UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CLASIFICACIÓN DE USUARIOS DE INSTAGRAM EN BASE A TEXTO E IMÁGENES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

MAXIMILIANO SEBASTIÁN KAUER MADRID

PROFESOR GUÍA:
JOSÉ SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
BARBARA POBLETE LABRA
JUAN BARRIOS NUÑEZ

SANTIAGO DE CHILE
2019

CLASIFICACIÓN DE USUARIOS DE INSTAGRAM EN BASE A TEXTO E IMÁGENES

El proyecto consiste en generar un modelo de clasificación de influenciadores. El contexto es el mercado de *marketing* digital en Instagram, red social basada en publicaciones de sus usuarios, donde se puede comentar las publicaciones de otras personas y dar *likes*. Un influenciador es un usuario que es considerado "influyente"; tiene muchos seguidores, normalmente más de 3.000, y realiza campañas publicitarias en las que promociona productos a través de su cuenta personal de Instagram.

Haip es una empresa dedicada a este mercado, que busca encontrar los influenciadores más apropiados para campañas publicitarias de, por ejemplo, productos deportivos o de belleza. Actualmente, buscar influenciadores es un proceso costoso porque es, en su mayoría, manual, y requiere de un gasto considerable de tiempo. Por esto es que se busca clasificar a los influenciadores en categorías como "deporte", "aire libre", etc. Conocer de antemano las categorías a las que pertenece cada influenciador permitirá agilizar las operaciones de la empresa y optimizar los procesos.

Para lograr el objetivo planteado, se usan modelos de aprendizaje de máquinas tales como redes neuronales, *word embeddings* y SVMs. Para ello se deben recolectar suficientes datos y entrenar los modelos de manera eficiente. Los datos para clasificar a los influenciadores consisten en sus publicaciones, los que incluyen una o más imágenes y un texto, sin embargo, para entrenar los modelos es necesario primero juntar una cantidad suficiente de datos previamente etiquetados, lo que se logra buscando fuentes de imágenes de acceso público en ImageNet y Pexels. Para clasificar una publicación, se usa ResNet, primero para clasificar la imagen y segundo para producir un vector en $\mathbb{R}^n$ a partir de la imagen, y fastText para hacer lo mismo a partir del texto. Luego se usa una red neuronal para la clasificación usando ambos vectores. Finalmente se compara la precisión de la clasificación usando solo las imágenes, solo los textos, y usando ambos.

Se logra diseñar un clasificados que es muy efectivo para clasificar en 5 categorías. El clasificador se compara con un conjunto de influenciadores ya clasificados por un grupo de 2 personas, y tiene un rendimiento suficientemente alto, por lo que pasa a ser usado en la plataforma de Haip. En un trabajo futuro, se puede extender el clasificador para aumentar la cantidad de categorías.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

El proyecto de memoria se enmarca en el contexto de la empresa Haip, emprendimiento chileno dedicado al rubro del mercado de influenciadores. Dicho mercado se centra en torno a personas influyentes (los influenciadores) en vez del mercado completo, y aborda campañas publicitarias en las que estos realizan publicaciones con publicidad en redes sociales, por ejemplo, Twitter e Instagram.

Instagram es una red social en la que los usuarios registrados pueden subir publicaciones, las que consisten en una foto y un texto, y otros usuarios pueden comentar y dar likes a las publicaciones. El usuario puede elegir entre un perfil público, en el cual sus publicaciones son visibles por cualquier persona, o privado, en el que sólo son visibles por sus seguidores, quienes previamente deben ser aceptados, pudiendo dar acceso a aplicaciones registradas para usar la API de Instagram. Un influenciador es un usuario popular, por ejemplo si tiene más de 3.000 seguidores.

Haip trabaja con empresas que realizan campañas principalmente en Chile a través de influenciadores en Instagram. Se buscan influenciadores apropiados para cada campaña de acuerdo a distintas métricas. Algunas de estas son las interacciones (suma entre cantidad de comentarios y de likes) promedio por publicación y la afinidad que éstos tengan con la campaña misma. Luego se le indica a las marcas cuáles son los influenciadores más afines a la campaña, así como el precio límite que conviene gastar, de acuerdo a sus características. Esta información se les pone a disposición a través de una aplicación web, que recopila información a través de la API mencionada. Algunas marcas que actualmente trabajan con Haip son Falabella y Puma.

En la situación actual, la búsqueda de los influenciadores es un proceso manual, lo que lo hace muy costoso, por lo que el propósito de este trabajo es optimizar el proceso mediante técnicas de *machine learning* a través de la clasificación de los influenciadores en distintas categorías, por ejemplo deporte, aire libre, etc., con el fin de conocer la afinidad de los influenciadores con las campañas publicitarias. Haip posee una base de datos con información de los influenciadores que trabajan en campañas de los clientes, que incluye sus publicaciones. Esta es la información que se pretende usar para la clasificación mediante inteligencia artificial.

1.1. Objetivos

Diseñar e implementar un sistema basado en aprendizaje de máquinas que permita clasificar influenciadores que forman parte de Haip a fin de optimizar el proceso de recomendación.

1.1.1. Objetivos Específicos

1. Definir métricas de evaluación para modelos de clasificación de imágenes y de texto.
2. Probar y evaluar los siguientes modelos:
   - Redes neuronales convolucionales para clasificar imágenes.
   - *Word embeddings* para clasificar texto.
3. Combinar ambos modelos, clasificación de texto e imágenes, y comparar los resultados.
4. Validar el funcionamiento de la clasificación a través de un prototipo.

1.2. Idea general de la solución

En este trabajo se desea clasificar influenciadores, usuarios de Instagram, dentro de ciertas posibles categorías: aire libre, animales, arte, comida, deporte, familia, moda, música, vida social y vida sana, entre otras. De estas categorías, inicialmente se seleccionan 5 para realizar pruebas y obtener los primeros resultados. Con este fin, la clasificación se realizará mediante el texto e imágenes de sus publicaciones.

Figura 1.1: Ejemplos de cinco categorías. De izquierda a derecha: deporte, aire libre, animales, comida y moda

Con las categorías ya definidas, se debe pasar a la recolección y curación de datos que puedan ser usados en el entrenamiento de los modelos de *machine learning* y en la validación de los resultados. La principal fuente de datos es Instagram, sin embargo sus datos no están clasificados, por lo que se usarán fuentes de datos alternativas. Es importante tener una gran cantidad de datos etiquetados, para lo que se buscará en distintas fuentes. ImageNet[1] es un repositorio de imágenes recolectadas con fines educativos y de investigación y clasificadas de acuerdo a la base de datos léxica de WordNet[2]. Desde el año 2010, ImageNet usa un

[1] http://image-net.org/
[2] https://wordnet.princeton.edu/

subconjunto de las imágenes, cada una de las cuales pertenece a una de mil posibles categorías, en una competencia de clasificación: ImageNet Large Scale Visual Recognition Challenge (ILSVRC). El problema es que solo están disponibles las url y muchas están obsoletas, pero aún así hay miles de imágenes etiquetadas disponibles para ser descargadas. Otra fuente es Pexels[3], página web que contiene imágenes gratuitas etiquetadas por ellos. Se buscaron las etiquetas que calzan con las categorías usadas en esta memoria. Por ejemplo, al buscar en el sitio web de Pexels según las etiquetas **cooking** y **chef** se obtienen imágenes pertenecientes a la categoría **comida** de este trabajo, y se puede repetir el mismo procedimiento para conseguir imágenes de todas las categorías.

Además de imágenes, es necesario tener texto clasificado para entrenar las redes neuronales que usen *word embeddings*, es decir, representaciones vectoriales del texto. Este tipo de datos es más difícil de conseguir, en parte porque las bases de datos de uso público que se encuentran en internet para inteligencia artificial se centran en texto en inglés, por lo que se seguirá otro camino, el cual consiste en aprovechar el hecho de que las publicaciones son datos que vinculan texto con imágenes. En base a esa propiedad de las publicaciones, después de haber entrenado los modelos de clasificación de imágenes, se clasificarán las imágenes de un conjunto de publicaciones, y posteriormente se usará el texto asociado como datos de entrenamiento para los modelos en base a *embeddings* de texto.

Python es el lenguaje más popular actualmente en el área, y el que se usa en Haip, por lo tanto se usará en la memoria. La librería PyTorch[4] tiene implementaciones de ResNet pre entrenado, de SVMs y de otros algoritmos usados en la memoria. Se realizarán pruebas con dichos modelos y distintos valores de hiperparámetros, que se ajustarán dependiendo de los resultados, iterando el proceso hasta llegar a la mejor solución. Una vez entrenada la red neuronal convolucional ResNet, ésta se usa para clasificar publicaciones y así obtener textos clasificados y tener datos para entrenar clasificadores usando word embeddings. Se puede usar otra red neuronal o una SVM, realizando pruebas similares a las anteriores entrenando el clasificador usando los vectores entregador por word2vec y fastText en vez de imágenes, ajustando los hiperparámetros de forma adecuada. Para realizar las pruebas se tiene disponible un servidor con GPU Nvidia Titan X Pascal de 12 GB, perteneciante a Orand[5], además de mi computador de uso personal, que cuenta con un procesador Intel Core i5-5200U y 8 GB de memoria RAM.

El objetivo final es clasificar al influenciador, no sus publicaciones, por lo que se debe idear una estrategia para obtener la clasificación final. Lo que se realizará es clasificar todas sus publicaciones y usar alguna forma de mayoría: se busca un umbral de tal forma que si un influenciador tiene al menos cierto porcentaje de sus publicaciones en cierta categoría, entonces el influenciador pertenece a la categoría. Finalmente, se debe implementar un prototipo que clasifique influenciadores de acuerdo a los datos de la plataforma de Haip para poder evaluar el sistema desde el punto de vista del usuario final.

[3]https://www.pexels.com/
[4]https://pytorch.org/
[5]http://www.orand.cl/

Sumario

- Se propone como objetivo principal clasificar influenciadores en base al texto e imágenes de sus publicaciones en Instagram.
- Cada influenciador puede pertenecer a más de una clase, por ejemplo deporte, moda, etc.
- Se experimentará con redes neuronales y word embeddings para clasificar las publicaciones, a partir de las cuales se clasificará al influenciador, y se implementará en Python.

Capítulo 2

Preliminares

En el trabajo de esta memoria se usan varios modelos de aprendizaje de máquinas descritos en esta sección. El más importante es el de red neuronal, ya que es el que finalmente permitirá clasificar a los influenciadores.

Una consecuencia relevante de la dependencia de algoritmos de aprendizaje de máquinas es que será muy importante poder recolectar un volumen de datos capaz de entrenas los modelos. En el capítulo siguiente se detalla el proceso para juntar los datos necesarios.

2.1. Redes neuronales

El principal modelo de aprendizaje de máquinas estudiado para esta memoria es el de redes neuronales (neural networks en inglés). El objetivo de una red neuronal es aproximar una función f^*. Por ejemplo si un clasificador $y = f^*(x)$ mapea x a una categoría y de un conjunto de c categorías, la red busca encontrar $y = f(x, \theta)$, donde θ son los parámetros de la red que buscan la mejor aproximación posible de f^*[2]. θ se busca mediante un proceso de entrenamiento a partir de datos de los cuales ya se conoce a que categoría pertenecen.

Las redes neuronales se llaman redes porque funcionan mediante un composición de funciones lineales llamadas **neuronas**, las cuales generalmente se agrupan en **capas**, las que a su vez le aplican una función no lineal a la salida de la neurona, por ejemplo $\sigma(x) = \frac{1}{1+e^{-x}}$ o $tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$, llamadas funciones de activación. La primera capa, o capa de entrada, recibe como input los datos de entrenamiento que se usan para aprender los parámetros θ, mientras que la última capa, o capa de salida es un vector en $\mathbb{R}^c$ donde todos sus elementos están en el rango $(0, 1]$ y suman 1, es decir, se interpretan como una distribución de probabilidad y la categoría con la probabilidad más alta es la predicha por el modelo. El resto de las capas de llaman capas ocultas. En las capas ocultas que se ven en la figura 2.1, la salida de todas las neuronas se usa como entrada de todas las neuronas de la capa siguiente, por lo que se llaman capas *fully connected*. Además de usar la salida de la red como una distribución de probabilidad, se puede extraer el vector de la capa anterior, llamado *feature vector*, que puede ser por ejemplo de tamaño 512, y usarlo como descriptor de la imagen.

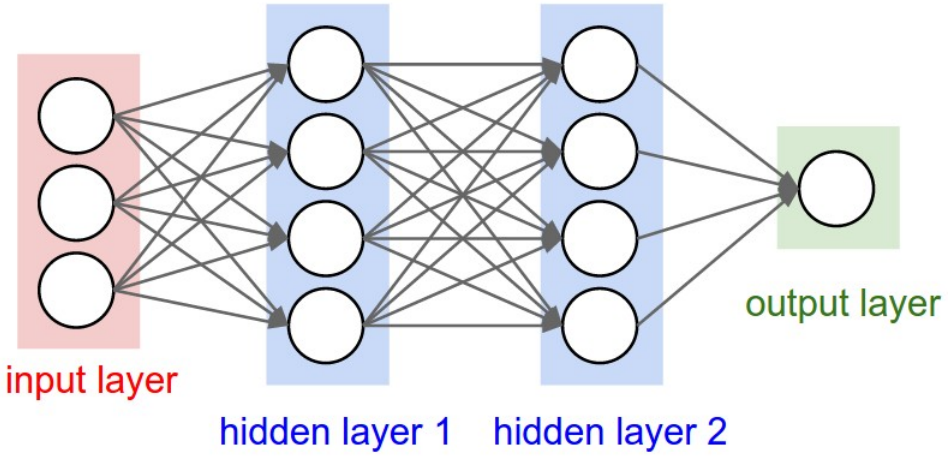

Figura 2.1: Red neuronal de 4 capas, cada una con 3, 4, 4 y 1 neurona respectivamente

Para medir los resultados de la red, se usa la precisión, que se define como el porcentaje de datos correctamente clasificados. Se divide en precisión de entrenamiento, que se mide con respecto a los datos que se usan para ajustar los parámetros θ de la red, y precisión de validación, que es con respecto a los datos que se usan durante el entrenamiento para verificar los resultados, pero no para ajustar los parámetros. La precisión de validación es más importante, porque predice mejor cómo se va a comportar la red con datos nuevos. Además se puede usar otro conjunto de datos de test, que sólo se evalúan con el modelo una vez terminado el entrenamiento.

Entrenamiento e hiperparámetros

Para entrenar una red neuronal se deben elegir valores para los hiperparámetros, que son los parámetros cuyo valor queda fijo una vez que se inicia el entrenamiento. Entre ellos se encuentran:

- Cantidad de datos de entrenamiento y de validación.
- Cantidad de épocas (o *epochs*), es decir, cuántas veces se le muestra el conjunto de entrenamiento completo a la red. Este parámetro debe ser lo suficientemente alto para que la red aprenda suficiente, pero no demasiado por restricciones de tiempo y porque puede causar que la precisión de entrenamiento sea mucho más alta que la de validación (overfitting).
- Tamaño de lote (o *batch*): es la cantidad de datos que se le pasan a la red cada vez que actualiza sus parámetros.
- Tasa de aprendizaje y momentum, que miden cuánto deben cambiar los parámetros θ mientras se entrena.

Como los hiperparámetros son fijos, hay que elegir su valor antes de entrenar, pero de todos modos hay que encontrar valores apropiados, por lo que se hacen varias pruebas con distintos valores y se eligen los que entreguen mejores resultados.

2.1.1. Redes neuronales convolucionales

Las redes convolucionales (convolutional neural networks) son un caso particular de las redes neuronales prealimentadas que funcionan bajo el supuesto de que los datos pertenecen a una topología tipo grilla, que es el caso de las imágenes si son vistas como matrices de pixeles [3]. La única diferencia con respecto a las redes neuronales no convolucionales es que una o más de las capas escondidas de la red son operaciones optimizadas para este tipo de datos organizando sus neuronas en 3 dimensiones, por tanto se les llama **capas convolucionales**. A diferencia de las capas fully connected, en las capas convolucionales las neuronas solo se conectan con las que están cerca espacialmente, a una distancia no mayor al tamaño de ventana, un parámetro interno del modelo.

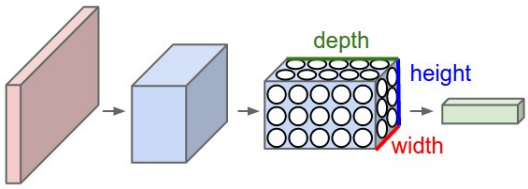

Figura 2.2: Ilustración de capas convolucionales

2.1.2. ResNet

Una red neuronal reciente y bastante eficaz es ResNet, propuesta en 2015 y descrita en [5]. ResNet tiene variantes con 18, 34, 50, 101 y 152 capas. Las variantes con más capas son más complejas y requieren mayor tiempo de entrenamiento y más datos para llegar a buenos resultados, por lo que en este caso es mejor probar con ResNet usando 18 y 34 capas. ResNet se usará con 2 fines: por un lado como clasificador de imágenes y por otro como *feature extractor*. Como clasificador, se entrena y usa de forma normal. Como feature extractor, significa que se ignora la última capa de la red, la encargada de clasificar, y se recoge el vector que se obtiene de la capa anterior, el feature vector, en este caso un vector de tamaño 512, que es una representación de la imagen.

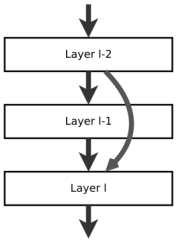

Figura 2.3: Ilustración de la capas dentro de la arquitectura ResNet.

La principal característica de ResNet que la distingue de otras redes neuronales es que usa *atajos* (identity shortcuts) entre las capas, es decir, conexiones entre capas que no son consecutivas. El objetivo de estas conexiones es permitir entrenar redes con mayor cantidad de capas sin que baje el rendimiento al actualizar tantos parámetros. Por esta razón, ResNet tiene variantes más profundas que otras arquitecturas, lo que permite aumentar la capacidad del modelo.

La arquitectura usada en este trabajo toma como entrada una matriz de $224 \times 224 \times 3$, es decir una imagen de 224×224 pixeles, y consiste, en la versión de 34 capas, de 32 capas convolucionales y una capa fully connected al final de tamaño igual

a la cantidad de clases en las que se clasifique, con identity shortcuts de forma intermitente. Las versiones de ResNet de 18 o 54 capas siguen la misma arquitectura pero con menos o más capas convolucionales. Como las imágenes que se usan no son de ese tamaño, 224×224, se les aplica una normalización que las comprime a ese tamaño antes de ser ingresadas al modelo.

2.2. SVM

Otro modelo de clasificación es el de SVM (support vector machine), clasificadores lineales que funciona mediante aprendizaje supervisado. Esto quiere decir que, al igual que las redes neuronales, necesita ejemplos clasificados a partir de los cuales ajustar sus parámetros internos para poder clasificar en base a combinaciones lineales de las representaciones de los ejemplos vistos durante el entrenamiento. La fortaleza de las SVM es que funcionan bien cuando los datos son de dimensiones altas, como es el caso de las representaciones vectoriales de las publicaciones.

A diferencia de las redes neuronales, las SVMs son clasificadores binarios, por lo que para trabajar con múltiples clases se debe adoptar alguna estrategia como *one vs the rest*, en la que se entrenan tantos clasificadores como clases, o *one vs one*, en la que se entrena un clasificador por cada par de clases, es decir, $n \times (n-1)/2$ clasificadores, si n es la cantidad de clases. La segunda opción es más costosa computacionalmente porque son más combinaciones, pero tiene mayor capacidad para encontrar relaciones entre los datos. Dado que el entrenamiento de SVM no fue un cuello de botella importante (en contraste con el entrenamiento de redes neuronales), se optó por la estrategia, *one vs one*.

Otra elección importante al trabajar con SVMs, es la de un *kernel* apropiado. Un *kernel* es una función que mapea los datos que se clasifican a un espacio de mayor dimensión. Esto se usa porque las SVM, al usar funciones lineales, separan unos datos de otros (i.e. clasifican) mediante hiperplanos, pero no siempre se pueden encontrar hiperplanos que funcionen bien en los datos originales, ya que rara vez hay una función lineal que separe inmediatamente los datos. Al tratar con datos de mayor dimensión, es más fácil encontrar esa función.

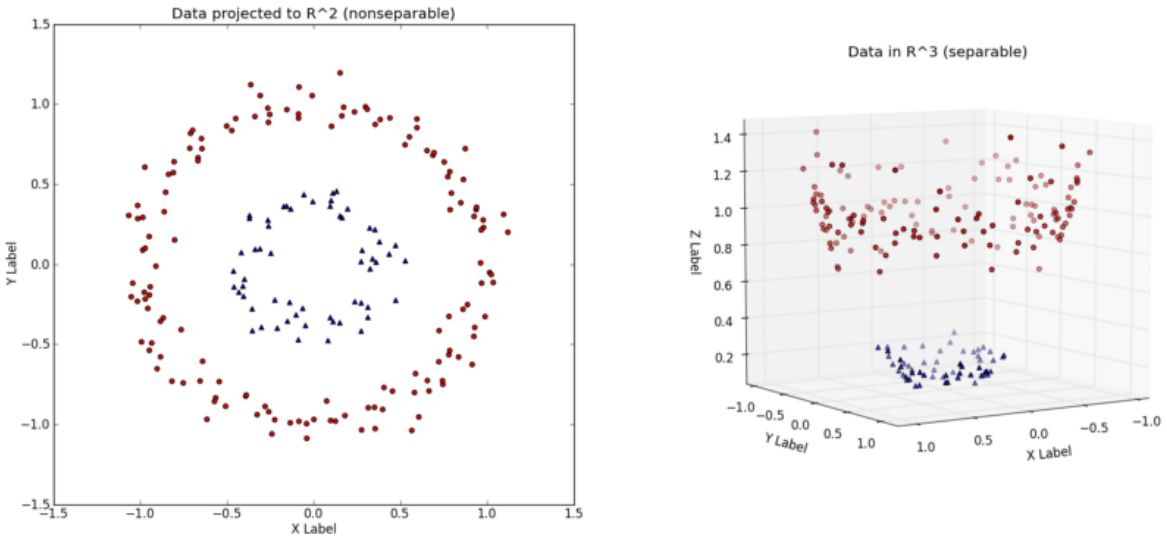

Figura 2.4: Fuente: https://towardsdatascience.com/understanding-the-kernel-trick-e0bc6112ef78

Como ejemplo, en la figura 2.4 se puede ver fácilmente que en los datos originales, en el gráfico de la izquierda, no existe un hiperplano (en $\mathbb{R}^2$ un hiperplano sería una recta) que separe los puntos rojos de los azules. En cambio, al aplicar la función *kernel* que lleva los puntos a $\mathbb{R}^3$, sí existe un hiperplano (un plano en el caso de $\mathbb{R}^3$) que separe los puntos. En esta trabajo, se usa la función *kernel* RBF (radial basis function), que tiene dos parámetros que deben ajustarse para que el clasificador funcione: γ, que define cuánto influye un solo dato de entrenamiento, y C, que es un parámetro de penalización de los errores. Realizar los entrenamientos, se probó con distintos valores de γ en el rango de 0,001 a 0,01, y C desde 1 a 100.

2.3. Word embeddings

El siguiente modelo importante es el de *word embeddings*, el cual codifica el significado semántico de palabras y sus relaciones en vectores n-dimensionales [7]. Así, por ejemplo, si se tiene una función $y = v(w)$, donde w es una palabra o texto e $y \in \mathbb{R}^n$ la representación del texto según el modelo, se puede usar y como entrada para entrenar una red neuronal. `Word2vec`[1] y `fastText`[2] son ejemplos de modelos de *word embeddings* en base a redes neuronales que, a partir de un conjunto de texto como las publicaciones de Instagram, permiten usar los vectores resultantes en redes neuronales o en un clasificador lineal como SVM [8]. Los vectores, o *word embeddings* que se obtienen de estos modelos, sirven para dos propósitos en este trabajo: en primer lugar, para clasificar una publicación a partir del texto de ésta, ya sea usando una red neuronal o una SVM, ignorando de momento la imagen de la publicación. Y en segundo lugar, los *word embeddings* se usan para, en conjunto con los *feature vectors* de los modelos de imágenes, ser la entrada de clasificadores usando redes neuronales o SVM que combinen texto e imágenes de las publicaciones.

[1] https://code.google.com/archive/p/word2vec/
[2] https://fasttext.cc/

2.3.1. FastText

Un modelo relativamente reciente de *word embeddings* es fastText, librería de código abierto que implementa el modelo propuesto el año 2016 con el fin de generar *word embeddings* robustos [1] y que puedan usarse dentro de un clasificador de texto [6]. Entre sus características destacan su velocidad de entrenamiento y la resistencia a palabras desconocidas. fastText, a diferencia de word2vec, genera buenos embeddings incluso para palabras que no había visto durante el entrenamiento. Esta última propiedad se tiene porque fastText no supone que la unidad más pequeña son las palabras, por el contrario, trata las palabras como la combinación de sus caracteres individuales, lo que le permite interpretar palabras que no reconoció durante el entrenamiento pero se parecen a otras que sí reconoce.

FastText provee su modelo pre entrenado en español usando texto recopilado de Wikipedia y CommonCrawl [4].

El modelo fastText incluye dos arquitecturas de entrenamiento para word embeddings, CBOW y Skip-gram. Ambas versiones funcionan tratando de predecir texto en oraciones, pero la diferencia radica en la forma en que lo hacen, CBOW trata de predecir una palabra a través del contexto (las palabras que lleva alrededor), mientras que Skip-gram trata de predecir el contexto a partir de la palabra.

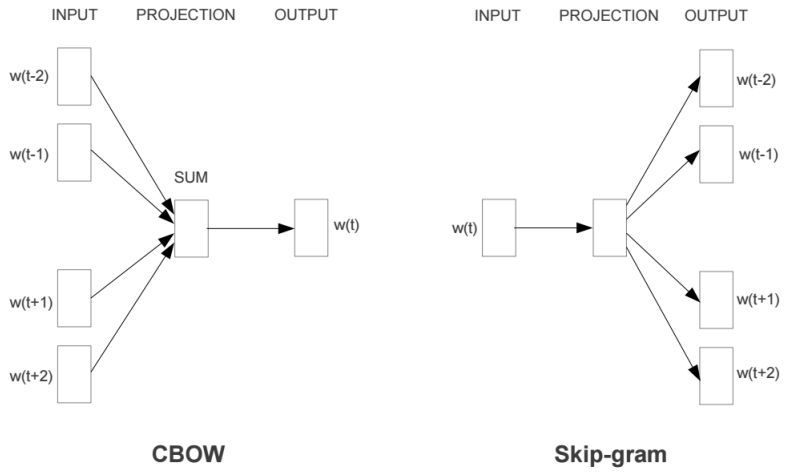

Figura 2.5: Ilustración de las arquitecturas de CBOW y Skip.gram.

2.4. PCA

Análisis de componentes principales (PCA por sus siglas en inglés) es un algoritmo estadístico que busca transformar un primer conjunto de datos de variables posiblemente correlacionadas en un segundo conjunto cuyos datos son de menor dimensión, de forma que las variables, llamadas componentes principales, tengan la mayor varianza posible entre sí. En este caso en particular, el primer conjunto de datos son los *feature vectors* generados

por ResNet a partir de las imágenes de las publicaciones que se desea clasificar. El segundo conjunto, el de datos de menor dimensión, son las mismas imágenes pero comprimidas en vectores de tamaño 128, es decir, son la proyección de las imágenes en un espacio de menor dimensión. Se toma por cierto el supuesto de que no se pierde mucha información, y por lo tanto estos vectores más pequeños mantienen suficientes características de las imágenes como para poder usarse en la clasificación.

El primer paso de PCA, es la normalización de los datos: a cada variable se le resta el valor promedio de esa variable y se divide por su desviación estándar. Luego de normalizar los vectores de ResNet, el conjunto de datos va a consistir en vectores pertenecientes al conjunto $[0, 1]^{512}$. Luego se calcula la matriz de covarianza, matriz de dimensión $p \times p$ ($p = 512$ es la dimensión de los datos), que contiene las covarianzas entre las dimensiones, y se calculan los vectores propios y valores propios de esta matriz. Los vectores se usan para transformar los datos de forma las dimensiones del maximicen la varianza, intuitivamente que cada variable capture más información que las variables originales, y se disminuye la covarianza entre las variables, es decir, el grado de variación conjunta entre las variables.

Este algoritmo permitirá adaptar la salida que se obtiene de una imagen en ResNet para que pueda ser usada como la entrada de la red neuronal que clasifique publicaciones considerando texto e imágenes. Para clasificar usando tanto texto como imágenes, se concatenarán vectores obtenidos mediante ResNet a partir de la imagen con vectores obtenidos mediante fastText, pero los de ResNet son de dimensión 512 y los de fastText 128. Como los vectores provenientes de las imágenes son 4 veces más grandes, se puede usar PCA para convertirlos en vectores del mismo tamaño que los de fastText y así clasificar publicaciones con una red neuronal que reciba como entrada vectores de tamaño 256.

Para poder usar PCA, primero se debe entrenar el modelo con un conjunto de datos de suficiente tamaño, y luego el resultado de este entrenamiento puede usarse con nuevos vectores provenientes de imágenes que se quiera clasificar. Por lo tanto, cuando se entrene ResNet, se usarán las mismas imágenes para entrenar PCA.

2.5. T-SNE

T-Distributed Stochastic Neighbor Embedding es un algoritmo no supervisado que, de forma similar a PCA, se usa como técnica de reducción de dimensión de datos, diseñada con el objetivo de visualizar conjuntos de datos de altas dimensiones [9]. Es importante notar esa diferencia; el objetivo de t-SNE es solamente poder visualizar los datos más fácilmente, y no es una función a partir de la cual se le calculen vectores de menor dimensión a nuevos posibles datos.

Usando t-SNE se podrán validar los modelos previos a la red neuronal para clasificar influenciadores, ya que la salida de estos corresponde a vectores de dimensión 128 o 512, si es fastText o ResNet respectivamente, y sin una técnica como t-SNE es difícil medir que tan útiles serán. Mediante esta reducción, se tomarán los vectores de alta dimensión y se proyectarán en vectores en $\mathbb{R}^2$, lo que permite graficarlos en una forma simple de entender.

De esta forma, se grafican publicaciones previamente clasificadas en un gráfico de puntos, cada punto de un color según la clase a la que pertenezca, y se espera observar agrupaciones o nubes de puntos del mismo color cercanos entre sí. Si esto ocurre, significa que los modelos están generando *feature vectors* que realmente reflejan las características de los datos y que diferencian unas clases de otras, ya que t-SNE está encontrando estas características. Si por el contrario no se forman nubes de puntos del mismo color, probablemente los vectores no son lo suficientemente buenos, ya que no se logran distinguir los de una clase de los de otra.

AL ejecutar el algoritmo, se usan como parámetros la cantidad de iteraciones, tasa de aprendizaje η, que controla cuánto se ajustan los parámetros del algoritmo y la perplejidad, que mide qué tan bien una distribución de probabilidad predice una muestra.

Sumario

Los modelos utilizados en este trabajo son:

- ResNet para clasificar las imágenes y generar *feature vectors* a partir de ellas.
- SVM, para clasificación de texto a partir de *word embeddings* y de imágenes a partir de *feature vectors*.
- PCA, para reducir la dimensión de los *feature vectors* y porder usarlos para clasificar influenciadores.
- T-SNE, para visualizar *feature vectors* y *word embeddings*.

Capítulo 3

Preparación de datos

Para empezar a recolectar los datos, se deben definir todas las categorías a las que pueden pertenecer las publicaciones:

- **Aire libre**: cualquier actividad al aire libre, deportiva o recreacional.
- **Animal**: publicaciones que demuestren afecto por los animales, principalmente mascotas.
- **Arte**: pinturas, esculturas, artesanías, artes plásticas, etc.
- **Belleza**: uso de productos de belleza, maquillaje, cuidado del pelo, etc.
- **Comida**: preparación o consumo de comida.
- **Deporte**: cualquier actividad deportiva. Una publicación de deporte al aire libre, surf por ejemplo, pertenece tanto a esta categoría como a la de aire libre.
- **Familia**: publicaciones con énfasis en relaciones familiares, niños, maternidad y paternidad.
- **Moda**: publicaciones con énfasis en la vestimenta, y particularmente la vestimenta elegante.
- **Música**: cualquier relación con la música.
- **Tatuajes**: personas con tatuajes.
- **Viajes**: viajes vacacionales. Normalmente no se puede identificar una publicación en esta categoría solo con la imagen, se necesita el texto también. Hay un traslape con aire libre.
- **Vida sana**: actividades relacionadas con el bienestar personal, como yoga, pilates y alimentación sana.
- **Videojuegos**: juegos electrónicos como los de PlayStation, computador, etc.

Éstas categorías agrupan a los influenciadores que se consideran en la plataforma, y un subconjunto de los influenciadores ya fueron etiquetados, manualmente, dentro de esas categorías. Con las categorías ya definidas, existen tres posibles tipos de datos; imágenes, texto y publicaciones, que combinan tanto imágenes como texto. Se empieza por la recolección de imágenes sin texto para poder entrenar ResNet, y de datos con texto e imágenes para entrenar el resto de los modelos, mediante las fuentes descritas a continuación.

ImageNet

ImageNet es una base de datos de imágenes organizada jerárquicamente según un subconjunto de la estructura WordNet, la que cuenta con más de 20.000 clases y desde cientos a miles de imágenes por clase, pero considerando solo clases que sean sustantivos. Los datos fueron recopilados especialmente para este tipo de usos y han sido usados durante varios años en competencias de inteligencia artificial.

El problema es que solo están disponibles las url de las imágenes y muchas están obsoletas, pero se descargaron cerca de 300.000 imágenes etiquetadas para entrenamiento, de las cuales se seleccionó un subconjunto apropiado para cada categoría. La forma de descargarlas fue seleccionar una categoría de ImageNet, asociarla a una de las categorías usadas en la memoria, y descargar todas las imágenes de esa clase de ImageNet y también las subclases. Por ejemplo, se descargaron exitosamente 1.020 imágenes dentro de la clase "Domestic animal, domesticated animal" de ImageNet, cuyas urls se encuentran disponibles en `http://www.image-net.org/synset?wnid=n01317541`, en la categoría **animal**, junto con las subclases "Domestic cat, house cat, Felis domesticus, Felis catus" y "Dog, domestic dog, Canis familiari", con 977 y 672 descargas exitosas respectivamente.

La descarga se realizó con un script de Python que, recibiendo el id de la clase, "n01317541" en el ejemplo anterior, descarga las imágenes de la clase y luego recursivamente consulta por cada una de las subclases correspondientes. Al hacer esto, descarga las imágenes en un computador personal y guarda la ubicación de los archivos junto con las categorías en la base de datos. En la figura 3.1 hay 3 imágenes de las descargadas pertenecientes a la categoría **animal**.

Figura 3.1: Imágenes descargadas desde ImageNet.

Pexels

Pexels es página web que contiene imágenes de acceso gratuito etiquetadas por ellos, de forma que se pueden buscar imágenes usando distintas clases. Se buscaron las etiquetas que calzan con las categorías usadas en esta memoria, por ejemplo, al buscar según las

etiquetas “cooking” y “chef” dentro del sitio web, se obtienen imágenes pertenecientes a la categoría “comida” de la memoria. De esta forma, se buscaron se descargaron en torno a 10.000 imágenes pertenecientes a cada categoría. La licencia de las imágenes se puede revisar en `https://www.pexels.com/photo-license/`.

Al igual que para ImageNet, se usó un script de Python que, a partir de un término de búsqueda como “cooking”, descarga automáticamente las imágenes del sitio web a través de las url https://www.pexels.com/search/cooking/?page=<n>, iterando sobre n para consultar múltiples páginas. En la figura 3.3 se ven imágenes descargadas según el tag **cooking**.

Figura 3.2: Publicaciones de Instagram subidas por influenciadores.

Instagram

La tercera fuente de datos es a través de la API de Instagram, la que permite consultar publicaciones por etiqueta, similar a como se hace en Pexels, con la diferencia de que los datos vienen con mucho más ruido porque no hay ninguna restricción en la plataforma de cómo usar las etiquetas, es decir, cualquier usuario puede subir contenido usando cualquier etiqueta, esté relacionada con la publicación o no.

Usando la misma estrategia que en Pexels, se buscaron etiquetas apropiadas para cada categorías, y usando otro script de Python se descargaron publicaciones asociadas a cierta categoría a través de la etiqueta usada en la búsqueda, por ejemplo buscando publicaciones de la categoría “comida” mediante la etiqueta de Instagram “cooking”. De esta forma se descargaron cerca de 180.000 publicaciones. La razón de usar un número tan alto de publicaciones, es que la mayoría de las publicaciones de descartó mediante el proceso descrito en el capítulo 3 y una solo parte pequeña de los datos se usó finalmente.

Figura 3.3: Imágenes descargadas desde Pexels.

Instagram es la única fuente de datos textuales usada, por lo que se deben usar a pesar de que tengan más ruido, y por lo tanto requieren mayor preprocesado para ser usados efectivamente.

Haip

Dentro de Haip se contrató a 2 personas para que clasificaran cerca de 3.000 influenciadores. Las categorías que se usaron no son exactamente las mismas que se usan en esta memoria, más bien son un superconjunto de las usadas. Es importante que a diferencia del resto de los datos clasificados, estos son los únicos en que se clasifica al *influenciador*, y no a ninguna publicación en particular. Por esta razón, estos datos solo se usan al final del trabajo para evaluar el clasificador final que combina tanto imágenes como texto y no para entrenar los modelos.

De los influenciadores clasificados, solo se consideran los que pertenezcan a alguna de las categorías usadas en la memoria, dejando de lado los que pertenezcan, por ejemplo, a "vida social" y "entretenimiento", categorías que no se consideraron por ser más subjetivas y difíciles de trabajar, pero que podrían considerarse en el futuro.

Clasificación manual

Por último, se clasificó para usar como conjunto de validación del modelo de imágenes un total de 1400 imágenes provenientes de las publicaciones, distribuidas equitativamente en cada categoría. Es decir, no se consideró el texto de las publicaciones para esta clasificación.

Además se clasificó un conjunto de 1204 publicaciones, considerando tanto texto como imágenes, para ser usado como validación del modelo de texto e imágenes. A diferencia del conjunto anterior, este se etiquetó usando múltiples categorías, por ejemplo, una publicación puede pertenecer a "belleza" y a "moda" simultáneamente.

Sumario

Las fuentes de datos para el entrenamiento, validación y test de los modelos son:

- ImageNet y Pexels para obtener imágenes etiquetadas. Se mapean las categorías esas fuentes a clases de este trabajo, por ejemplo, imágenes que en Pexels aparecen como "cooking", se usan en la clase "comida".
- Publicaciones de Instagram, única fuente de datos textuales. Bajo el supuesto de que tanto el texto como la imagen pertenecen a la misma clase, se usa el texto para entrenamiento de *word embeddings* usando como clase la predicha según la imagen.
- Cerca de 3.000 influenciadores y 1.400 publicaciones fueron clasificadas manualmente y por separado.

Capítulo 4

Desarrollo de los modelos

A continuación se detalla la realización del entrenamiento de los modelos con datos de las clases “aire libre”, “animal”, “comida”, “deporte” y “moda”.

4.1. ResNet

Se eligió la versión de ResNet de 34 capas que se entrenó a partir de una combinación de imágenes elegidas al azar tanto de las de Pexels como de las de ImageNet. El procedimiento fue mirar las imágenes de Pexels e ImageNet para encontrar las etiquetas de ambos conjuntos de datos que mejor representaran cada categoría, y a a partir de esas juntar 10.000 imágenes por categoría, 50.000 en total. A modo de ejemplo, para “comida” se usaron imágenes de Pexels con las etiquetas “chef”, “cooking” y “food”, e imágenes de ImageNet dentro de las clase “Food, nutrient” y algunas de sus subclases, entre ellas “Soul food” y “Fare”.

Para la programación, se usó la librería PyTorch, la que implementa ResNet en la clase resnet34[1]. A esta clase se le cambió la última, una capa lineal de tamaño 512×1000, 512 el tamaño de salida de la capa anterior y 1000 la cantidad de clases del modelo preentrenado, por una capa lineal con un tamaño 512×5. De esta forma, se cambió solo la última capa al modelo para que clasificara en 5 clases.

Como función de pérdida se usó la función de entropía cruzada, definida en la clase CrossEntropyLoss[2], definida por la fórmula:

$$\text{loss}(x, class) = -\log\left(\frac{\exp(x[class])}{\sum_j \exp(x[j])}\right) \tag{4.1}$$

donde x es el vector de salida de la última capa, en este caso de $x \in \mathbb{R}^5$, $class \in \{0, 1, 2, 3, 4\}$ es la posición en x de la clase correcta y $j = 5$ es la cantidad de clases. La expresión dentro del

[1] https://pytorch.org/docs/stable/torchvision/models.html#torchvision.models.resnet34
[2] https://pytorch.org/docs/stable/nn.html#crossentropyloss

logaritmo, que se interpreta como la probabilidad que el modelo le asigna a la clase correcta, vale 1 si x predice con certeza de 100 % la clase correcta, por ejemplo si $x = [1, 0, 0, 0, 0]$ y $class = 0$, y entonces la pérdida es $\log(1) = 0$ en este caso. Para cualquier otro valor de x, $\text{loss}(x, class) \in (0, +\infty)$

Luego, como optimizador se usó descenso de gradiente estocástico, implementado en la clase SGD[3], con taza de aprendizaje inicial de $0,001$ y decaimiento de $0,1$ cada 7 iteraciones, luego la taza de aprendizaje es $lr = 0,001 \times 0,9^{\lfloor \text{i}/7 \rfloor}$, con i = cantidad de iteraciones, y momentum $\rho = 0,9$.

Con dichos parámetros, en cada iteración del entrenamiento los parámetros p de ResNet se ajustan según las fórmulas:

$$v = \rho * v + g$$

$$p = p - lr * v$$

Donde g es el gradiente calculado a partir de la función de pérdida (4.1).

La red se entrenó por 15 iteraciones. Se siguió entrenando hasta 20 iteraciones, pero entre 13 y 15 dejó de disminuir la función de pérdida estancándose en $0,16$ para el conjunto de test y $0,2$ para el de validación, e igualmente la precisión alcanzó 93 %, por lo que las iteraciones se dejaron en 15. Terminado este proceso, se agregó a los datos de entrenamiento otro conjunto de imágenes de tamaño 10.000 no pertenecientes a ninguna de las 5 categorías, etiquetadas como "otra", las que se seleccionaron al azar de distintas selecciones de Pexels e ImageNet. Con este conjunto de datos extendido a 6 categorías, se repitió el entrenamiento. Ambos entrenamientos se realizaron en el servidor de Orand, demorándose aproximadamente 36 minutos cada uno, y generando como salido 2 objetos de Python serializados en archivos de 82MB cada uno. El rendimiento medido de ambos modelos se puede observar a continuación.

[3]https://pytorch.org/docs/stable/_modules/torch/optim/sgd.html

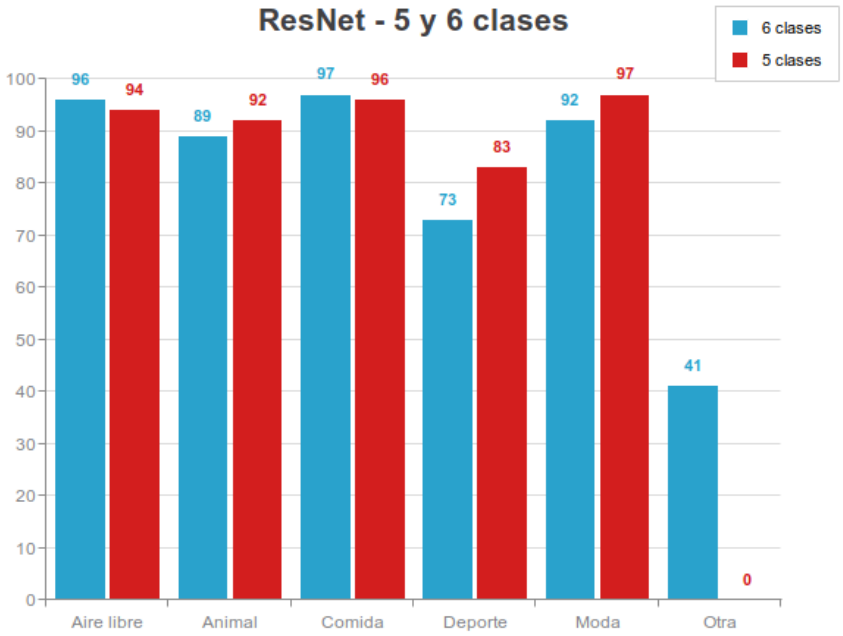

Figura 4.1: Validación de ResNet.

La precisión (porcentaje de imágenes clasificadas correctamente) de ambos modelos es bastante alta, sin embargo se debe considerar que el conjunto de datos solo contiene imágenes no ambiguas, claramente pertenecientes a alguna categoría en particular, muy distinto a los datos que se querrá categorizar más adelante, sin embargo es un buen comienzo.

Por último, para evaluar los vectores generados en la penúltima capa de ResNet (en vez de evaluar la clasificación), se usa t-SNE, que reduce la dimensión de los vectores de 512 a 2 para poder visualizarlos. Se eligen 7.000 publicaciones al azar clasificadas por ResNet, que se ven en la figura 4.2. Las nubes de puntos están claramente separadas; los puntos de una misma clase se encuentran juntos entre si y distantes a los puntos de otras clases. t-SNE no recibe como parte de los datos la clase a la que pertenecen, si no que solo se usa para colorear el gráfico, lo que quiere decir que t-SNE logra distinguir efectivamente los feature vectors de una clase de los de otra, y por lo tanto los vectores codifican correctamente la información de cada clase.

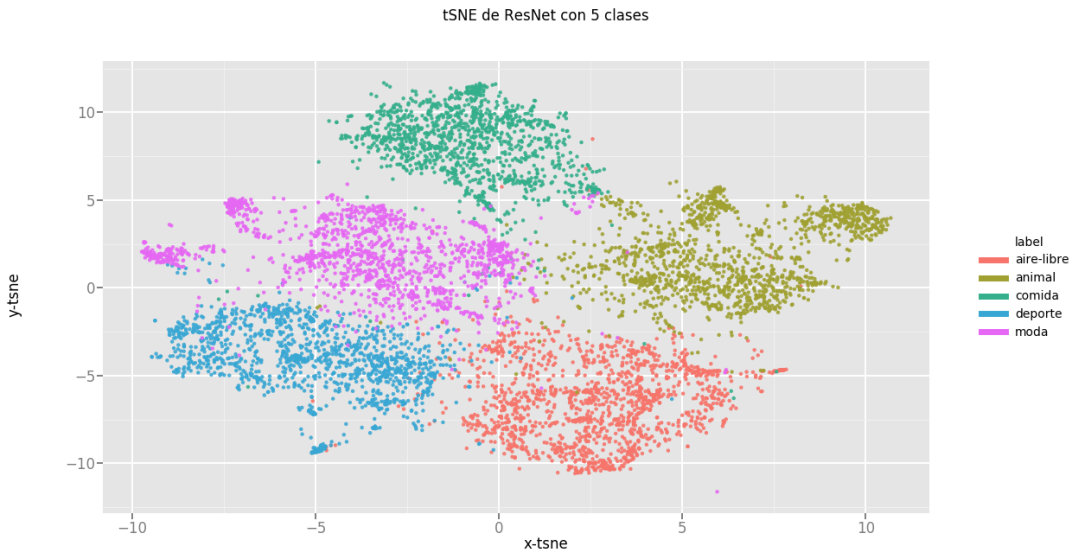

Figura 4.2: T-SNE sobre ResNet de 5 clases.

4.2. FastText

Para entrenar fastText primero usaron los modelos obtenidos mediante el proceso explicado en la sección anterior: se clasificaron 74.029 publicaciones de Instagram obtenidas mediante consultas por etiquetas usando ResNet a partir solo de la imagen, y se marcaron todas aquellas en que coincidió la etiqueta de Instagram con las de ambas versiones de ResNet. Así, por ejemplo si una publicación se obtuvo mediante la etiqueta "maratón" y se marcó como "deporte", se usó en esta sección si y solo si además ResNet de 5 y 6 clases marcaron la imagen de esa publicación como "deporte". En caso contrario esa publicación de marco como "indefinido". De esta forma, 35.751 publicaciones fueron marcadas como "indefinido" y 38.278 dentro de alguna categoría, obteniéndose la siguiente distribución:

Categoría	número de publicaciones
Aire libre	8.238
Animales	7.649
Comida	7.471
Deporte	7.816
Moda	7.104

Tabla 4.1: Distribución de categorías en publicaciones de entrenamiento.

En casi la mitad de las publicaciones no coincidieron las categorías de ResNet y la etiqueta de Instagram, esto porque el uso de estas etiquetas es muy libre, lo que significa que cualquier publicación puede tener cualquier etiqueta. Por eso es que se usan ambos modelos de ResNet para clasificar dichas publicaciones, para tener datos los más limpios posibles al entrenar fastText.

Con las publicaciones a utilizar elegidas, se debe primero limpiar el texto de las publicaciones mediante algunas operaciones sencillas. Se eliminan algunos caracteres, como "!?()", se eliminan los saltos de linea ya que fastText espera cada texto como una sola linea, y se convierte todo el texto a minúsculas. Otra posible operación es remover todos los "#" y "@", que en las publicaciones de Instagram tienen significado extra, ya que los "#" se usan como etiquetas (las que se usaron para buscar publicaciones) y los "@" sirven para menciones, es decir, vínculos a la cuanta de otros usuarios. Los resultados obtenidos fueron levemente mejores sacando "#" y "@", por lo que en las secciones siguientes se optó por este camino.

Por último, se realizaron 3 tipos de pruebas: fastText no supervisado, en la que no se usaron las categorías como parte de los datos, y fastText supervisado, en que la categoría si se usó como parte de los datos, tanto con la versión de *skipgram* como la de *continuous bag of words*. Tanto en el conjunto de entrenamiento como en el de test *skipgram* obtuvo mayor precisión, y se optó por seguir las pruebas solo con éste modelo.

4.2.1. FastText supervisado

Para esta sección, se prepararon 2 archivos a partir del texto de las 38.278 publicaciones obtenidas aplicando las operaciones descritas en la sección anterior, además de agregando las etiquetas al principio del texto de la forma "__etiqueta__". Un texto con 30.000 publicaciones se usó como entrenamiento, y otro de 8.278 como validación. Los primeros 2 textos del archivo de entrenamiento se ve así:

```
__label__moda • + ⭐remes cancheras• 🔥podes reservar con  seña previa . podes
pagar en efectivo /  tranferencia / deposito . . . . indumentaria instamoda
fashion fashiongirl argentina nuevacoleccion ss19 caseros congreso villabosch
villadevoto villadelparque moda ellagritamodaind
__label__comida suculentas hojarascas , si las vez , tienes que probarlas . al
comprar nuestros productos , puedes estar seguro que tu paladar se deleitará
y a su vez , estarás ayudando a diferentes factividades sociales . síguenos y
entérate de este movimiento . suculentas hojarascas hojarascas galletas
cookies yumny regias monterrey artesanales instagood instadiary instafood cafe
coffe love love amor postre merienda like viernes
```

Notar que los *emojis* no se eliminaron, pues son parte importante del texto. Se entrenó por 15 iteraciones, lo que duró aproximadamente 3 minutos en mi computador personal. Al evaluar en los 8.278 textos de validación, se obtuvo 94 % de precisión, aunque hay que notar, al igual que con ResNet, que todos los textos necesariamente pertenecen a alguna de las categorías en particular, lo que no necesariamente se cumple con una publicación al azar de Instagram.

Una vez entrenado el modelo, se obtuvieron los *word embeddings* de las publicaciones, tanto para graficarlos usando t-SNE, como para usarlos más adelante para entrenar la red neuronal final. El resultado de aplicar t-SNE sobre 7.000 *word embeddings* al azar es el siguiente:

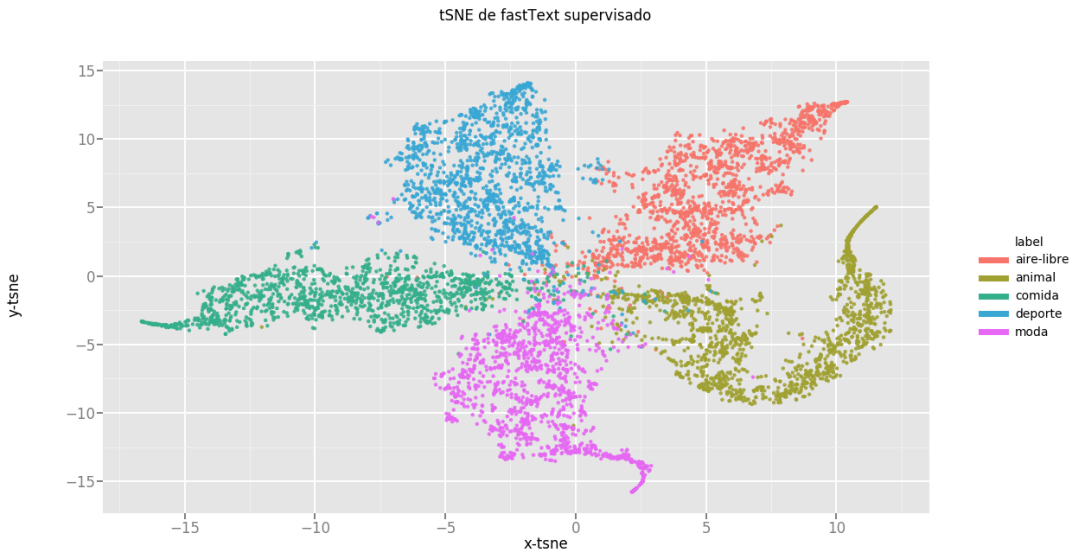

Figura 4.3: T-SNE sobre fastText supervisado.

Se puede observar claramente que t-SNE fue capaz de distinguir las categorías a pesar de que solo vio los *word embeddings*, lo que indica que fastText genera buenos *embeddings*, lo que va a ser fundamental para que la red neuronal más adelante clasifique influenciadores.

4.2.2. FastText no supervisado

Esta sección es idéntica a la anterior, se usa el mismo conjunto de datos, con la única diferencia de que no se usan las clases, y por lo tanto no hay validación ni precisión del modelo. Los primeros 2 textos del archivo sin clases se ve así:

```
• + ⭐remes cancheras• 🔥podes reservar con  seña previa . podes pagar en
efectivo /  tranferencia / deposito . . . . indumentaria instamoda fashion
fashiongirl argentina nuevacoleccion ss19 caseros congreso villabosch
villadevoto villadelparque moda ellagritamodaind
suculentas hojarascas , si las vez , tienes que probarlas . al
comprar nuestros productos , puedes estar seguro que tu paladar se deleitará
y a su vez , estarás ayudando a diferentes factividades sociales . síguenos y
entérate de este movimiento . suculentas hojarascas hojarascas galletas
cookies yumny regias monterrey artesanales instagood instadiary instafood
cafe coffe love love amor postre merienda like viernes
```

Al igual que el caso anterior, la forma de determinar donde termina un texto y empieza el siguiente es por los salto de línea. Nuevamente se grafican los vectores que entrega el modelo luego de aplicar t-SNE a un subconjunto de 7.000 *embeddings* seleccionados al azar.

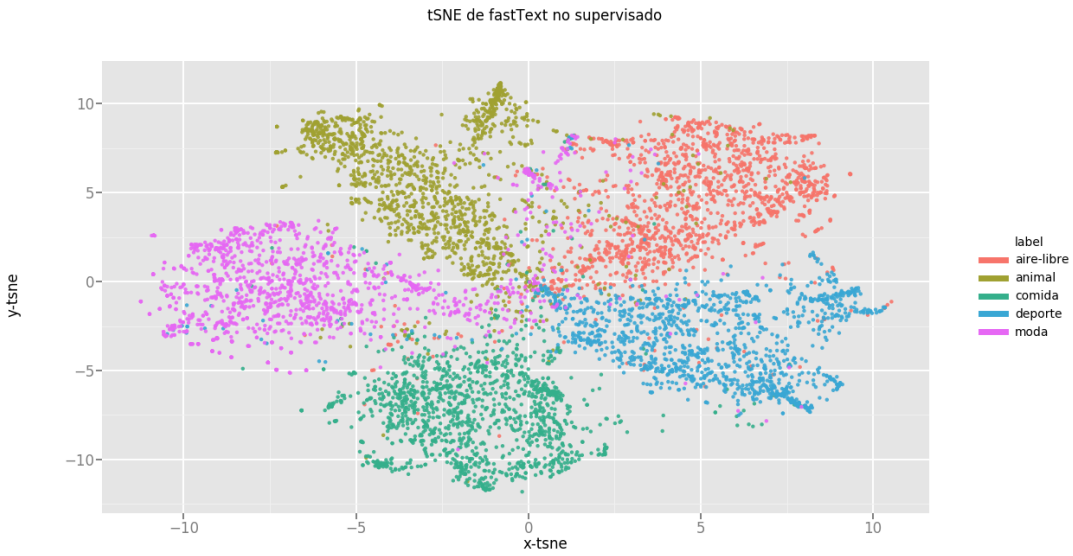

Figura 4.4: T-SNE sobre fastText no supervisado.

Los puntos están más dispersos que en el caso supervisado, pero aún así se notan claramente las clases agrupadas. Es interesante que en este caso, a diferencia del anterioir, ni fastText ni t-SNE pudieron ver a qué categorías pertenecían los textos y *embeddings*, respectivamente.

4.3. Clasificador de texto usando SVM

Además de usar los vectores de fastText no supervisado para t-SNE, se usan para entrenar un clasificador lineal. Esto tiene 2 propósitos: evaluar los vectores (lo que ya se hizo además con t-SNE), y comparar al final la clasificación usando solo texto versus usando texto e imágenes.

Para este clasificador lineal se usa la implementación de SVM de scikit-learn, que provee la clase SVC[4], con el kernel RBF y los parámetros $\gamma = 0{,}002$, $C = 90$. Se seleccionan al azar 35.000 de las 38.278 publicaciones que se usaron previamente, y se le pasan al SVM para que las use como conjunto de entrenamiento. Luego de evaluar en los 3.278 textos restantes, correspondientes al conjunto de test, se obtiene una precisión de 94 %. De nuevo, hay que considerar que la precisión es mejor en un conjunto de datos en que cada publicación pertenece a exactamente una de las clases que al tomar textos de publicaciones al azar.

[4]https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

4.4. Clasificación de publicaciones con red neuronal I+T

El último modelo que se entrena es una red neuronal que usa simultáneamente las imágenes y el texto de las publicaciones. Al igual que los casos anteriores, es necesario manipular los datos que se tienen disponibles para poder usarlos. Esta vez, no se usan directamente los datos de las publicaciones, si no que se usa la salida tanto de ResNet como de fastText supervisado y no supervisado. Esto se hace según el siguiente procedimiento: primero, se usa ResNet para obtener un vector de tamaño 512 a partir de la imagen, y se le aplica PCA para reducirlo a otro de tamaño 128. En segundo lugar, se limpia el texto de la publicación (convertir a minúsculas, etc.), y se usa fastText para obtener otro vector de tamaño 128. Finalmente, se concatenan ambos vectores y se obtiene así la entrada de la red I+T, de tamaño 256. Se usan las 38.278 publicaciones que se han estado utilizando para entrenar como observaciones iniciales de PCA, obteniendo así una función PCA : $\mathbb{R}^{512} \to \mathbb{R}^{128}$ para reducir la dimensión de la salida de ResNet.

Se realizarán 2 entrenamientos independientes de la red I+T, uno usando la concatenación del vector que es salida de PCA con la salida de fastText supervisado, y otro con la salida de fastText no supervisado, recibiendo como entrada un vector de tamaño 256. En ambos casos la arquitectura es la siguiente:

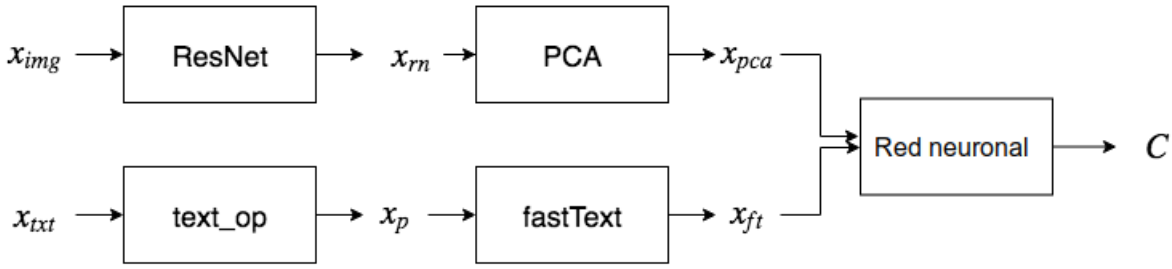

Figura 4.5: Arquitectura de la red neuronal.

En el diagrama, x_{img} es la imagen de una publicación de Instagram, $x_{rn} \in \mathbb{R}^{512}$ es la salida de la última capa de ResNet, $x_{pca} \in \mathbb{R}^{128}$ es la salida de PCA, x_{txt} es el texto de la misma publicación, x_p es el texto preprocesado con las operaciones previamente descritas (sacar saltos de linea, convertir todo a minúsculas, etc.), $x_{ft} \in \mathbb{R}^{128}$ es la salida de fastText, y $C \in \{$aire libre, animales, comida, deporte, moda$\}$ es la categoría predicha.

Siguiendo la estructura del diagrama, se usaron las 38.278 publicaciones para entrenar la red neuronal I+T. Al igual que en ResNet, se usó descenso de gradiente estocástico y entropía cruzada como función de pérdida, y se entrenó por 25 iteraciones, lo que demoró poco menos de 1 minuto en el servidor de Orand. A pesar de lo rápido del entrenamiento, se decidió no aumentar las iteraciones, o épocas, porque no se observaron mejores resultados, y hacerlo podría haber aumentado el riesgo de sufrir sobreajuste.

4.5. Clasificación de publicaciones usando una SVM

Como segunda alternativa de clasificación, se puede usar una SVM en vez de una red neuronal que tome como entrada la concatenación de los vectores que se obtienen a partir de ResNet y fastText. Conceptualmente, la idea es exactamente que la descrita en la sección anterior, solo que cambia el modelo usado para clasificar, por lo tanto sigue el mismo esquema de aplicar ResNet en conjunto con PCA para imagenes, y aplicar fastText para texto. El esquema es el siguiente:

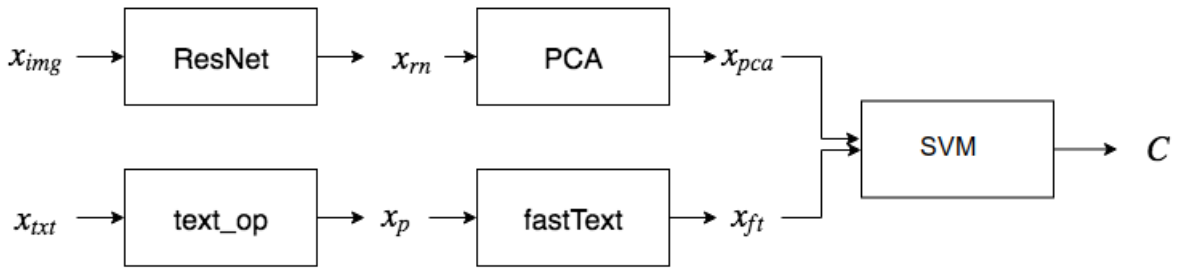

Figura 4.6: Arquitectura de la SVM.

La elección de usar SVM o red neuronal depende de la que obtenga una mayor precisión cuando se evalúen los resultados.

4.6. Clasificación de publicaciones ponderando las predicciones de ResNet y fastText

En ambos modelos de clasificación de publicaciones usando red neuronal o SVM, tanto en ResNet como en fastText se usa como salida de estos modelos un vector que describe al *input*, la imagen o el texto, pero además sirven por si solos como clasificadores entregando como salida una probabilidad de que cada publicación pertenezca a cada categoría. Esto permite una tercera alternativa de clasificación de publicaciones ponderando ambos valores por una constante λ. Dicho de otra forma, si para una categoría c la probabilidad de que la imagen de una publicación sea de esa categoría es $I(c)$ y la probabilidad de que el texto pertenezca a la categoría es $T(c)$, se puede estimar la probabilidad $p(c)$ de que la publicación, considerando tanto imagen como texto, pertenezca a c como $p(c) = T(c) * \lambda + I(c) * (1 - \lambda)$, usando un $\lambda \in [0, 1]$ apropiado. A partir de esto, se clasifica una publicación eligiendo la clase $c = \arg\max_{c_i} p(c_i)$.

Para aplicar este método, se debe encontrar un λ apropiado. Para eso se usa el conjunto de 702 publicaciones manualmente clasificadas, y se elige el λ que maximice la precisión.

A modo de ejemplo, supongamos que a la imagen de una publicación ResNet asigna las probabilidades [0,78, 0,01, 0,006, 0,104, 0,1] y al texto fastText le asigna [0,35, 0,55, 0,03, 0,02, 0,05],

donde cada posición en los vectores representa la probabilidad de pertenecer a **aire libre**, **animal**, **comida**, **deporte** y **moda** respectivamente. Entonces, si te toma $\lambda = 0{,}8$, es decir se le da a la imagen una ponderación del 80 %, las probabilidades combinadas son $p = [0{,}694, 0{,}118, 0{,}0108, 0{,}0872, 0{,}09]$, y por lo tanto dicha publicación se clasifica como **aire libre**, que es la clase correspondiente a la primera posición en p y tiene el mayor valor. Siguiendo este procedimiento, se prueba con distintos λ entre 0 y 1 evaluando la precisión en las 702 publicaciones, y se toma el que la maximice.

4.7. Clasificación de influenciadores

Las 2 estrategias recién descritas permites clasificar publicaciones, sin embargo el objetivo final no es ese, si no más bien clasificar influenciadores. Para esto, se clasifican todas las publicaciones disponibles del influenciador que se quiera clasificar, y se consideran como categorías del influenciador aquellas categorías en las que hayan suficientes publicaciones dentro de dicha categoría, de forma que superen un umbral que se define a partir de los resultados. Aquí se puede usar tanto la clasificación de publicaciones según la red neuronal, o bien ponderando las predicciones de ResNet y fastText. Simplemente se usa la alternativa que se tenga mejores resultados en los conjuntos de datos de prueba.

En el caso de que se clasifica usando la red neuronal, como se tienen 2 versiones de la red, una a partir de fastText supervisado y otra a partir de fastText no supervisado, se usan ambas para clasificar. Una publicación se clasifica en cierta categoría solo si ambas redes neuronales predicen la misma categoría, y en caso contrario simplemente se considera desconocida la categoría de una publicación, ya que es preferible tener menos publicaciones clasificadas que tener publicaciones incorrectamente clasificadas. Observando los resultados se observa que a pesar de que no todas las publicaciones terminen clasificándose, sí se clasifican suficientes como para saber las categorías del influenciador.

Con esta estrategia, puede ocurrir que un influenciador tenga publicaciones clasificadas en cierta categoría sin pertenecer a ella. Esto corrige 2 dificultades de la clasificación: primero el error de la red neuronal que podría atribuirle a un influenciador una categoría incorrecta, y segundo el hecho de que puede publicar dentro de una categoría y no pertenecer a ella si es que solo es una proporción muy baja de sus publicaciones pertenece a esa categoría.

De esta forma, si ocurre que un influenciador no tiene suficientes publicaciones clasificadas como para ser clasificado, se interpreta o bien como que no pertenece a ninguna de las categorías consideradas, o bien como que sus categorías son desconocidas.

Sumario

- Se procede a clasificar las publicaciones de tres formas distintas: usando las imágenes, usando el texto, y usando ambos. A su vez, la clasificación usando ambos tipos de datos se divide en: usando un clasificador lineal, una red neuronal, o ponderando las predicciones de la clasificación por imagen y por texto.

- Se usa ResNet tanto para clasificar las imágenes por si solas, como para generar los *feature vectors* que se usan en el clasificador lineal y en el de red neuronal para clasificar combinando texto e imagen.
- Se usa fastText, variantes supervisado y no supervisado, para clasificar los textos de las publicaciones, y para generar los *word embeddings* para la clasificación usando texto e imágenes.
- Usando la concatenación de los *feature vectors* reducidos con PCA, y de los *word embeddings*, se clasifican las publicaciones usando, por separado, una SVM y una red neuronal.
- Además, se clasifica la publicación ponderando los porcentajes que fastText y ResNet le atribuyen a cada clase.
- Se decide que un influenciador pertenece a una clase si al menos una parte de sus publicaciones pertenecen a la clase, usando alguna forma de clasificación para publicaciones. El umbral a partir del cual se decide que pertenece a la clase se encuentra a partir de los influenciadores clasificados.

Capítulo 5

Resultados

La evaluación de los resultados se dividió en etapas; primero se analizan los resultados de clasificar solo las publicaciones, y después la clasificación de los influenciadores (considerando un subconjunto de sus publicaciones). En ambos casos, se usó como métrica de evaluación el *recall*, r, que se define, por clase, como la cantidad de elementos (publicaciones o influenciadores, según corresponda) correctamente clasificados como pertenecientes a la clase, dividido por la suma entre elementos correctamente clasificados como pertenecientes a la clase y elementos incorrectamente clasificados como no pertenecientes. La segunda métrica es *precision*, p, que es la cantidad de elementos correctamente clasificados como pertenecientes a la clase, dividido por la suma entre elementos correctamente clasificados como pertenecientes a la clase y elementos incorrectamente clasificados como pertenecientes a la clase. De forma más simple:

$$r = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, p = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}$$

donde TP son los verdaderos positivos, es decir el clasificador predijo correctamente que el elemento *si* pertenece a la clase c, FN son los falsos negativos, es decir el clasificador predijo incorrectamente que el elemento *no* pertenece a la clase c, y FP son los falsos positivos, es decir el clasificador predijo incorrectamente que el elemento *si* pertenece a la clase c.

Una última métrica útil para evaluar la clasificación de influenciadores es la pérdida de *hamming*, definida por la siguiente fórmula.

$$\mathrm{hloss}(h) = \frac{1}{p * j} \sum_{\mathrm{i}=1}^{p} |h(X_{\mathrm{i}}) \Delta Y_{\mathrm{i}}|$$

donde h es el clasificador que se está evaluando, p es la cantidad de datos, j es la cantidad de clases, X_{i} es el influenciador i e Y_{i} las categorías a las que pertenece el influenciador i. Luego, $|h(X_{\mathrm{i}}) \Delta Y_{\mathrm{i}}|$ es el tamaño del de la diferencia simétrica entre $h(X_{\mathrm{i}})$ e Y_{i}, es decir, la cantidad de veces que un par influenciador-categoría es asignado erróneamente. La pérdida de

hamming mide cuantas veces ocurre esto a lo largo del conjunto de datos, y debiera ser lo más cercano a 0 posible. Si se clasifica perfectamente vale exactamente 0 y si a cada influenciador se le clasifica exactamente en las clases a las que no pertenece, vale 1.

5.1. Clasificación de publicaciones

En esa sección, se muestran los resultados de los modelos sobre un conjunto de 702 publicaciones de Instagram, clasificadas a mano en exactamente una de las clases "aire libre", "animal", "comida", "deporte" o "moda". Primero se analiza el caso de usar solo el texto de las publicaciones, después usar solo la imagen, y finalmente combinando ambos datos, tanto usando los vectores en red neuronal y SVM como ponderando las predicciones.

Se espera que combinando los datos de texto e imágenes la precisión sea lo tan alta como para poder usar el clasificador de publicaciones para después poder clasificar influenciadores.

5.1.1. Usando solo el texto

El texto por si solo no es suficiente para clasificar las publicaciones. Intuitivamente, esto es de esperarse porque al observar las publicaciones, frecuentemente ocurre que el texto refleja emociones, pero no necesariamente hay una relación directa con el contenido, como sí ocurre con la imagen de la publicación. En la figura 5.1 se ve tanto la precisión como el *recall* si es que se usa una red neuronal entrenada a partir de los vectores que entrega fastText.

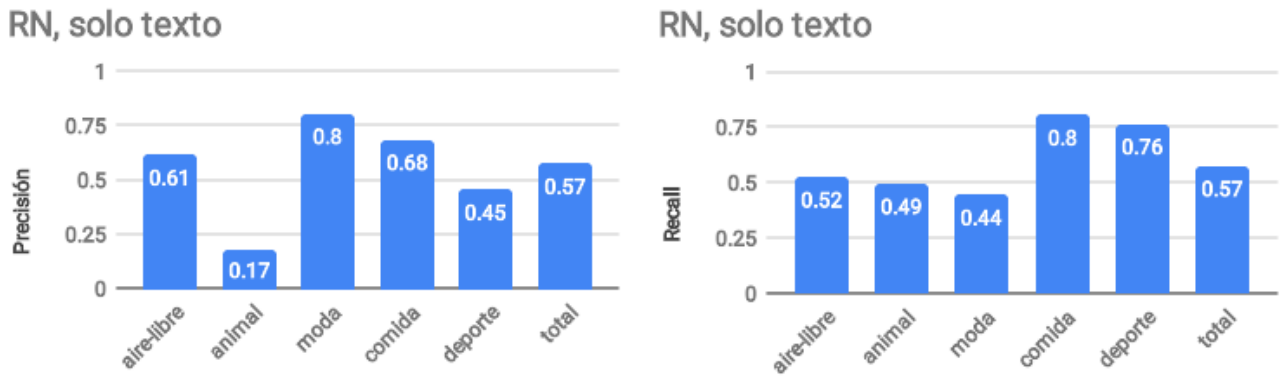

Figura 5.1: Clasificación usando red neuronal a partir de fastText.

Si en cambio se una una SVM a partir de los vectores de fastText, los resultados que se muestran en la figura 5.2 muestran una disminución en la precisión y el recall. Esto indica que independiente del modelo que se elija, SVM o red neuronal, el texto por si solo no es suficiente para clasificar.

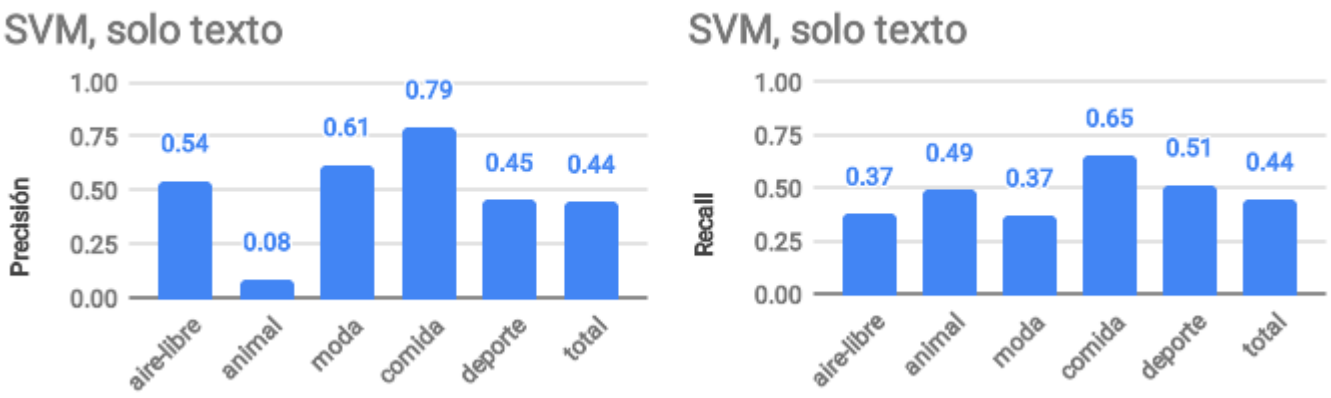

Figura 5.2: Clasificación usando red neuronal a partir de fastText.

5.1.2. Usando solo las imágenes

Las imágenes por si solas demuestran ser bastante más efectivas que el texto, alcanzando un 23 % más de precisión que el texto si se usa una red neuronal, como se observa en la figura 5.3. La diferencia en este caso entre usar como clasificador una SVM o red neuronal no es significativa, como se ve comparando las figuras 5.3 y 5.4. Una diferencia que sí es más importante entre usar SVM y red neuronal es en la clase "comida", en la que la SVM tiene precisión particularmente baja.

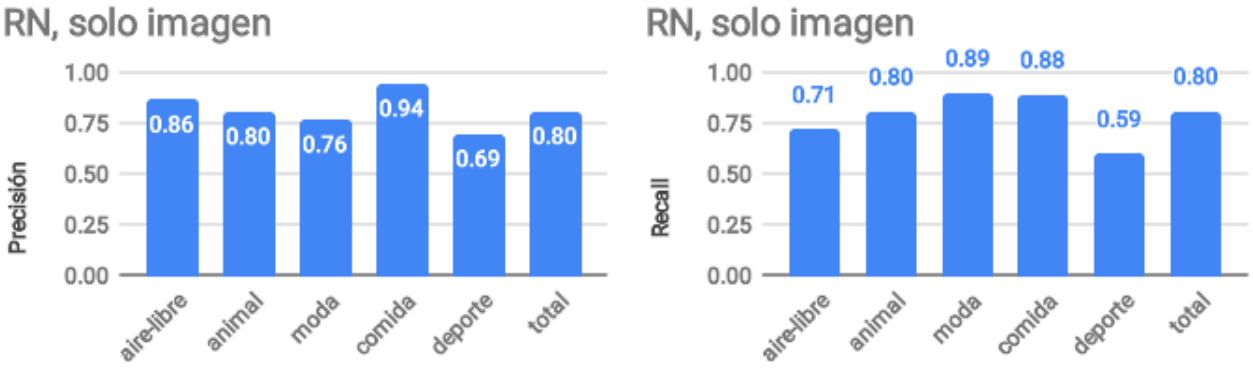

Figura 5.3: Clasificación usando red neuronal a partir de ResNet.

Teniendo los resultados por separado de clasificar usando texto y clasificar usando imágenes, el siguiente paso es ver qué ocurre cuando ambos se toman en cuenta.

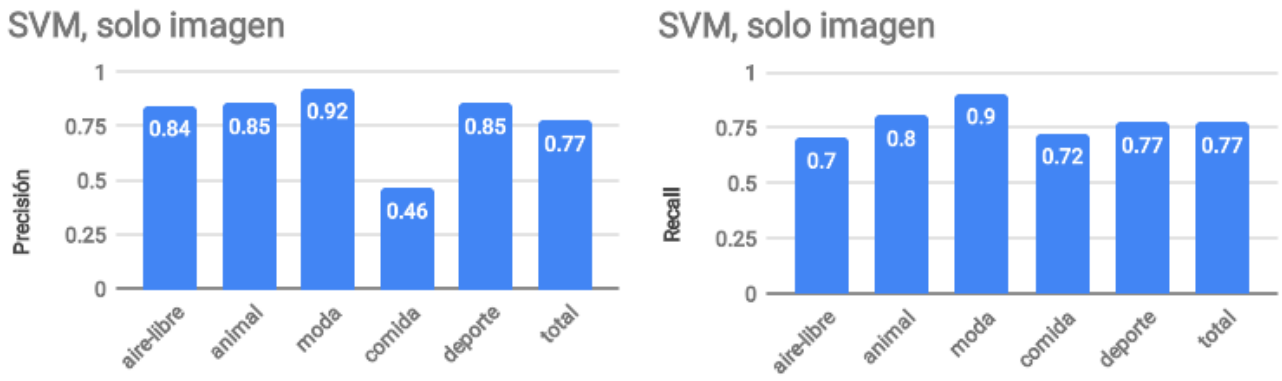

Figura 5.4: Clasificación usando SVM a partir de ResNet.

5.1.3. Usando texto e imágenes

Tomando en cuenta los resultados de clasificar usando el texto por si solo y usando la imagen por si sola, lo importante ahora es analizar si usar ambos permite mejorar los resultados obtenidos con imagen por si sola. Esto se hace de 2 formas por separado: ponderando las predicciones de ResNet y de fastText, y con una red neuronal; ponderar las predicciones es el camino más simple, y por lo tanto el que se analiza primero, mientras que agregar una red neuronal sobre los 2 modelos que ya se usan le agrega complejidad a la solución, pero que puede aumentar la precisión de manera considerable. Si la precisión no mejora, quiere decir que no tiene sentido usar el texto como predictor y es mejor usar solamente la imagen.

Para combinar texto e imágenes, se usan los modelos individuales, lo que se puede hacer de varias maneras. Por ejemplo, para calcular la probabilidad de que un texto pertenezca a una categoría, se puede usar el clasificador de red neuronal o el de SVM a partir de los *embeddings* de fastText, y lo mismo para las imágenes y ResNet. En este caso, se opta por partir de los modelos de texto e imágenes que funcionan mejor en vez de intentar probar con todas las combinaciones.

Combinar texto e imágenes ponderando predicciones

Este es el método descrito en la sección 4.6, es decir, se calcula la probabilidad de que una publicación pertenezca a una clases c como $p(c) = I(c) * \lambda + T * (1 - \lambda)$. Para encontrar un λ adecuado, se calculó la precisión obtenida en las publicaciones de prueba para todos los valores de λ entre 0 y 1 con saltos de 0,01, es decir, $\lambda = 0\,\%, 1\,\%, ..., 99\,\%, 100\,\%$, que se pueden ver en la figura 5.5. Para $\lambda = 0$, el método es equivalente a solo usar el texto, pues se pondera por 100 % del texto y 0 % de la imagen, por lo que da el mismo valor que en el caso de usar texto con SVM. Con $\lambda = 1$ ocurre exactamente lo contrario, corresponde a ponderar por 100 % de la imagen.

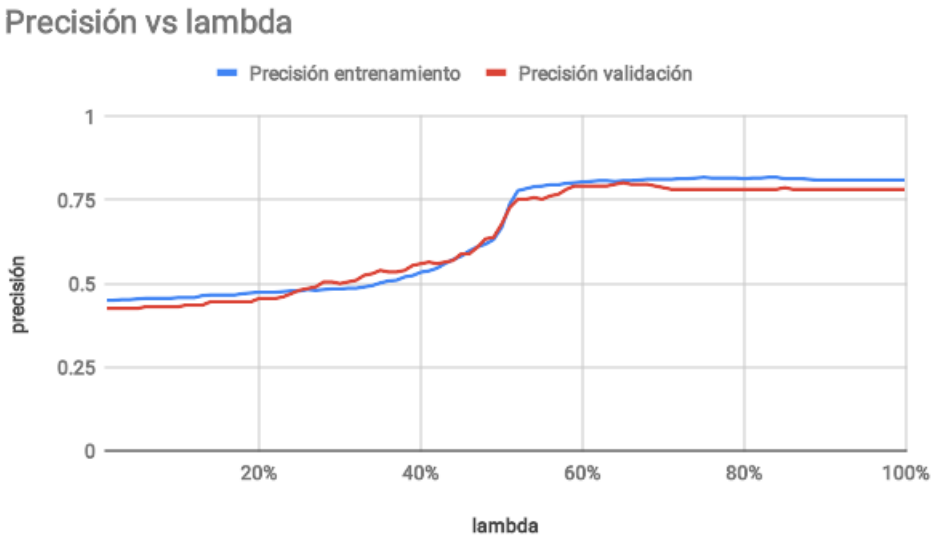

Figura 5.5: Precisión al ponderar imagen y texto usando distintos valores de λ.

Se puede ver que cuando la ponderación pasa del 50 % a la imagen, ésta toma el mayor peso e inmediatamente se mantiene más o menos constante, lo que quiere decir que de esta forma el texto aporta muy poco. El máximo se alcanza en $\lambda = 0{,}82$, pero es apenas un poco mejor que usar solo la imagen, como se ve en los resultados de la figura 5.6, y no es una alternativa que valga la pena.

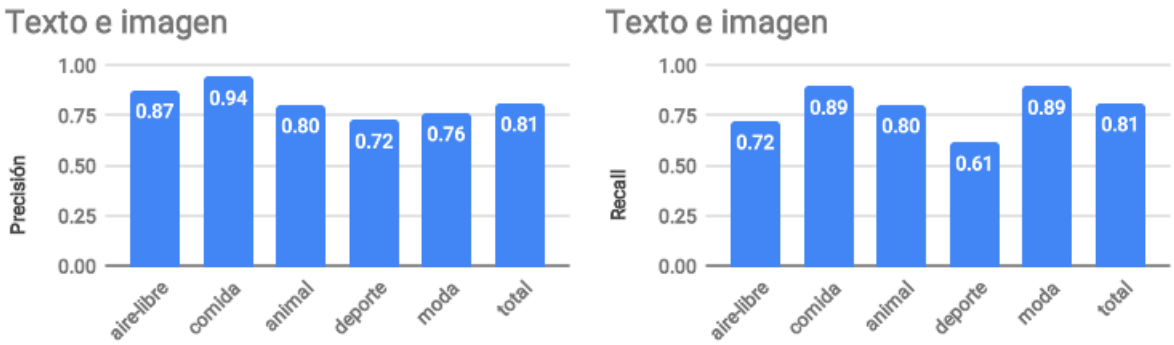

Figura 5.6: Precisión y recall por clase al ponderar imagen y texto usando $\lambda = 0{,}82$.

Combinar texto e imágenes en red neuronal

En la figura 5.7 se ven los resultados de la clasificación usando una red neuronal que toma como entrada la concatenación de los vectores de ResNet y de fastText. Aumenta de 80 % usando solo la imagen a 83 %, lo que es una diferencia suficiente como para considerar el texto. Además en la figura 5.8 se compara por clase, y lo que ocurre es que se mantiene una diferencia similar en cada clase.

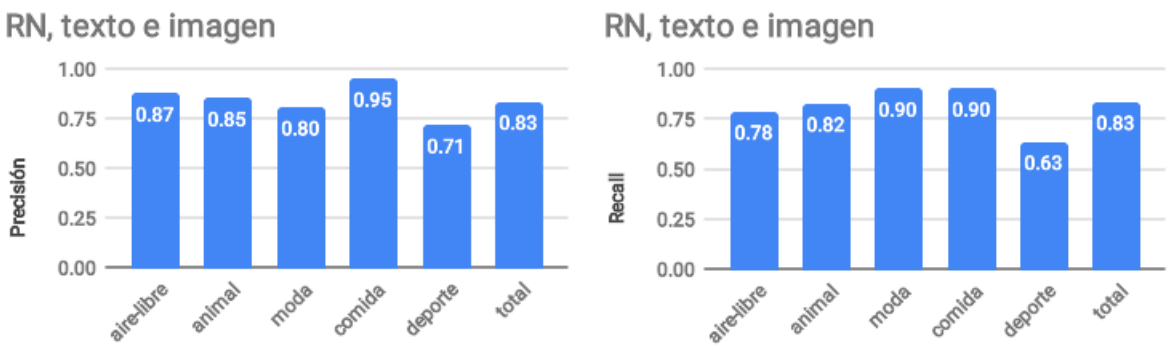

Figura 5.7: Precisión y recall por clase usando red neuronal.

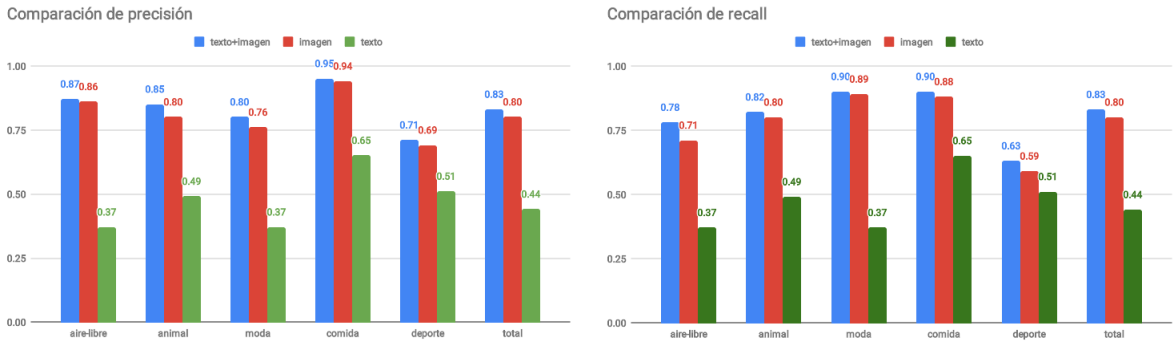

Figura 5.8: Comparación entre texto solo, imagen solo y ambos.

5.2. Clasificación de influenciadores

Con este último modelo, el de red neuronal, es que se clasificaron las publicaciones de los influenciadores para así clasificar a los mismos. Para decidir qué porcentaje de sus publicaciones tiene que pertenecer a una clase para que el influenciador pertenezca a la clase, un umbral de decisión, se observa en la figura 5.9 el efecto que tiene el umbral en la precisión y el recall; si el umbral es 0 % de las publicaciones, la precisión es muy baja y el recall es alto, porque no hay falsos negativos (ya que no hay negativos). En el otro extremo, si se toma un umbral del 100 %, es decir, un influenciador pertenece a una clase c solo si todas sus publicaciones de clasificaron como c, el recall es 0 pero la precisión es casi 1, ya que no hay positivos y por lo tanto tampoco hay falsos positivos. El equilibrio se alcanza cuando se decide que un influenciador está en la clase c si al menos 28 % de sus publicaciones son de esa clase, es decir, en ese punto la precisión y el recall son iguales. En la figura 5.10 se ve mejor el tradeoff, ya que al aumentar uno necesariamente disminuye el otro.

Si bien esta elección del umbral permite elegir entre una mejor precisión, que lo que hace es castigar los falsos positivos, y un mejor recall, que castiga falsos negativos, desde el punto de vista del contexto en el que se encuentra este trabajo, se considera como peor un falso positivo que un falso negativo, entonces es más importante priorizar la precisión que el recall.

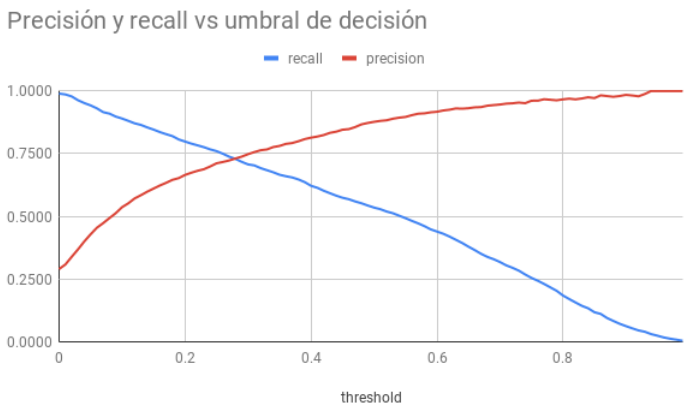

Figura 5.9: Efecto del umbral de decisión en la precisión y el recall.

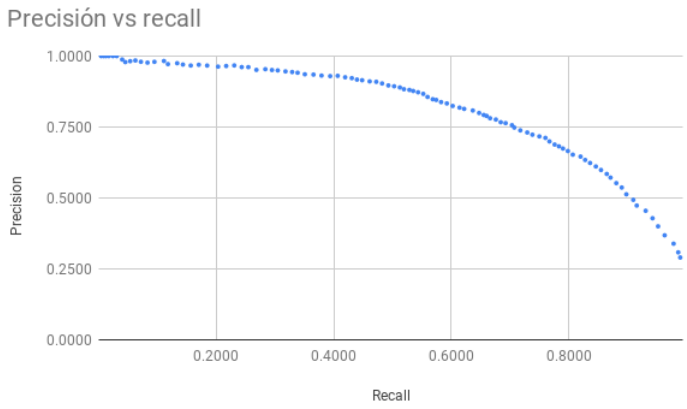

Figura 5.10: Tradeoff entre precisión y recall.

Si en vez de tratar de maximizar el mínimo entre recall y precisión, que se logra cuando el umbral es 28 % se busca optimizar la función de pérdida de *hamming*, se alcanza cuando el umbral es 39 % y se obtienen los resultados de la figura 5.11, y un mínimo de pérdida de *hamming*:

$$\text{hloss}_5 = 0,221$$

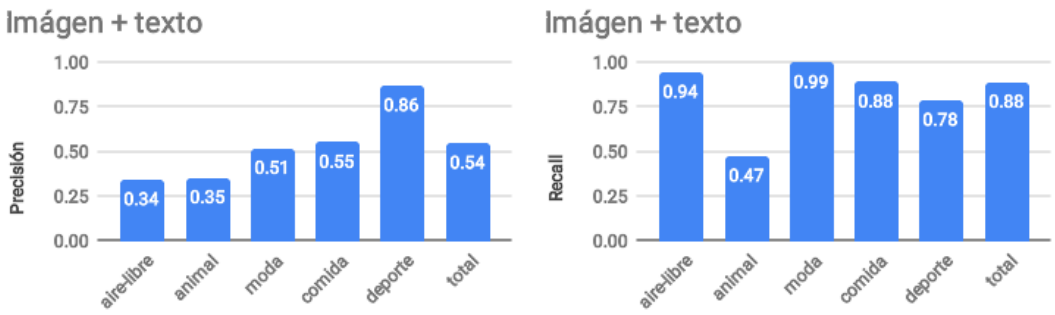

Figura 5.11: Tradeoff entre precisión y recall.

5.3. Clasificación con 8 y 13 categorías

Se intentó obtener resultados cercanos a los obtenidos con 5 clases pero clasificando con 8 y 13, sin embargo los resultados fueron bastante más bajos. Se repitió el entrenamiento según el mismo procedimiento que para 5 clases, pero con mayor cantidad de datos y categorías, y se obtuvieron resultados mucho menos útiles. Se evaluó la red neuronal entrenada para 8 categorías frente a un conjunto de 100 influenciadores:

$$\text{hloss}_8 = 0,501$$

Y el porcentaje de influenciadores para los que se predijo al menos una categoría de forma correcta fue de apenas $11,7\,\%$ en este caso.

5.4. Prototipo

Como prototipo, se clasificaron a los influenciadores de la base de datos a partir de los datos ya presentes, y se guardó la clasificación para poder ser revisada consultando directamente en la base de datos.

Por ejemplo, algunos se los resultados se pueden consultar en el siguiente formato:

```
[
  {
    "username": "pancha_lara",
    "categories": {
      "nn": ["deporte"]
    }
  },
  {
    "username": "hardessenjavi",
```

```
    "categories": {
      "nn": ["aire-libre", "moda"]
    }
  },
  {
    "username": "jansport_cl",
    "categories": {
      "nn": ["moda"]
    }
  }
]
```

Capítulo 6

Conclusiones

El objetivo de esta memoria radica en poder clasificar automáticamente influenciadores, es decir usuarios de Instagram con miles de seguidores, para agilizar las operaciones dentro de Haip. Dicho objetivo se cumplió con las categorías "aire libre", "animal", "comida", "deporte" y "moda" con precisión suficiente, por lo que el modelo será útil a pesar de funcionar con pocas categorías.

Los modelos usados demostraron ser altamente eficientes, y clasificar a un influenciador toma menos de un minuto. Considerando que la cantidad de influenciadores nuevos que ingresar a la plataforma es relativamente baja, el tiempo que toma es suficientemente bueno. Como habría de suponerse observando las publicaciones de los influenciadores, las imágenes aportan muchas más información para poder predecir las categorías, ya que el texto muchas veces es muy corto, o es muy difícil de interpretar sin ver la imagen, sin embargo demostró ser útil al aumentar la precisión en casos en que la imagen no era suficiente y el texto sí podía aportar información extra. Para combinar la información del texto y de la imagen, usar las probabilidades obtenidas desde ambos modelos, ResNet y fastText, no demostró ser efectivo; sí funcionó usar los *feature vectors* y *word embeddings* obtenidos a partir de los respectivos modelos, entrenando con ellos otra red neuronal.

Para poder evaluar la clasificación, no solo en términos de precisión y otras métricas, se clasifican a todos los influenciadores de la plataforma, de forma que los usuarios puedan ver los resultados y se pueda recibir una evaluación de parte de éstos. La clasificación en 5 categorías demuestra ser precisa y útil.

Para lograr que funcione el entrenamiento, fue fundamental concentrar gran parte del trabajo en preparar, descargar y limpiar suficientes datos, lo que tiene la dificultad de que muchos datos no son aptos para usarse por incluir mucho ruido. Tomando en cuenta que no es factible juntar una cantidad alta de datos clasificados uno a uno, son fundamentales fuentes de datos como ImageNet. Para datos textuales, es aún más difícil, en parte porque importa el idioma y la mayoría de las fuentes de datos que se encuentran están en inglés, por lo que es importante tener un proceso con el que generar datos confiables.

6.1. Trabajo futuro

Se puede extender el trabajo realizado en la memoria para aumentar la cantidad de clases en las que se clasifica a los influenciadores. Aumentar la cantidad de clases implica nuevas dificultades que no están tan presentes en un principio. Por ejemplo, se vuelve difícil distinguir entre clases similares, como es el caso de "deporte" y "aire libre", sin embargo son dificultades que se pueden superar.

6.1.1. Cantidad de datos

Un primer paso al aumentar la cantidad de clases es aumentar también la cantidad de datos de entrenamiento, no solo agregando datos de las nuevas clases, si no que obteniendo más dato por cada clase. Una dificultad importante de esta estrategia es que el entrenamiento de los modelos se vuelve costoso rápidamente. No es solo entrenar como tal, si no que, para la red que usa texto e imágenes, se vuelve costoso pre calcular los *word embeddings* y los vectores de ResNet a partir de imágenes.

6.1.2. Ajustes en los modelos

Tanto ResNet, como fastText y los SVM se pueden ajustar cambiando sus hiperparámetros. Se puede probar con distintos valores de, por ejemplo, taza de aprendizaje, otro optimizador distinto a descenso de gradiente estocástico, entre otras variaciones. Esta alternativa es menos costosa que la anterior, pero difícilmente puede ser lo suficiente efectiva y los más probable es que de todas formas deba ser acompañada con un incremento en los datos de entrenamiento.

El trabajo hecho definitivamente se puede extender añadiendo clases a las que actualmente tiene. Idealmente también debería considerar clases como "familia" y "vida sana", lo que no se alcanzó a lograr en este trabajo.

Capítulo 7

Bibliografía

[1] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

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

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

[4] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *ArXiv e-prints*, December 2015.

[6] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[7] L. Kerem Senel, I. Utlu, V. Yucesoy, A. Koc, and T. Cukur. Semantic Structure and Interpretability of Word Embeddings. *ArXiv e-prints*, November 2017.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, January 2013.

[9] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

Capítulo 8

Anexos

Etiquetas usadas para buscar publicaciones en Pexels

- **Aire libre**: camping, climbing, hiking, outdoor, park.
- **Comida**: chef, cooking, food
- **Animal**: no se Pexels para esta categoría.
- **Deporte**: athlete, crossfit, exercise, fitness, gym, jogging, marathon, sport, workout.
- **Moda**: fashion, suit

Etiquetas usadas para buscar publicaciones en ImageNet

- **Aire libre**: Campsite, campground, camping site, camping ground, bivouac, encampment, camping area, beach, plage, natural elevation, elevation, highland upland, hill, mountain mount, ridge, gum tree.
- **Comida**: food, nutrient, comfort food, soul food, fare, diet, vegetarianism.
- **Animal**: animal, animate being, beast, creature, fauna, domestic animal, domesticated animal, domestic dog, canis familiaris, domestic cat, house cat, felis domesticus.
- **Deporte**: sport, athletics, archery, athletic game, blood sport, contact sport, cycling, gymnastics, gymnastic exercise, acrobatics, tumbling, judo, outdoor sport, field sport, racing, riding horseback, rock climbing, rowing row, skating, skiing, team sport, track and field, water sports, aquatics.
- **Moda**: clothing, article of clothing, vesture, wear, wearable, habiliment, attire, garb, dress, black, loungewear, slops, tailor made, work clothing, work clothes.

Etiquetas usadas para buscar publicaciones en Instagram

- **Aire libre**: airelibre, airelibrechile, outdoor, outdoorchile, naturaleza, trekking, trekkingday, trekkingchile, hikking, nature, hikkingchile, glaciar, dunas, montaña, monta-

ñas, paisaje, landscape.

- **Comida**: comida, comidasana, hamburguesa, cheff, cocina, postre, parrilla, asado, veganfood, vegetariano, eat, sandwiches, sandwich, tallarines, comidacasera, banqueteria, sabor, comidachilena, platotipico, food, healthyfood.
- **Animal**: perro, perros, gato, gatos, mascota, mascotas, pet, pets, animales, vidaanimal, vidasalvaje, wildlife, mascotagram, petsofinstagram, doglover, cats, catlover, cachorro, instagato.
- **Deporte**: training, fitness, atlethic, crossfit, workout, snowboarding, triatlon, triathlon, strong, ejercicios, basquetbol, natacion, protriathlete, runner, maraton, deportechile, calistenia, calisteniachile, fit, deporte, deporteextremo, deporteysalud, entrenamiento, ejercicio, deporteesvida, entrenamientofuncional, crossfitchile.
- **Moda**: moda, modamujer, modahombre, fashion, style, lookingood, gala, alfombraroja, vestido, traje, elegante, tux, tuxedo, outfit, menwithstyle, womenwithstyle, estilo, instamoda.