UNIVERSIDAD AUTONOMA DE MADRID

Escuela Politécnica Superior

18|19

Trabajo fin de grado

Estudio y aplicación de redes neuronales a predicción de hashtags en dominios cruzados

Ricardo Sánchez-Guzmán Hitti

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

Campus Internacional excelencia UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería informática

TRABAJO FIN DE GRADO

Estudio y aplicación de redes neuronales a predicción de hashtags en dominios cruzados

Autor: Ricardo Sánchez-Guzmán Hitti
Tutor: Alejandro Bellogín Kouki
Ponente: Iván Cantador Gutiérrez

junio 2019

Ricardo Sánchez-Guzmán Hitti
Estudio y aplicación de redes neuronales a predicción de hashtags en dominios cruzados

Ricardo Sánchez-Guzmán Hitti
C\ Francisco Tomás y Valiente Nº 11

IMPRESO EN ESPAÑA – PRINTED IN SPAIN

Agradecimientos

Me gustaría dar las gracias a Alejandro Bellogín, porque desde el primer momento que hablé con él se interesó por ayudarme y no ha parado hasta ver finalizado este trabajo, siempre con ganas de ayudar y dando un apoyo que siempre le agradeceré.

A mi familia que desde que empezó este camino siempre me ha apoyado y sin ellos nunca habría llegado tan lejos.

Resumen

Las redes sociales son a día de hoy usadas por gran cantidad de personas debido a la facilidad con la que permiten conocer gente o para enterarse de las últimas noticias. En estos tiempos, sobre todo entre la gente joven, es poco habitual ver a gente sin cuenta en dominios como Instagram o Twitter. Además, esta también es una oportunidad para las grandes compañías que quieren aprovechar la información que proporcionan estos dominios para hacer recomendaciones más personalizadas a sus nuevos usuarios.

Este Trabajo de Fin de Grado consiste en el estudio y aplicación de algoritmos de aprendizaje automático como redes neuronales y vecinos próximos para la predicción de hashtags en dominios cruzados y su posterior evaluación. Se utilizarán algoritmos de aprendizaje supervisado, empleándose los datos de Instagram para la fase de entrenamiento y los datos de Twitter para la de test. Esto nos permitirá conocer en qué medida el vocabulario activo de un usuario en un dominio es similar al vocabulario utilizado en otro distinto.

Por otro lado, hemos estudiado el subtitulado de imágenes, las técnicas más extendidas a día de hoy y las métricas más utilizadas en este ámbito. Hemos generado varios tipos de algoritmos KNN para ver su potencial en sus distintas versiones y hemos utilizado una librería de redes neuronales para ver su potencial frente a estos algoritmos. Para conseguir los usuarios de test de Twitter se han identificado en primer lugar los usuarios de Instagram y después se ha comprobado si el nombre del usuario era el mismo para descargar toda esta información, se ha realizado un preprocesado de los datos y, por último, se han formado los distintos ficheros de test para su evaluación.

Como conclusión, hemos observado que la red neuronal utilizada en sus distintas versiones es superior a los algoritmos KNN, aunque en algunos casos no con mucha diferencia. Aquellos usuarios que utilizábamos en los datos de entrenamiento y que luego aparecían en los de test aún siendo dominios distintos daban muy buenos resultados en la predicción de hashtags, esto nos indica que existe una alta correlación en el vocabulario de un usuario en ambos dominios ya que si está condición no se cumplía los resultados disminuían en un 45 %. Por último, señalar que esta tendencia se cumple e incluso mejora con usuarios que poseen mayor cantidad de imágenes, siempre y cuando se cumpla que el usuario aparezca en la fase de entrenamiento.

Palabras clave

Dominio, Red neuronal, Hashtag, Subtitulado de imágenes, Vecinos más cercanos, Twitter, Instagram.

Abstract

Social networks are currently used by large numbers of people because of the ease with which they can meet people there or to find the latest news. In these times, especially among young people, it is rare to see someone without an account in domains like Instagram or Twitter. In addition, this is also an opportunity for large companies that want to take advantage of the information provided by these domains to make more personalized recommendations to their new users.

This Bachelor's Thesis consists in the study and application of automatic learning algorithms such as neural networks and nearest neighbours for the prediction of hashtags in crossed domains including their corresponding evaluation. Supervised learning will be used and Instagram data will be used for the training phase and Twitter data for the test. This will allow us to know to what extent the active vocabulary of a user in a domain is similar to the vocabulary used in a different one.

Additionally, we have studied the problem of image captioning, the most widespread techniques to date and the most used metrics in this area. We have generated several types of KNN algorithms to see their potential in their different versions and we have used a library of neural networks to analyse their efficiency against these algorithms. To obtain Twitter test users, Instagram users have been identified first and then it has been verified if the username was the same to download all their information, a preprocessing of the data has been done and, finally, the different test files for evaluation have been formed.

As a conclusion, we have found that the neural network used in its different versions is superior to the KNN algorithms, although in some cases with not much difference. Those users that we used in the training data and that later appeared in the test (even being different domains) gave very good results in the prediction of hashtags, this indicates that there is a high correlation in the vocabulary of a user in both domains since if this condition was not met, the results decreased by 45 %. Finally, we want to note that this trend is met (and is even improved) with users who have more images, provided that the user appears in the training phase.

Keywords

Domain, Neural network, Hashtag, Image captioning, Nearest neighbors, Twitter, Instagram.

Índice

Listas

Lista de ecuaciones

Lista de figuras

Lista de tablas

1

Introducción

En este primer capítulo se describe la motivación por la cual se realiza este proyecto, los objetivos del mismo y la estructura del documento final.

1.1. Motivación del proyecto

Durante los últimos años el auge de las redes sociales ha permitido que cientos de personas se conozcan y puedan compartir sus gustos y aficiones. Este crecimiento de las redes sociales ha hecho que se conviertan en grandes almacenes de información, en especial, dos redes sociales bastante extendidas hasta el momento y que en los últimos años han empezado a crecer con bastante fuerza, son: Instagram y Twitter. Estas dos redes sociales se han convertido en una oportunidad para el ámbito de la minería web, suponiendo nuevos retos para la extracción de información y la combinación de ambas en un reto aún mayor.

Un desafío habitual en el ámbito de los sistemas de recomendación es el de la falta de información a la hora de recomendar un ítem a un usuario del que no tenemos ninguna información. Este problema se denomina arranque en frío *(cold start)* [1] y es algo que a día de hoy sigue siendo un problema en las grandes compañías como Netflix, Spotify o YouTube.

El uso de las huellas digitales [2] que podría tener un usuario en Internet permitiría mitigar este problema, un ejemplo muy claro de estas huellas digitales las podríamos encontrar en las redes sociales. Mediante los perfiles de los usuarios podemos tener una información base con la que recomendar productos, debido a que la gran parte de estos perfiles son públicos y permiten acceder a esta información.

En concreto es muy habitual ver que un usuario tenga cuenta en dominios como Instagram y Twitter al mismo tiempo, en la figura 1.1 se puede apreciar este hecho, y por eso hemos aprovechado esta propiedad para cruzar información de ambos dominios.

En este proyecto se usa un elemento común existente tanto en Instagram como en Twitter, y son las imágenes, las cuales usaremos para la predicción de hashtags mediante el uso de las redes neuronales artificiales. También hemos estudiado el subtitulado de imágenes con redes neuronales y las métricas utilizadas en la evaluación de estos algoritmos.

Figura 1.1: Uso simultáneo de las redes sociales por parte de los usuarios [3].

1.2. Objetivos y enfoque

El objetivo de este Trabajo Fin De Grado es el de predecir hashtags mediante el cruce de imágenes de Twitter e Instagram, para esta finalidad usaremos una red neuronal que entrena utilizando las imágenes de Instagram y el vocabulario activo de los usuarios en esta red social y usando como Test las imágenes contenidas en los tweets de los perfiles de esos mismos usuarios en Twitter.

Estos resultados serán evaluados y analizados para ver la relevancia de los componentes de dicha red neuronal. También se evaluará la eficacia de los métodos según los tipos de usuario, es decir, sus niveles de acierto en usuarios con muchas o pocas imágenes, así como con una muestra aleatoria de los usuarios.

1.3. Estructura del documento

El presente documento está dividido en 5 capítulos, los cuales se detallan a continuación:

Capítulos

Capítulo 1. Introducción: Breve descripción de la motivación del proyecto, los objetivos a cumplir y la estructura del documento.

Capítulo 2. Estado del Arte: Se hace un repaso de los aspectos más importantes en este área en los últimos años. Centrándonos en el uso de las redes neuronales, una revisión del subtitulado de imágenes y las métricas de evaluación utilizadas.

Capítulo 3. Diseño e Implementación: En este capítulo se detalla la obtención de los datos de los usuarios, el procesado de estos, las librerías utilizadas y el resto de herramientas implicadas en el proceso.

Capítulo 4. Pruebas y Evaluación: Se analizan y comparan los resultado obtenidos con distintos conjuntos de datos para obtener conclusiones relevantes.

Capítulo 5. Conclusiones y Trabajo Futuro: Se exponen las conclusiones obtenidas y el potencial del proyecto en futuros ámbitos de investigación.

2

Estado del arte

En este capítulo se abordan las redes neuronales más importantes a la hora de trabajar con el subtitulado de imágenes describiendo brevemente cada una de ellas, un repaso de las ultimas metodologías utilizadas para la elaboración de subtítulos y las métricas para su evaluación.

2.1. Redes Neuronales

En este subapartado se explican tres de las redes neuronales más importantes a la hora de generar subtítulos a partir de una imagen.

Cuando hablamos de una red neuronal nos referimos a un sistema de procesamiento bioinspirado en el que el procesamiento de la información tiene lugar en elementos llamados neuronas.

Las neuronas están conectadas entre si y trasmiten información entre ellas, estás conexiones entre neuronas tienen ponderaciones y además cada neurona tiene una función de activación, que en caso de que se supere un cierto umbral disparara la salida de la neurona o la inhibirá.

2.1.1. CNN

Las redes CNN (del inglés, Convolutional Neural Network) [4] han ganado fuerza en la última década y se han utilizado sobretodo en el ámbito del procesamiento de imágenes, ya que son capaces de analizar características muy complejas con gran precisión.

La red Convolucional esta compuesta por tres tipos de capas: Capa Convolucional, Capa de Pooling y la Capa Clasificadora.

La Capa Convolucional es la encargada de realizar la operación de convolución, su funcionamiento es el siguiente, la red recibe como patrón de entrada una imagen, sobre esa imagen se aplica un filtro o kernel que nos devuelve las características de la imagen, un filtro muy común suele ser el de detectar los bordes de una imagen.

La Capa de Pooling es la encargada de reducir las dimensiones de los valores de entrada

para la siguiente capa convolucional, se suele aplicar una operación bastante común llamada maxpooling, que se encarga de dividir la imagen en subregiones y se queda con la característica más representativa de la región.

La Capa Clasificadora donde cada píxel obtenido se considera una neurona, esta capa tendrá tantas neuronas como clases queramos clasificar (softmax), y será la encargada de darnos la predicción final.

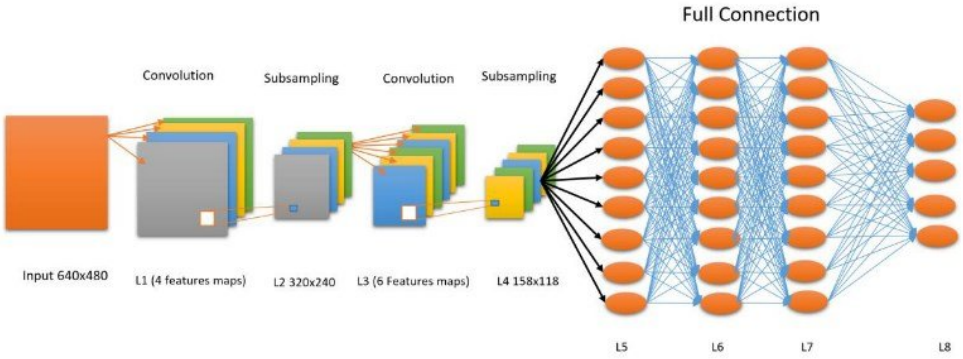

Figura 2.1: Arquitectura de una red neuronal convolucional [5].

2.1.2. RNN

Cuando hablamos de las RNN (del inglés, Recurrent Neural Network) [6] nos referimos a aquellas redes que son capaces de guardar el estado interno y combinarlo con los valores de entrada de la red para producir una salida, es decir, son capaces de tener memoria, para lograr esto se utiliza la retroalimentación. Una misma neurona podría estar conectada a las neuronas de la capa posterior, anterior e incluso a ella misma. Por este motivo el uso de las redes neuronales recurrentes las hace idóneas para la predicción de secuencias, por ejemplo: para la predicción de cadenas de texto o de sonido. En la figura 2.2 se puede apreciar la arquitectura de la red.

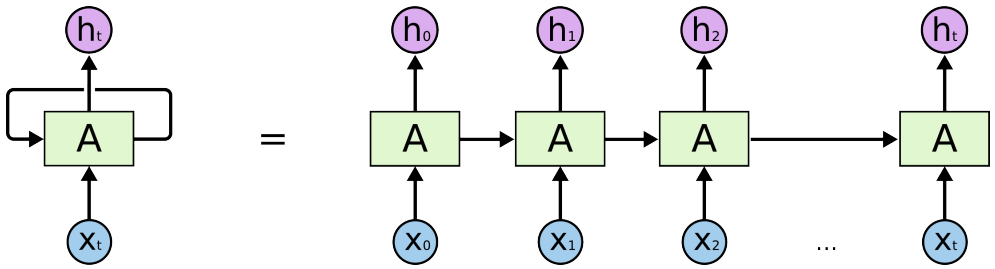

Figura 2.2: Arquitectura de una red neuronal recurrente [7].

Una de las diferencias más significativas de las RNN en comparación a otras redes, es que las capas que forman la red a la hora de predecir la siguiente palabra de una frase comparten los mismos

parámetros de entrada, esto permite utilizar menos parámetros a la hora de trabajar con este tipo de redes.

2.1.3. LSTM

Existe un gran problema a la hora de utilizar las redes neuronales recurrentes, este problema reside en la cantidad de información que puede "guardar" la red, las RNN son capaces de guardar pequeñas cantidades de información mediante bucles pero en el caso de necesitar guardar datos que tengan una dependencia temporal muy extensa la red comenzaría a fallar en sus predicciones.

Por ejemplo, imaginemos que queremos predecir un punto en una serie temporal caótica, la red utiliza como valores de entrenamiento los últimos ocho puntos de la serie para predecir el noveno, pero ¿y si necesitáramos los últimos cien puntos para predecir el punto ciento uno?, la red comenzaría a fallar, esto es debido a que podemos considerar cada intervalo de tiempo como una capa oculta conectada con el resto, la información pasa de capa en capa para después propagar el error hacia atrás, el problema surge en que si tenemos muchos intervalos de tiempo se produce el problema del desvanecimiento del gradiente.

La red utiliza el descenso por gradiente para entrenar la red, basándose en los errores actuales y pasados para calcular los pesos de las sinapsis, esto hace que el entrenamiento sea muy costoso y los cambios en los pesos no lleguen a producirse por perdida de información (desvanecimiento del gradiente), esto fue un problema hasta que en el año 1997 es propuesta una nueva variante de la red RNN llamada LSTM (del inglés, LONG Short-Term Memory) [8] por Sepp Hochreiter y Jürgen Schmidhuber.

Las LSTM resuelven el problema de las RNN ya que consiguen tener una memoria a más largo plazo y esto permite almacenar, por ejemplo, el contexto de un usuario, algo fundamental en la personalización.

La red LSTM esta formada por bloques de memoria o unidades LSTM, donde cada unidad está formada por tres puertas multiplicativas con retroalimentación: puerta de entrada, puerta de salida y una puerta de olvido, y además incluye una célula de memoria que será la encargada de guardar el valor.

La puerta de entrada se encarga de controlar en qué medida van apareciendo nuevos valores, actúa como filtro en el módulo de memoria, protegiéndola de valores irrelevantes.

La puerta de salida se encarga de regular el peso del valor contenido en la célula de memoria a la hora de calcular la salida de ese bloque, con esto conseguimos evitar la generación de información irrelevante por parte de nuestra unidad LSTM.

La puerta de olvido es la encargada de mantener el valor en la célula de memoria en

cada tiempo, su función sería la de elegir qué valores y por cuánto tiempo permanecen en el módulo de memoria.

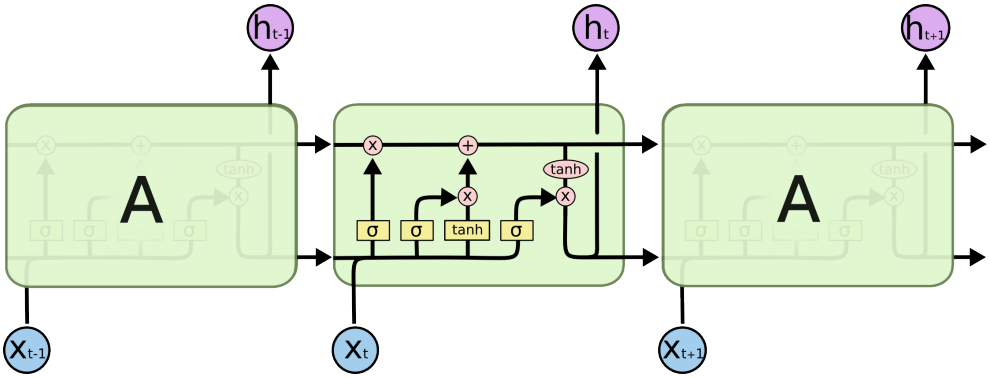

Figura 2.3: Arquitectura de una red Long-Short Term Memory [7].

2.2. Subtitulado de imágenes

El subtitulado de imágenes ha sido siempre un área que se ha resistido en el ámbito del aprendizaje automático, mientras que otras áreas como la clasificación de imágenes o el reconocimiento de voz prácticamente ya están muy avanzadas, el subtitulado de imágenes aún no ha conseguido alcanzar esta progresión pero esta tendencia esta cambiando debido a los nuevos métodos que están empezando a surgir en los últimos años.

Existen tres tipos de metodologías a la hora de abordar un problema de subtitulado de imágenes:

Método I: se realiza el análisis de la imagen y **se detectan los objetos o atributos más relevantes de la imagen**, a partir de ese reconocimiento de objetos se elabora la descripción de la imagen mediante las palabras que hacen referencia a esos objetos.

Método II: en este método tenemos una descripción que está hecha por una persona que describe la imagen, se usan las características de la imagen y se hace una descripción en base a la **combinación de las descripciones de las imágenes más semejantes** a la imagen pasada como parámetro.

Método III: el **método de secuencias de palabras** es el más novedoso y que últimamente se ha convertido en el estándar para los algoritmos de generación de subtitulado, este método utiliza las palabras más relevantes para una imagen y todas las palabras generadas previamente, es decir, cuando se genera una nueva descripción se utilizan todas las palabras generadas previamente para formular la siguiente palabra, esto permite usar combinaciones de palabras que nunca se podrían haber generado en los datos de entrenamiento, las palabras que normalmente utiliza son las generadas en un vocabulario el cual se crea a partir de las descripciones que las imágenes tienen hechas por los usuarios.

Vamos a enfocarnos en este tercer método al ser el más relevante de los tres.

En primer lugar necesitamos un modelo que sea capaz de recordar las palabras previas generadas y predecir nuevas en un orden correcto, esto lo podemos conseguir con las redes neuronales recurrentes **RNN o LSTM**, estas redes permiten almacenar secuencias cortas o largas de palabras y generar nuevas predicciones como se ha explicado anteriormente.

En la fase de entrenamiento utilizamos miles de imágenes acompañadas de un título que las caracterice, una vez tengamos las características de las imágenes buscamos optimizar los parámetros que lleven a la descripción objetivo.

Para hacer la predicción en un tiempo t de una palabra se utiliza un vector oculto de longitud fija y una capa que tiene como salidas el número de palabras del vocabulario, usando una función Softmax sacamos la palabra con mayor probabilidad y la añadimos a la siguiente predicción en t + 1.

Algo que conviene mencionar es que está empezando a surgir en los últimos años un nuevo tipo de planteamiento para eliminar los elementos irrelevantes a la hora de subtitular imágenes, son los llamados **Mecanismos de atención**.

Estos mecanismos de atención permite que los elementos más importantes de una imagen tengan más prioridad que otros a la hora de describir una imagen, es decir, a la hora de entrenar nuestra red RNN o LSTM le pasaremos las características de la imagen que tengan más peso y no la imagen completa. Esto podría generar palabras importantes para zonas especificas de la imagen pero no para la descripción general de toda la imagen, por eso es importante que el vocabulario en el que nos basemos sea el de las palabras generadas en regiones previamente exploradas de la imagen. En la figura 2.4 podemos ver un ejemplo, donde por ejemplo podemos dar una prioridad a zonas de la imagen y dar más peso a esas palabras generadas entre otras muchas posibilidades.

Figura 2.4: En la esquina superior izquierda de cada imagen se puede observar la palabra que se genera según a qué parte de la imagen se le esté dando más prioridad [9].

2.3. Métricas de Evaluación

En este subapartado vamos hablar de las métricas utilizadas para la evaluación de los hashtags y subtítulos.

2.3.1. Métricas para la evaluación de hashtags

Precisión: La precisión indica el ratio entre el número de elementos relevantes devueltos y el número de elementos totales obtenidos (Ver ecuación 2.1). La precisión final será la media de todos los patrones de entrada.

$$Precision = \frac{\#TP}{\#TP + \#FP} \tag{2.1}$$

Recall: El recall indica el ratio entre el número de elementos relevantes devueltos y el número de elementos relevantes totales (Ver ecuación 2.2). El recall final será la media de todos los patrones de entrada.

$$Recall = \frac{\#TP}{\#TP + \#FN} \tag{2.2}$$

F1-score o media armónica: Está medida se suele utilizar en el área de la recuperación de información, permite combinar la Precisión y el Recall y su valor esta acotado entre 0 y

1, siendo 0 el peor resultado y 1 el mejor.

$$F1 = 2 * \frac{1}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{2.3}$$

Recomendado	No recomendado
Positivo Verdadero (TP)	Falso Negativo (FN)
Falso Postivo (FP)	Negativo Verdadero (TN)

Tabla 2.1: Notación utilizada en las métricas de Precisión y Recall.

2.3.2. Métricas para la evaluación de subtítulos

BLEUr-n [10] es uno de los primeros métodos que surgió para la traducción automática, mide el nivel de precisión de los n-gramas entre dos oraciones, donde un n-grama es la subsecuencia de palabras de una oración con respecto a la secuencia completa, por ejemplo, si usáramos bigramas serían conjuntos de dos palabras. Esta métrica compara las oraciones y penaliza la brevedad en las descripciones de referencia, por ejemplo, si tenemos una descripción de referencia de tres palabras y nuestra descripción obtenida tiene nueve palabras seguramente acertemos todas las palabras, pero las frases no serán similares. BLEUr acepta más de una oración para comparar con la oración de referencia y así poder tener un resultado más robusto.

$$p_n = \frac{\sum_{c \epsilon Candidatos} \sum_{n-gramas \epsilon c} Count(n-gramas)}{\sum_{c' \epsilon Candidatos} \sum_{n-gramas \epsilon c'} Count(n-gramas')} \tag{2.4a}$$

El primer paso es calcular la precisión de los n-gramas, BLEUr utiliza una versión distinta de la precisión original, esto es debido a que es posible que en la descripción devuelta para una oración de referencia aparezca una palabra muchas veces que se encuentra en la oración de referencia, con lo cual nuestra precisión seria alta pero la descripción no tendría ningún sentido. En un primer paso se calculan las coincidencias de cada oración con la oración de referencia y se contabilizan, si la ocurrencia de una palabra aparece más de una vez solo se queda con el número de veces que aparezca esa palabra en la oración de referencia y se divide por el número total de las palabras candidatas.

$$BP = \begin{cases} \text{si } c > r & 1 \\ \text{si } c \leq r & \exp^{1-r/c} \end{cases} \tag{2.4b}$$

BLEUr aplica una penalización en caso de que la longitud de la oración generada sea mayor que la de la original, donde c es la longitud de la oración generada y r la longitud de la oración de referencia.

$$BLEUr = BP * \exp\left(\sum_{n=1}^{N} w_n log p_n\right) \tag{2.4c}$$

Para el calculo final BLEUr aplica la media geometrica para los n-gramas utilizados, donde W_n es el peso que se le da a cada n-grama, generalmente se suele dar el mismo peso a todos los n-grama $W_n = 1/N$.

CIDEr [11] para el cálculo de esta métrica aplica una pequeña derivación inicial, donde cada frase se representa en un conjunto de 1 a 4 n-gramas. Después se calculan las similitudes entre la descripción de referencia y las obtenidas, usando TF-IDF para que las palabras que sean muy comunes tengan una ponderación baja, por último, se calcula la similitud coseno entre los n-gramas obtenidos y los de referencia.

$$g_k(s_{ij}) = \frac{h_k(s_{ij})}{\sum_{w \epsilon \beta^{h_l(s_{ij})}}} \log\left(\frac{I}{\sum_{I_p \epsilon I} min(1, \sum_q h_k(S_{pq}))}\right) \tag{2.5a}$$

Está métrica evalúa la descripción generada a partir de una imagen con respecto a un conjunto de descripciones de referencia, las oraciones se representan utilizando n-gramas, donde un n-grama esta denotado con w_k e indica un conjunto de palabras ordenadas.
Para asignar ponderaciones a los n-gramas se utiliza TF-IDF, donde el número de veces que aparece un n-grama en una oración de referencia es $h_k(s_{ij})$ o $h_k(c_i)$ si es para una oración candidata.
Para calcular el score TF-IDF $g_k(s_{ij})$, de cada n-grama utilizado w_k, se utiliza la fórmula 2.5a, donde ϵ es el vocabulario utilizado por todos los n-gramas e I es el conjunto de descripciones de imágenes en el dataset. El primer componente mide el TF de cada n-grama y el segundo el IDF. El IDF se calcula haciendo el logaritmo entre el número de imágenes |I|, dividido por el número de imágenes en las que aparece w_k en las oraciones de referencia.

$$CIDEr_s(c_i, S_i) = \frac{1}{m} \sum_j \frac{g^n(c_i) * g^n(s_{ij})}{\parallel g^n(c_i) \parallel * \parallel g^n(s_{ij}) \parallel} \tag{2.5b}$$

Donde c_i es la oración candidata y s_i las oraciones de referencia, $g^n(c_i)$ es un vector TF-IDF formado por $g^k(c_i)$ que corresponde a todos los n-gramas de longitud n y $\parallel g^n(c_i) \parallel$ que es el modulo de ese vector, se aplica el mismo principio a $g^n(s_{ij})$.

$$CIDEr_s(c_i, S_i) = \sum_{n=1}^{N} w_n CIDEr_n(c_i, S_i) \tag{2.5c}$$

Como en la métrica BLEUr el peso que se le da a cada n-grama es uniforme $w_n = \frac{1}{N}$.

ROGUE-L [12] está métrica es una variante de ROUGE y mide la secuencia de palabras que sea más larga y que concuerde mejor entre dos oraciones. No requiere que las coinci-

dencias sean consecutivas solo que la secuencia de las palabras estén en el orden correcto a nivel de oración.

$$R_{lcs} = \frac{LCS(X,Y)}{m} \tag{2.6a}$$

$$P_{lcs} = \frac{LCS(X,Y)}{n} \tag{2.6b}$$

Donde X es la oracion de referencia e Y es la oracion candidata, m sería la longitud de X y n la longitud de Y, LCS (X,Y) es la longitud de la subsecuencia de palabras más larga de X en Y, y $\beta = P_{lcs}/R_{lcs}$.

$$F_{lcs} = \frac{(1+\beta^2)R_{lcs}P_{lcs}}{R_{lcs} + \beta^2 P_{lcs}} \tag{2.6c}$$

METEOR [13] surge para suplir algunas deficiencias de métricas anteriores como BLEUr. Difiere de BLEUr en varios puntos, mientras que BLEUr trabaja con n-gramas, METEOR lo hace con unigramas, BLEUr utiliza la precisión en varios n-gramas para después utilizar la penalización por longitud para ajustar la puntuación final, en cambio, METEOR utiliza una función F1 parametrizada, donde todos los parámetros se ajustan para que exista una mayor correlación.

$$Fmean = \frac{10PR}{R + 9P} \tag{2.7a}$$

Se utiliza la media armónica para combinar la precisión y el recall, solo que en este caso el recall es ponderado nueve veces más que la precisión.

Meteor también aplica penalizaciones a las oraciones como hacia BLEUr, en un primer paso se agrupan los unigramas en conjuntos lo más pequeños posibles, donde un conjunto sería los unigramas que están de forma consecutiva con respecto a la descripción de referencia (muy similar a una búsqueda proximal, donde se hacen intervalos mínimos que contengan todas las palabras de una búsqueda), por ejemplo, una descripción que fuera idéntica totalmente generaría un solo conjunto puesto que todas las palabras estarían de forma consecutiva y corresponden con la descripción de referencia.

$$Penalty = 0{,}5 * \left(\frac{\#chunks}{\#unigrams\_matched}\right)^3 \tag{2.7b}$$

Donde #chunks es el número de conjuntos, y #unigrams_matched son los unigramas utilizados.

Por último se calcularía el Score final en la fórmula 2.7c.

$$Score = Fmean * (1 - Penalty) \tag{2.7c}$$

3

Diseño e Implementación

En este capítulo se explica el desarrollo técnico del proyecto, describiendo el proceso de extracción de datos, su procesado y almacenamiento de estos. Después se hará una descripción de la red neuronal utilizada en el proceso de predicción de subtítulos y hashtags, y una explicación detallada del flujo del proyecto.

3.1. Diseño

En este apartado se detallan los recursos utilizados en la elaboración del proyecto, las librerías externas utilizadas, el proceso seguido para la extracción, preprocesado y almacenamiento de los datos, y, por último, una breve descripción de la red neuronal utilizada. En la siguiente figura 3.1 se puede apreciar el esquema del proyecto.

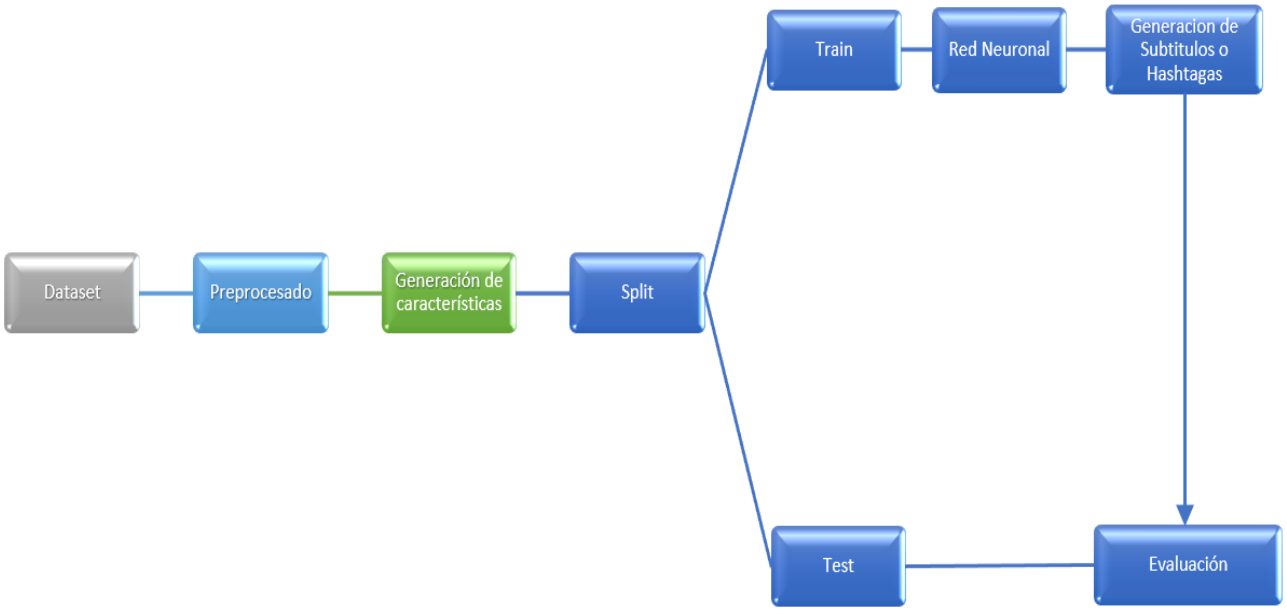

Figura 3.1: Diagrama de flujo.

En un primer momento partimos de un dataset formado por posts de usuarios de Instagram y tweets de usuarios de Twitter, se da por hecho que un mismo nombre de usuario (o username) en ambos dominios corresponde a un mismo usuario para facilitar el desarrollo de este proyecto. Se analizaron varios usuarios aleatorios con tal de comprobar si la temática que utilizaba un usuario en un dominio era la misma que utilizaba en el otro, en todos los casos se cumplió este hecho, por ejemplo, si un usuario mayoritariamente subía fotos de comidas o moda en su perfil de Instagram, en su perfil de Twitter la información mayoritaria también era sobre esos temas en forma de hashtags o imágenes.

Una vez obtenidos los datos de Twitter en crudo pasamos a un preprocesamiento de los mismos, donde se llevará a cabo un normalizado de las cadenas de texto y una extracción de las características de las imágenes relacionadas con los posts y tweets.

Con el dataset generado se hace un particionado de los datos, formando un conjunto de datos para entrenamiento (training) y otro para test. Los datos de training permiten entrenar la red neuronal para posteriormente evaluar los datos de test y mostrar tablas comparativas con los resultados obtenidos, en nuestro caso los datos de training serán las imágenes de los posts de Instagram y los datos de test las imágenes de los tweets de Twitter.

3.1.1. Modificaciones y descripción del entorno de trabajo

Los recursos utilizados en este proyecto son los que aparecen en la tabla 3.1, un aspecto importante a la hora de realizar predicción de subtítulos con redes neuronales es la de contar con un equipo de altas prestaciones sobretodo en la parte de GPUs puesto que es lo que utiliza la red principalmente para entrenar.

Recursos	Características
CPU	i7-8550U
GPU	GeForce 1070 ti 8GB GDRR5
S.O	Kubuntu 18.04
Versión python	2.7
RAM	32GB

Tabla 3.1: Equipo utilizado para de entranimiento y pruebas.

En nuestro caso no contamos con un ordenador de prestaciones tan altas, por este motivo hemos tenido que cambiar la configuración de la librería de attend2u (ver más en Sección 3.2.2) para poder ejecutar la red neuronal sobre el equipo.

A la hora de entrenar una red neuronal se pueden utilizar dos recursos, la CPU o la GPU, está acreditado [14] que el uso de una CPU para el entrenamiento de una red neuronal es mucho más lento que el uso de la GPU, el uso de la librería tensorflow permite el entrenamiento con GPUs pero para

ello debemos tener instalado previamente las librerías de Cuda y Cudnn, tanta es la diferencia que si no hubiéramos tenido una GPU de gama media no podríamos llevar a cabo la ejecución de la red neuronal.

En la ejecución normal de la red se cuenta con cuatro GPUs pero en nuestro caso solo contamos con una y con la mitad de potencia que la usada en el paper de attend2u, por este motivo se llevan a cabo estas modificaciones.

Cambios realizados en la ejecución de **predicción de subtítulos**:

Ejecución normal

python m train num_gpus 4 batch size 200

tf.GPUOptions(per_process_gpu_memory_fraction=0.95)

Ejecución adaptada

python m train num_gpus 1 batch size 150

tf.GPUOptions(per_process_gpu_memory_fraction=0.85)

Cambios realizados en la ejecución de **predicción de hashtags**:

Ejecución normal

python m train num_gpus 4 batch size 200

tf.GPUOptions(per_process_gpu_memory_fraction=0.95)

Ejecución adaptada

python m train num_gpus 1 batch size 90

tf.GPUOptions(per_process_gpu_memory_fraction=0.85)

Donde per_process_gpu_memory_fraction indica la cantidad de memoria que se va a utilizar de la tarjeta grafica y batch_size indica el número de ejemplos de entrenamiento que se utilizará para el cálculo del error y la actualización de pesos.

Se requiere tener un espacio en disco para imágenes de Instagram y Twitter de al menos 40 GB y otros 20 GB para la generación de los vectores de características de las imágenes y el guardado de los ckpt (del inglés, Checkpoints) que vaya generando la red en la fase de entrenamiento.

Debido a la escasez de recursos como GPUs se ha disminuido el número de épocas de entrenamiento de la red, dando prioridad a la cantidad de datos en lugar de al número de épocas de entrenamiento, este cambio inevitablemente sí llevará a una pérdida de rendimiento de la red puesto que se ha intentado ajustar el entrenamiento de la red a 24 horas para llevar a cabo las pruebas y no estar

lastrado por el tiempo de entrenamiento de la red.

Como herramienta auxiliar se ha utilizado la plataforma **Google Cloud computing**. Es muy fácil de utilizar y permite utilizar recursos a gran escala como Tarjetas gráficas de gama alta o memoria RAM de más de 100 GB, permite crearnos un entorno de trabajo, seleccionar los recursos que nos interesan y crear un equipo potente. Los componentes básicos son totalmente accesibles por una cuenta básica, en cambio, para el uso de componentes de gama alta como tarjetas gráficas es necesario la solicitud de estos permisos, se cuenta con un saldo gratuito de 300 dólares con una duración máxima de 12 meses.

3.2. Librerías externas utilizadas

3.2.1. Tweepy

Para la extracción de información de Twitter se ha utilizado la librería Tweepy [15], esta librería proporciona acceso a todos los métodos de la REST APIs de Twitter, previamente para poder usar esta librería debemos tener una cuenta en Twitter y crear una aplicación, una vez tengamos creada la cuenta y registrada la aplicación se generarán una serie de parámetros como el token identificador que nos permitirá validarnos desde nuestra aplicación y extraer la información de Twitter. Existen dos tipos de cuentas en twitter: la estándar y la premium, en nuestro caso trabajaremos con la cuenta estándar que es gratuita pero que tiene algunas limitaciones, estas limitaciones no afectaran a los datos en sí, solo al número de tweets que podamos obtener y al tiempo de peticiones por minuto.

3.2.2. attend2u

Attend2u [16] es la librería principal sobre la que basaremos este proyecto, la hemos elegido por dos motivos fundamentales: el dataset de Instagram que posee y los buenos resultados que tiene en la predicción de subtítulos y de hashtags con redes neuronales. La red utiliza como query una imagen y el vocabulario activo del usuario, basándose en esto devuelve una descripción o una serie de hashtags que describen la imagen.

Se probaron otras librerías semejantes como Seq2seq [17], ShowTell [18] o AttendTell [19] pero fueron descartadas por motivos como el rendimiento que ofrecían o requisitos demasiado elevados para su ejecución.

Debido al reciente caso de Facebook, donde los datos de los usuarios fueron usados de forma inapropiada por parte de la compañía y la llegada de la nueva ley de protección de datos, Instagram restringió la funcionalidad de su API, este ha sido el motivo por el cual hemos preferido buscar un dataset ya creado en vez de crearlo nosotros, como ha sido en el caso de Twitter donde la API sigue

manteniendo toda su funcionalidad.

3.2.3. Resnet101

Para la extracción de características relevantes de las imágenes attend2u utiliza la librería Resnet101 [20], se trata de una red convolucional formada por 101 capas ocultas y que está entrenada con más de 1 millón de imágenes del dataset ImageNet 2012.

3.2.4. COCO

Para la evaluación de los subtítulos generados por la red neuronal se utilizan las métricas contenidas en la librería MS COCO [21].

3.3. Extracción, procesado y almacenamiento de los datos

3.3.1. Extracción

Para la extracción de información de Twitter se ha utilizado la librería Tweepy, esta librería nos permite descargar cualquier información accesible mediante el API de Twitter como: followers, following, hashtags y tweets.

Antes de extraer la información contenida en Twitter tenemos que saber qué usuarios nos interesan, puesto que queremos una correlación entre usuarios de Twitter e Instagram. El dataset de Instagram está accesible en la librería de attend2u y será el punto de partida. En un primer paso se extraen los identificadores de los usuarios que están contenidos en Instagram, obteniendo así un total de **6315 usuarios** únicos en Instagram de los cuales ya tenemos sus imágenes descargadas.

El siguiente paso es comprobar que ese username de usuario de Instagram exista en el dominio de Twitter, para eso hacemos una petición a la url de Twitter y en caso de que nos devuelva un código de OK o un código de ERROR lo contabilizaremos como un usuario válido o no válido, puede ser que el usuario exista pero no tenga un perfil público en Twitter este usuario también es descartado, una vez hechas estas peticiones el número total de usuarios con el mismo username que están en Instagram y qué con una cuenta pública en Twitter es de **3102 usuarios**.

Podemos ver que al menos la mitad de los usuarios de Instagram tienen perfil en Twitter, esto es un factor a tener en cuenta ya que nos indica que los usuarios suelen ser propensos a tener cuenta en ambos dominios, incluso a la hora de comprobar si la temática de los perfiles se correspondía con la de Instagram, se observaba cómo en muchos perfiles de Twitter se mostraba la url del perfil del usuario en Instagram.

Ahora es donde entra en juego la librería Tweepy, una vez tenemos una lista de usuarios válidos (sabemos que existe el username y que su perfil es público), tenemos que comprobar si hay suficiente información en estos perfiles para hacer pruebas, para esto se hace un filtrado de los usuarios que al menos tengan más de 5 tweets en las últimas semanas, puesto que la cuenta estándar no permite obtener tweets muy antiguos. El total de usuarios con este filtrado es de **1115 usuarios**.

Se procede a descargar la información de estos usuarios, para no repetir varias descargas (puesto que tenemos una cuenta estándar y esto podría ralentizar el proceso), descargamos toda la información relevante de un usuario aunque no vaya a ser utilizada después como: followers, following, hashtags y tweets, toda esta información quedará guardada en archivos JSON para su posterior procesado, donde cada archivo JSON contendrá la información de su respectivo usuario.

3.3.2. Procesado

Con toda la información descargada y guardada, se procede a formatearla y guardala de una forma que podamos trabajar con ella. Nos interesan las imágenes del perfil de un usuario en Twitter que tengan hashtags, es decir, necesitamos tweets que contengan una imagen y hashtags vinculados a esa imagen.

Se recorren los tweets de cada usuario y se comprueba si un tweet tiene una imagen y hashtags, como resultado obtenemos que el número de usuarios con imágenes y hashtags en Tweets es **844**. Eliminamos toda la información que no sea relevante mediante una expresión regular, por ejemplo la presencia de emoticonos entre otros caracteres. En las siguiente tablas 3.2 se pueden ver las estadísticas obtenidas de los usuarios finales.

Tipo	Nº usuarios
Imagenes $>$ 50	451
Followers $>$ 500	337
Following $>$ 500	321
Hashtags $>$ 50	747
Tweets $>$ 800	552

(a) Estadísticas de todos los atributos descargados.

hashtags/imagen	Nº de usuarios
h/i $>$ 1	844
h/i $>$ 10	337
h/i $>$ 50	293
h/i $>$ 1000	1

(b) Número de usuarios con mas de N tweets que contengan imágenes y hashtags al mismo tiempo.

Tabla 3.2: Estadísticas de los usuarios obtenidos de Twitter.

Twitter	tweets	usuarios	tweet/usuario	hashtag/tweet
hashtags	1,127,871	1115	1011.54	1.87

Tabla 3.3: Medias de los usuarios y hashtags de nuestro dataset.

3.3.3. Almacenado

El resultado final es guardado en un archivo JSON, donde están contenidos todos los usuarios, los parámetros almacenados son: el username del usuario, la url de la imagen y los hashtags vinculados a esa imagen. Todo esto nos permite realizar un aprendizaje supervisado.

Las imágenes de cada usuario son descargadas y renombradas con el id del tweet en el que estaban contenidas y el username del usuario para saber en todo momento a qué usuario pertenecen. Todas estas imágenes se guardan en un directorio para su posterior uso en la fase de entrenamiento y evaluación, con un total de **89936 imágenes**.

3.4. Funcionamiento de la Red CSMN

Para la generación de subtítulos o hashtags se ha utilizado una red neuronal con el nombre CSMN (del inglés, Context Sequence Memory network), este nombre es debido a que utiliza la información del contexto de un usuario como su vocabulario activo (palabras previas o hashtags), utiliza la secuencia de palabras que va generando previamente para generar las palabras futuras, y las características almacenadas de las imágenes para devolver un resultado basado en este contexto. En la figura 3.2 se puede ver la estructura de la red.

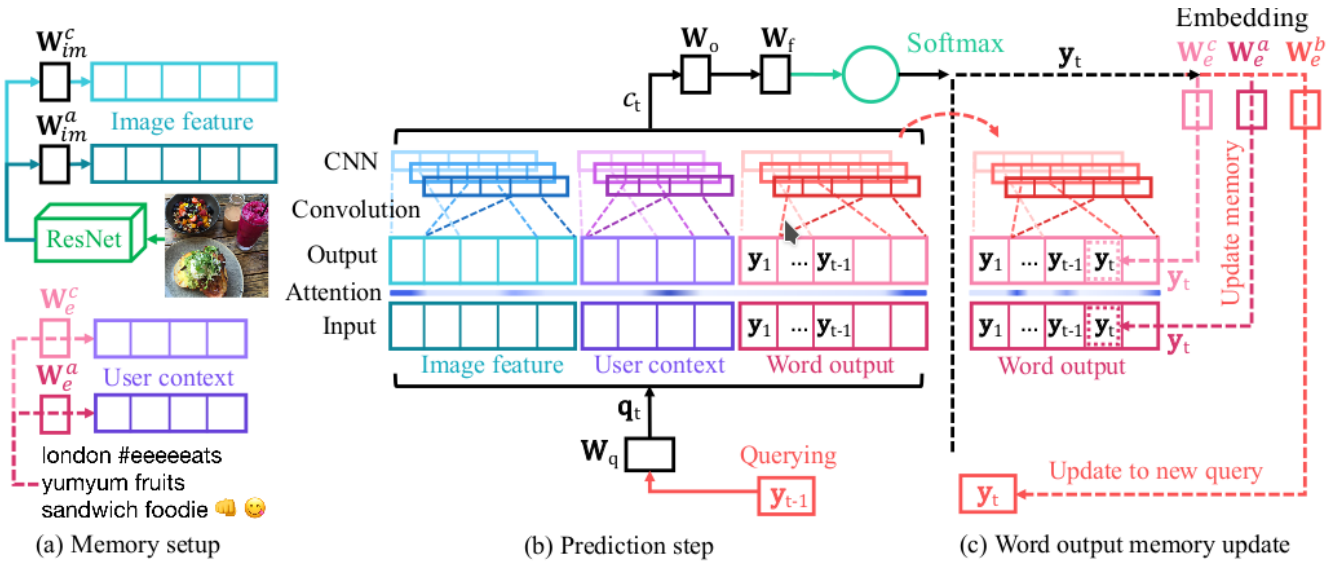

Figura 3.2: Arquitectura de una red CSMN [16].

La red utiliza una imagen como parámetro de entrada y un vocabulario activo, según la configuración que esté establecida devolverá hashtags o subtítulos para esa imagen. La red tiene 3 componentes principales: las características de las imágenes, el contexto del usuario y la retroalimentación de palabras generadas.

Las **características de las imágenes**, son vectores de características de las imágenes generados por la red Resnet101.

El **contexto del usuario**, donde guardamos las palabras que ha utilizado previamente el usuario en su perfil de Twitter o Instagram, a estas palabras se les asigna una ponderación TF-IDF, y se ordenan de manera decreciente según esta puntuación, al usar TF-IDF disminuimos la posibilidad de utilizar palabras que se usen con mucha frecuencia y permite un enfoque más personalizado al usuario puesto que tenemos los identificadores de los usuarios.

La **retroalimentación de las palabras generadas**, cuando la red genera una palabra para la descripción de una imagen esa palabra no se elimina, sino que se reutiliza para predecir las siguientes palabras, de esta forma se guarda un registro de la secuencia de palabras previas.

3.5. Implementación

En este apartado comentamos el flujo del proyecto desde la formación del dataset hasta la predicción final de los datos.

3.5.1. Generación del dataset

A la hora de ejecutar el proyecto debemos tener previamente almacenadas las imágenes con las que vamos a trabajar y los archivos JSON que vamos a utilizar, las imágenes deben estar guardadas en un directorio con el nombre images, donde el nombre de la imagen estará identificado de la siguiente forma **id-usuario-twitter_@_id-tweet_id-usuario-twitter**, esto permitiria tener identificada la imagen y al usuario al que pertenece en todo momento. Los archivos JSON se dividirían en dos uno para training y otro para test.

En primer lugar (ver la figura 3.3) tenemos una estructura donde el primer componente es el id del usuario en Instagram, donde están contenidos todos los posts de los usuarios están identificados por el id del post al que pertenezcan, dentro de cada post se guarda: el id del post, los hashtags utilizados en ese post, la oración que acompaña a ese post y finalmente la url para descargar la imagen.

En segundo lugar (ver la figura 3.4) tenemos el username que identifica al usuario, en segundo lugar el identificador de Tweet que nos valdrá como identificador de la imagen y, por último, los hashtags ligados a esa imagen.

Hay que recordar que los usuarios y Tweets seleccionados son aquellos en los que existe al menos una imagen y un hashtag para poder tener un aprendizaje supervisado. Una vez seleccionadas las imágenes de los usuarios de test las guardamos en el directorio data/images, y generamos los ficheros

Figura 3.3: Estructura de un archivo JSON para train.

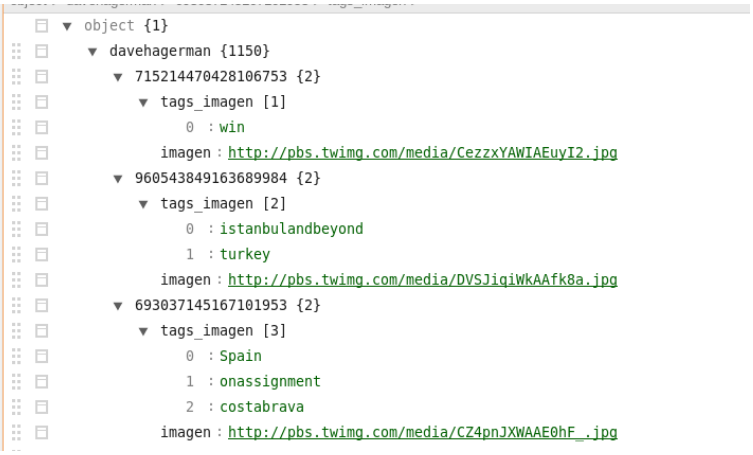

Figura 3.4: Estructura de un archivo JSON para test.

.txt con los datos de entrenamiento que serán los post de los usuarios en Instagram y los datos de test que serán los tweets de los usuarios en Twitter.

Comenzamos a generar las características del dataset:

En primer lugar se realiza un **cargado de los archivos JSON** tanto de entrenamiento como de test y parsearemos las palabras por una expresión regular para crear los tokens normalizados.

En segundo lugar **se generan los diccionarios**, para el subtitulado de imágenes se utiliza un diccionario de 40,000 palabras y para la predicción de hashtags un diccionario de 60,000, en ambos casos estos diccionarios se crearán a partir de las palabras utilizadas en los posts de Instagram (Training), en ningún caso se utilizarán las palabras utilizadas en Twitter (Test).

A continuación **se calculan las frecuencias TF-IDF** de las palabras utilizadas, esto permite ayudar a la red neuronal a ponderar las palabras a la hora de utilizarlas.

Por último, **se realiza el guardado de los datos** en un fichero .txt donde se almacenan los datos relacionados con cada imagen de post, como la matriz TF-IDF, la longitud total de la cadena o el nombre del vector numpy correspondiente a esa imagen.

Una vez configurados los ficheros de training y test, el siguiente paso es el mapeado de características de las imágenes, para ello se utiliza una red neuronal residual Resnet101.

Utilizando la red pre-entrenada Resnet 101 se logra almacenar las características de las imágenes en vectores que son almacenados en archivo numpy para su posterior uso, la identificación de estos ficheros es la misma que las usadas en las imágenes.

Se realizará este proceso con todas las imágenes que están contenidas en el directorio images/, se guardaran los los fichero .npy en el directorio /data/resnet_pool5_features/.

3.5.2. Fase de Entrenamiento

En el proceso de entrenamiento se ajustan los parámetros, como el número de épocas, valor de la constante de aprendizaje, contexto máximo del usuario, entrenamiento para subtítulos o para hashtags entre otros muchos.

En cada época se le pasa el lote de imágenes de entrenamiento y se calcula la pérdida de entropía cruzada, la entropía cruzada es una función continua que permite saber cómo de acertada está siendo la predicción, un valor muy alto indicará que a la red le queda mucho por aprender o que no está aprendiendo y un valor muy bajo indicará que la red está lista, siendo cero el valor óptimo.

Una vez calculado el error para cada lote de imágenes se propaga el error y se actualizan los pesos

w y sesgos b para toda la red.

Por último debido a que la red tarda mucho en entrenarse se realiza un punto de salvado (checkpoint) cada 500 iteraciones, se ha tenido que subir a 1000 debido a las limitaciones de espacio en disco. Una vez terminadas las 100,000 iteraciones, guardamos el último .ckpt que contiene los pesos de la red actualizados y borramos el resto.

3.5.3. Fase de Evaluación

Una vez generado el ckpt de la red neuronal de attend2u llega la parte de evaluación, donde lo primero que hace la librería es cargar el punto de salvado y los datos de test, el tiempo que tarda en evaluar es de apenas un minuto y devuelve la descripción o hashtags obtenidos para cada imagen.

Estas predicciones van acompañadas de su objetivo (la frase o hashtags objetivos), en caso de ser frases se utilizan las funciones de las métricas de COCO mencionadas en el estado del arte para evaluar el resultado y en caso de ser hashtags se utiliza f1_score (ver Ecuación 2.3).

3.5.4. Otros modelos implementados

Al no tener acceso a los baselines del paper de attend2u se hizo una versión lo más similar posible con la información disponible en el paper de algunos de estos baselines entre ellos KNN y alguna modificación de la red CSMN.

Como opción para comparar el rendimiento de la red Neuronal con otros algoritmos del aprendizaje automático se ha optado por **KNN**. Mediante la librería de Scikit-learn se ha utilizado la función **KNeighborsClassifier** para el uso del algoritmo KNN, para el trabajo con matrices el paquete **numpy** y para la conversión de palabras a vectores representativos **TfidfTransformer**.

La primera variante **KNN-image** utiliza la distancia euclídea entre dos imágenes, se realiza el entrenamiento con las imágenes de train utilizando los vectores numpy que caracterizan a esa imagen (obtenidos por la red ResNet101), donde el atributo serán los valores del vector de numpy y su clase el identificador de la imagen. Una vez entrenada predecimos la imagen de test y nos devolverá el identificador de la imagen más cercana, una vez obtenido el identificador se extrae su descripción.

La segunda variante busca el usuario más cercano **KNN-user**, dada una imagen se identifica al dueño de esa imagen y se consulta su vocabulario activo, es decir, las palabras utilizadas por ese usuario en todos sus posts, se compara ese vector de palabras con el resto de usuarios y se obtiene el usuario más cercano, de ese usuario elegimos una imagen al azar de su compendio de posts y la descripción de esa imagen es la elegida para la evaluación.

La tercera variante es una combinación de las dos anteriores **KNN-user-image**, en un primer paso se busca al usuario más cercano por vocabulario activo y una vez seleccionado ese usuario se cogen todas las imágenes de ese usuario y se busca la más similar mediante los vectores de numpy, devolviendo esa descripción para su evaluación.

Una versión alternativa que probamos para medir el rendimiento de la predicción de hashtags es la de variar la memoria de contexto de la red neuronal, es decir, el número de palabras del vocabulario activo que registra la red, este número de palabras se elige mediante ponderaciones TF-IDF del vocabulario total del usuario y de las palabras que se le añaden en la retroalimentación.

4

Pruebas y Evaluación

En este capítulo se detallan los resultados obtenidos en los experimentos para su posterior análisis. En un primer lugar se realiza una comparativa de los resultados originales de la red neuronal y los obtenidos con nuestra configuración tanto en el subtitulado de imágenes como en la predicción de hashtags y en segundo lugar se realizan distintas pruebas con los datasets cruzados de Instagram y Twitter, para la posterior discusión de los mismos.

4.1. Comparativa de los resultados para la evaluación de subtítulos

Antes de utilizar la red neuronal para la predicción de hashtags se estuvo trabajando en el subtitulado para comprender el funcionamiento de este tipo de algoritmos puesto que ambas funciones son muy similares, por este motivo nos ha parecido buena idea reflejarlo en el proyecto y ver los resultados obtenidos con la nueva configuración.

Se han intentado realizar los menos cambios posibles en la red neuronal para hacer viable el trabajo con la misma, el único parámetro que se ha modificado ha sido el número de épocas de entrenamiento para el cálculo de los pesos de la red, donde el número de épocas original era de 500,000, **pasando en nuestra configuración a 100,000 épocas**, este cambio será algo estándar en el resto de pruebas. Con esta configuración hemos obtenido mejores resultados que reduciendo la cantidad de datos de training, esto no es del todo extraño puesto que como hemos visto en la asignatura de neurocomputación los datos son la parte más importate a la hora de entrenar una red neuronal donde se necesitan grandes cantidades de los mismos para hacer un entrenamiento efectivo.

Se ha utilizado el dataset de Instagram proporcionado por la librería attend2u para replicar los archivos de training y de test. Para esta primera prueba el dataset pertenece a un dominio único. Hemos seleccionado una serie de baselines (definidos en la Sección 3.5.4) y hemos ejecutado de nuevo los experimentos, el porcentaje de datos de training es del 90 % y el de test del 10 %, otro dato importante es que los usuarios que aparecen en los datos de training y los usuarios que aparecen en

los de test no son los mismos en esta primera prueba.

Métodos	B-1	B-2	B-3	B-4	Meteor	CIDEr	ROGUE-L
1 NN-Im	0.071	0.020	0.007	0.004	0.032	0.059	0.069
1 NN-Usr	0.063	0.014	0.002	0.000	0.028	0.025	0.059
1 NN-UsrIm	0.106	0.032	0.011	0.005	0.046	0.084	0.104
(CSMN-W60-P5)	**0.171**	**0.068**	**0.029**	**0.013**	**0.064**	**0.214**	**0.177**
(CSMN-W20-P5)	0.116	0.041	0.018	0.007	0.044	0.119	0.123

(a) Resultados de los métodos de evaluación para el subtitulado de imágenes dataset de Instagram con la red neuronal original de attend2u [16].

Métodos	B-1	B-2	B-3	B-4	Meteor	CIDEr	ROGUE-L
1 NN-Im	0.065	0.018	0.009	0.007	0.030	**0.064**	0.066
1 NN-Usr	0.073	0.020	0.004	0.001	0.028	0.033	0.067
1 NN-UsrIm	0.088	0.024	0.000	0.003	0.040	0.056	0.072
(CSMN-W60-P5)	0.087	0.032	0.014	**0.008**	0.114	0.036	0.109
(CSMN-W20-P5)	**0.095**	**0.035**	**0.015**	0.007	**0.115**	0.040	**0.112**

(b) Resultados de los métodos de evaluación para el subtitulado de imágenes del dataset de Instagram con la red neuronal adaptada.

Tabla 4.1: Resultados con la red neuronal original 4.1(a) y con la red neuronal adaptada 4.1(b) usando el dataset de Instragram.

Si nos fijamos en la Tabla 4.1(b), podemos ver que un descenso del rendimiento general en los resultados obtenidos, esto es normal puesto que hemos reducido en un 80 % el número de épocas de entrenamiento, con lo cual nos parece que los resultados obtenidos son aceptables. Se puede ver cómo existe una tendencia similar en los algoritmos KNN, siendo el peor KNN-image y siendo el mejor KNN-User-image. Al no tener acceso a los baselines hemos hecho una replica de los mismos con los datos del paper y es normal que los resultados varíen.

Algo novedoso es que el baseline CSMN-W20-P5 con menos memoria de contexto del usuario es el que mejor rendimiento tiene en comparación con los resultados del original donde el que mejor resultado tenia era CSMN-W60-P5, recordemos que la memoria de contexto era el número de palabras que se almacenaban del vocabulario activo de un usuario ordenadas mediante una ponderación TF-IDF. Además, podemos ver que los resultados de la red neuronal CSMN-W20-P5 de los autores del paper y el de nuestra nueva configuración se obtienen resultados muy aproximados.

Hemos seleccionado seis imágenes de test de Instagram donde se pueden ver la descripción objetivo, los resultados obtenidos por los autores del paper de attend2u y los resultados obtenidos con nuestra configuración utilizando la red neuronal CSMN-W60-P5, como se puede ver en la siguiente figura 4.1, se aprecian palabras similares en ambas descripciones aunque la calidad de las mismas disminuyan.

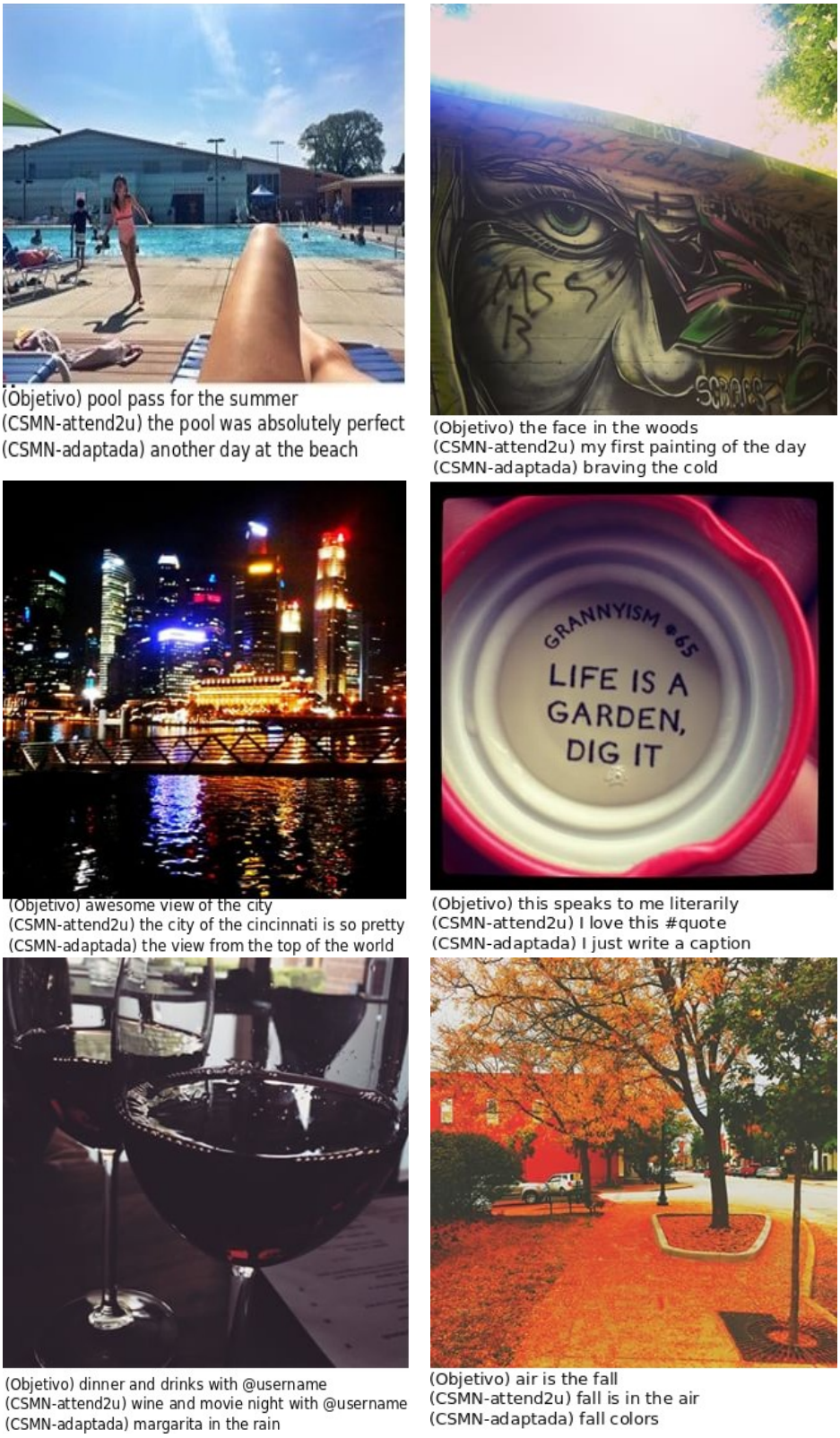

Figura 4.1: Ejemplos de subtítulos para imágenes.

4.2. Comparativa de los resultados para la evaluación de hashtags

Como hemos mencionado en anteriores capítulos (ver más en la Sección 3.5.1) a la hora de predecir hashtags realizaremos la misma disminución de épocas de entrenamiento que realizamos en el subtitulado de imágenes y utilizaremos un diccionario con un tamaño de 60,000 palabras en vez de las 40,000 utilizadas en los subtítulos. Este diccionario solo incluirá palabras que sean hashtags sin contar los subtítulos de referencia de las imágenes.

Métodos	F1-SCORE
1 NN-Im	0.049
1 NN-Usr	0.054
1 NN-UsrIm	0.109
(CSMN-W60-P5)	**0.230**
(CSMN-W20-P5)	0.147

(a) Resultados de los métodos de evaluación para la predicción de hashtags dataset de Instagram con la red neuronal original de attend2u [16].

Métodos	F1-SCORE
1 NN-Im	0.0379
1 NN-Usr	0.1345
1 NN-UsrIm	0.1635
(CSMN-W60-P5)	0.1745
(CSMN-W20-P5)	**0.1979**

(b) Resultados de los métodos de evaluación para la predicción de hashtags del dataset de Instagram con la red neuronal adaptada.

Tabla 4.2: Resultados con la red neuronal original 4.2(a) y con la red neuronal adaptada 4.2(b) usando el dataset de Instragram.

Como podemos ver en la Tabla 4.2, la progresión de los algoritmos de vecinos próximos se sigue cumpliendo, incluso se mejoran estos resultados, este motivo puede ser porque a la hora de utilizar KNN-user los creadores del paper utilizan un vocabulario con un máximo de 60 palabras del vocabulario activo del usuario al que pertenece la imagen consultada y así encontrar los usuarios más cercanos a ese vocabulario, pero en nuestro caso cogemos el vocabulario completo del usuario y lo comparamos con el resto de vocabularios completos de los usuarios de training, esto hace que la fase de entrenamiento sea más lenta puesto que aumenta la dimensionalidad de los vectores que identifican a estas palabras pero a cambio ofrece una mayor representación del vocabulario de los usuarios.

El rendimiento de la red CSMN-60 adaptada empeora aunque no tanto como en el caso del subtitulado, y como dato relevante destacamos que la red neuronal con una menor memoria de contexto CSMN-W20-P5 mejora los resultados originales de la red con un 0.1979 frente al 0.147 de la red original.

A continuación en la figura 4.2 se puede ver una pequeña comparativa de los resultados obtenidos en la predicción de hashtags con la red neuronal CSMN-W60-P5 original y la predicción hecha por la red adaptada. En este caso los resultados son mucho más similares que en el subtitulado de imágenes obteniendo palabras muy relacionadas en la mayoría de descripciones.

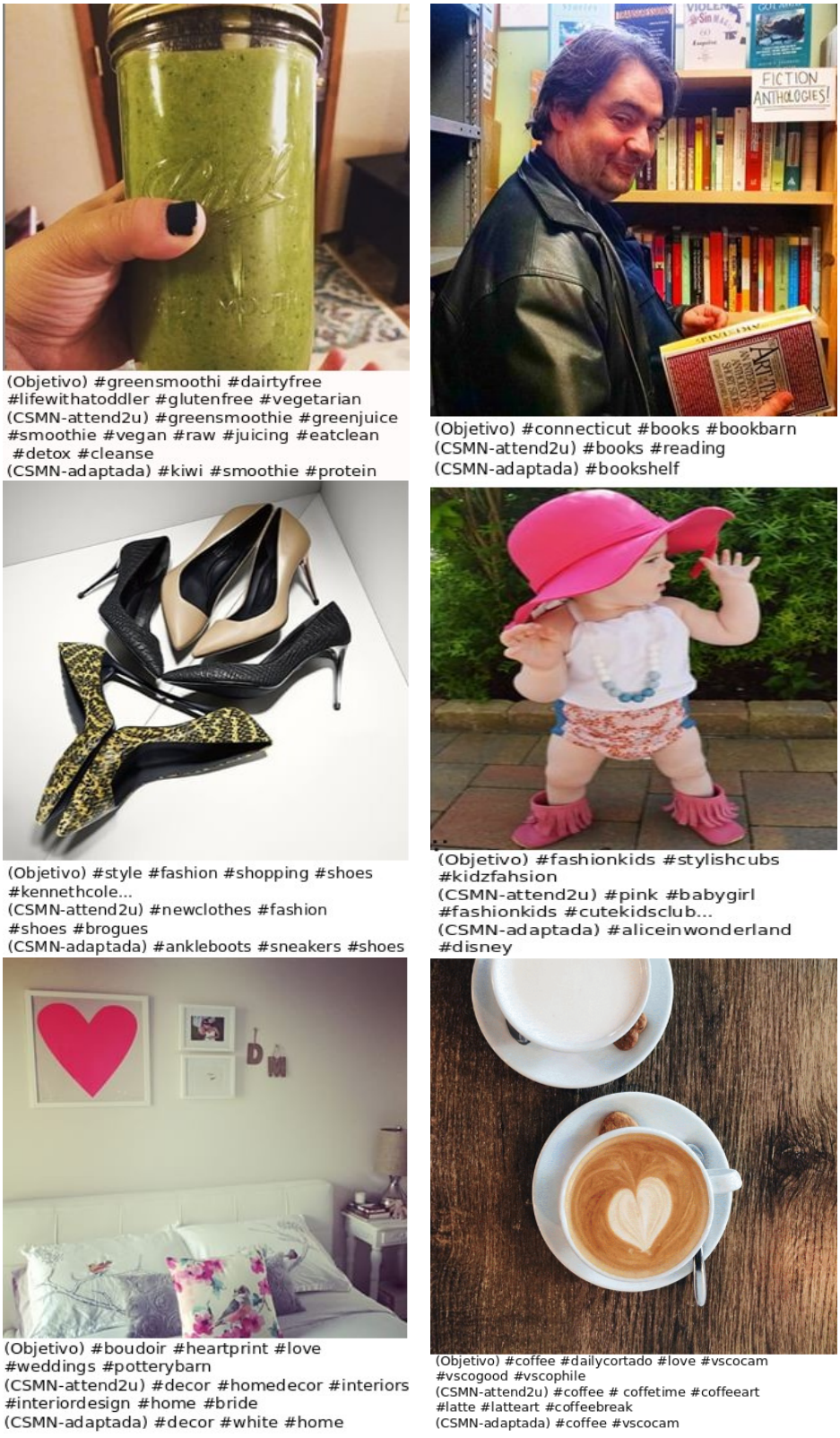

Figura 4.2: Ejemplos de predicción de hashtags para imágenes.

4.3. Resultados obtenidos con dataset cruzado

En este capítulo se realizan las diferentes pruebas con el dataset de Instagram y Twitter, a diferencia de como se ha usado en los experimentos anteriores donde, al igual que en el artículo de referencia [16], sólo se usaban datos de Instagram. Se llevan a cabo dos pruebas para cada tipo de usuario, la primera prueba consiste en un fichero de test con un único usuario y las imágenes relacionadas con este, y una segunda prueba con las imágenes de varios usuarios siendo estos del mismo tipo que el usuario único.

4.3.1. Usuario con un conjunto de imágenes pequeño

En un primer lugar a la hora de seleccionar las imágenes y hashtags vinculados a esas imágenes se puso como criterio que para ser datos válidos debían tener como mínimo una imagen y un hashtag ligados a esa imagen, este criterio no dio buen resultado. Esto no es del todo extraño puesto que había imágenes que solo tenían un hashtag vinculado y si no se acertaba el error era absoluto; por este motivo se añadió una nueva regla: la imagen debía tener como mínimo tres hashtags vinculados, es decir, los Tweets válidos de un usuario eran aquellos que tenían tres hashtags y una imagen, este criterio fue clave en relación a los resultados obtenidos.

Otro aspecto es el de añadir una nueva configuración de la red neuronal, debido a los resultados obtenidos en la comparativa de hashtags de la sección anterior se añadió una nueva configuración CSMN-W40-P5 para ver qué resultados obteníamos con una medida de contexto intermedia entre W20 y W60 utilizadas en el paper.

Para la primera prueba hemos utilizado un usuario con un conjunto de imágenes reducido, seleccionando un usuario con **69** imágenes para después aplicarle la métrica f1_score a los métodos utilizados.

Para la segunda prueba hemos utilizado usuarios con un rango de imágenes comprendido entre **1 y 169**, debido a la cantidad de imágenes de que disponemos en el dataset de Twitter hemos limitado el número de imágenes para todas las pruebas en **cuatro mil imágenes**. Esto permitirá que la cantidad de datos que tenga cada archivo no sea determinante a la hora de analizar los resultados.

Como se puede ver en la tabla 4.3(a), los algoritmos KNN no funcionan bien con un conjunto reducido de imágenes, esto es debido a que las distintas variantes de KNN no funcionan bien con poca información puesto que se busca la imagen más cercana, el usuario más cercano o las imágenes del usuario más cercano, y los hashtags vinculados a esas imágenes deben ser idénticos para obtener un buen resultado ya que en caso de que no haya ningún hashtag igual la métrica indicará cero o un número muy bajo.

Por otro lado las redes neuronales dan muy buenos resultados ya que trabajan con un diccionario

Métodos	F1-SCORE
1 NN-Im	0.006
1 NN-Usr	0.000
1 NN-UsrIm	0.000
(CSMN-W60-P5)	0.303
(CSMN-W40-P5)	0.301
(CSMN-W20-P5)	**0.313**

(a) Resultados de los métodos de evaluación para la predicción de hashtags con el dataset de Instagram y Twitter para un único usuario con un conjunto reducido de imágenes.

Métodos	F1-SCORE
1 NN-Im	0.054
1 NN-Usr	0.073
1 NN-UsrIm	0.085
(CSMN-W60-P5)	**0.162**
(CSMN-W40-P5)	0.157
(CSMN-W20-P5)	**0.162**

(b) Resultados de los métodos de evaluación para la predicción de hashtags con el dataset de Instagram y Twitter para varios usuarios con un conjunto reducido de imágenes.

Tabla 4.3: Resultados de la predicción de hashtags con el dataset cruzado de Instagram y Twitter con una cantidad de imágenes reducida.

que contiene las palabras más frecuentes de los usuarios, si recordamos en el caso del paper los usuarios que aparecían en training no aparecían en test pero en este caso el usuario único está en training y en test aunque los datos pertenezcan a dominios diferentes, por lo que se puede apreciar un usuario que utiliza un determinado vocabulario activo en Instagram tendrá un vocabulario muy similar en Twitter aún siendo un conjunto de imágenes tan reducido.

En la tabla 4.3(b) partimos con un fichero de test mucho más grande que está compuesto por usuarios con un conjunto pequeño de imágenes, la principal diferencia con respecto a la anterior prueba es en los algoritmos KNN donde vemos que con gran cantidad de imágenes los resultados obtenidos son mucho mejores, se mantiene la tendencia vista en la comparativa de la sección 4.2 siendo el mejor de ellos 1NN-UsrIm aunque no por mucho de 1NN-Usr, una explicación es que 1NN-usr utiliza un random para seleccionar la imagen del usuario más cercano, a diferencia de 1NN-UsrIm que una vez encontrado el usuario más cercano, busca la imagen más cercana de ese usuario, pero al existir tan pocas imágenes vinculadas a un usuario es normal que esa diferencia no sea tan grande.

En las redes neuronales seguimos obteniendo muy buenos resultados aunque no tan buenos como en la anterior tabla, esto es debido a que no ocurre como en el caso anterior donde todos los usuarios contenidos en training están en el test de Twitter, por eso el porcentaje de f1_score es menor, otro aspecto relevante es que la red neuronal ideal para los autores de attend2u CSMN-W60-P5, empata con CSMN-W20-P5 y nuestra nueva configuración queda algo peor pero no por mucho, esto es algo que ya empezamos a apreciar en la comparativa de hashtags y es que la red neuronal con menos épocas de entrenamiento es más eficaz con menor memoria de contexto.

En la siguiente figura 4.3 se pueden ver una serie de ejemplos con la red neuronal CSMN-W20-P5. Estos resultados cuentan con una gran tasa de acierto, mayor que la vista en la comparativa de hashtags llegando a tener un f1_score de 0.67 (siendo 1 el valor máximo de la métrica).

Figura 4.3: Ejemplos de predicción de hashtags para usuarios con un conjunto reducido de imágenes.

4.3.2. Usuario con un conjunto de imágenes intermedio

Como en el caso anterior para mantener un equilibrio entre los test de pruebas hemos trabajado con dos archivos de test: un archivo donde esta representando un usuario con **286** imágenes y otro fichero con un tamaño de cuatro mil imágenes donde están contenidas las imágenes de los usuarios con un rango de **170 y 349** imágenes.

El usuario único está contenido en los datos de training y test pero como siempre perteneciendo a un dominio distinto.

Métodos	F1-SCORE
1 NN-Im	0.000
1 NN-Usr	0.000
1 NN-UsrIm	0.000
(CSMN-W60-P5)	0.398
(CSMN-W40-P5)	0.420
(CSMN-W20-P5)	**0.457**

(a) Resultados de los métodos de evaluación para la predicción de hashtags con el dataset de Instagram y Twitter para un único usuario con un conjunto intermedio de imágenes.

Métodos	F1-SCORE
1 NN-Im	0.079
1 NN-Usr	0.140
1 NN-UsrIm	0.152
(CSMN-W60-P5)	0.173
(CSMN-W40-P5)	0.192
(CSMN-W20-P5)	**0.207**

(b) Resultados de los métodos de evaluación para la predicción de hashtags con el dataset de Instagram y Twitter para varios usuarios con un conjunto intermedio de imágenes.

Tabla 4.4: Resultados de la predicción de hashtags con el dataset cruzado de Instagram y Twitter con una cantidad de imágenes intermedia.

Se repite el hecho explicado anteriormente debido a que el primer fichero de test trabaja con un número pequeño de imágenes los resultados de KNN no son relevantes, en cambio las redes neuro-

nales mejoran en gran medida. Esto puede ser debido a que ahora contamos con más imágenes del usuario único, esto hace pensar que la tendencia se mantiene y mejora para un mismo usuario aunque el número de imágenes aumente los resultados no disminuyen sino que mejoran.

Para el segundo fichero de test vemos cómo los resultados también mejoran en gran medida, se observa que al hacer el test con usuarios con más imágenes se obtiene unos resultados más relevantes, llegando a alcanzar los resultados obtenidos por los autores del paper de attend2u con un 80 % menos de épocas de entrenamiento.

Figura 4.4: Ejemplos de predicción de hashtags para usuarios con un conjunto intermedio de imágenes.

4.3.3. Usuario con un conjunto de imágenes grande

Para la última prueba de usuarios hemos seleccionado un usuario único con **919** imágenes, como novedad este usuario no está contenido en los datos de training de Instagram. Por otro lado tenemos el segundo fichero que guarda usuarios con un conjunto de imágenes mayor de **350**.

Métodos	F1-SCORE
1 NN-Im	0.030
1 NN-Usr	0.068
1 NN-UsrIm	0.074
(CSMN-W60-P5)	0.086
(CSMN-W40-P5)	0.127
(CSMN-W20-P5)	**0.150**

(a) Resultados de los métodos de evaluación para la predicción de hashtags con el dataset de Instagram y Twitter para un único usuario con un conjunto grande de imágenes.

Métodos	F1-SCORE
1 NN-Im	0.082
1 NN-Usr	0.085
1 NN-UsrIm	0.096
(CSMN-W60-P5)	0.141
(CSMN-W40-P5)	0.139
(CSMN-W20-P5)	**0.145**

(b) Resultados de los métodos de evaluación para la predicción de hashtags con el dataset de Instagram y Twitter para varios usuarios con un conjunto grande de imágenes.

Tabla 4.5: Resultados de la predicción de hashtags con el dataset cruzado de Instagram y Twitter con una cantidad de imágenes grande.

Como se puede ver en la tabla 4.5(a) nos encontramos unos resultados no tan altos como los que nos esperábamos, esto seguramente sea debido al motivo que distingue a este fichero del resto de ficheros únicos y es que no se han utilizado los datos de Instagram de este usuario para entrenar la red neuronal, con lo cual los resultados obtenidos son más cercanos a los vistos en la sección 4.2 donde los usuarios de test no estaban en los datos de training.

Los algoritmos KNN comienzan a dar resultados más aceptables, esto es comprensible ya que el usuario único ya cuenta con un número considerable de imágenes lo que hace viable el uso de KNN para la predicción de hashtags, la tendencia que apreciamos en la sección 4.2 se sigue respetando, mantiéndose KNN-usr-image como el mejor método de los vecinos próximos.

Algo bastante llamativo es que parece ser que una cantidad excesiva de imágenes no produce los mejores resultados, puesto que en el segundo fichero de test existen usuarios contenidos y no contenidos en training, los algoritmos basados en vecinos próximos obtienen buenos resultados y los basados en redes neuronales también pero no los esperados, puesto que se pensaba que la tendencia se mantendría e incluso mejoraría con usuarios con muchas imágenes pero esto no es así, es posible que la mayoria de los usuarios no pertenezcan a los datos utilizados de training y por eso los resultados no sean los esperados.

(Objetivo) #cat #catnap #catagram #catsofinstagram
(CSMN-W20-P5) #catagram #catsofinstagram
#tuxedocat #caturday #cat

(Objetivo) #guineapig #animal #pet #cute #canon
(CSMN-W40-P5) #pet #cute #guineapig #pets

Figura 4.5: Ejemplos de predicción de hashtags para usuarios con un conjunto grande de imágenes.

4.3.4. Conjunto aleatorio de imágenes

Por último hemos realizado una división distinta de los datos de test, en vez de dividir el archivo por conjuntos de imágenes ligadas a los usuarios, hemos seleccionado imágenes aleatorias de todo el conjunto de usuarios sin tener en cuenta el tipo de usuario al que pertenece. Para hacerlo lo más similar posible a la sección 4.2 hemos utilizado un test formado por cinco mil imágenes.

Métodos	F1-SCORE
1 NN-Im	0.057
1 NN-Usr	0.111
1 NN-UsrIm	0.123
(CSMN-W60-P5)	0.179
(CSMN-W40-P5)	**0.191**
(CSMN-W20-P5)	0.188

Tabla 4.6: Resultados obtenidos en la predicción de hashtags haciendo una división por imágenes aleatorias y no por usuarios.

Al ver los datos obtenidos en la tabla 4.6 podemos ver cómo los datos son muy similares o casi idénticos a los obtenidos en la tabla 4.2, esto nos hace pensar que la tendencia con imágenes aleatorias de los usuarios es la misma al usar imágenes aleatorias de un dominio único y dominios cruzados como es el caso.

KNN nos muestra buenos resultados aunque varían un poco con respecto a la comparativa de hashtags, y en este caso la red neuronal que sale más beneficiada es la red propuesta por nosotros: la CSMN-W40-P5.

4.4. Discusión de los resultados

Al hacer la comparativa entre los datos obtenidos por los autores del paper de attend2u y los cambios hechos en el entrenamiento de la red neuronal para hacer viable la ejecución de la misma en nuestro ordenador, se ha podido ver una disminución del rendimiento algo que no nos extraña ya que el entrenamiento ahora es mucho menor, pero por otro lado hemos visto cómo la red neuronal CSMN-W20-P5 es mejor trabajando con menos épocas que la CSMN-W60-P5, esto seguramente sea debido a que con más palabras de contexto sea necesario entrenar la red con más épocas para obtener mejores resultados. Por otro lado hemos visto cómo la tendencia en los algoritmos KNN se ha respetado en todas las pruebas siendo el peor de ellos KNN-image y el mejor KNN-user-image, el rendimiento de este último es muy sorprendente puesto que el tiempo de entrenamiento es de una hora y treinta minutos aproximadamente, esto hace que KNN sea un algoritmo útil en caso que queramos utilizar una alternativa a las redes neuronal ya que estas aún siendo mejores necesitan muchos más recursos en la fase de entrenamiento (de media, tardan entre 18 y 24 horas). En este caso KNN será válido siempre que se cuente con las suficientes imágenes de test para hacerlo viable

A la hora de realizar las pruebas en dominios cruzados hemos visto cómo el usuario con un número de imágenes intermedio es el que mejor resultados ha dado con respecto al pequeño y al grande. Algo vital a la hora de obtener buenos resultados es que los usuarios contenidos en los datos de test de Twitter estén contenidos en los datos de training de Instagram, puesto que como hemos visto el vocabulario activo de los usuarios es muy similar aún siendo dos dominios distintos.

5

Conclusiones y Trabajo Futuro

En este capítulo se recogen las conclusiones de este documento donde se explican los resultados obtenidos y el potencial del proyecto en futuros trabajos.

5.1. Conclusión

Hemos podido ver cómo a la hora de utilizar redes neuronales para la predicción de subtítulos o hashtags es necesario tener un equipo hardware con grandes prestaciones, de lo contrario esta tarea puede ser imposible de realizar debido a la alta carga computacional de los métodos estudiados en este trabajo (redes neuronales).

Hemos comprobado como al disminuir los datos de training de una red neuronal se obtienen peores resultados que modificando el número de épocas de entrenamiento, esto nos hace pensar que es mejor tener datos de gran calidad a la hora de entrenar que el número de iteraciones que se realice sobre la red neuronal para entrenar los pesos, siempre y cuando exista un equilibrio aceptable en estos dos aspectos.

Por otro lado hemos visto cómo la red neuronal con menos épocas de entrenamiento es más eficaz utilizando un conjunto pequeño de palabras del vocabulario del usuario antes que trabajar con una gran cantidad de palabras, puesto que el número de épocas de entrenamiento debería ser mayor para ajustar estos parámetros de manera adecuada.

El aspecto más determinante a la hora de obtener buenos resultados ha sido el de entrenar la red con los datos de Instagram de un usuario y luego utilizar los datos de Twitter de ese mismo usuario para el test, los resultados obtenidos han sido muy satisfactorios indicándonos que existe una alta correlación entre los vocabularios activos de ambos usuarios a la hora de comentar las imágenes en los distintos dominios.

También se ha visto que los algoritmos KNN resultan ser alternativas con mucho potencial a la hora de predecir subtítulos o hashtags para una imagen puesto que su fase de entrenamiento es muy corta, aunque siempre será necesario contar con una gran cantidad de imágenes de test para tener

resultados válidos puesto que con las métricas utilizadas no basta con tener palabras similares a la del objetivo: deben ser idénticas.

Por último, todo el código y los scripts desarrollados se encuentran disponibles en el repositorio: `https://bitbucket.org/Rsanchezguzman/crosshashtags`.

5.2. Trabajo Futuro

En el futuro, nos gustaría realizar pruebas en otros dominios y ver qué resultados podríamos obtener, por ejemplo, en el ámbito de la música, sería interesante ver cómo las redes neuronales u otro tipo de algoritmos de aprendizaje automático son capaces de trabajar con otros tipos de datos y ver si se obtienen los mismos resultados (o similares) que hemos observado en este TFG.

Otra idea interesante sería recoger información de dos dominios mediante estos algoritmos para utilizarlo en un tercero, por ejemplo a la hora de que un usuario se registre en un nuevo dominio utilizar la información de los dominios auxiliares para recomendarle algún producto y lidiar con el problema del arranque en frío (cold start) [1], esto permitirá una recomendación muy personalizada y evaluar si la recomendación mediante está información es útil.

Otro aspecto interesante para sortear el problema del hardware habría sido el de utilizar en vez de imágenes un tipo de dato menos pesado, como es el de las palabras para el uso de redes neuronales, por ejemplo, para estudiar la correlación en posts de Instagram y Facebook.

Bibliografía

[1] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Syst. Appl.*, vol. 41, pp. 2065–2073, Mar. 2014.

[2] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Comput. Surv.*, vol. 52, pp. 5:1–5:38, Feb. 2019.

[3] eLogia, "Estudio anual de redes sociales." `http://proceedings.mlr.press/v32/graves14.pdf`, 2018.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[5] S. Contreras and F. De la Rosa, "Aplicación de deep learning en robótica móvil para exploración y reconocimiento de objetos basados en imágenes," in *Aplicación de Deep Learning en Robótica Móvil para Exploración y Reconocimiento de Objetos basados en Imágenes*, 09 2016.

[6] Z. C. Lipton, "A critical review of recurrent neural networks for sequence learning," *CoRR*, vol. abs/1506.00019, 2015.

[7] S. F. Felix Gers, Fred Cummins, "Understanding lstm networks." `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[9] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015.

[10] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, (Stroudsburg, PA, USA), pp. 311–318, Association for Computational Linguistics, 2002.

[11] R. Vedantam, C. L. Zitnick, and D. Parikh, "Cider: Consensus-based image description evaluation.," in *CVPR*, pp. 4566–4575, IEEE Computer Society, 2015.

[12] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Proc. ACL workshop on Text Summarization Branches Out*, p. 10, 2004.

[13] S. Banerjee and A. Lavie, "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments," in *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, (Ann Arbor, Michigan), pp. 65–72, Association for Computational Linguistics, June 2005.

[14] "Accelerate machine learning with the cudnn deep neural network library." https://devblogs.nvidia.com/accelerate-machine-learning-cudnn-deep-neural-network-library/. Accessed 28 April 2019.

[15] "Tweepy documentation — tweepy 3.7.0 documentation." https://tweepy.readthedocs.io/en/latest. Accessed 5 May 2019.

[16] C. C. Park, B. Kim, and G. Kim, "Attend to You: Personalized Image Captioning with Context Sequence Memory Networks," in *CVPR*, 2017.

[17] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. E. Hinton, "Grammar as a foreign language," *CoRR*, vol. abs/1412.7449, 2014.

[18] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: Lessons learned from the 2015 MSCOCO image captioning challenge," *CoRR*, vol. abs/1609.06647, 2016.

[19] K. Xu, J. Ba, R. Kiros, K. Cho, A. C. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *CoRR*, vol. abs/1502.03044, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[21] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context.," *CoRR*, vol. abs/1405.0312, 2014.

Definiciones

API Se trata de un conjunto de procedimientos y subrutinas que ofrecen a un programa software el acceso a un servicio.

Dominio Nombre que identifica un sitio web.

Hashtags Cadena de caracteres precedida por una almohadilla que puede estar compuesta por una o varias palabras concatenadas.

JSON Formato de intercambio de texto conocido por ser más sencillo que XML.

Test Conjunto de imágenes y subtítulos utilizado para evaluar el modelo entrenado.

TF-IDF Métrica utilizada para medir la frecuencia de los términos.

Tweet Cadena de texto con una longitud máxima de 280 caracteres que puede contener: hipervínculos, imágenes, videos y texto.


UAM
UNIVERSIDAD AUTONOMA
DE MADRID