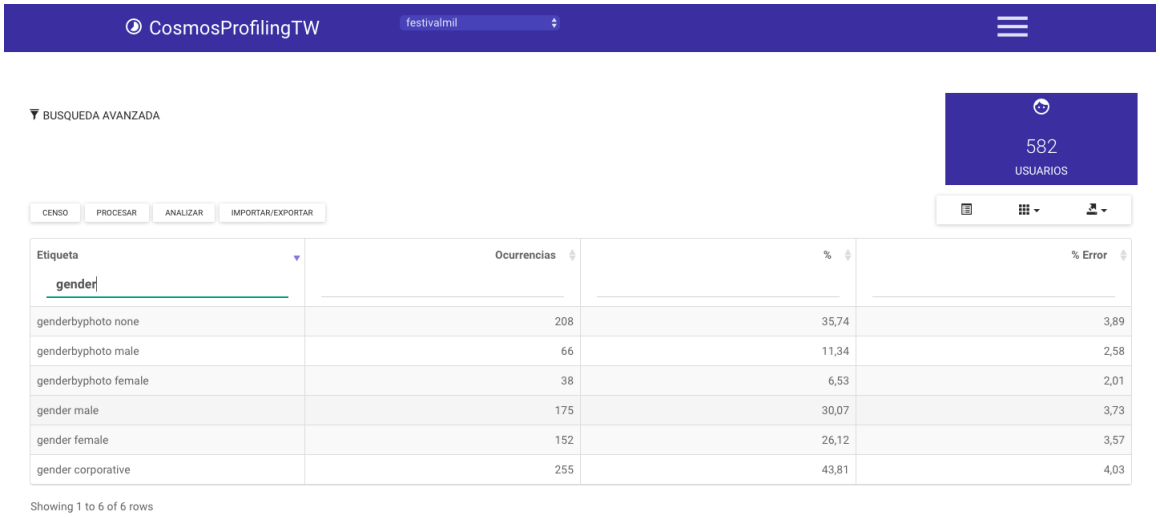

Figura 6.9: Análisis de las etiquetas utilizadas en el censo empleado

CAPÍTULO 7
Conclusiones

La gran cantidad de información textual y gráfica que existe en redes sociales, junto con la posibilidad de ser analizada con el fin de extraer conocimiento, ha marcado un punto de inflexión en campos como el marketing o la seguridad.

Con esta motivación surge el objetivo principal de nuestro trabajo: construir un sistema capaz de determinar el género de los usuarios a partir del contenido que publican en Twitter. Para ello se propuso elaborar sistemas capaces de realizar la detección sobre las imágenes y el texto y, una vez elaborados, intentar combinarlos para obtener una predicción combinada.

Para construir el sistema textual se han abordado distintas técnicas como el uso de n-gramas, los cuales han demostrado obtener los mejores resultados para la tarea, así como el uso de *word embeddings* que pese a no haber sido explicado durante la carrera ha resultado especialmente interesante debido a la gran potencia que están logrando en numerosas tareas en el campo de la lingüística computacional.

En cuanto al sistema basado en imágenes se ha propuesto principalmente el uso de dos tecnologías, no explicadas durante la realización del grado, y basadas en el uso de técnicas novedosas en el campo del aprendizaje profundo. Por una parte, se ha propuesto un sistema basado en el uso del algoritmo HaarCascade y redes neuronales profundas para la detección del género en caras. Por otra parte, se ha propuesto un sistema que utiliza tecnología de vanguardia (VGG) para extraer etiquetas descriptivas en las imágenes.

Una vez elaborados los reconocedores basados en imagen y texto se ha tratado de combinar las salidas de ambos subsistemas con el fin de obtener una salida común. Para ello se ha propuesto dos tipos distintos de combinación (votación mayoritaria y *Stacking*). Esta tarea ha sido especialmente ardua y, aunque se ha conseguido mejorar la precisión de algunos clasificadores, no ha sido posible mejorar la precisión global del sistema.

Por ultimo, se ha implementado un reconocedor de género en caras en la aplicación SocialCensus desarrollada por la empresa Autoritas Consulting. Esta tarea ha sido especialmente interesante, ya que ha permitido implementar un sistema aparentemente pensado para un trabajo académico en una solución práctica con el fin de resolver un problema real, como es el de segmentar a los usuarios en Twitter a partir del contenido que comparten.

Además, a nivel profesional este trabajo ha supuesto un reto debido a la necesidad de profundizar en un gran número de tecnologías no comentadas en el grado, como es el caso del algoritmo HaarCascade, la arquitectura VGG o los *word embeddings*, así como aspectos aparentemente teóricos, como es la construcción de un sistema completo de reconocimiento de formas. También ha permitido profundizar en el uso de Python, así

como de algunas bibliotecas en el campo del *machine learning* utilizadas por empresas como Google, Intel o Twitter.

CAPÍTULO 8
Relación con los estudios cursados

La rama de computación ha permitido cursar distintas asignaturas que han tenido un impacto directo sobre este trabajo.

En primer lugar, asignaturas como sistemas de almacenamiento y recuperación de la información (SAR) han permitido profundizar en el uso de Python para el tratamiento de texto así como para su representación.

Por otra parte, asignaturas como percepción (PER) o aprendizaje automático (APR) han sentado las bases teóricas y prácticas sobre el desarrollo de sistemas de reconocimiento de formas y la aplicación de distintos algoritmos como las máquinas de vectores soporte o las redes neuronales.

Han sido necesarios conocimientos básicos de programación y estructuras de datos para desarrollar los programas. Asignaturas como estructuras de datos y algoritmos (EDA) han sido de gran ayuda en la representación y procesado de la información.

Por último, ha sido necesario conocimientos proporcionados por la asignatura de estadística y probabilidad para comprender los aspectos que subyacen por debajo de la mayoría de las técnicas de aprendizaje automático.

Cabe destacar que durante el desarrollo de este proyecto se han desarrollado numerosas competencias transversales como ”aprendizaje permanente”, ”diseño y proyecto” , ”planificación y gestión del tiempo” o ”comunicación efectiva” entre otras.

Bibliografía

[1] Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimoni. Gender, genre, and writing style in formal written texts. Text-The Hague Then Amsterdam Then Berlin- 2003, vol. 23, no 3, p. 321-346.

[2] Dasha Bogdanova, Paolo Rosso, Thamar Solorio On the Impact of Sentiment and Emotion Based Features in Detecting Online Sexual Predators. En Proceedings of the 3rd workshop in computational approaches to subjectivity and sentiment analysis. Association for Computational Linguistics, 2012. p. 110-118.

[3] Paul Viola, Michael J. Jones Rapid Object Detection using a Boosted Cascade of Simple Features. CVPR (1), 2001, vol. 1, p. 511-518.

[4] Karen Simonyan, Andrew Zisserman Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556, 2014.

[5] Saman Daneshvar, Diana Inkpen Gender Identification in Twitter using N-grams and LSA Notebook for PAN at CLEF 2018

[6] Corinna Cortes, Vladimir Vapnik Support-vector networks, Machine learning, 1995, vol. 20, no 3, p. 273-297.

[7] Takumi Takahashi, Takuji Tahara, Koki Nagatani, Yasuhide Miura, Tomoki Taniguchi, and Tomoko Ohkuma Text and Image Synergy with Feature Cross Technique for Gender Identification. Working Notes Papers of the CLEF, 2018.

[8] Francisco Rangel, Paolo Rossol, J Manuel Montes-y-Gómez, Martin Potthast, Benno Stein . Overview of the 6th Author Profiling Task at PAN 2018: Multimodal Gender Identification in Twitter.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean . Distributed Representations of Words and Phrasesand their Compositionality. En Advances in neural information processing systems. 2013. p. 3111-3119.

[10] David H. Wolpert STACKED GENERALIZATION Neural networks, 1992, vol. 5, no 2, p. 241-259.

[11] José Hernández Orallo, MªJosé Ramírez Quintana, Cèsar Ferri Ramírez *Introducción a la Minería de Datos*. Pearson Prentice Hall

[12] Scikit learn Documentación Consultado en https://scikit-learn.org/.

[13] NLTK Consultado en https://www.nltk.org/.

[14] Tensorflow Consultado en https://www.tensorflow.org/.

[15] McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model. Consultado en http://www.mccormickml.com.

[16] OpenCV Consultado en https://opencv.org/.

[17] Caffe Consultado en http://caffe.berkeleyvision.org/.

[18] Gensim Consultado en https://radimrehurek.com/gensim/index.html.

[19] Clasificador en cascada para detección de objetos Consultado en https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html.

[20] Modelos HaarCascade Consultado en https://github.com/opencv/opencv/tree/master/data/haarcascades.

[21] Modelo de una neurona propuesto por McCulloch-Pitts Consultado en https://es.wikipedia.org/wiki/Neurona_de_McCulloch-Pitts.

[22] Redes neuronales convolucionales Consultado en http://www.aprendemachinelearning.com/como-funcionan-las.-convolutional-neural-networks-vision-por-ordenador/.

[23] Autoritas Consulting http://www.autoritas.net/.

[24] Electron Consultado en https://electronjs.org/.

[25] Lucene Consultado en https://lucene.apache.org/core/.

[26] Luke Consultado en http://www.getopt.org/luke/.

[27] Instalación de OPENCV Consultado en https://opencv-java-tutorials.readthedocs.io/en/latest/01-installing-opencv-for-java.html.

APÉNDICE A
Instalación de OpenCV

Se ha realizado una instalación de OpenCV para JAVA siguiendo el tutorial proporcionado en [27].

A.1 Instalación para MacOS

Para realizar la instalación en MacOS es necesario tener instalado HomeBrew [1] y la línea de órdenes de XCode.

Una vez se satisfacen los requisitos, es necesario instalar Apache Ant mediante Brew; para ello se debe ejecutar la siguiente línea de código.

```
$ brew install ant
```

Una vez instalado ANT se modificará el fichero OpenCV utilizando la siguiente orden:

```
$ brew edit opencv
```

y se modificará la siguiente linea:

```
-DBUILD_opencv_java=OFF
```

por la siguiente:

```
-DBUILD_opencv_java=ON
```

Una vez guardado el fichero se ejecutará la siguiente orden que instalará OPENCV y montará las bibliotecas necesarias para poder ejecutarlo desde JAVA

```
$ brew install --build-from-source opencv
```

Si todo se ha ejecutado correctamente, en la carpeta `\usr\local\Cellar\opencv\3.x.x\share\OpenCV\java\` se dispone de los ficheros necesarios para utilizar OpenCV desde JAVA.

A.2 Instalación para Windows

Para realizar la instalación en Windows, en primer lugar es necesario descargar un ejecutable mediante la https://opencv.org/releases/. Tras realizar la instalación del ejecutable serán necesarios dos ficheros:

[1] https://brew.sh/index_es

- El fichero `opencv-3xx.jar` localizado en la carpeta `\opencv\build\java`
- El fichero `opencv_java3xx.dll` localizado en la carpeta `\opencv\build\java\x86` o `\opencv\build\java\x64` dependiendo de la arquitectura del sistema